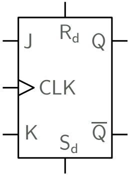
Digital Circuits: Part 5

M. B. Patil
mbpatil@ee.iitb.ac.in
www.ee.iitb.ac.in/~sequel

Department of Electrical Engineering
Indian Institute of Technology Bombay

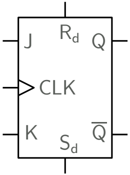
JK flip-flop: asynchronous inputs



S_d	R_d	CLK	J	K	Q_{n+1}
0	1	X	X	X	0
1	0	X	X	X	1
1	1	X	X	X	invalid
0	0	$\uparrow$	0	0	Q_n
0	0	$\uparrow$	0	1	0
0	0	$\uparrow$	1	0	1
0	0	$\uparrow$	1	1	$\overline{Q_n}$

normal operation

JK flip-flop: asynchronous inputs

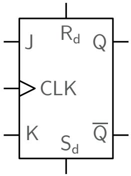


S_d	R_d	CLK	J	K	Q_{n+1}
0	1	X	X	X	0
1	0	X	X	X	1
1	1	X	X	X	invalid
0	0	$\uparrow$	0	0	Q_n
0	0	$\uparrow$	0	1	0
0	0	$\uparrow$	1	0	1
0	0	$\uparrow$	1	1	$\overline{Q_n}$

normal operation

* Clocked flip-flops are also provided with *asynchronous* or *direct* Set and Reset inputs, S_d and R_d, (also called Preset and Clear, respectively) which override all other inputs (J, K, CLK).

JK flip-flop: asynchronous inputs

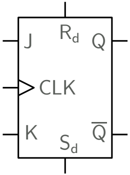


S_d	R_d	CLK	J	K	Q_{n+1}
0	1	X	X	X	0
1	0	X	X	X	1
1	1	X	X	X	invalid
0	0	$\uparrow$	0	0	Q_n
0	0	$\uparrow$	0	1	0
0	0	$\uparrow$	1	0	1
0	0	$\uparrow$	1	1	$\overline{Q_n}$

normal operation

* Clocked flip-flops are also provided with *asynchronous* or *direct* Set and Reset inputs, S_d and R_d, (also called Preset and Clear, respectively) which override all other inputs (J, K, CLK).
* The S_d and R_d inputs may be active low; in that case, they are denoted by $\overline{S_d}$ and $\overline{R_d}$.

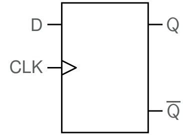
JK flip-flop: asynchronous inputs



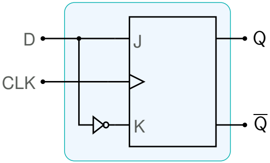
S_d	R_d	CLK	J	K	Q_{n+1}
0	1	X	X	X	0
1	0	X	X	X	1
1	1	X	X	X	invalid
0	0	↑	0	0	Q_n
0	0	↑	0	1	0
0	0	↑	1	0	1
0	0	↑	1	1	$\overline{Q_n}$

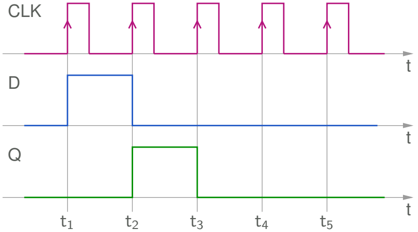
normal operation

* Clocked flip-flops are also provided with *asynchronous* or *direct* Set and Reset inputs, S_d and R_d, (also called Preset and Clear, respectively) which override all other inputs (J, K, CLK).
* The S_d and R_d inputs may be active low; in that case, they are denoted by $\overline{S_d}$ and $\overline{R_d}$.
* The asynchronous inputs are convenient for starting up a circuit in a known state.

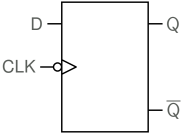


CLK	D	Q_{n+1}
↑	0	0
↑	1	1

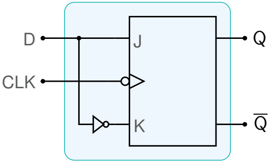


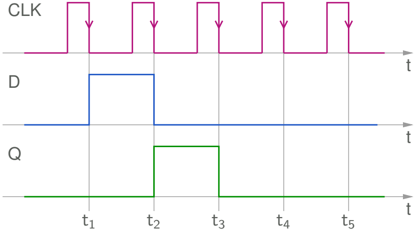


positive edge–triggered D flip–flop



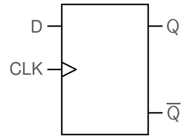
CLK	D	Q_{n+1}
↓	0	0
↓	1	1



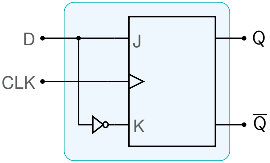


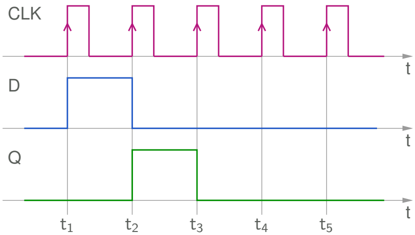
negative edge–triggered D flip–flop

D flip-flop

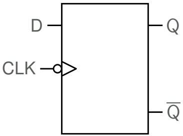


CLK	D	Q_{n+1}
↑	0	0
↑	1	1

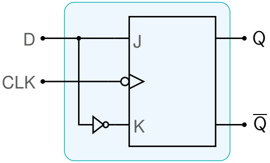


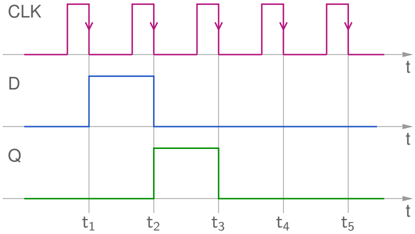


positive edge–triggered D flip–flop



CLK	D	Q_{n+1}
↓	0	0
↓	1	1

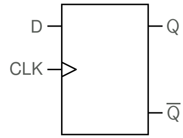




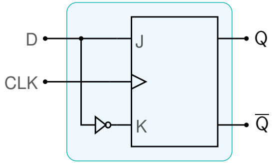
negative edge–triggered D flip–flop

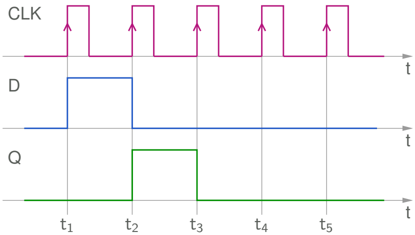
* The D flip-flop can be used to *delay* the Data (D) signal by one clock period.

D flip-flop

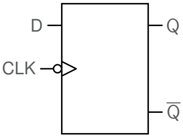


CLK	D	Q_{n+1}
↑	0	0
↑	1	1

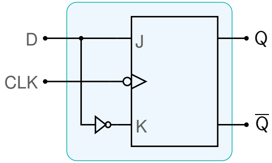


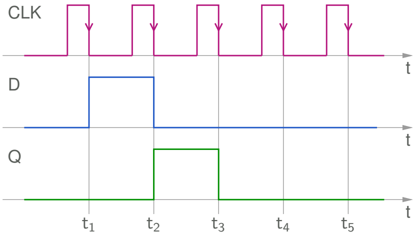


positive edge–triggered D flip–flop



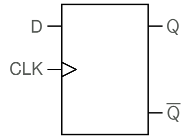
CLK	D	Q_{n+1}
↓	0	0
↓	1	1



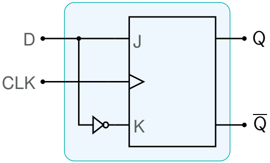


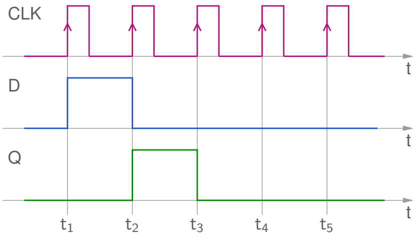
negative edge–triggered D flip–flop

* The D flip-flop can be used to *delay* the Data (D) signal by one clock period.
* With $J = D$, $K = \overline{D}$, we have either $J = 0$, $K = 1$ or $J = 1$, $K = 0$; the next Q is 0 in the first case, 1 in the second case.

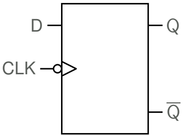


CLK	D	Q_{n+1}
↑	0	0
↑	1	1

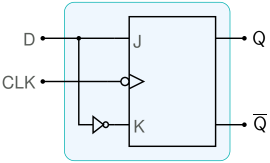


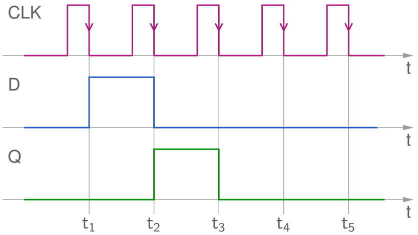


positive edge–triggered D flip–flop



CLK	D	Q_{n+1}
↓	0	0
↓	1	1

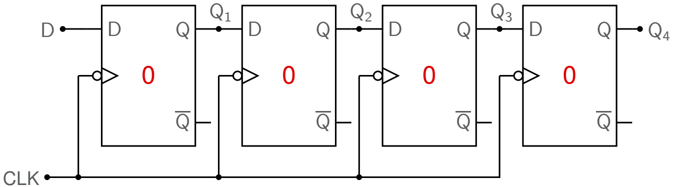




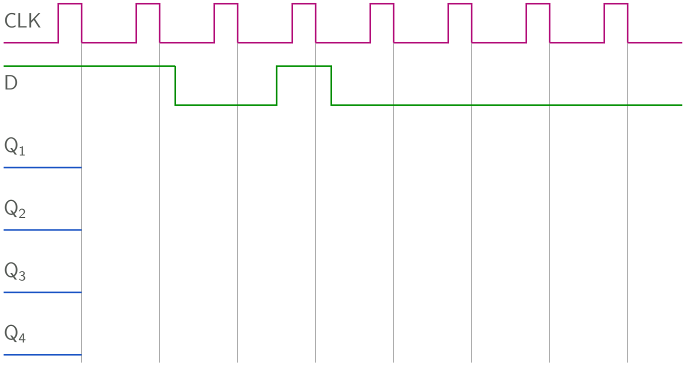
negative edge–triggered D flip–flop

* The D flip-flop can be used to *delay* the Data (D) signal by one clock period.
* With $J = D$, $K = \overline{D}$, we have either $J = 0$, $K = 1$ or $J = 1$, $K = 0$; the next Q is 0 in the first case, 1 in the second case.
* Instead of a JK flip-flop, an RS flip-flop can also be used to make a D flip-flop, with $S = D$, $R = \overline{D}$.

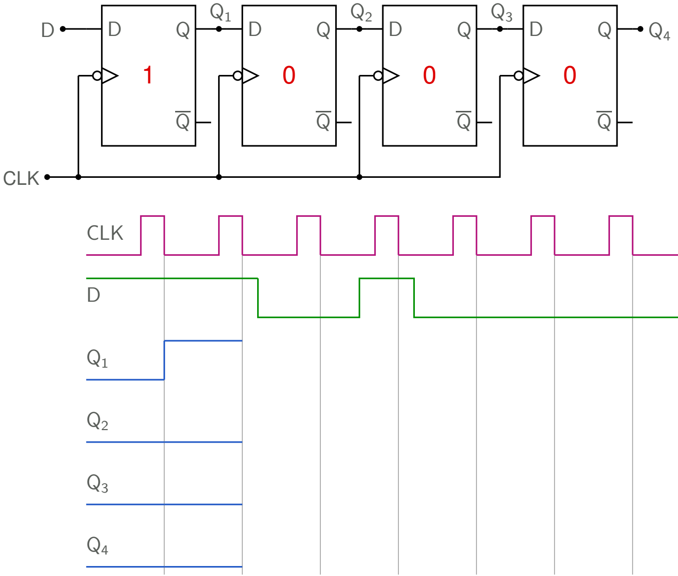
Shift register



CLK	D	Q_{n+1}
$\downarrow$	0	0
$\downarrow$	1	1

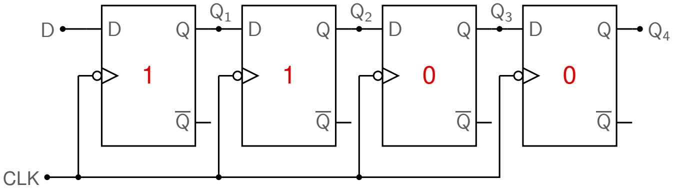


Shift register

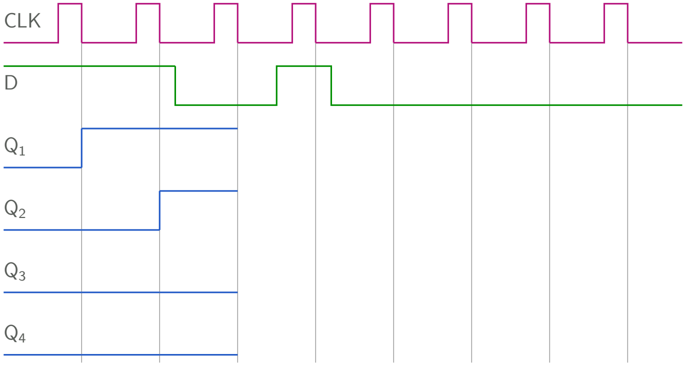


CLK	D	Q_{n+1}
↓	0	0
↓	1	1

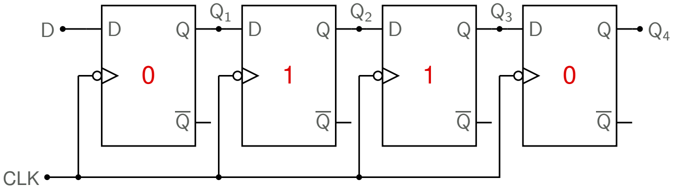
Shift register



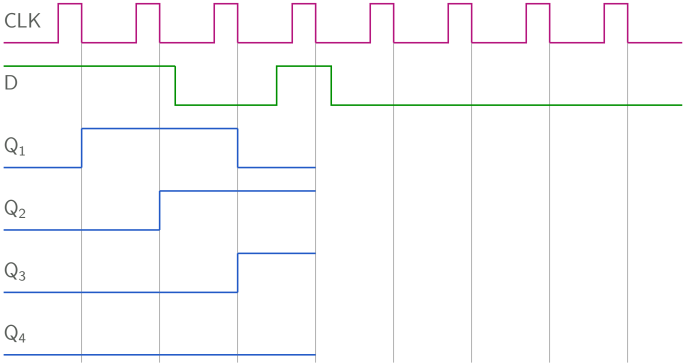
CLK	D	Q_{n+1}
$\downarrow$	0	0
$\downarrow$	1	1



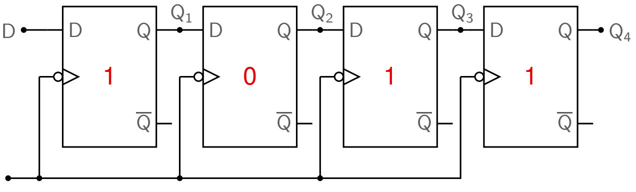
Shift register



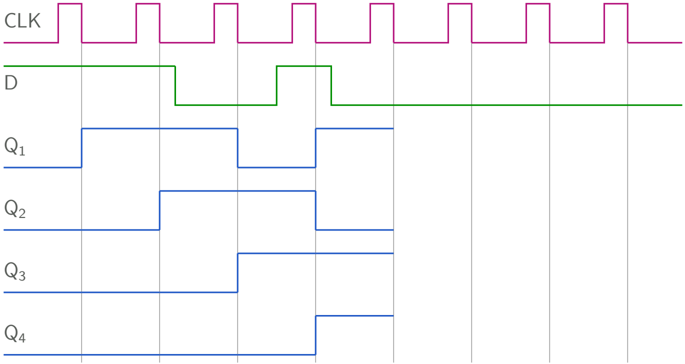
CLK	D	Q_{n+1}
$\downarrow$	0	0
$\downarrow$	1	1



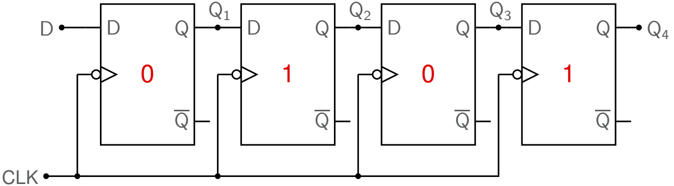
Shift register



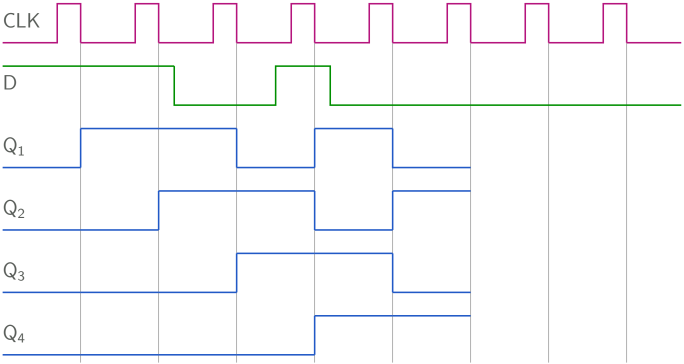
CLK	D	Q_{n+1}
↓	0	0
↓	1	1



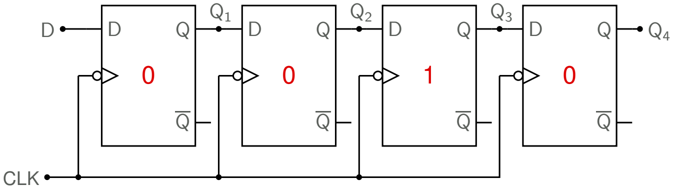
Shift register



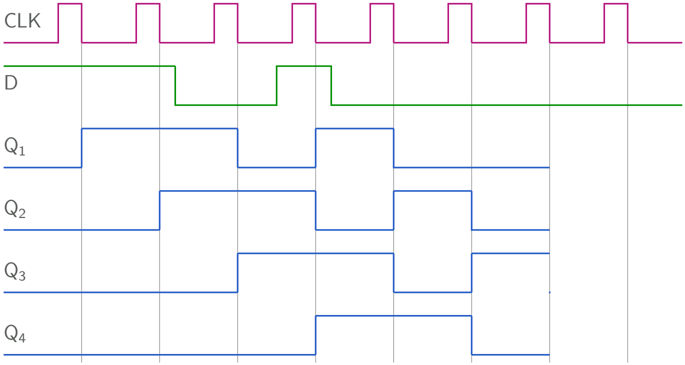
CLK	D	Q_{n+1}
$\downarrow$	0	0
$\downarrow$	1	1



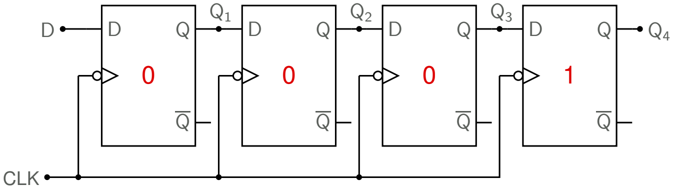
Shift register



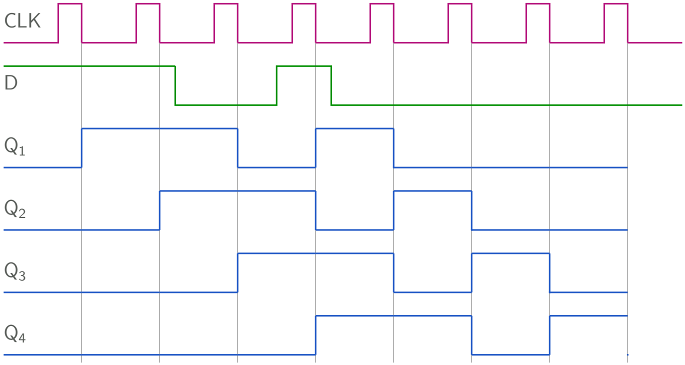
CLK	D	Q_{n+1}
↓	0	0
↓	1	1



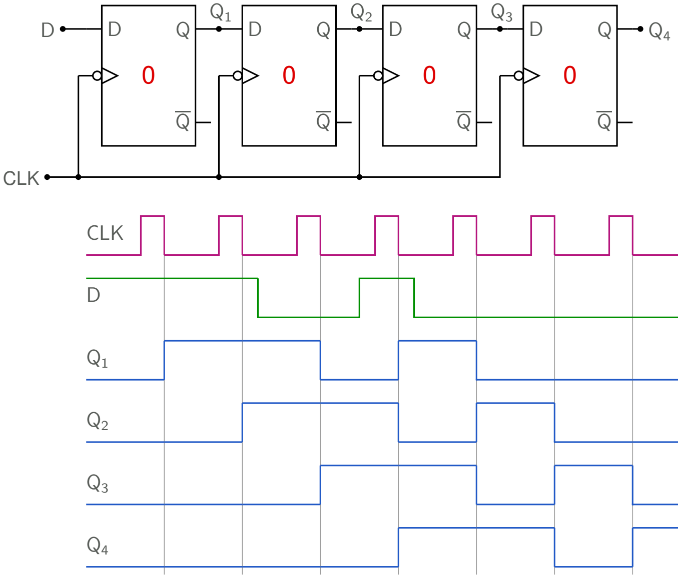
Shift register



CLK	D	Q_{n+1}
↓	0	0
↓	1	1

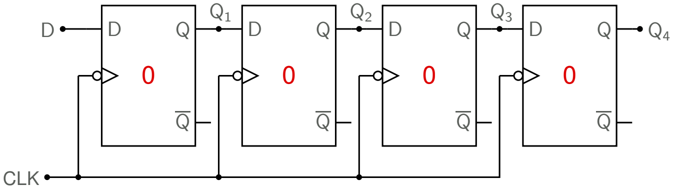


Shift register

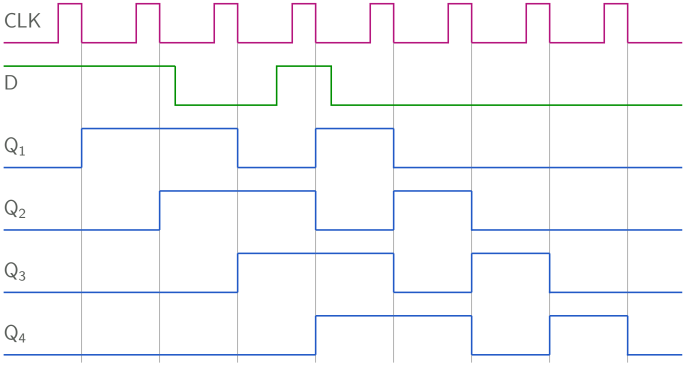


CLK	D	Q_{n+1}
↓	0	0
↓	1	1

Shift register

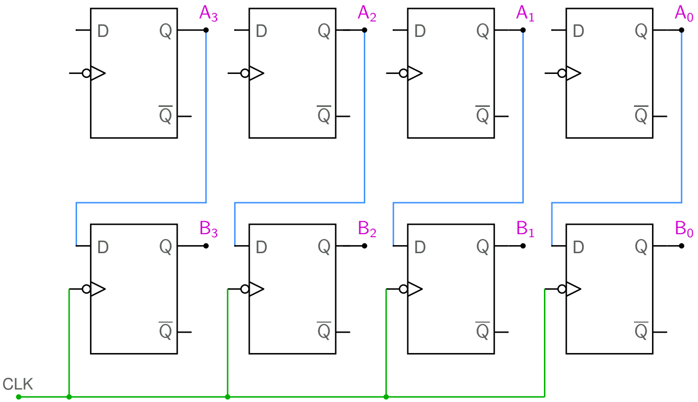


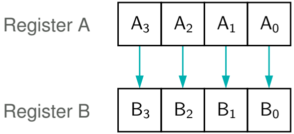
CLK	D	Q_{n+1}
↓	0	0
↓	1	1



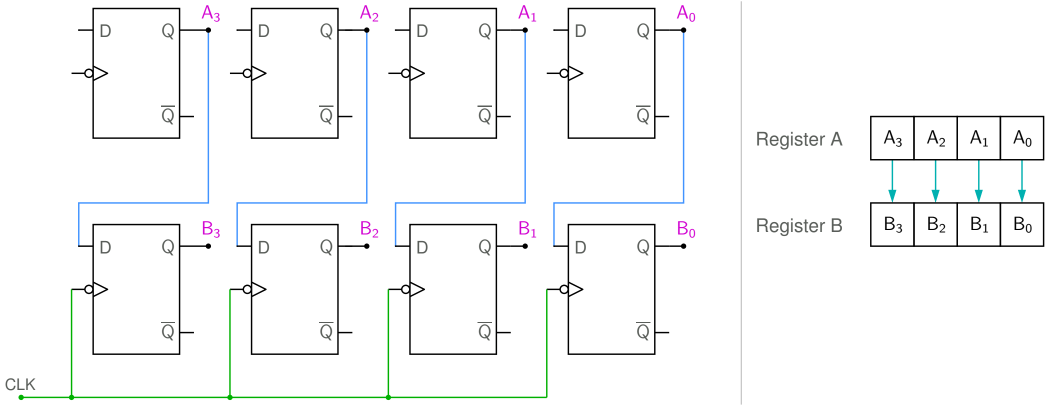
SEQUEL file: `ee101_shift_reg_1.sqproj`

Parallel transfer between shift registers



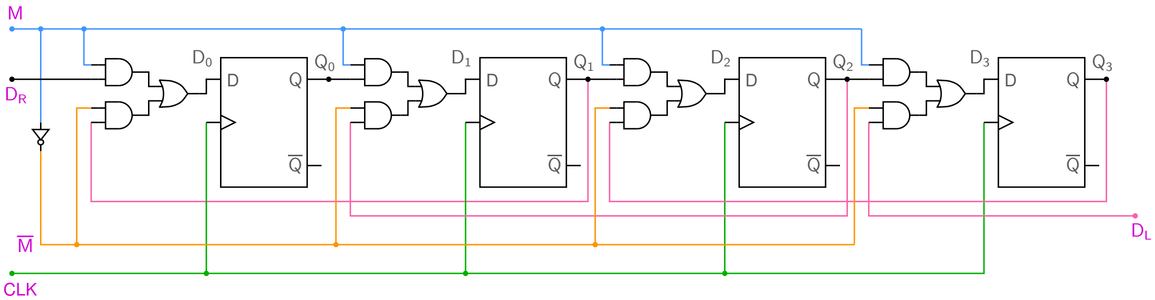


Parallel transfer between shift registers

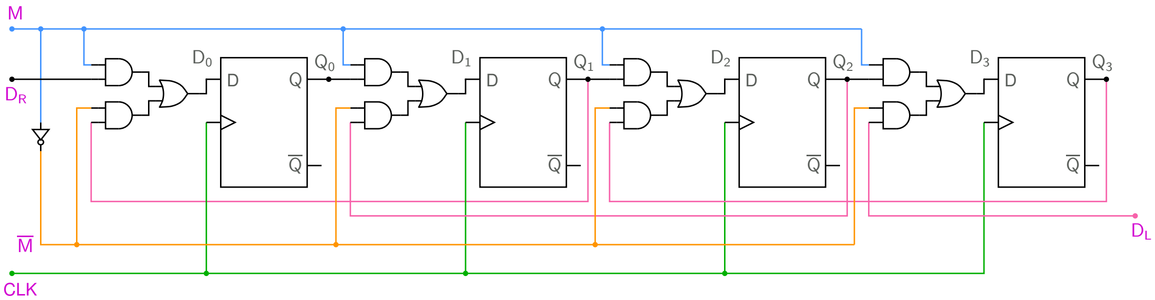


* After the active clock edge, the contents of the A register ($A_3A_2A_1A_0$) are copied to the B register.

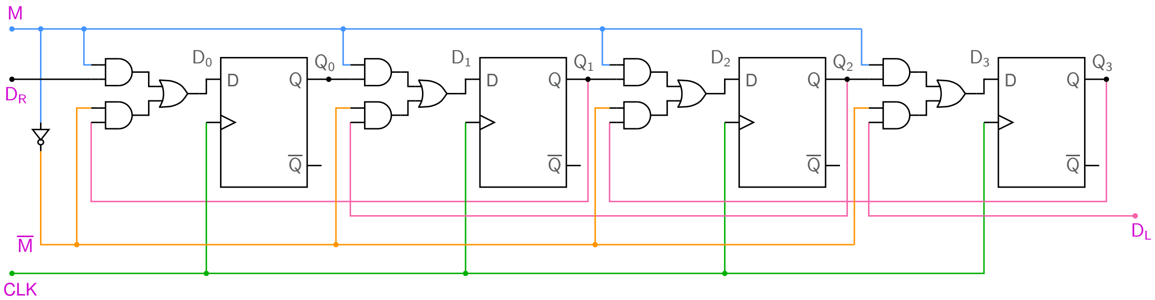
Bidirectional shift register



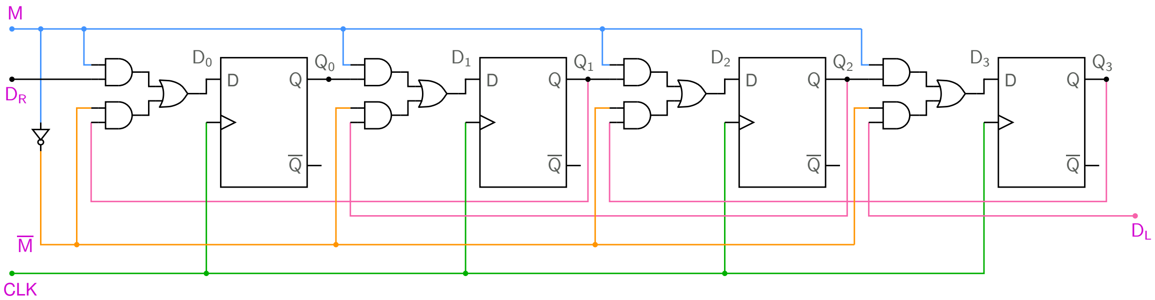
Bidirectional shift register



* When the mode input (M) is 1, we have
  $D_0 = D_R, \; D_1 = Q_0, \; D_2 = Q_1, \; D_3 = Q_2.$



* When the mode input (M) is 1, we have
  $D_0 = D_R,\ D_1 = Q_0,\ D_2 = Q_1,\ D_3 = Q_2$.
* When the mode input (M) is 0, we have
  $D_0 = Q_1,\ D_1 = Q_2,\ D_2 = Q_3,\ D_3 = D_L$.



* When the mode input (M) is 1, we have
  $D_0 = D_R$, $D_1 = Q_0$, $D_2 = Q_1$, $D_3 = Q_2$.
* When the mode input (M) is 0, we have
  $D_0 = Q_1$, $D_1 = Q_2$, $D_2 = Q_3$, $D_3 = D_L$.
* M = 1 $\rightarrow$ shift right operation.
  M = 0 $\rightarrow$ shift left operation.

Shift left operation

	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0		
original number	0	0	0	0	1	1	0	1	0	dec. 13

Shift left operation

	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0		
original number	0	0	0	0	1	1	0	1	0	dec. 13
after shift left	0	0	0	1	1	0	1	0		dec. 26

Shift left operation

	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0		
original number	0	0	0	0	1	1	0	1	0	dec. 13
after shift left	0	0	0	1	1	0	1	0		dec. 26

Shift left $\rightarrow\ \times 2$

Multiplication using shift and add

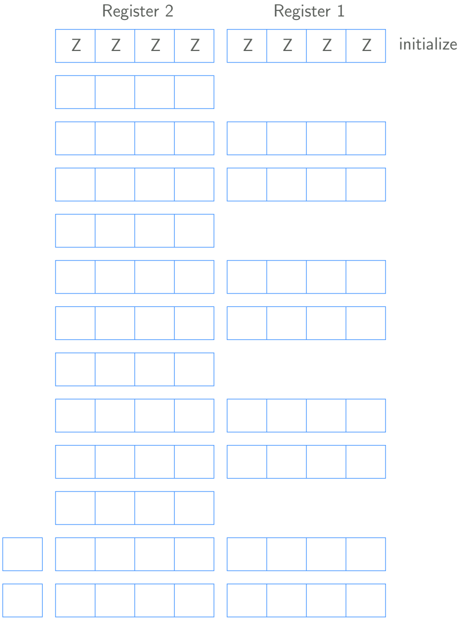
					1	0	1	1	$A_3A_2A_1A_0$	(decimal 11)
				$\times$	1	1	0	1	$B_3B_2B_1B_0$	(decimal 13)
					1	0	1	1	since $B_0 = 1$	
+				0	0	0	0	Z	since $B_1 = 0$	
				0	1	0	1	1	addition	
+			1	0	1	1	Z	Z	since $B_2 = 1$	
			1	1	0	1	1	1	addition	
+		1	0	1	1	Z	Z	Z	since $B_3 = 1$	
	1	0	0	0	1	1	1	1	addition	(decimal 143)

Note that $Z = 0$. We use Z to denote 0s which are independent of the numbers being multiplied.

Multiplication using shift and add

					1	0	1	1	$A_3A_2A_1A_0$	(decimal 11)
				×	1	1	0	1	$B_3B_2B_1B_0$	(decimal 13)
					1	0	1	1	since $B_0 = 1$	
+				0	0	0	0	Z	since $B_1 = 0$	
				0	1	0	1	1	addition	
+			1	0	1	1	Z	Z	since $B_2 = 1$	
			1	1	0	1	1	1	addition	
+		1	0	1	1	Z	Z	Z	since $B_3 = 1$	
	1	0	0	0	1	1	1	1	addition	(decimal 143)

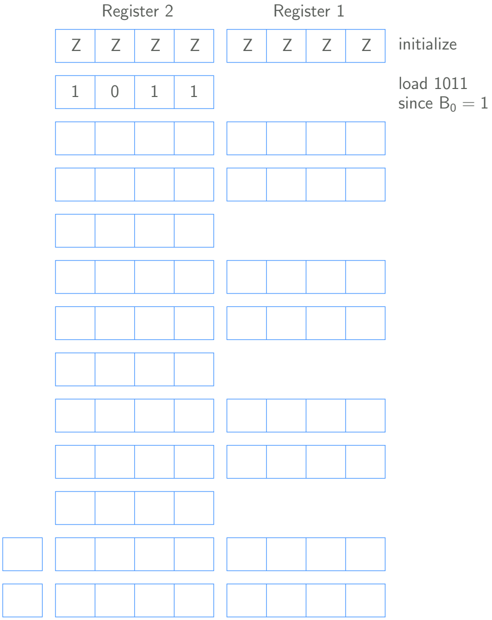
Note that $Z = 0$. We use Z to denote 0s which are independent of the numbers being multiplied.



Multiplication using shift and add

					1	0	1	1	$A_3A_2A_1A_0$	(decimal 11)
				×	1	1	0	1	$B_3B_2B_1B_0$	(decimal 13)
					1	0	1	1	since $B_0 = 1$	
+				0	0	0	0	Z	since $B_1 = 0$	
				0	1	0	1	1	addition	
+			1	0	1	1	Z	Z	since $B_2 = 1$	
			1	1	0	1	1	1	addition	
+		1	0	1	1	Z	Z	Z	since $B_3 = 1$	
1	0	0	0	1	1	1	1		addition	(decimal 143)

Note that $Z = 0$. We use Z to denote 0s which are independent of the numbers being multiplied.



Multiplication using shift and add

					1	0	1	1	$A_3A_2A_1A_0$	(decimal 11)
				$\times$	1	1	0	1	$B_3B_2B_1B_0$	(decimal 13)
					1	0	1	1	since $B_0 = 1$	
+				0	0	0	0	Z	since $B_1 = 0$	
				0	1	0	1	1	addition	
+			1	0	1	1	Z	Z	since $B_2 = 1$	
			1	1	0	1	1	1	addition	
+		1	0	1	1	Z	Z	Z	since $B_3 = 1$	
	1	0	0	0	1	1	1	1	addition	(decimal 143)

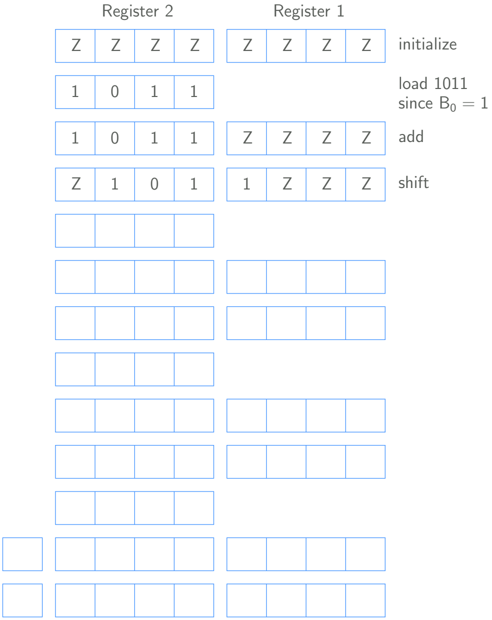
Note that $Z = 0$. We use Z to denote 0s which are independent of the numbers being multiplied.

	Register 2				Register 1				
	Z	Z	Z	Z	Z	Z	Z	Z	initialize
	1	0	1	1					load 1011 since $B_0 = 1$
	1	0	1	1	Z	Z	Z	Z	add

Multiplication using shift and add

					1	0	1	1	$A_3A_2A_1A_0$	(decimal 11)
				$\times$	1	1	0	1	$B_3B_2B_1B_0$	(decimal 13)
					1	0	1	1	since $B_0 = 1$	
+				0	0	0	0	Z	since $B_1 = 0$	
				0	1	0	1	1	addition	
+			1	0	1	1	Z	Z	since $B_2 = 1$	
			1	1	0	1	1	1	addition	
+		1	0	1	1	Z	Z	Z	since $B_3 = 1$	
	1	0	0	0	1	1	1	1	addition	(decimal 143)

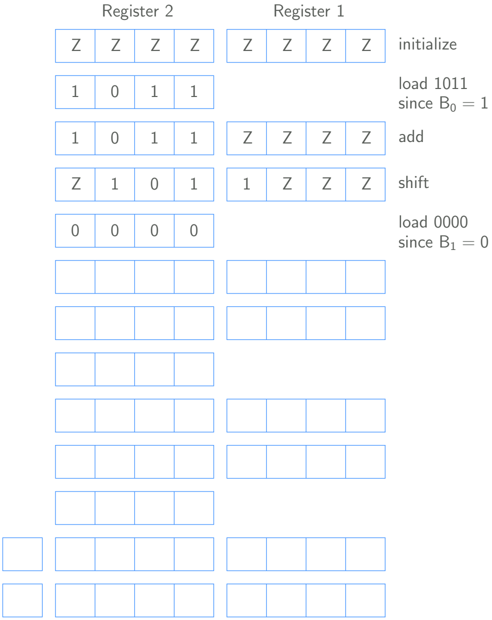
Note that Z = 0. We use Z to denote 0s which are independent of the numbers being multiplied.



Multiplication using shift and add

					1	0	1	1	$A_3A_2A_1A_0$ (decimal 11)
				$\times$	1	1	0	1	$B_3B_2B_1B_0$ (decimal 13)
					1	0	1	1	since $B_0 = 1$
$+$				0	0	0	0	Z	since $B_1 = 0$
				0	1	0	1	1	addition
$+$			1	0	1	1	Z	Z	since $B_2 = 1$
			1	1	0	1	1	1	addition
$+$		1	0	1	1	Z	Z	Z	since $B_3 = 1$
1	0	0	0	1	1	1	1		addition (decimal 143)

Note that $Z = 0$. We use Z to denote 0s which are independent of the numbers being multiplied.



Multiplication using shift and add

$$
\begin{array}{rcccccccl}
 & & & & 1 & 0 & 1 & 1 & A_3A_2A_1A_0 \quad \text{(decimal 11)} \\
 & & & \times & 1 & 1 & 0 & 1 & B_3B_2B_1B_0 \quad \text{(decimal 13)} \\
\hline
 & & & & 1 & 0 & 1 & 1 & \text{since } B_0 = 1 \\
+ & & & 0 & 0 & 0 & 0 & Z & \text{since } B_1 = 0 \\
\hline
 & & & 0 & 1 & 0 & 1 & 1 & \text{addition} \\
+ & & 1 & 0 & 1 & 1 & Z & Z & \text{since } B_2 = 1 \\
\hline
 & & 1 & 1 & 0 & 1 & 1 & 1 & \text{addition} \\
+ & 1 & 0 & 1 & 1 & Z & Z & Z & \text{since } B_3 = 1 \\
\hline
1 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & \text{addition} \quad \text{(decimal 143)}
\end{array}
$$

Note that $Z = 0$. We use Z to denote 0s which are independent of the numbers being multiplied.

	Register 2				Register 1				
	Z	Z	Z	Z	Z	Z	Z	Z	initialize
	1	0	1	1					load 1011 since $B_0 = 1$
	1	0	1	1	Z	Z	Z	Z	add
	Z	1	0	1	1	Z	Z	Z	shift
	0	0	0	0					load 0000 since $B_1 = 0$
	0	1	0	1	1	Z	Z	Z	add

Multiplication using shift and add

					1	0	1	1	$A_3A_2A_1A_0$	(decimal 11)
				×	1	1	0	1	$B_3B_2B_1B_0$	(decimal 13)
					1	0	1	1	since $B_0 = 1$	
+				0	0	0	0	Z	since $B_1 = 0$	
				0	1	0	1	1	addition	
+			1	0	1	1	Z	Z	since $B_2 = 1$	
			1	1	0	1	1	1	addition	
+		1	0	1	1	Z	Z	Z	since $B_3 = 1$	
	1	0	0	0	1	1	1	1	addition	(decimal 143)

Note that $Z = 0$. We use Z to denote 0s which are independent of the numbers being multiplied.

	Register 2				Register 1				
	Z	Z	Z	Z	Z	Z	Z	Z	initialize
	1	0	1	1					load 1011 since $B_0 = 1$
	1	0	1	1	Z	Z	Z	Z	add
	Z	1	0	1	1	Z	Z	Z	shift
	0	0	0	0					load 0000 since $B_1 = 0$
	0	1	0	1	1	Z	Z	Z	add
	Z	0	1	0	1	1	Z	Z	shift

Multiplication using shift and add

					1	0	1	1	$A_3A_2A_1A_0$ (decimal 11)
				$\times$	1	1	0	1	$B_3B_2B_1B_0$ (decimal 13)
					1	0	1	1	since $B_0 = 1$
+				0	0	0	0	Z	since $B_1 = 0$
				0	1	0	1	1	addition
+			1	0	1	1	Z	Z	since $B_2 = 1$
			1	1	0	1	1	1	addition
+		1	0	1	1	Z	Z	Z	since $B_3 = 1$
	1	0	0	0	1	1	1	1	addition (decimal 143)

Note that $Z = 0$. We use Z to denote 0s which are independent of the numbers being multiplied.

	Register 2				Register 1				
	Z	Z	Z	Z	Z	Z	Z	Z	initialize
	1	0	1	1					load 1011 since $B_0 = 1$
	1	0	1	1	Z	Z	Z	Z	add
	Z	1	0	1	1	Z	Z	Z	shift
	0	0	0	0					load 0000 since $B_1 = 0$
	0	1	0	1	1	Z	Z	Z	add
	Z	0	1	0	1	1	Z	Z	shift
	1	0	1	1					load 1011 since $B_2 = 1$

Multiplication using shift and add

					1	0	1	1	$A_3A_2A_1A_0$	(decimal 11)
				×	1	1	0	1	$B_3B_2B_1B_0$	(decimal 13)
					1	0	1	1	since $B_0 = 1$	
+				0	0	0	0	Z	since $B_1 = 0$	
				0	1	0	1	1	addition	
+			1	0	1	1	Z	Z	since $B_2 = 1$	
			1	1	0	1	1	1	addition	
+		1	0	1	1	Z	Z	Z	since $B_3 = 1$	
	1	0	0	0	1	1	1	1	addition	(decimal 143)

Note that $Z = 0$. We use Z to denote 0s which are independent of the numbers being multiplied.

	Register 2				Register 1				
	Z	Z	Z	Z	Z	Z	Z	Z	initialize
	1	0	1	1					load 1011 since $B_0 = 1$
	1	0	1	1	Z	Z	Z	Z	add
	Z	1	0	1	1	Z	Z	Z	shift
	0	0	0	0					load 0000 since $B_1 = 0$
	0	1	0	1	1	Z	Z	Z	add
	Z	0	1	0	1	1	Z	Z	shift
	1	0	1	1					load 1011 since $B_2 = 1$
	1	1	0	1	1	1	Z	Z	add

Multiplication using shift and add

					1	0	1	1	$A_3A_2A_1A_0$	(decimal 11)
				$\times$	1	1	0	1	$B_3B_2B_1B_0$	(decimal 13)
					1	0	1	1	since $B_0 = 1$	
+				0	0	0	0	Z	since $B_1 = 0$	
				0	1	0	1	1	addition	
+			1	0	1	1	Z	Z	since $B_2 = 1$	
			1	1	0	1	1	1	addition	
+		1	0	1	1	Z	Z	Z	since $B_3 = 1$	
	1	0	0	0	1	1	1	1	addition	(decimal 143)

Note that Z = 0. We use Z to denote 0s which are independent of the numbers being multiplied.

	Register 2				Register 1				
	Z	Z	Z	Z	Z	Z	Z	Z	initialize
	1	0	1	1					load 1011 since $B_0 = 1$
	1	0	1	1	Z	Z	Z	Z	add
	Z	1	0	1	1	Z	Z	Z	shift
	0	0	0	0					load 0000 since $B_1 = 0$
	0	1	0	1	1	Z	Z	Z	add
	Z	0	1	0	1	1	Z	Z	shift
	1	0	1	1					load 1011 since $B_2 = 1$
	1	1	0	1	1	1	Z	Z	add
	Z	1	1	0	1	1	1	Z	shift

Multiplication using shift and add

				1	0	1	1	$A_3A_2A_1A_0$	(decimal 11)
			$\times$	1	1	0	1	$B_3B_2B_1B_0$	(decimal 13)
				1	0	1	1	since $B_0 = 1$	
+			0	0	0	0	Z	since $B_1 = 0$	
			0	1	0	1	1	addition	
+		1	0	1	1	Z	Z	since $B_2 = 1$	
		1	1	0	1	1	1	addition	
+	1	0	1	1	Z	Z	Z	since $B_3 = 1$	
1	0	0	0	1	1	1	1	addition	(decimal 143)

Note that $Z = 0$. We use Z to denote 0s which are independent of the numbers being multiplied.

	Register 2				Register 1				
	Z	Z	Z	Z	Z	Z	Z	Z	initialize
	1	0	1	1					load 1011 since $B_0 = 1$
	1	0	1	1	Z	Z	Z	Z	add
	Z	1	0	1	1	Z	Z	Z	shift
	0	0	0	0					load 0000 since $B_1 = 0$
	0	1	0	1	1	Z	Z	Z	add
	Z	0	1	0	1	1	Z	Z	shift
	1	0	1	1					load 1011 since $B_2 = 1$
	1	1	0	1	1	1	Z	Z	add
	Z	1	1	0	1	1	1	Z	shift
	1	0	1	1					load 1011 since $B_3 = 1$

Multiplication using shift and add

					1	0	1	1	$A_3A_2A_1A_0$	(decimal 11)
				$\times$	1	1	0	1	$B_3B_2B_1B_0$	(decimal 13)
					1	0	1	1	since $B_0 = 1$	
+				0	0	0	0	Z	since $B_1 = 0$	
				0	1	0	1	1	addition	
+			1	0	1	1	Z	Z	since $B_2 = 1$	
			1	1	0	1	1	1	addition	
+		1	0	1	1	Z	Z	Z	since $B_3 = 1$	
1	0	0	0	1	1	1	1		addition	(decimal 143)

Note that $Z = 0$. We use Z to denote 0s which are independent of the numbers being multiplied.

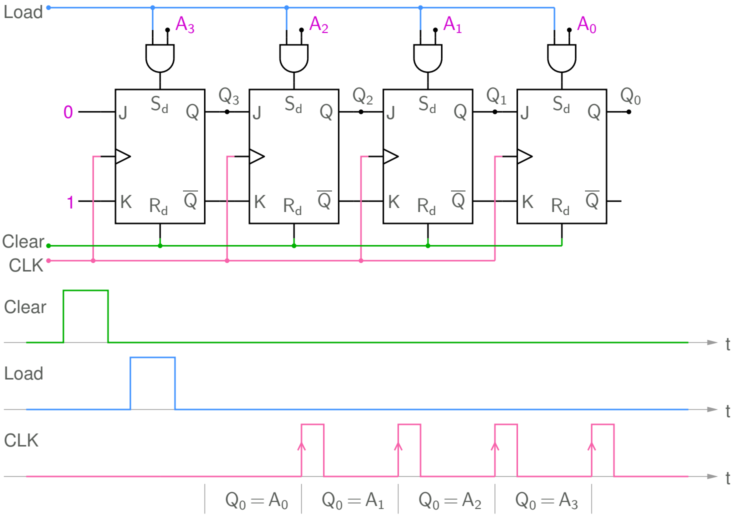
	Register 2				Register 1				
	Z	Z	Z	Z	Z	Z	Z	Z	initialize
	1	0	1	1					load 1011 since $B_0 = 1$
	1	0	1	1	Z	Z	Z	Z	add
	Z	1	0	1	1	Z	Z	Z	shift
	0	0	0	0					load 0000 since $B_1 = 0$
	0	1	0	1	1	Z	Z	Z	add
	Z	0	1	0	1	1	Z	Z	shift
	1	0	1	1					load 1011 since $B_2 = 1$
	1	1	0	1	1	1	Z	Z	add
	Z	1	1	0	1	1	1	Z	shift
	1	0	1	1					load 1011 since $B_3 = 1$
1	0	0	0	1	1	1	1	Z	add

Multiplication using shift and add

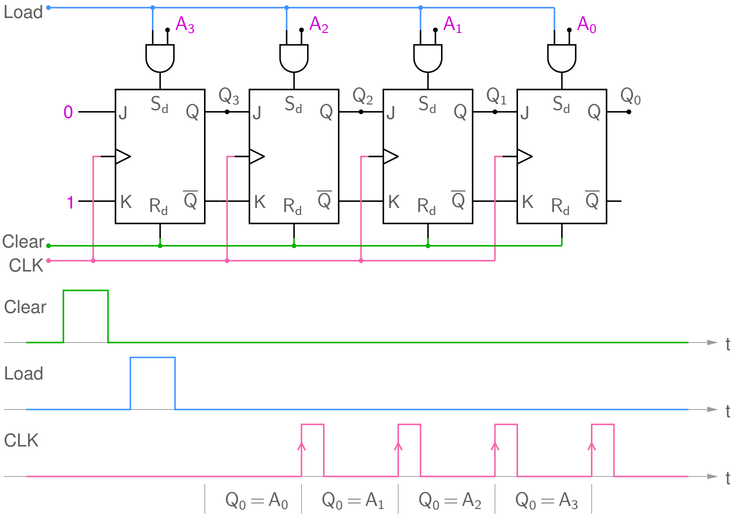
					1	0	1	1	$A_3A_2A_1A_0$	(decimal 11)
				×	1	1	0	1	$B_3B_2B_1B_0$	(decimal 13)
					1	0	1	1	since $B_0 = 1$	
+				0	0	0	0	Z	since $B_1 = 0$	
				0	1	0	1	1	addition	
+			1	0	1	1	Z	Z	since $B_2 = 1$	
			1	1	0	1	1	1	addition	
+		1	0	1	1	Z	Z	Z	since $B_3 = 1$	
	1	0	0	0	1	1	1	1	addition	(decimal 143)

Note that $Z = 0$. We use Z to denote 0s which are independent of the numbers being multiplied.

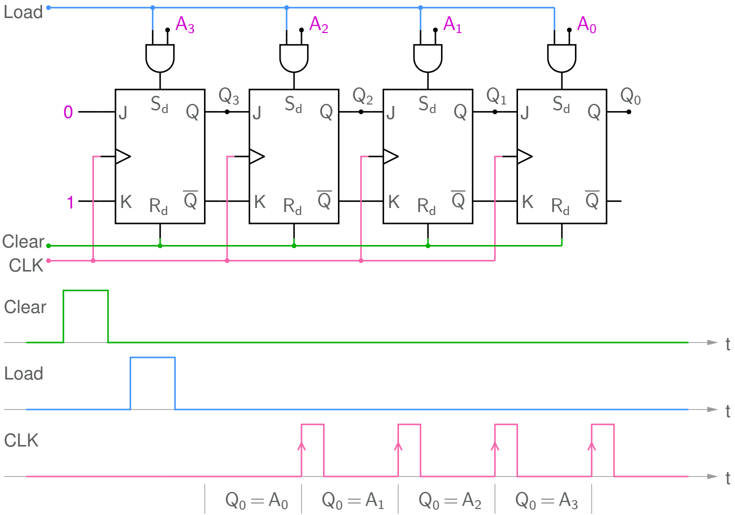
	Register 2				Register 1				
	Z	Z	Z	Z	Z	Z	Z	Z	initialize
	1	0	1	1					load 1011 since $B_0 = 1$
	1	0	1	1	Z	Z	Z	Z	add
	Z	1	0	1	1	Z	Z	Z	shift
	0	0	0	0					load 0000 since $B_1 = 0$
	0	1	0	1	1	Z	Z	Z	add
	Z	0	1	0	1	1	Z	Z	shift
	1	0	1	1					load 1011 since $B_2 = 1$
	1	1	0	1	1	1	Z	Z	add
	Z	1	1	0	1	1	1	Z	shift
	1	0	1	1					load 1011 since $B_3 = 1$
1	0	0	0	1	1	1	1	Z	add
Z	1	0	0	0	1	1	1	1	shift



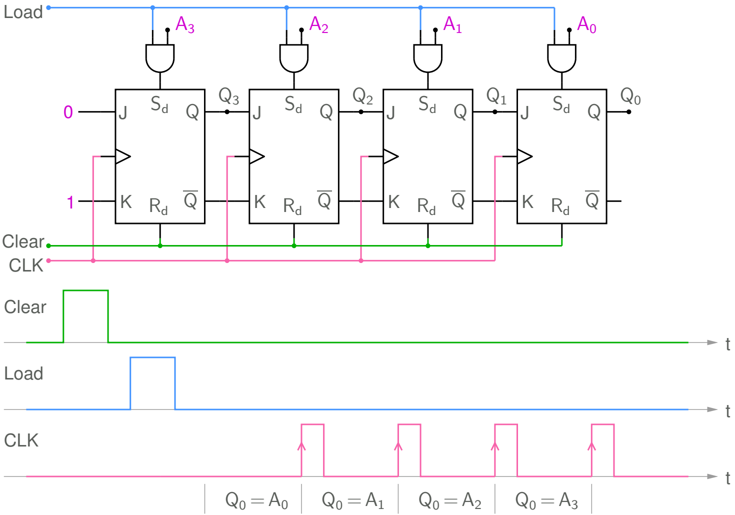
Load
A3
A2
A1
A0
Sd
Q
Q3
Q2
Q1
Q0
0
1
J
K
Rd
Q̄
Clear
CLK
t
Q0 = A0
Q0 = A1
Q0 = A2
Q0 = A3



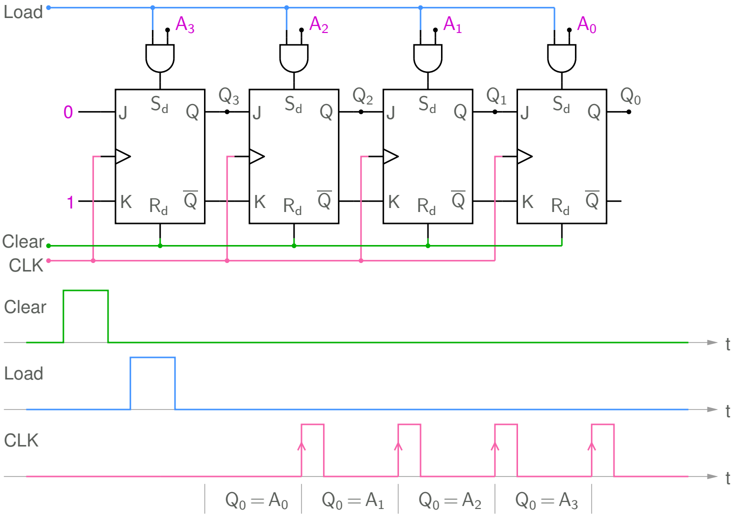
∗ All flip-flops are cleared in the beginning (with $R_d = \text{Clear} = 1,\ S_d = 0$).

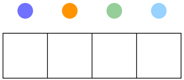


* All flip-flops are cleared in the beginning (with $R_d = \text{Clear} = 1,\ S_d = 0$).
* When Load $= 1,\ S_d = A_i,\ R_d = 0 \rightarrow A_i$ gets loaded into the i^{th} flip-flop.

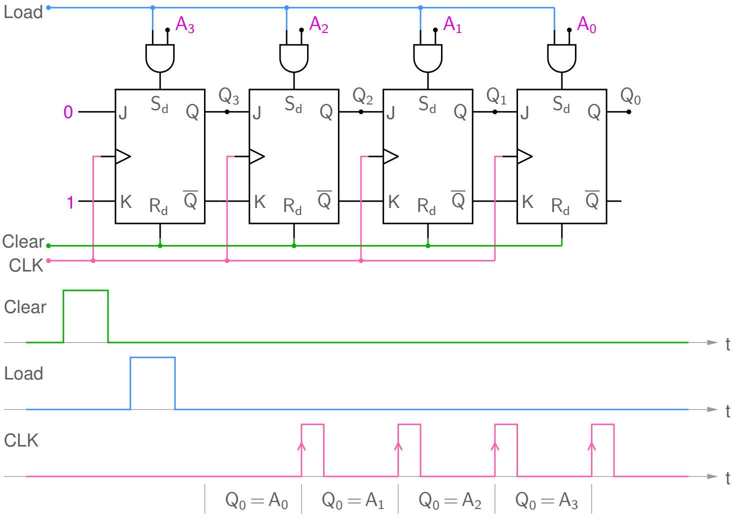


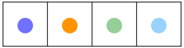
* All flip-flops are cleared in the beginning (with $R_d = \text{Clear} = 1$, $S_d = 0$).
* When Load = 1, $S_d = A_i$, $R_d = 0 \rightarrow A_i$ gets loaded into the i^{th} flip-flop.
* Subsequently, with every clock pulse, the data shifts right and appears *serially* at the output Q_0.
  $\rightarrow$ parallel in-serial out data movement



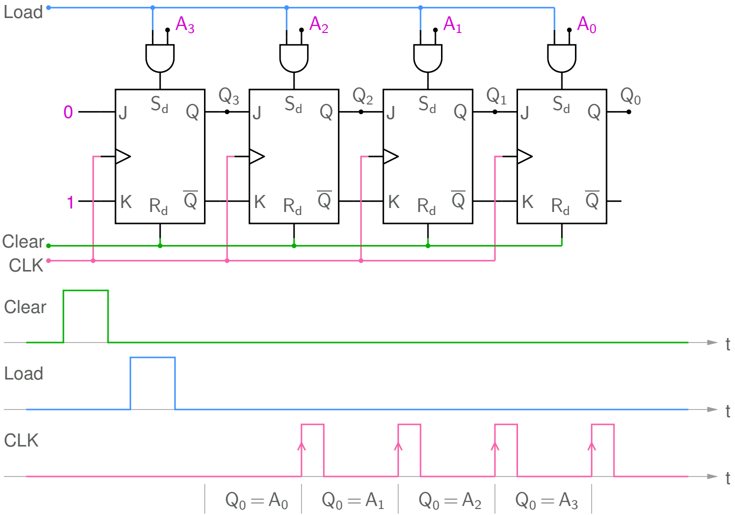


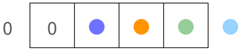
* All flip-flops are cleared in the beginning (with $R_d = \text{Clear} = 1$, $S_d = 0$).
* When Load $= 1$, $S_d = A_i$, $R_d = 0$ $\rightarrow$ A_i gets loaded into the i^{th} flip-flop.
* Subsequently, with every clock pulse, the data shifts right and appears *serially* at the output Q_0.
  $\rightarrow$ parallel in-serial out data movement



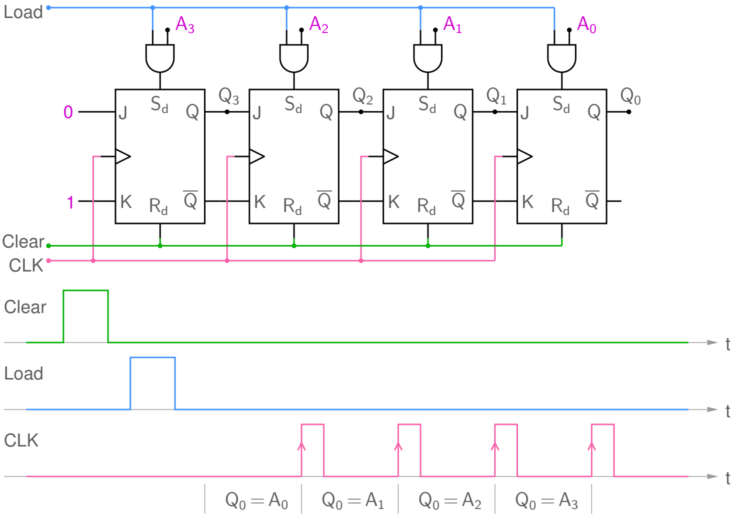


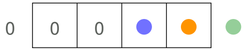
- $*$ All flip-flops are cleared in the beginning (with $R_d = \text{Clear} = 1,\ S_d = 0$).
- $*$ When Load $= 1$, $S_d = A_i$, $R_d = 0 \rightarrow A_i$ gets loaded into the i^{th} flip-flop.
- $*$ Subsequently, with every clock pulse, the data shifts right and appears *serially* at the output Q_0.
  $\rightarrow$ parallel in-serial out data movement



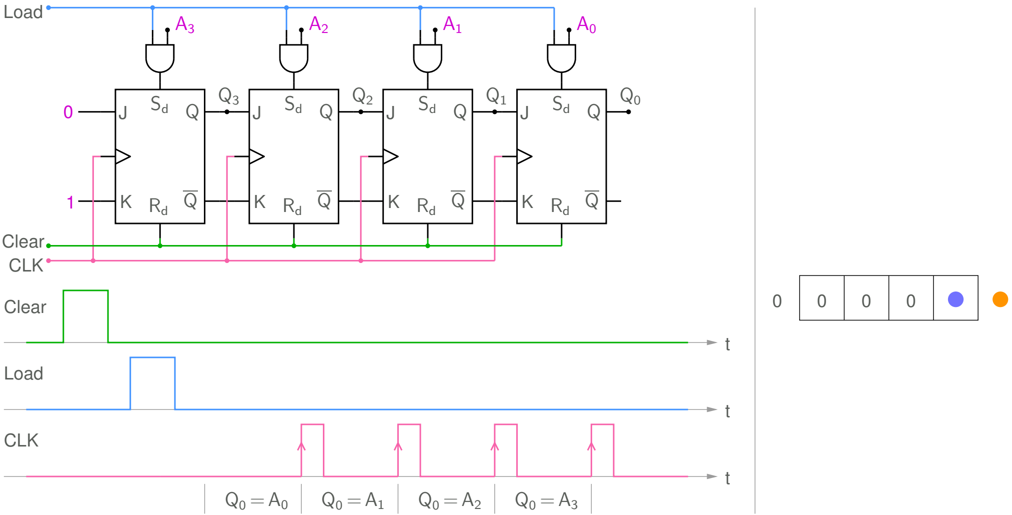


- ∗ All flip-flops are cleared in the beginning (with $R_d = \text{Clear} = 1$, $S_d = 0$).
- ∗ When $\text{Load} = 1$, $S_d = A_i$, $R_d = 0 \rightarrow A_i$ gets loaded into the i^{th} flip-flop.
- ∗ Subsequently, with every clock pulse, the data shifts right and appears *serially* at the output Q_0.
  $\rightarrow$ parallel in-serial out data movement

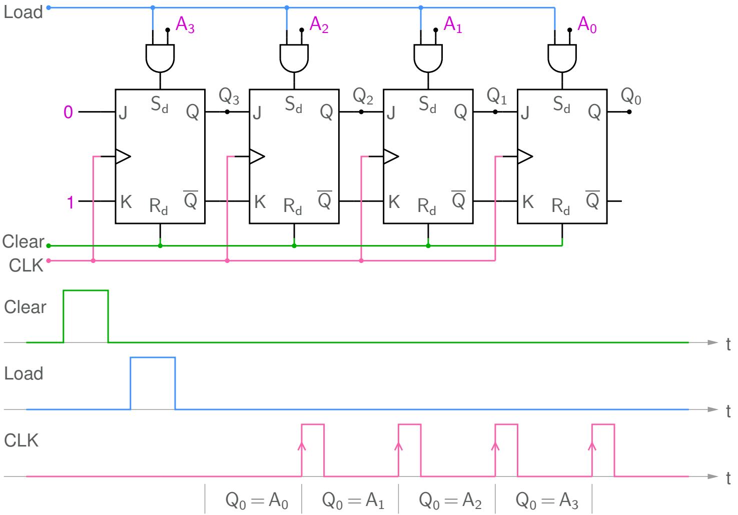


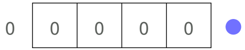


* All flip-flops are cleared in the beginning (with $R_d = \text{Clear} = 1$, $S_d = 0$).
* When $\text{Load} = 1$, $S_d = A_i$, $R_d = 0 \rightarrow A_i$ gets loaded into the i^{th} flip-flop.
* Subsequently, with every clock pulse, the data shifts right and appears *serially* at the output Q_0.
  $\rightarrow$ parallel in-serial out data movement

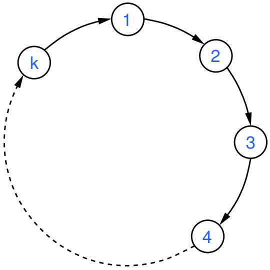


* All flip-flops are cleared in the beginning (with $R_d = \text{Clear} = 1,\ S_d = 0$).
* When Load $= 1$, $S_d = A_i$, $R_d = 0 \rightarrow A_i$ gets loaded into the i^{th} flip-flop.
* Subsequently, with every clock pulse, the data shifts right and appears *serially* at the output Q_0.
  $\rightarrow$ parallel in-serial out data movement

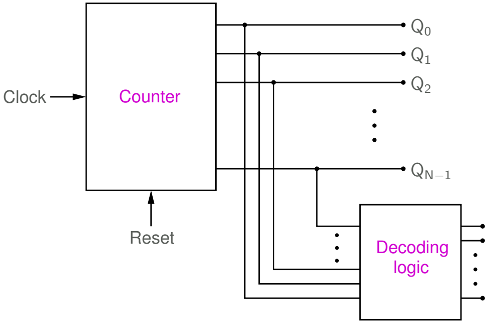




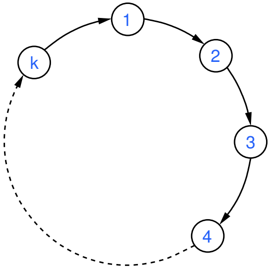
* All flip-flops are cleared in the beginning (with $R_d = \text{Clear} = 1,\ S_d = 0$).
* When Load $= 1$, $S_d = A_i$, $R_d = 0 \rightarrow A_i$ gets loaded into the i^{th} flip-flop.
* Subsequently, with every clock pulse, the data shifts right and appears *serially* at the output Q_0.
  $\rightarrow$ parallel in-serial out data movement



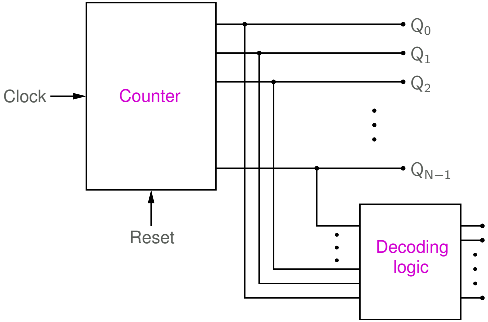
State transition diagram



General configuration

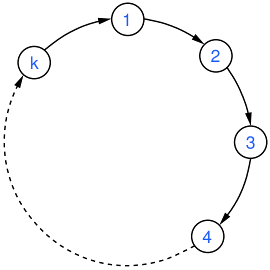


State transition diagram

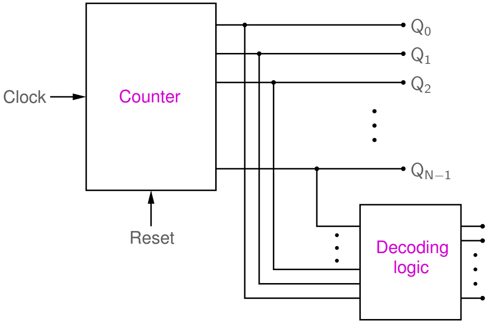


General configuration

* A counter with k states is called a modulo-k (mod-k) counter.

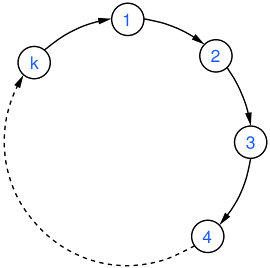


State transition diagram

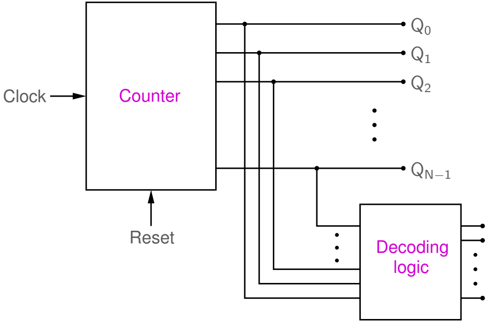


General configuration

* A counter with k states is called a modulo-k (mod-k) counter.
* A counter can be made with flip-flops, each flip-flop serving as a memory element with two states (0 or 1).

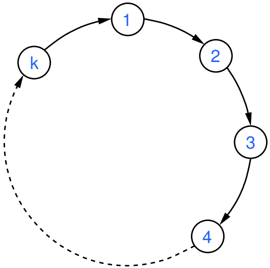


State transition diagram

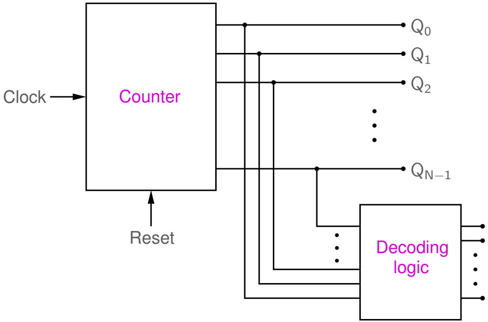


General configuration

* A counter with k states is called a modulo-k (mod-k) counter.
* A counter can be made with flip-flops, each flip-flop serving as a memory element with two states (0 or 1).
* If there are N flip-flops in a counter, there are 2^N possible states (since each flip-flop can have $Q = 0$ or $Q = 1$). It is possible to exclude some of these states.
  $\rightarrow$ N flip-flops can be used to make a mod-k counter with $k \leq 2^N$.



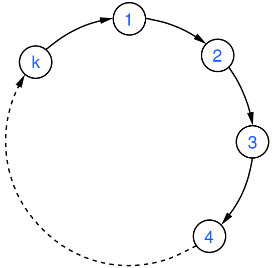
State transition diagram



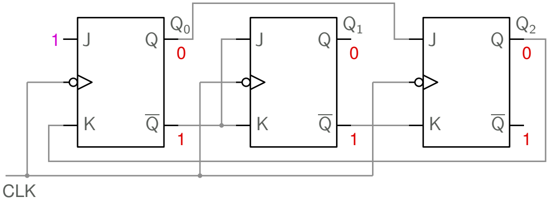
General configuration

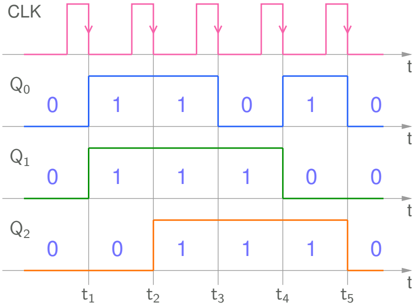
* A counter with k states is called a modulo-k (mod-k) counter.
* A counter can be made with flip-flops, each flip-flop serving as a memory element with two states (0 or 1).
* If there are N flip-flops in a counter, there are 2^N possible states (since each flip-flop can have $Q=0$ or $Q=1$). It is possible to exclude some of these states.
  $\rightarrow$ N flip-flops can be used to make a mod-k counter with $k \leq 2^N$.
* Typically, a reset facility is also provided, which can be used to force a certain state to initialize the counter.

Counters: example

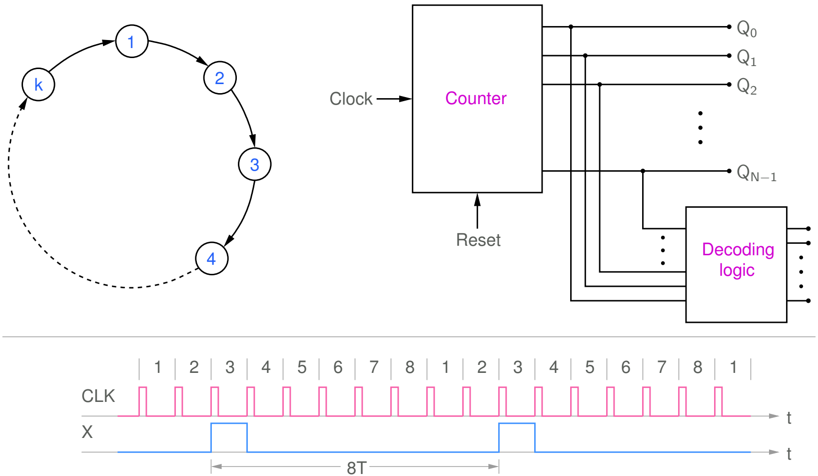


State transition diagram

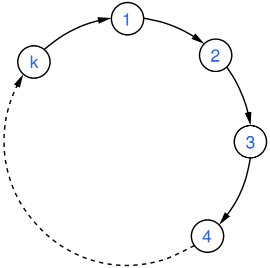


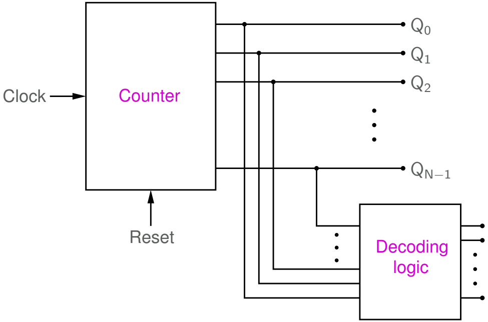


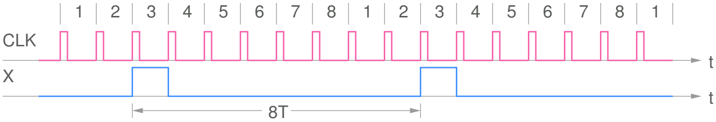
state	Q_0	Q_1	Q_2
1	0	0	0
2	1	1	0
3	1	1	1
4	0	1	1
5	1	0	1
1	0	0	0



X is 1 for state 3; else, it is 0.



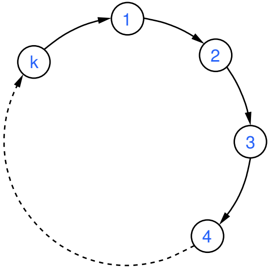


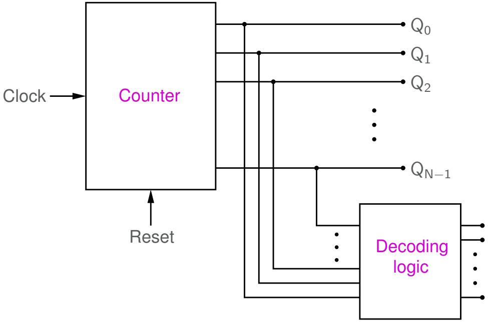


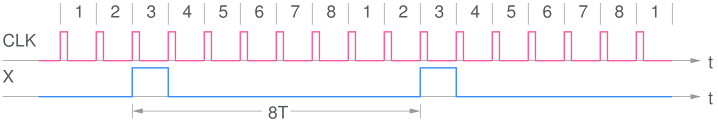
X is 1 for state 3; else, it is 0.

* The counter outputs (i.e., the flip-flop outputs, Q_0, Q_1, $\cdots$ Q_{N-1}) can be decoded using appropriate logic.

Counters



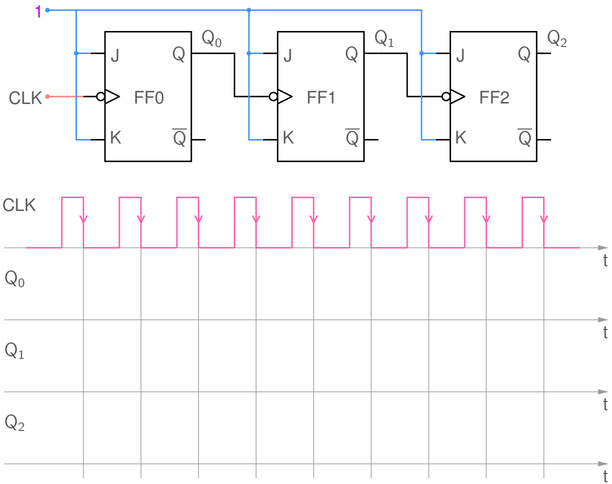




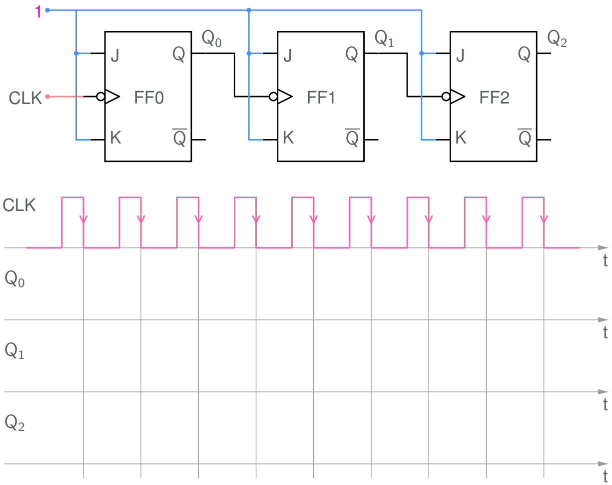
X is 1 for state 3; else, it is 0.

* The counter outputs (i.e., the flip-flop outputs, Q_0, Q_1, $\cdots$ Q_{N-1}) can be decoded using appropriate logic.
* In particular, it is possible to have a decoder output (say, X) which is 1 only for state i, and 0 otherwise.
  $\rightarrow$ For k clock pulses, we get a single pulse at X, i.e., the clock frequency has been divided by k. For this reason, a mod-k counter is also called a divide-by-k counter.

A binary ripple counter

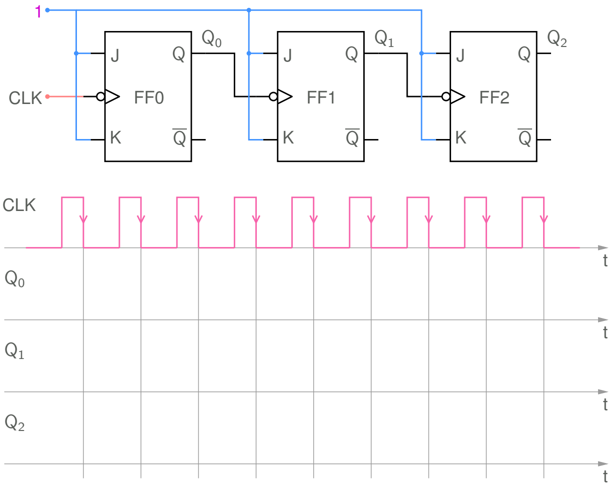


A binary ripple counter



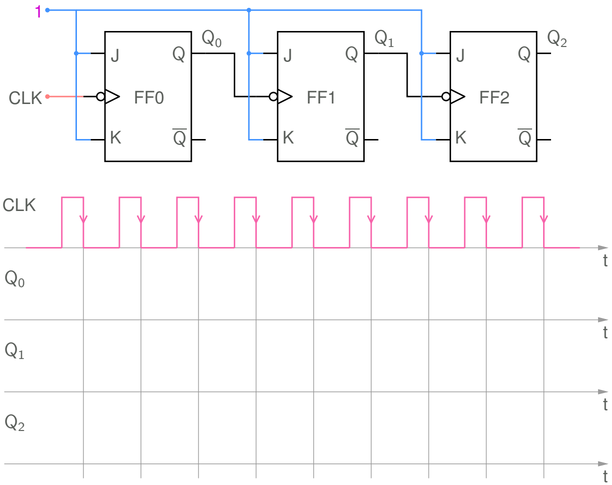
* $J = K = 1$ for all flip-flops. Let $Q_0 = Q_1 = Q_2 = 0$ initially.

A binary ripple counter



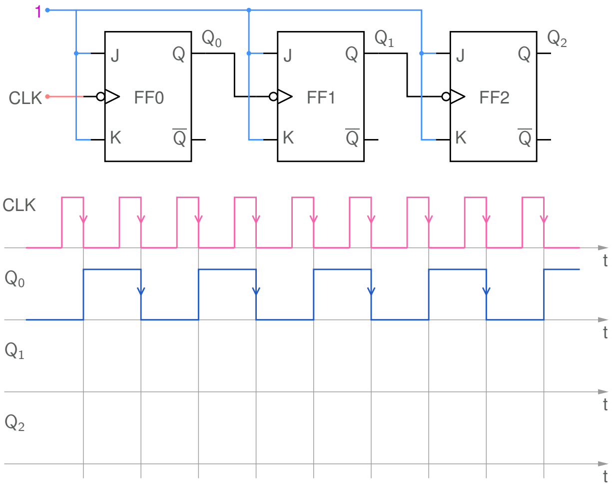
* $J = K = 1$ for all flip-flops. Let $Q_0 = Q_1 = Q_2 = 0$ initially.
* Since $J = K = 1$, each flip-flop will toggle when an active (in this case, negative) clock edge arrives.

A binary ripple counter



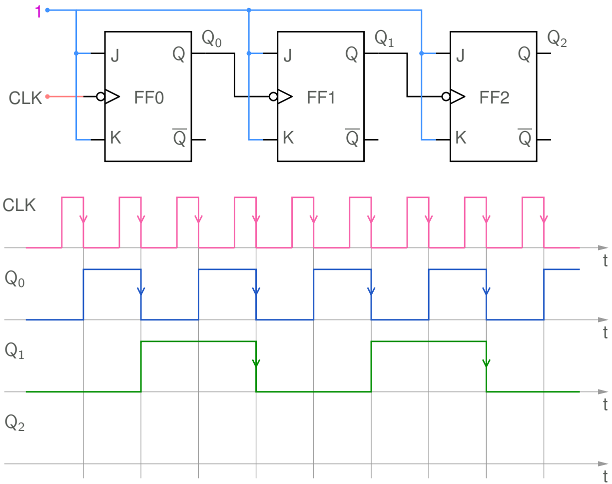
* $J = K = 1$ for all flip-flops. Let $Q_0 = Q_1 = Q_2 = 0$ initially.
* Since $J = K = 1$, each flip-flop will toggle when an active (in this case, negative) clock edge arrives.
* For FF1 and FF2, Q_0 and Q_1, respectively, provide the clock.

A binary ripple counter



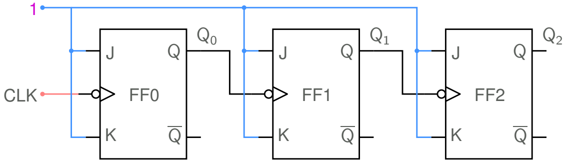
* $J = K = 1$ for all flip-flops. Let $Q_0 = Q_1 = Q_2 = 0$ initially.
* Since $J = K = 1$, each flip-flop will toggle when an active (in this case, negative) clock edge arrives.
* For FF1 and FF2, Q_0 and Q_1, respectively, provide the clock.

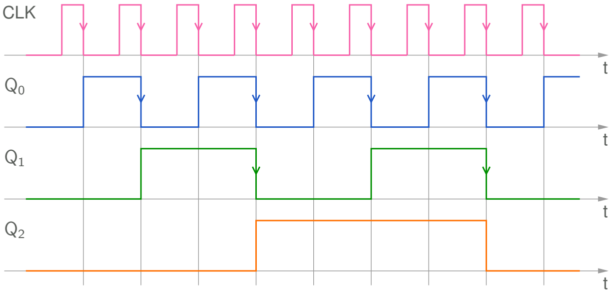
A binary ripple counter



* $J = K = 1$ for all flip-flops. Let $Q_0 = Q_1 = Q_2 = 0$ initially.
* Since $J = K = 1$, each flip-flop will toggle when an active (in this case, negative) clock edge arrives.
* For FF1 and FF2, Q_0 and Q_1, respectively, provide the clock.

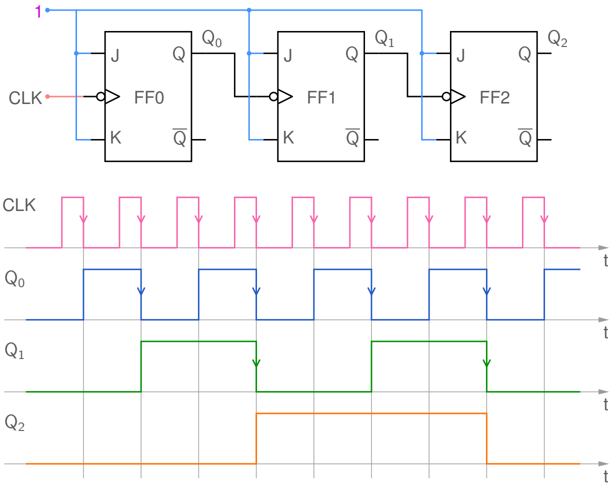
A binary ripple counter





* $J = K = 1$ for all flip-flops. Let $Q_0 = Q_1 = Q_2 = 0$ initially.
* Since $J = K = 1$, each flip-flop will toggle when an active (in this case, negative) clock edge arrives.
* For FF1 and FF2, Q_0 and Q_1, respectively, provide the clock.

A binary ripple counter

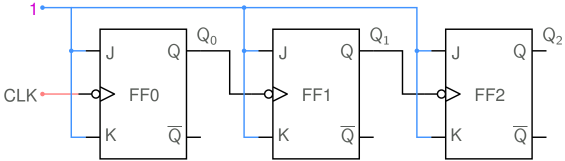


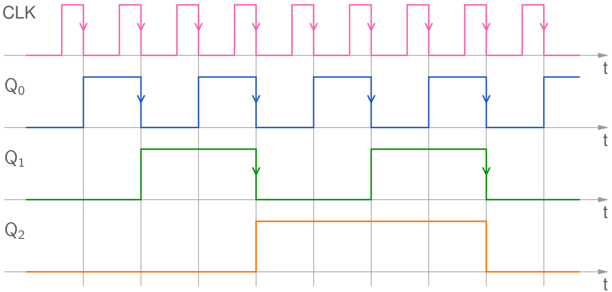
Q_2	Q_1	Q_0
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1
0	0	0

repeats

* $J = K = 1$ for all flip-flops. Let $Q_0 = Q_1 = Q_2 = 0$ initially.
* Since $J = K = 1$, each flip-flop will toggle when an active (in this case, negative) clock edge arrives.
* For FF1 and FF2, Q_0 and Q_1, respectively, provide the clock.

A binary ripple counter

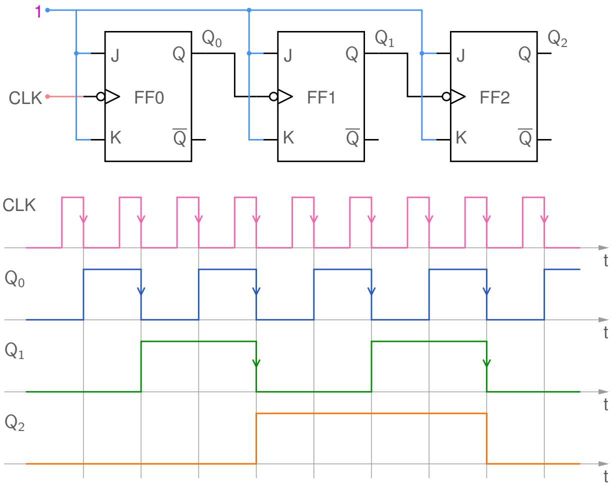




Q_2	Q_1	Q_0	
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	
0	0	0	repeats

* $J = K = 1$ for all flip-flops. Let $Q_0 = Q_1 = Q_2 = 0$ initially.
* Since $J = K = 1$, each flip-flop will toggle when an active (in this case, negative) clock edge arrives.
* For FF1 and FF2, Q_0 and Q_1, respectively, provide the clock.
* Note that the direct inputs S_d and R_d (not shown) are assumed to be $S_d = R_d = 0$ for all flip-flops, allowing normal flip-flip operation.

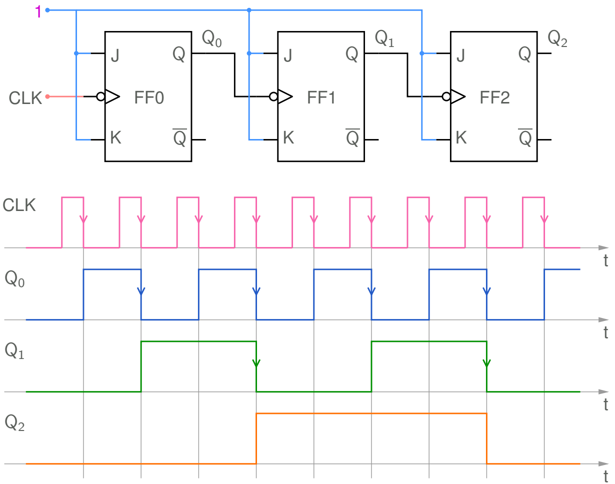
A binary ripple counter



Q_2	Q_1	Q_0
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1
0	0	0

repeats

A binary ripple counter

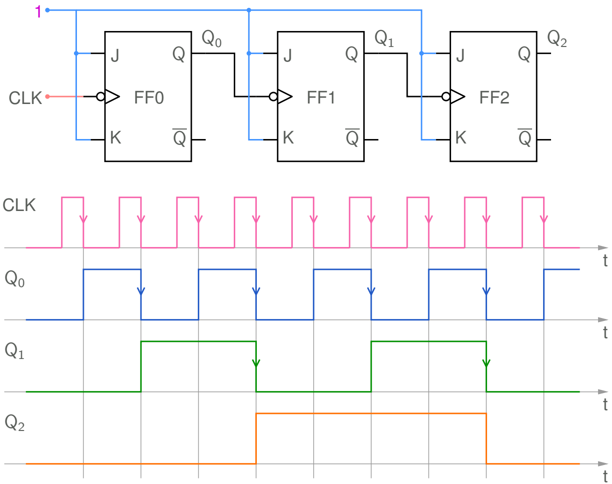


Q_2	Q_1	Q_0
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1
0	0	0

repeats

* The counter has 8 states, $Q_2\,Q_1\,Q_0 = 000, 001, 010, 011, 100, 101, 110, 111$.
  $\rightarrow$ it is a mod-8 counter. In particular, it is a *binary, mod-8, up* counter (since it counts *up* from 000 to 111).

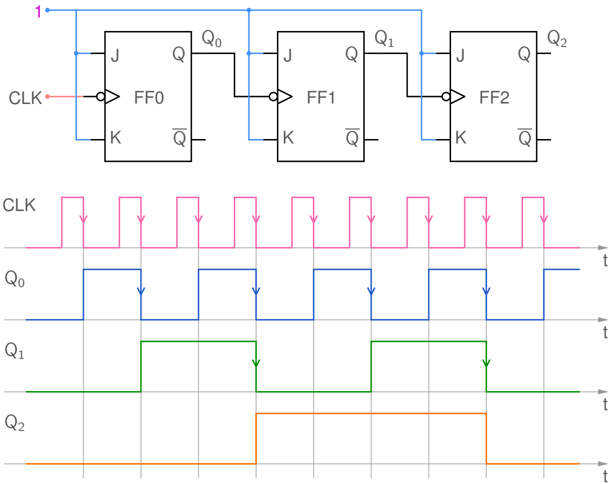
A binary ripple counter



Q_2	Q_1	Q_0	
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	
0	0	0	repeats

* The counter has 8 states, $Q_2\,Q_1\,Q_0 = 000$, 001, 010, 011, 100, 101, 110, 111.
  $\rightarrow$ it is a mod-8 counter. In particular, it is a *binary, mod-8, up* counter (since it counts *up* from 000 to 111).
* If the clock frequency is f_c, the frequency at the Q_0, Q_1, Q_2 outputs is $f_c/2$, $f_c/4$, $f_c/8$, respectively. For this counter, therefore, div-by-2, div-by-4, div-by-8 outputs are already available, without requiring decoding logic.

A binary ripple counter

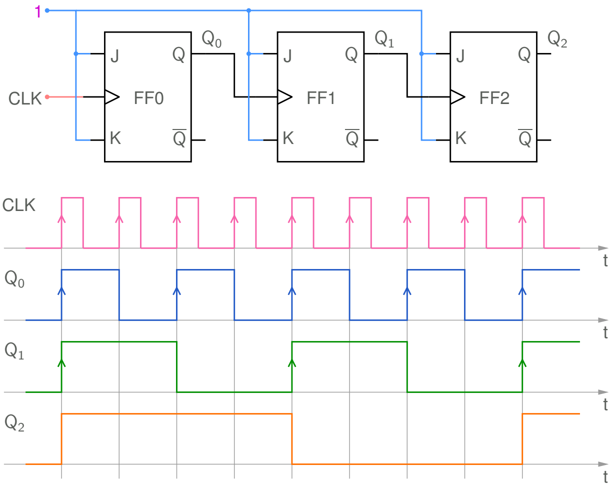


Q_2	Q_1	Q_0
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1
0	0	0

repeats

* The counter has 8 states, $Q_2\,Q_1\,Q_0 = 000,\ 001,\ 010,\ 011,\ 100,\ 101,\ 110,\ 111$.
  $\rightarrow$ it is a mod-8 counter. In particular, it is a *binary, mod-8, up* counter (since it counts *up* from 000 to 111).
* If the clock frequency is f_c, the frequency at the Q_0, Q_1, Q_2 outputs is $f_c/2$, $f_c/4$, $f_c/8$, respectively. For this counter, therefore, div-by-2, div-by-4, div-by-8 outputs are already available, without requiring decoding logic.
* This type of counter is called a "ripple" counter since the clock transitions *ripple* through the flip-flops.

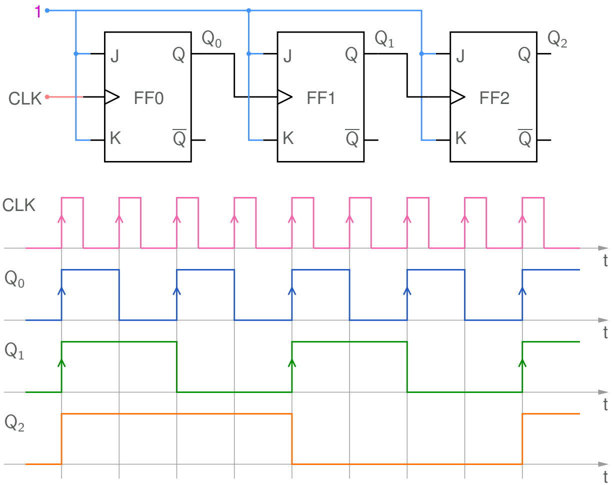
A binary ripple counter



Q_2	Q_1	Q_0
0	0	0
1	1	1
1	1	0
1	0	1
1	0	0
0	1	1
0	1	0
0	0	1
0	0	0

repeats

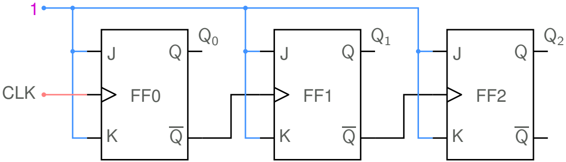
A binary ripple counter

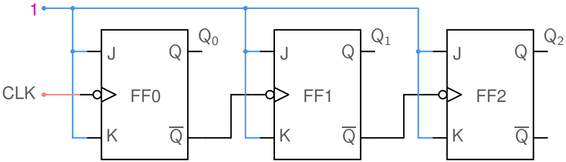


Q_2	Q_1	Q_0
0	0	0
1	1	1
1	1	0
1	0	1
1	0	0
0	1	1
0	1	0
0	0	1
0	0	0

repeats

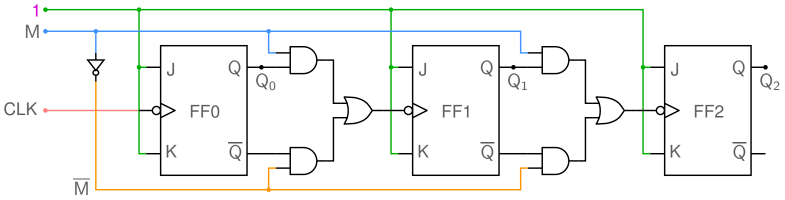
* If positive edge-triggered flip-flops are used, we get a binary *down* counter (counting down from 111 to 000).

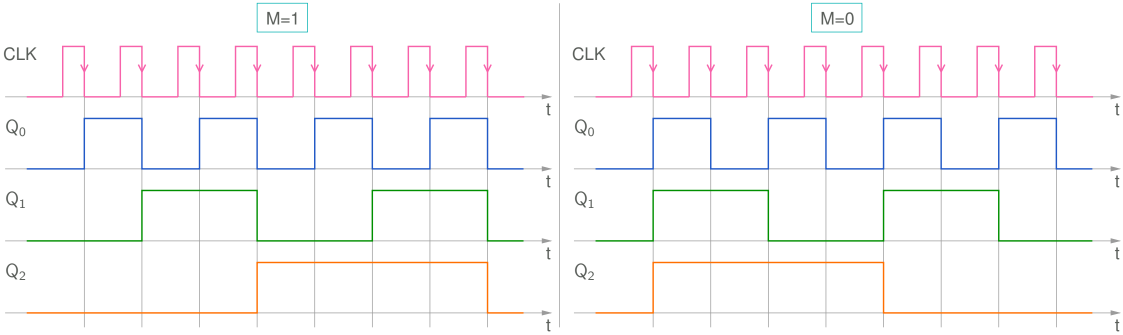




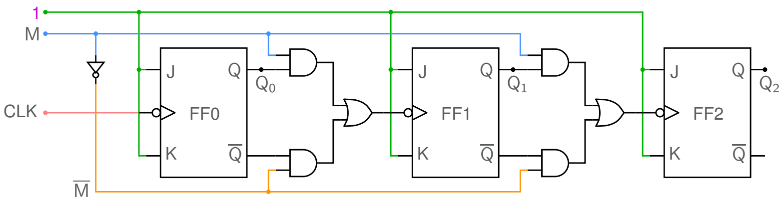
* Home work: Sketch the waveforms (CLK, Q_0, Q_1, Q_2), and tabulate the counter states in each case.

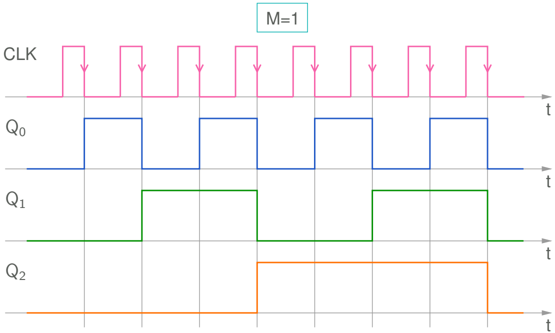
Up-down binary ripple counters

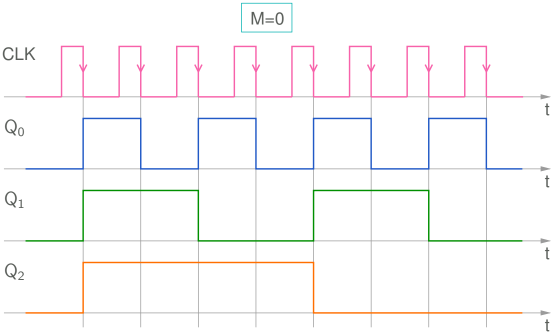




Up-down binary ripple counters

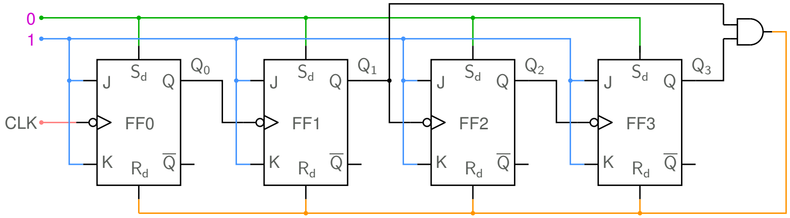


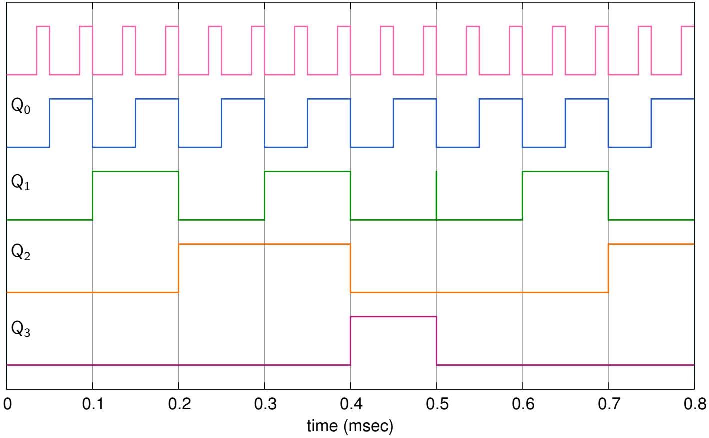




* When Mode (M) = 1, the counter counts up; else, it counts down. (SEQUEL file: `ee101_counter_3.sqproj`)

Decade counter using direct inputs

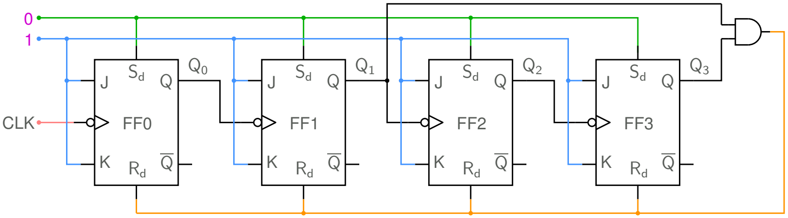




Q_3	Q_2	Q_1	Q_0
0	0	0	0
0	0	0	1
0	0	1	0
0	0	1	1
0	1	0	0
0	1	0	1
0	1	1	0
0	1	1	1
1	0	0	0
1	0	0	1
0	0	0	0

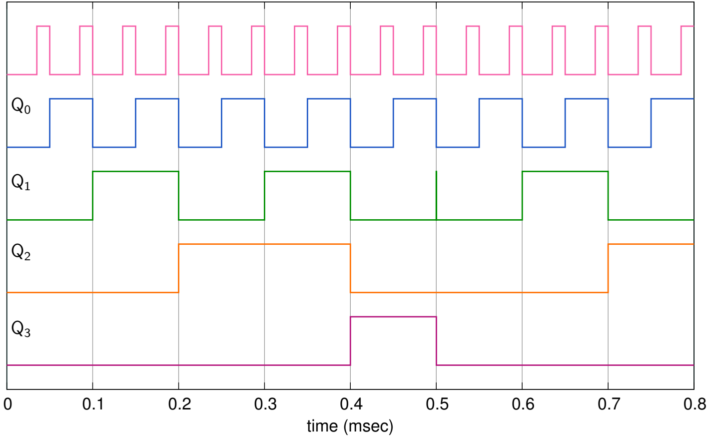
repeats

SEQUEL file: `ee101_counter_5.sqproj`



Decade counter using direct inputs

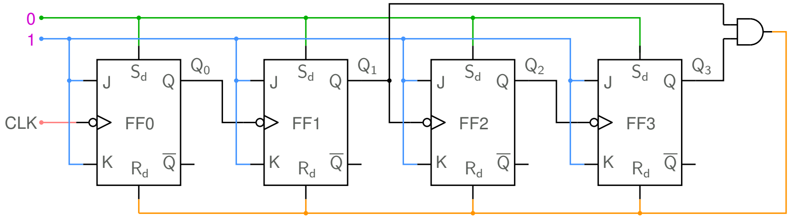
* When the counter reaches $Q_3\,Q_2\,Q_1\,Q_0 = 1010$ (i.e., decimal 10), $Q_3\,Q_1 = 1$, and the flip-flops are cleared to $Q_3\,Q_2\,Q_1\,Q_0 = 0000$.



Q_3	Q_2	Q_1	Q_0
0	0	0	0
0	0	0	1
0	0	1	0
0	0	1	1
0	1	0	0
0	1	0	1
0	1	1	0
0	1	1	1
1	0	0	0
1	0	0	1
0	0	0	0

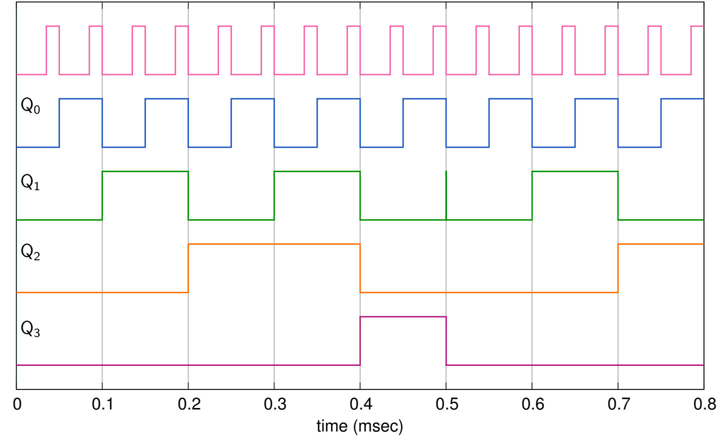
repeats

SEQUEL file: `ee101_counter_5.sqproj`



Decade counter using direct inputs

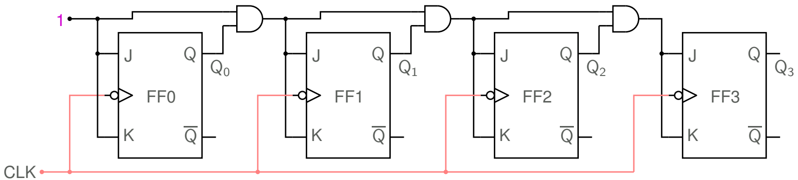
* When the counter reaches $Q_3 Q_2 Q_1 Q_0 = 1010$ (i.e., decimal 10), $Q_3 Q_1 = 1$, and the flip-flops are cleared to $Q_3 Q_2 Q_1 Q_0 = 0000$.
* The counter counts from 0000 (decimal 0) to 1001 (decimal 9) $\rightarrow$ "decade counter."

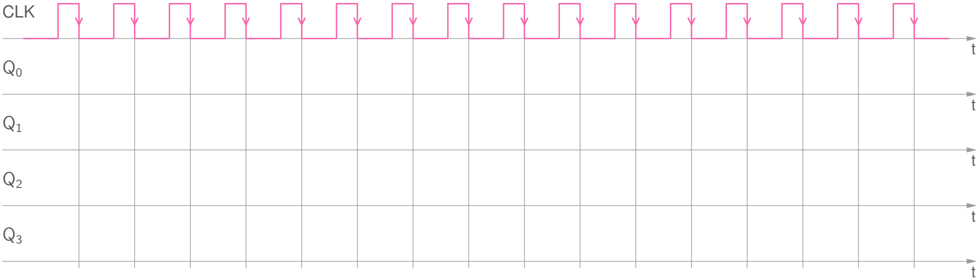


Q_3	Q_2	Q_1	Q_0	
0	0	0	0	
0	0	0	1	
0	0	1	0	
0	0	1	1	
0	1	0	0	
0	1	0	1	
0	1	1	0	
0	1	1	1	
1	0	0	0	
1	0	0	1	
0	0	0	0	repeats

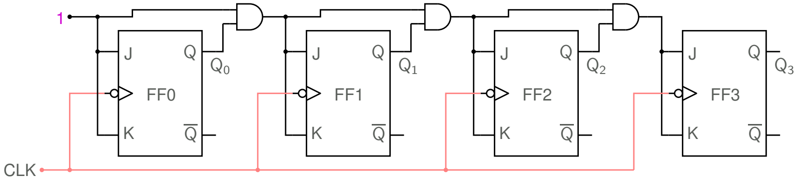
SEQUEL file: `ee101_counter_5.sqproj`

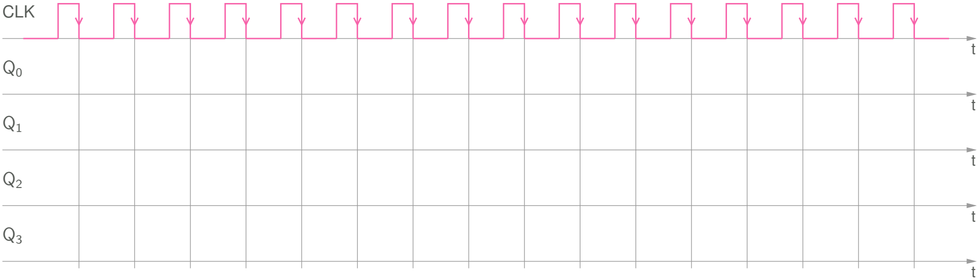
A synchronous counter





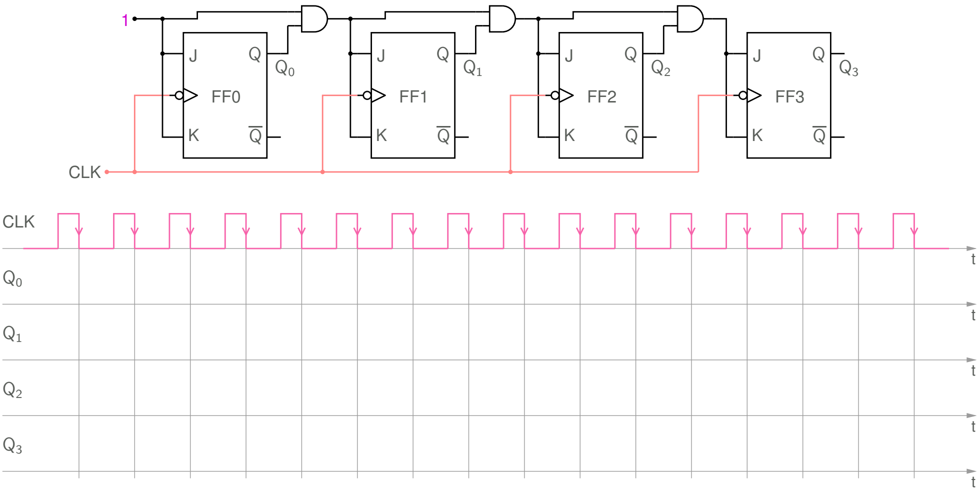
A synchronous counter





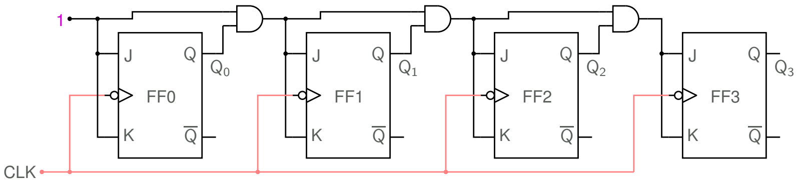
* Since all flip-flops are driven by the same clock, the counter is called a “synchronous” counter.

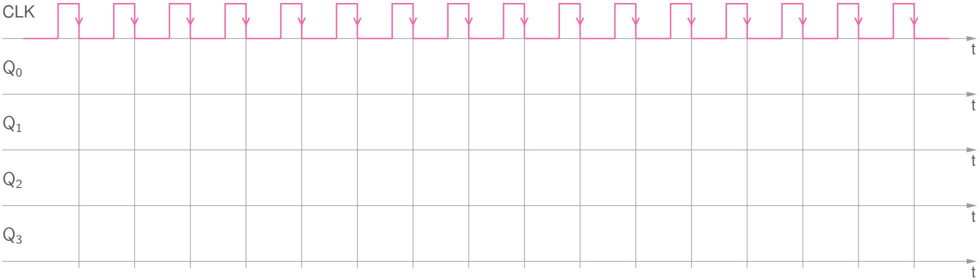
A synchronous counter



* Since all flip-flops are driven by the same clock, the counter is called a "synchronous" counter.
* $J_0 = K_0 = 1,\ J_1 = K_1 = Q_0,\ J_2 = K_2 = Q_1 Q_0,\ J_3 = K_3 = Q_2 Q_1 Q_0.$

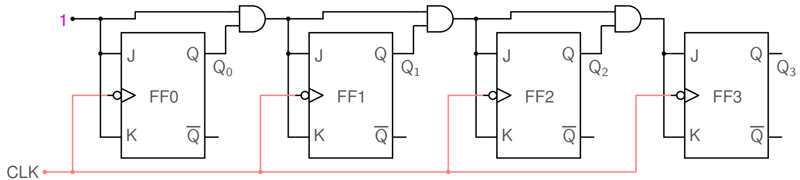
A synchronous counter

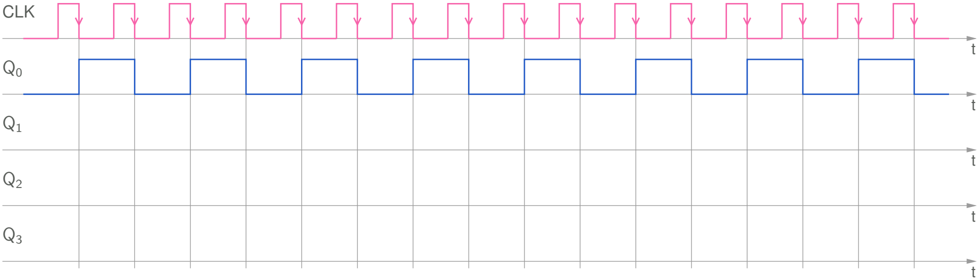




* Since all flip-flops are driven by the same clock, the counter is called a "synchronous" counter.
* $J_0 = K_0 = 1$, $J_1 = K_1 = Q_0$, $J_2 = K_2 = Q_1 Q_0$, $J_3 = K_3 = Q_2 Q_1 Q_0$.
* FF0 toggles after every active edge.
  FF1 toggles if $Q_0 = 1$ (just before the active clock edge); else, it retains its previous state. (Similarly, for FF2 and FF3)

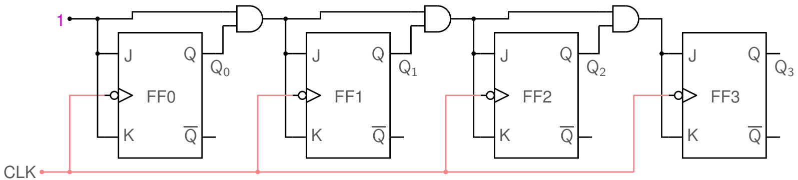
A synchronous counter

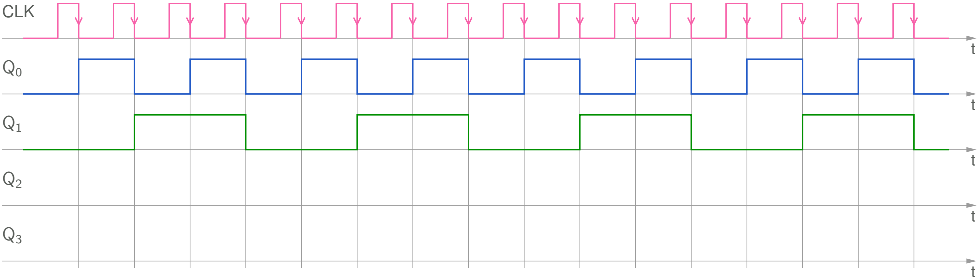




* Since all flip-flops are driven by the same clock, the counter is called a "synchronous" counter.
* $J_0 = K_0 = 1,\ J_1 = K_1 = Q_0,\ J_2 = K_2 = Q_1 Q_0,\ J_3 = K_3 = Q_2 Q_1 Q_0$.
* FF0 toggles after every active edge.
  FF1 toggles if $Q_0 = 1$ (just before the active clock edge); else, it retains its previous state. (Similarly, for FF2 and FF3)

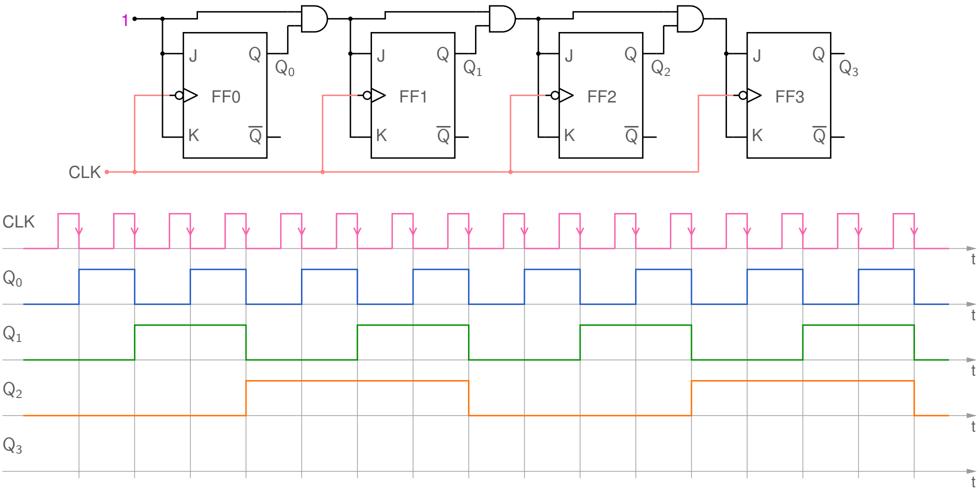
A synchronous counter





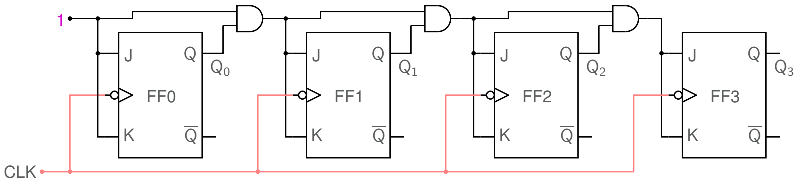
* Since all flip-flops are driven by the same clock, the counter is called a "synchronous" counter.
* $J_0 = K_0 = 1,\; J_1 = K_1 = Q_0,\; J_2 = K_2 = Q_1\,Q_0,\; J_3 = K_3 = Q_2\,Q_1\,Q_0.$
* FF0 toggles after every active edge.
  FF1 toggles if $Q_0 = 1$ (just before the active clock edge); else, it retains its previous state. (Similarly, for FF2 and FF3)

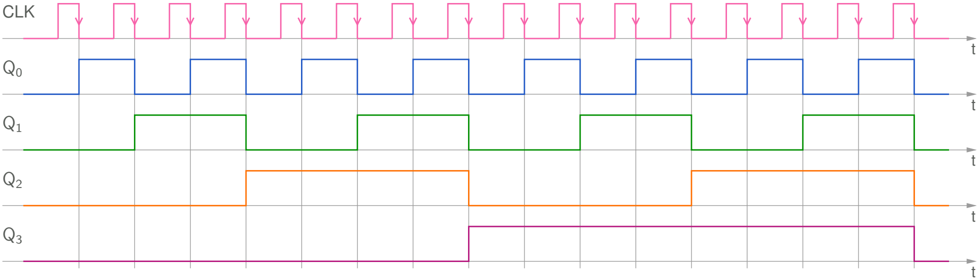
A synchronous counter



* Since all flip-flops are driven by the same clock, the counter is called a "synchronous" counter.
* $J_0 = K_0 = 1,\ J_1 = K_1 = Q_0,\ J_2 = K_2 = Q_1\,Q_0,\ J_3 = K_3 = Q_2\,Q_1\,Q_0.$
* FF0 toggles after every active edge.
  FF1 toggles if $Q_0 = 1$ (just before the active clock edge); else, it retains its previous state. (Similarly, for FF2 and FF3)

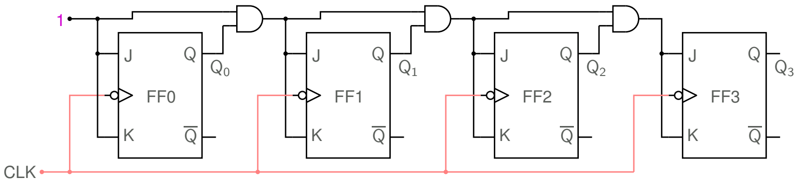
A synchronous counter

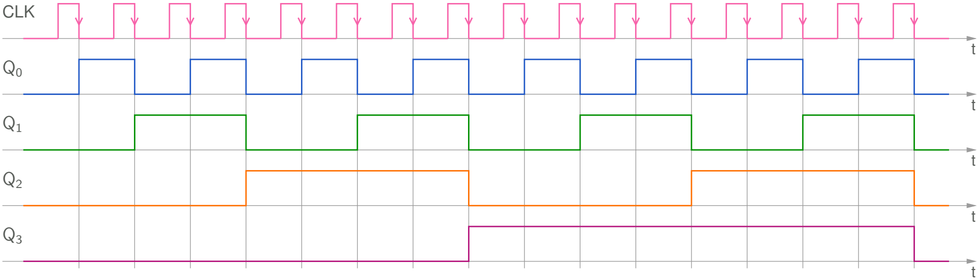




* Since all flip-flops are driven by the same clock, the counter is called a "synchronous" counter.
* $J_0 = K_0 = 1$, $J_1 = K_1 = Q_0$, $J_2 = K_2 = Q_1 Q_0$, $J_3 = K_3 = Q_2 Q_1 Q_0$.
* FF0 toggles after every active edge.
  FF1 toggles if $Q_0 = 1$ (just before the active clock edge); else, it retains its previous state. (Similarly, for FF2 and FF3)

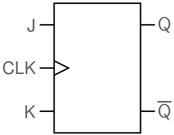
A synchronous counter



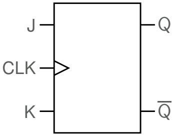


* Since all flip-flops are driven by the same clock, the counter is called a "synchronous" counter.
* $J_0 = K_0 = 1, \; J_1 = K_1 = Q_0, \; J_2 = K_2 = Q_1 Q_0, \; J_3 = K_3 = Q_2 Q_1 Q_0$.
* FF0 toggles after every active edge.
  FF1 toggles if $Q_0 = 1$ (just before the active clock edge); else, it retains its previous state. (Similarly, for FF2 and FF3)
* From the waveforms, we see that it is a binary up counter.

Design of synchronous counters



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

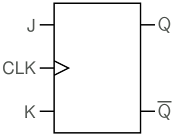


CLK	J	K	Q_{n+1}
$\uparrow$	0	0	Q_n
$\uparrow$	0	1	0
$\uparrow$	1	0	1
$\uparrow$	1	1	$\overline{Q_n}$

CLK	Q_n	Q_{n+1}	J	K

* Consider the *reverse* problem: We are given Q_n and the next desired state (Q_{n+1}). What should J and K be in order to make that happen?

Design of synchronous counters

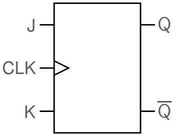


CLK	J	K	Q_{n+1}
$\uparrow$	0	0	Q_n
$\uparrow$	0	1	0
$\uparrow$	1	0	1
$\uparrow$	1	1	$\overline{Q_n}$

CLK	Q_n	Q_{n+1}	J	K

* Consider the *reverse* problem: We are given Q_n and the next desired state (Q_{n+1}). What should J and K be in order to make that happen?
* $Q_n = 0$, $Q_{n+1} = 0$: We can either force $Q_{n+1} = 0$ with $J = 0$, $K = 1$, or let $Q_{n+1} = Q_n$ by making $J = 0$, $K = 0$.

Design of synchronous counters

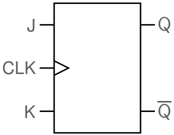


CLK	J	K	Q_{n+1}
$\uparrow$	0	0	Q_n
$\uparrow$	0	1	0
$\uparrow$	1	0	1
$\uparrow$	1	1	$\overline{Q_n}$

CLK	Q_n	Q_{n+1}	J	K

* Consider the *reverse* problem: We are given Q_n and the next desired state (Q_{n+1}). What should J and K be in order to make that happen?
* $Q_n = 0$, $Q_{n+1} = 0$: We can either force $Q_{n+1} = 0$ with $J = 0$, $K = 1$, or let $Q_{n+1} = Q_n$ by making $J = 0$, $K = 0$.
  $\rightarrow$ $J = 0$, $K = X$ (i.e., K can be 0 or 1).

Design of synchronous counters

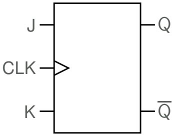


CLK	J	K	Q_{n+1}
↑	0	0	Q_n
↑	0	1	0
↑	1	0	1
↑	1	1	$\overline{Q_n}$

CLK	Q_n	Q_{n+1}	J	K
↑	0	0	0	X

* Consider the *reverse* problem: We are given Q_n and the next desired state (Q_{n+1}). What should J and K be in order to make that happen?
* $Q_n = 0$, $Q_{n+1} = 0$: We can either force $Q_{n+1} = 0$ with $J = 0$, $K = 1$, or let $Q_{n+1} = Q_n$ by making $J = 0$, $K = 0$.
  $\rightarrow$ $J = 0$, $K = X$ (i.e., K can be 0 or 1).

Design of synchronous counters

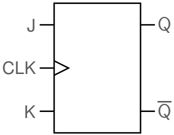


CLK	J	K	Q_{n+1}
$\uparrow$	0	0	Q_n
$\uparrow$	0	1	0
$\uparrow$	1	0	1
$\uparrow$	1	1	$\overline{Q_n}$

CLK	Q_n	Q_{n+1}	J	K
$\uparrow$	0	0	0	X

* Consider the *reverse* problem: We are given Q_n and the next desired state (Q_{n+1}). What should J and K be in order to make that happen?
* $Q_n = 0$, $Q_{n+1} = 0$: We can either force $Q_{n+1} = 0$ with $J = 0$, $K = 1$, or let $Q_{n+1} = Q_n$ by making $J = 0$, $K = 0$.
  $\rightarrow J = 0$, $K = X$ (i.e., K can be 0 or 1).
* Similarly, work out the other entries in the table.

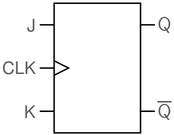
Design of synchronous counters



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

CLK	Q_n	Q_{n+1}	J	K
↑	0	0	0	X
↑	0	1		

* Consider the *reverse* problem: We are given Q_n and the next desired state (Q_{n+1}). What should J and K be in order to make that happen?
* $Q_n = 0$, $Q_{n+1} = 0$: We can either force $Q_{n+1} = 0$ with $J = 0$, $K = 1$, or let $Q_{n+1} = Q_n$ by making $J = 0$, $K = 0$.
  $\rightarrow J = 0$, $K = X$ (i.e., K can be 0 or 1).
* Similarly, work out the other entries in the table.

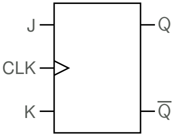


CLK	J	K	Q_{n+1}
↑	0	0	Q_n
↑	0	1	0
↑	1	0	1
↑	1	1	$\overline{Q_n}$

CLK	Q_n	Q_{n+1}	J	K
↑	0	0	0	X
↑	0	1	1	X

* Consider the *reverse* problem: We are given Q_n and the next desired state (Q_{n+1}). What should J and K be in order to make that happen?
* $Q_n = 0$, $Q_{n+1} = 0$: We can either force $Q_{n+1} = 0$ with $J = 0$, $K = 1$, or let $Q_{n+1} = Q_n$ by making $J = 0$, $K = 0$.
  $\rightarrow$ $J = 0$, $K = X$ (i.e., K can be 0 or 1).
* Similarly, work out the other entries in the table.

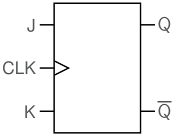
Design of synchronous counters



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

CLK	Q_n	Q_{n+1}	J	K
↑	0	0	0	X
↑	0	1	1	X
↑	1	0		

* Consider the *reverse* problem: We are given Q_n and the next desired state (Q_{n+1}). What should J and K be in order to make that happen?
* $Q_n = 0$, $Q_{n+1} = 0$: We can either force $Q_{n+1} = 0$ with $J = 0$, $K = 1$, or let $Q_{n+1} = Q_n$ by making $J = 0$, $K = 0$.
  $\rightarrow J = 0$, $K = X$ (i.e., K can be 0 or 1).
* Similarly, work out the other entries in the table.

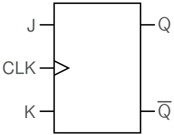


CLK	J	K	Q_{n+1}
↑	0	0	Q_n
↑	0	1	0
↑	1	0	1
↑	1	1	$\overline{Q_n}$

CLK	Q_n	Q_{n+1}	J	K
↑	0	0	0	X
↑	0	1	1	X
↑	1	0	X	1

* Consider the *reverse* problem: We are given Q_n and the next desired state (Q_{n+1}). What should J and K be in order to make that happen?
* $Q_n = 0$, $Q_{n+1} = 0$: We can either force $Q_{n+1} = 0$ with $J = 0$, $K = 1$, or let $Q_{n+1} = Q_n$ by making $J = 0$, $K = 0$.
  $\rightarrow J = 0$, $K = X$ (i.e., K can be 0 or 1).
* Similarly, work out the other entries in the table.

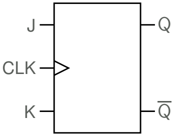
Design of synchronous counters



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

CLK	Q_n	Q_{n+1}	J	K
↑	0	0	0	X
↑	0	1	1	X
↑	1	0	X	1
↑	1	1		

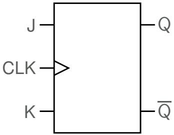
* Consider the *reverse* problem: We are given Q_n and the next desired state (Q_{n+1}). What should J and K be in order to make that happen?
* $Q_n = 0$, $Q_{n+1} = 0$: We can either force $Q_{n+1} = 0$ with $J = 0$, $K = 1$, or let $Q_{n+1} = Q_n$ by making $J = 0$, $K = 0$.
  $\rightarrow$ $J = 0$, $K = X$ (i.e., K can be 0 or 1).
* Similarly, work out the other entries in the table.



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

CLK	Q_n	Q_{n+1}	J	K
↑	0	0	0	X
↑	0	1	1	X
↑	1	0	X	1
↑	1	1	X	0

* Consider the *reverse* problem: We are given Q_n and the next desired state (Q_{n+1}). What should J and K be in order to make that happen?
* $Q_n = 0$, $Q_{n+1} = 0$: We can either force $Q_{n+1} = 0$ with $J = 0$, $K = 1$, or let $Q_{n+1} = Q_n$ by making $J = 0$, $K = 0$.
$\rightarrow$ $J = 0$, $K = X$ (i.e., K can be 0 or 1).
* Similarly, work out the other entries in the table.



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

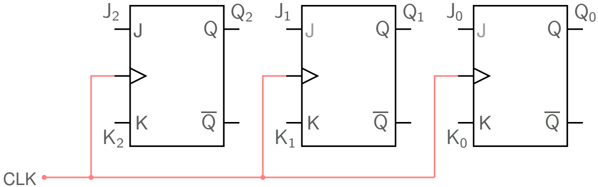
CLK	Q_n	Q_{n+1}	J	K
↑	0	0	0	X
↑	0	1	1	X
↑	1	0	X	1
↑	1	1	X	0

* Consider the *reverse* problem: We are given Q_n and the next desired state (Q_{n+1}). What should J and K be in order to make that happen?
* $Q_n = 0$, $Q_{n+1} = 0$: We can either force $Q_{n+1} = 0$ with $J = 0$, $K = 1$, or let $Q_{n+1} = Q_n$ by making $J = 0$, $K = 0$.
  $\rightarrow J = 0$, $K = X$ (i.e., K can be 0 or 1).
* Similarly, work out the other entries in the table.
* The table for a negative edge-triggered flip-flop would be identical except for the active edge.

Design of synchronous counters

state	Q_2	Q_1	Q_0
1	0	0	0
2	0	0	1
3	0	1	0
4	0	1	1
5	1	0	0
1	0	0	0

repeats



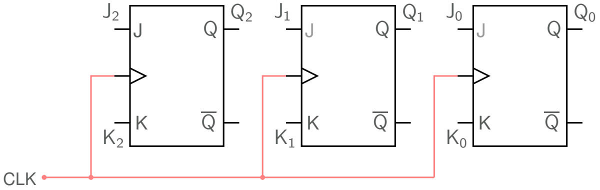
CLK	Q_n	Q_{n+1}	J	K
↑	0	0	0	X
↑	0	1	1	X
↑	1	0	X	1
↑	1	1	X	0

Design a synchronous mod-5 counter with the given state transition table.

Design of synchronous counters

state	Q_2	Q_1	Q_0
1	0	0	0
2	0	0	1
3	0	1	0
4	0	1	1
5	1	0	0
1	0	0	0

repeats



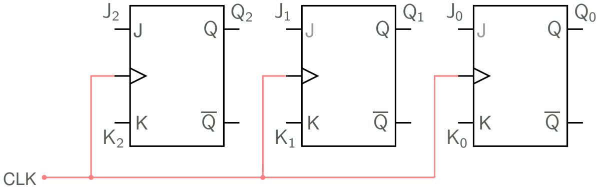
CLK	Q_n	Q_{n+1}	J	K
↑	0	0	0	X
↑	0	1	1	X
↑	1	0	X	1
↑	1	1	X	0

Design a synchronous mod-5 counter with the given state transition table.

Outline of method:

state	Q_2	Q_1	Q_0
1	0	0	0
2	0	0	1
3	0	1	0
4	0	1	1
5	1	0	0
1	0	0	0

repeats



CLK	Q_n	Q_{n+1}	J	K
↑	0	0	0	X
↑	0	1	1	X
↑	1	0	X	1
↑	1	1	X	0

Design a synchronous mod-5 counter with the given state transition table.

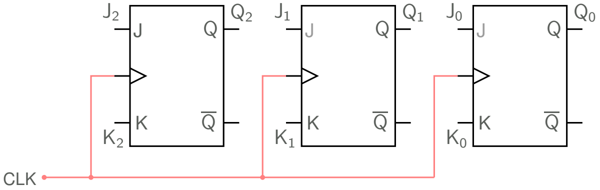
Outline of method:

* State 1 $\rightarrow$ State 2 means
  Q_2: $0 \rightarrow 0$,
  Q_1: $0 \rightarrow 0$,
  Q_0: $0 \rightarrow 1$.

M. B. Patil, IIT Bombay

state	Q_2	Q_1	Q_0
1	0	0	0
2	0	0	1
3	0	1	0
4	0	1	1
5	1	0	0
1	0	0	0

repeats



CLK	Q_n	Q_{n+1}	J	K
↑	0	0	0	X
↑	0	1	1	X
↑	1	0	X	1
↑	1	1	X	0

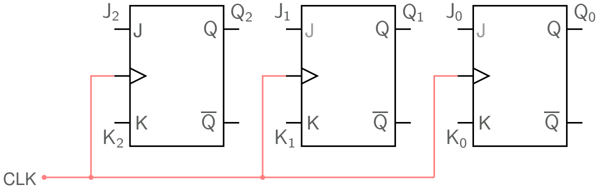
Design a synchronous mod-5 counter with the given state transition table.

Outline of method:

* State 1 $\rightarrow$ State 2 means
  Q_2: $0 \rightarrow 0$,
  Q_1: $0 \rightarrow 0$,
  Q_0: $0 \rightarrow 1$.
* Refer to the right table. For Q_2: $0 \rightarrow 0$, we must have $J_2 = 0$, $K_2 = X$, and so on.

state	Q_2	Q_1	Q_0
1	0	0	0
2	0	0	1
3	0	1	0
4	0	1	1
5	1	0	0
1	0	0	0

repeats



CLK	Q_n	Q_{n+1}	J	K
↑	0	0	0	X
↑	0	1	1	X
↑	1	0	X	1
↑	1	1	X	0

Design a synchronous mod-5 counter with the given state transition table.

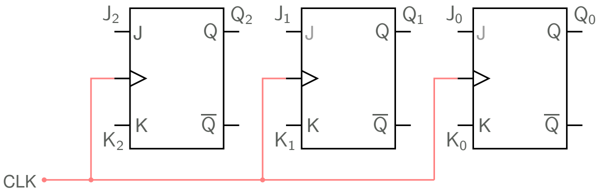
Outline of method:

* State 1 $\longrightarrow$ State 2 means
  Q_2: $0 \rightarrow 0$,
  Q_1: $0 \rightarrow 0$,
  Q_0: $0 \rightarrow 1$.
* Refer to the right table. For Q_2: $0 \rightarrow 0$, we must have $J_2 = 0$, $K_2 = X$, and so on.
* When we cover all transitions in the left table, we have the truth tables for J_0, K_0, J_1, K_1, J_2, K_2 in terms of Q_0, Q_1, Q_2.

Design of synchronous counters

state	Q_2	Q_1	Q_0
1	0	0	0
2	0	0	1
3	0	1	0
4	0	1	1
5	1	0	0
1	0	0	0

repeats



CLK	Q_n	Q_{n+1}	J	K
↑	0	0	0	X
↑	0	1	1	X
↑	1	0	X	1
↑	1	1	X	0

Design a synchronous mod-5 counter with the given state transition table.

Outline of method:

* State 1 $\longrightarrow$ State 2 means
  Q_2: $0 \rightarrow 0$,
  Q_1: $0 \rightarrow 0$,
  Q_0: $0 \rightarrow 1$.
* Refer to the right table. For Q_2: $0 \rightarrow 0$, we must have $J_2 = 0$, $K_2 = X$, and so on.
* When we cover all transitions in the left table, we have the truth tables for J_0, K_0, J_1, K_1, J_2, K_2 in terms of Q_0, Q_1, Q_2.
* The last step is to come up with suitable functions for J_0, K_0, J_1, K_1, J_2, K_2 in terms of Q_0, Q_1, Q_2. This can be done with K-maps. (If the number of flip-flops is more than 4, other techniques can be employed.)

state	Q_2	Q_1	Q_0	J_2	K_2	J_1	K_1	J_0	K_0
1	0	0	0						
2	0	0	1						
3	0	1	0						
4	0	1	1						
5	1	0	0						
1	0	0	0						

CLK	Q_n	Q_{n+1}	J	K
$\uparrow$	0	0	0	X
$\uparrow$	0	1	1	X
$\uparrow$	1	0	X	1
$\uparrow$	1	1	X	0

state	Q_2	Q_1	Q_0	J_2	K_2	J_1	K_1	J_0	K_0
1	0	0	0						
2	0	0	1						
3	0	1	0						
4	0	1	1						
5	1	0	0						
1	0	0	0						

CLK	Q_n	Q_{n+1}	J	K
$\uparrow$	0	0	0	X
$\uparrow$	0	1	1	X
$\uparrow$	1	0	X	1
$\uparrow$	1	1	X	0

state	Q_2	Q_1	Q_0	J_2	K_2	J_1	K_1	J_0	K_0
1	0	0	0	0	X				
2	0	0	1						
3	0	1	0						
4	0	1	1						
5	1	0	0						
1	0	0	0						

CLK	Q_n	Q_{n+1}	J	K
$\uparrow$	0	0	0	X
$\uparrow$	0	1	1	X
$\uparrow$	1	0	X	1
$\uparrow$	1	1	X	0

Design of synchronous counters

state	Q_2	Q_1	Q_0	J_2	K_2	J_1	K_1	J_0	K_0
1	0	0	0	0	X	0	X		
2	0	0	1						
3	0	1	0						
4	0	1	1						
5	1	0	0						
1	0	0	0						

CLK	Q_n	Q_{n+1}	J	K
$\uparrow$	0	0	0	X
$\uparrow$	0	1	1	X
$\uparrow$	1	0	X	1
$\uparrow$	1	1	X	0

Design of synchronous counters

state	Q_2	Q_1	Q_0	J_2	K_2	J_1	K_1	J_0	K_0
1	0	0	0	0	X	0	X	1	X
2	0	0	1						
3	0	1	0						
4	0	1	1						
5	1	0	0						
1	0	0	0						

CLK	Q_n	Q_{n+1}	J	K
$\uparrow$	0	0	0	X
$\uparrow$	0	1	1	X
$\uparrow$	1	0	X	1
$\uparrow$	1	1	X	0

state	Q_2	Q_1	Q_0	J_2	K_2	J_1	K_1	J_0	K_0
1	0	0	0	0	X	0	X	1	X
2	0	0	1						
3	0	1	0						
4	0	1	1						
5	1	0	0						
1	0	0	0						

CLK	Q_n	Q_{n+1}	J	K
$\uparrow$	0	0	0	X
$\uparrow$	0	1	1	X
$\uparrow$	1	0	X	1
$\uparrow$	1	1	X	0

Design of synchronous counters

state	Q_2	Q_1	Q_0	J_2	K_2	J_1	K_1	J_0	K_0
1	0	0	0	0	X	0	X	1	X
2	0	0	1	0	X				
3	0	1	0						
4	0	1	1						
5	1	0	0						
1	0	0	0						

CLK	Q_n	Q_{n+1}	J	K
$\uparrow$	0	0	0	X
$\uparrow$	0	1	1	X
$\uparrow$	1	0	X	1
$\uparrow$	1	1	X	0

Design of synchronous counters

state	Q_2	Q_1	Q_0	J_2	K_2	J_1	K_1	J_0	K_0
1	0	0	0	0	X	0	X	1	X
2	0	0	1	0	X	1	X		
3	0	1	0						
4	0	1	1						
5	1	0	0						
1	0	0	0						

CLK	Q_n	Q_{n+1}	J	K
$\uparrow$	0	0	0	X
$\uparrow$	0	1	1	X
$\uparrow$	1	0	X	1
$\uparrow$	1	1	X	0

state	Q_2	Q_1	Q_0	J_2	K_2	J_1	K_1	J_0	K_0
1	0	0	0	0	X	0	X	1	X
2	0	0	1	0	X	1	X	X	1
3	0	1	0						
4	0	1	1						
5	1	0	0						
1	0	0	0						

CLK	Q_n	Q_{n+1}	J	K
$\uparrow$	0	0	0	X
$\uparrow$	0	1	1	X
$\uparrow$	1	0	X	1
$\uparrow$	1	1	X	0

Design of synchronous counters

state	Q_2	Q_1	Q_0	J_2	K_2	J_1	K_1	J_0	K_0
1	0	0	0	0	X	0	X	1	X
2	0	0	1	0	X	1	X	X	1
3	0	1	0						
4	0	1	1						
5	1	0	0						
1	0	0	0						

CLK	Q_n	Q_{n+1}	J	K
$\uparrow$	0	0	0	X
$\uparrow$	0	1	1	X
$\uparrow$	1	0	X	1
$\uparrow$	1	1	X	0

state	Q_2	Q_1	Q_0	J_2	K_2	J_1	K_1	J_0	K_0
1	0	0	0	0	X	0	X	1	X
2	0	0	1	0	X	1	X	X	1
3	0	1	0	0	X				
4	0	1	1						
5	1	0	0						
1	0	0	0						

CLK	Q_n	Q_{n+1}	J	K
$\uparrow$	0	0	0	X
$\uparrow$	0	1	1	X
$\uparrow$	1	0	X	1
$\uparrow$	1	1	X	0

Design of synchronous counters

state	Q_2	Q_1	Q_0	J_2	K_2	J_1	K_1	J_0	K_0
1	0	0	0	0	X	0	X	1	X
2	0	0	1	0	X	1	X	X	1
3	0	1	0	0	X	X	0		
4	0	1	1						
5	1	0	0						
1	0	0	0						

CLK	Q_n	Q_{n+1}	J	K
$\uparrow$	0	0	0	X
$\uparrow$	0	1	1	X
$\uparrow$	1	0	X	1
$\uparrow$	1	1	X	0

Design of synchronous counters

state	Q_2	Q_1	Q_0	J_2	K_2	J_1	K_1	J_0	K_0
1	0	0	0	0	X	0	X	1	X
2	0	0	1	0	X	1	X	X	1
3	0	1	0	0	X	X	0	1	X
4	0	1	1						
5	1	0	0						
1	0	0	0						

CLK	Q_n	Q_{n+1}	J	K
$\uparrow$	0	0	0	X
$\uparrow$	0	1	1	X
$\uparrow$	1	0	X	1
$\uparrow$	1	1	X	0

Design of synchronous counters

state	Q_2	Q_1	Q_0	J_2	K_2	J_1	K_1	J_0	K_0
1	0	0	0	0	X	0	X	1	X
2	0	0	1	0	X	1	X	X	1
3	0	1	0	0	X	X	0	1	X
4	0	1	1						
5	1	0	0						
1	0	0	0						

CLK	Q_n	Q_{n+1}	J	K
$\uparrow$	0	0	0	X
$\uparrow$	0	1	1	X
$\uparrow$	1	0	X	1
$\uparrow$	1	1	X	0

Design of synchronous counters

state	Q_2	Q_1	Q_0	J_2	K_2	J_1	K_1	J_0	K_0
1	0	0	0	0	X	0	X	1	X
2	0	0	1	0	X	1	X	X	1
3	0	1	0	0	X	X	0	1	X
4	0	1	1	1	X				
5	1	0	0						
1	0	0	0						

CLK	Q_n	Q_{n+1}	J	K
↑	0	0	0	X
↑	0	1	1	X
↑	1	0	X	1
↑	1	1	X	0

Design of synchronous counters

state	Q_2	Q_1	Q_0	J_2	K_2	J_1	K_1	J_0	K_0
1	0	0	0	0	X	0	X	1	X
2	0	0	1	0	X	1	X	X	1
3	0	1	0	0	X	X	0	1	X
4	0	1	1	1	X	X	1		
5	1	0	0						
1	0	0	0						

CLK	Q_n	Q_{n+1}	J	K
↑	0	0	0	X
↑	0	1	1	X
↑	1	0	X	1
↑	1	1	X	0

Design of synchronous counters

state	Q_2	Q_1	Q_0	J_2	K_2	J_1	K_1	J_0	K_0
1	0	0	0	0	X	0	X	1	X
2	0	0	1	0	X	1	X	X	1
3	0	1	0	0	X	X	0	1	X
4	0	1	1	1	X	X	1	X	1
5	1	0	0						
1	0	0	0						

CLK	Q_n	Q_{n+1}	J	K
$\uparrow$	0	0	0	X
$\uparrow$	0	1	1	X
$\uparrow$	1	0	X	1
$\uparrow$	1	1	X	0

state	Q_2	Q_1	Q_0	J_2	K_2	J_1	K_1	J_0	K_0
1	0	0	0	0	X	0	X	1	X
2	0	0	1	0	X	1	X	X	1
3	0	1	0	0	X	X	0	1	X
4	0	1	1	1	X	X	1	X	1
5	1	0	0						
1	0	0	0						

CLK	Q_n	Q_{n+1}	J	K
↑	0	0	0	X
↑	0	1	1	X
↑	1	0	X	1
↑	1	1	X	0

Design of synchronous counters

state	Q_2	Q_1	Q_0	J_2	K_2	J_1	K_1	J_0	K_0
1	0	0	0	0	X	0	X	1	X
2	0	0	1	0	X	1	X	X	1
3	0	1	0	0	X	X	0	1	X
4	0	1	1	1	X	X	1	X	1
5	1	0	0	X	1				
1	0	0	0						

CLK	Q_n	Q_{n+1}	J	K
$\uparrow$	0	0	0	X
$\uparrow$	0	1	1	X
$\uparrow$	1	0	X	1
$\uparrow$	1	1	X	0

Design of synchronous counters

state	Q_2	Q_1	Q_0	J_2	K_2	J_1	K_1	J_0	K_0
1	0	0	0	0	X	0	X	1	X
2	0	0	1	0	X	1	X	X	1
3	0	1	0	0	X	X	0	1	X
4	0	1	1	1	X	X	1	X	1
5	1	0	0	X	1	0	X		
1	0	0	0						

CLK	Q_n	Q_{n+1}	J	K
↑	0	0	0	X
↑	0	1	1	X
↑	1	0	X	1
↑	1	1	X	0

state	Q_2	Q_1	Q_0	J_2	K_2	J_1	K_1	J_0	K_0
1	0	0	0	0	X	0	X	1	X
2	0	0	1	0	X	1	X	X	1
3	0	1	0	0	X	X	0	1	X
4	0	1	1	1	X	X	1	X	1
5	1	0	0	X	1	0	X	0	X
1	0	0	0						

CLK	Q_n	Q_{n+1}	J	K
$\uparrow$	0	0	0	X
$\uparrow$	0	1	1	X
$\uparrow$	1	0	X	1
$\uparrow$	1	1	X	0

state	Q_2	Q_1	Q_0	J_2	K_2	J_1	K_1	J_0	K_0
1	0	0	0	0	X	0	X	1	X
2	0	0	1	0	X	1	X	X	1
3	0	1	0	0	X	X	0	1	X
4	0	1	1	1	X	X	1	X	1
5	1	0	0	X	1	0	X	0	X
1	0	0	0						

CLK	Q_n	Q_{n+1}	J	K
↑	0	0	0	X
↑	0	1	1	X
↑	1	0	X	1
↑	1	1	X	0

* We now have the truth tables for J_0, K_0, J_1, K_1, J_2, K_2 in terms of Q_0, Q_1, Q_2. The next step is to find logical functions for each of them.

state	Q_2	Q_1	Q_0	J_2	K_2	J_1	K_1	J_0	K_0
1	0	0	0	0	X	0	X	1	X
2	0	0	1	0	X	1	X	X	1
3	0	1	0	0	X	X	0	1	X
4	0	1	1	1	X	X	1	X	1
5	1	0	0	X	1	0	X	0	X
1	0	0	0						

CLK	Q_n	Q_{n+1}	J	K
↑	0	0	0	X
↑	0	1	1	X
↑	1	0	X	1
↑	1	1	X	0

* We now have the truth tables for J_0, K_0, J_1, K_1, J_2, K_2 in terms of Q_0, Q_1, Q_2. The next step is to find logical functions for each of them.
* Note that we have not tabulated the J and K values for those combinations of Q_0, Q_1, Q_2 which do not occur in the state transition table (such as $Q_2 Q_1 Q_0 = 110$). We treat these as don't care conditions.

state	Q_2	Q_1	Q_0	J_2	K_2	J_1	K_1	J_0	K_0
1	0	0	0	0	X	0	X	1	X
2	0	0	1	0	X	1	X	X	1
3	0	1	0	0	X	X	0	1	X
4	0	1	1	1	X	X	1	X	1
5	1	0	0	X	1	0	X	0	X
1	0	0	0						

J_2

Q_0 \ Q_2Q_1	00	01	11	10
0	0	0	X	X
1	0	1	X	X

K_2

Q_0 \ Q_2Q_1	00	01	11	10
0	X	X	X	1
1	X	X	X	X

J_1

Q_0 \ Q_2Q_1	00	01	11	10
0	0	X	X	0
1	1	X	X	X

K_1

Q_0 \ Q_2Q_1	00	01	11	10
0	X	0	X	X
1	X	1	X	X

J_0

Q_0 \ Q_2Q_1	00	01	11	10
0	1	1	X	0
1	X	X	X	X

K_0

Q_0 \ Q_2Q_1	00	01	11	10
0	X	X	X	X
1	1	1	X	X

Design of synchronous counters

state	Q_2	Q_1	Q_0	J_2	K_2	J_1	K_1	J_0	K_0
1	0	0	0	0	X	0	X	1	X
2	0	0	1	0	X	1	X	X	1
3	0	1	0	0	X	X	0	1	X
4	0	1	1	1	X	X	1	X	1
5	1	0	0	X	1	0	X	0	X
1	0	0	0						

J_2

Q_0 \ Q_2Q_1	00	01	11	10
0	0	0	X	X
1	0	1	X	X

K_2

Q_0 \ Q_2Q_1	00	01	11	10
0	X	X	X	1
1	X	X	X	X

J_1

Q_0 \ Q_2Q_1	00	01	11	10
0	0	X	X	0
1	1	X	X	X

K_1

Q_0 \ Q_2Q_1	00	01	11	10
0	X	0	X	X
1	X	1	X	X

J_0

Q_0 \ Q_2Q_1	00	01	11	10
0	1	1	X	0
1	X	X	X	X

K_0

Q_0 \ Q_2Q_1	00	01	11	10
0	X	X	X	X
1	1	1	X	X

* We treat the unused states ($Q_2\,Q_1\,Q_0 = 101$, 110, 111) as (additional) don't care conditions. Since these are different from the don't care conditions arising from the state transition table, we mark them with a different colour.

Design of synchronous counters

state	Q_2	Q_1	Q_0	J_2	K_2	J_1	K_1	J_0	K_0
1	0	0	0	0	X	0	X	1	X
2	0	0	1	0	X	1	X	X	1
3	0	1	0	0	X	X	0	1	X
4	0	1	1	1	X	X	1	X	1
5	1	0	0	X	1	0	X	0	X
1	0	0	0						

J_2

Q_0 \ Q_2Q_1	00	01	11	10
0	0	0	X	X
1	0	1	X	X

K_2

Q_0 \ Q_2Q_1	00	01	11	10
0	X	X	X	1
1	X	X	X	X

J_1

Q_0 \ Q_2Q_1	00	01	11	10
0	0	X	X	0
1	1	X	X	X

K_1

Q_0 \ Q_2Q_1	00	01	11	10
0	X	0	X	X
1	X	1	X	X

J_0

Q_0 \ Q_2Q_1	00	01	11	10
0	1	1	X	0
1	X	X	X	X

K_0

Q_0 \ Q_2Q_1	00	01	11	10
0	X	X	X	X
1	1	1	X	X

* We treat the unused states ($Q_2\,Q_1\,Q_0 = 101$, 110, 111) as (additional) don't care conditions. Since these are different from the don't care conditions arising from the state transition table, we mark them with a different colour.
* We will assume that a suitable initialization facility is provided to ensure that the counter starts up in one of the five allowed states (say, $Q_2\,Q_1\,Q_0 = 000$).

Design of synchronous counters

state	Q_2	Q_1	Q_0	J_2	K_2	J_1	K_1	J_0	K_0
1	0	0	0	0	X	0	X	1	X
2	0	0	1	0	X	1	X	X	1
3	0	1	0	0	X	X	0	1	X
4	0	1	1	1	X	X	1	X	1
5	1	0	0	X	1	0	X	0	X
1	0	0	0						

J_2

Q_0 \ Q_2Q_1	00	01	11	10
0	0	0	X	X
1	0	1	X	X

K_2

Q_0 \ Q_2Q_1	00	01	11	10
0	X	X	X	1
1	X	X	X	X

J_1

Q_0 \ Q_2Q_1	00	01	11	10
0	0	X	X	0
1	1	X	X	X

K_1

Q_0 \ Q_2Q_1	00	01	11	10
0	X	0	X	X
1	X	1	X	X

J_0

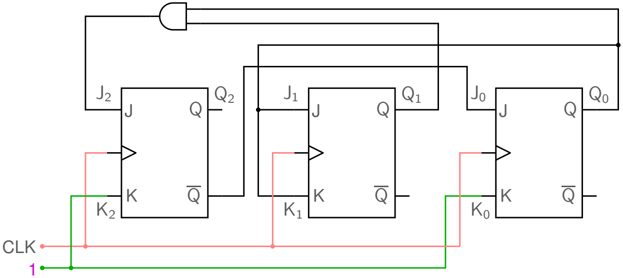
Q_0 \ Q_2Q_1	00	01	11	10
0	1	1	X	0
1	X	X	X	X

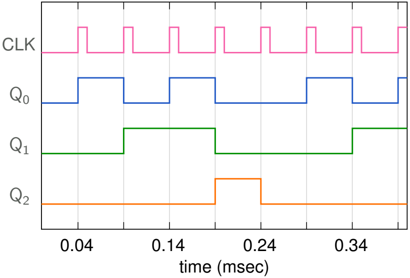
K_0

Q_0 \ Q_2Q_1	00	01	11	10
0	X	X	X	X
1	1	1	X	X

* We treat the unused states ($Q_2\,Q_1\,Q_0 = 101,\ 110,\ 111$) as (additional) don't care conditions. Since these are different from the don't care conditions arising from the state transition table, we mark them with a different colour.
* We will assume that a suitable initialization facility is provided to ensure that the counter starts up in one of the five allowed states (say, $Q_2\,Q_1\,Q_0 = 000$).
* From the K-maps, $J_2 = Q_1\,Q_0$, $K_2 = 1$, $J_1 = Q_0$, $K_1 = Q_0$, $J_0 = \overline{Q_2}$, $K_0 = 1$.

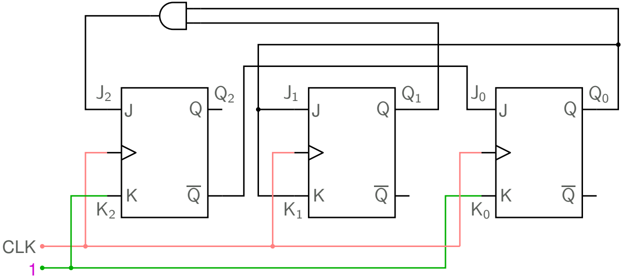
Design of synchronous counters: verification



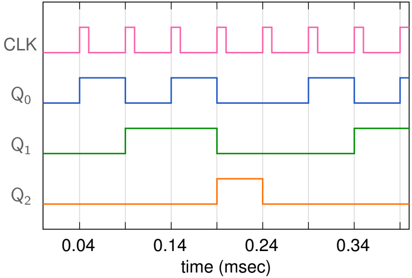


SEQUEL file: `ee101_counter_6.sqproj`

Design of synchronous counters: verification

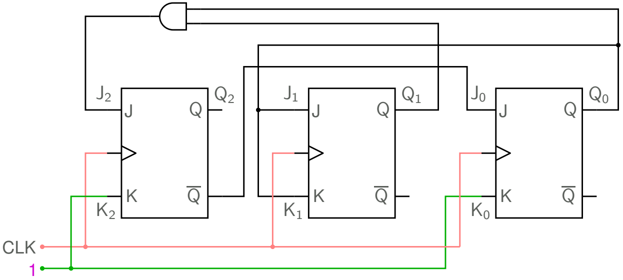


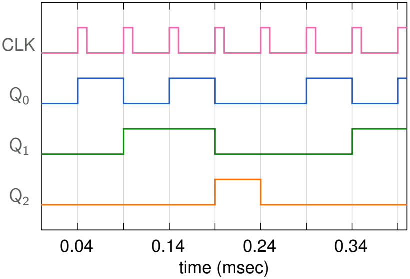
* $J_2 = Q_1 Q_0$,
  $K_2 = 1$,
  $J_1 = Q_0$,
  $K_1 = Q_0$,
  $J_0 = \overline{Q_2}$,
  $K_0 = 1$.



SEQUEL file: `ee101_counter_6.sqproj`

Design of synchronous counters: verification

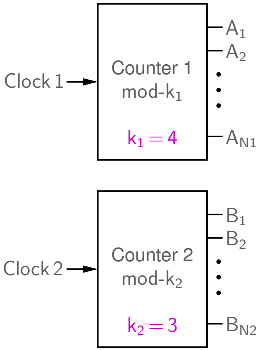




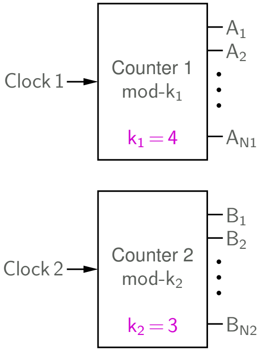
SEQUEL file: `ee101_counter_6.sqproj`

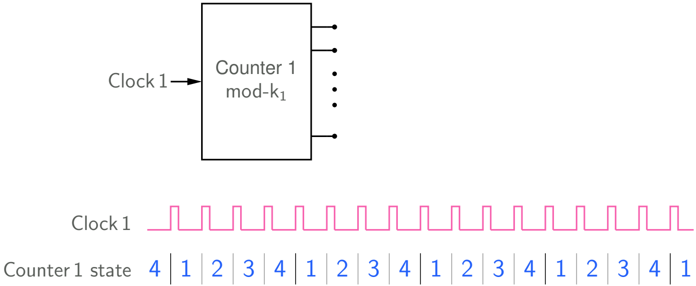
* $J_2 = Q_1 Q_0$,
  $K_2 = 1$,
  $J_1 = Q_0$,
  $K_1 = Q_0$,
  $J_0 = \overline{Q_2}$,
  $K_0 = 1$.
* Note that the design is independent of whether positive or negative edge-triggered flip-flops are used.

Combination of counters: Approach 1

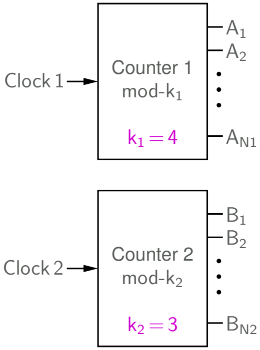


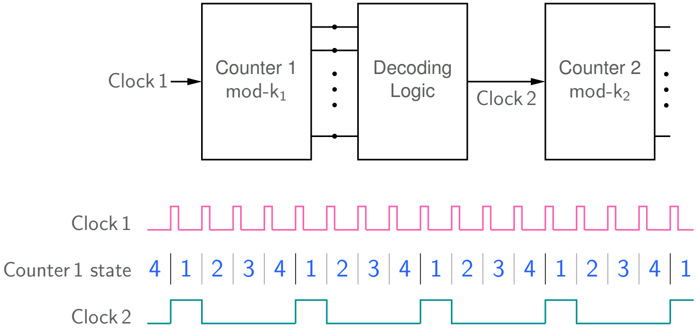
Combination of counters: Approach 1



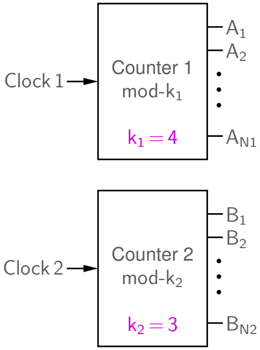


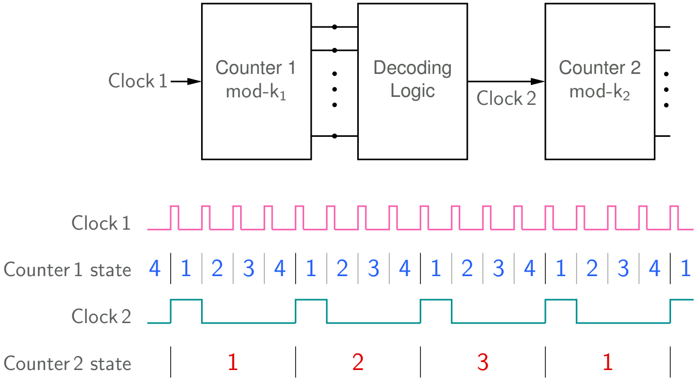
Combination of counters: Approach 1



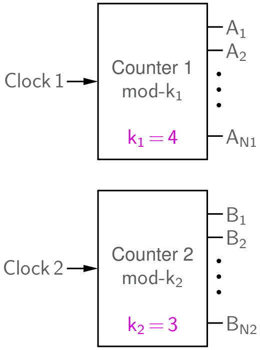


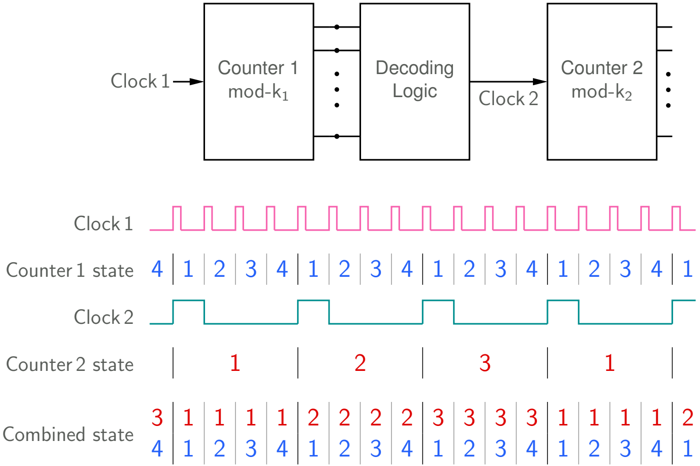
Combination of counters: Approach 1



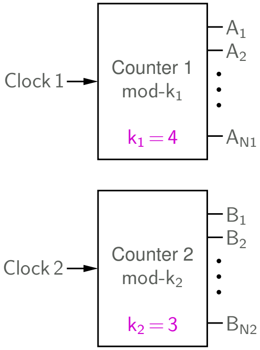


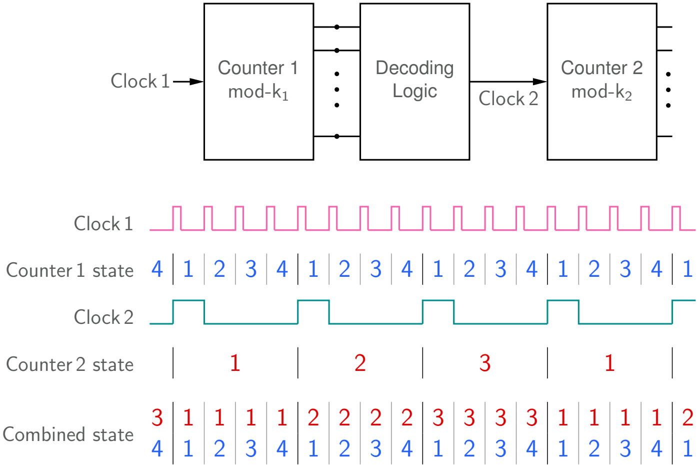
Combination of counters: Approach 1





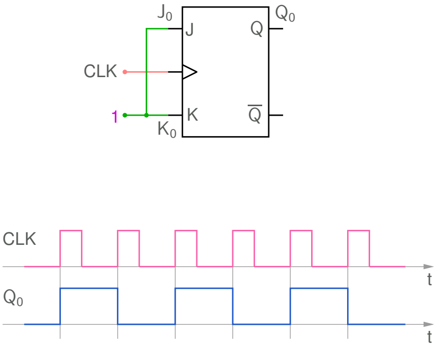
Combination of counters: Approach 1



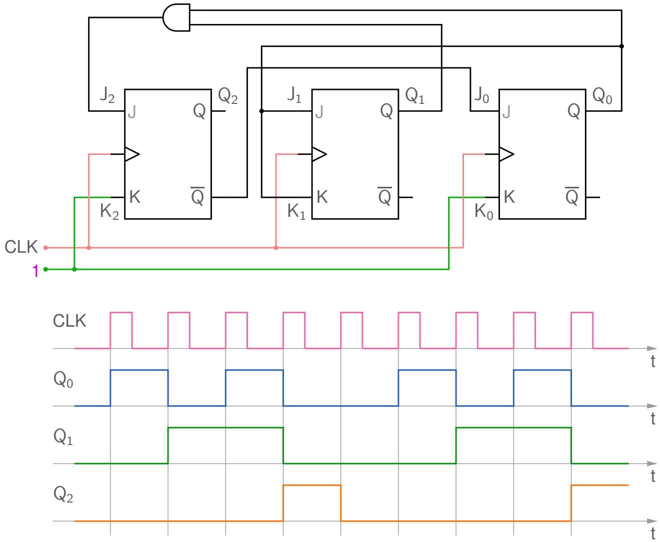


→ the combined counter is a mod-$k_1 k_2$ counter.

Combination of counters: example



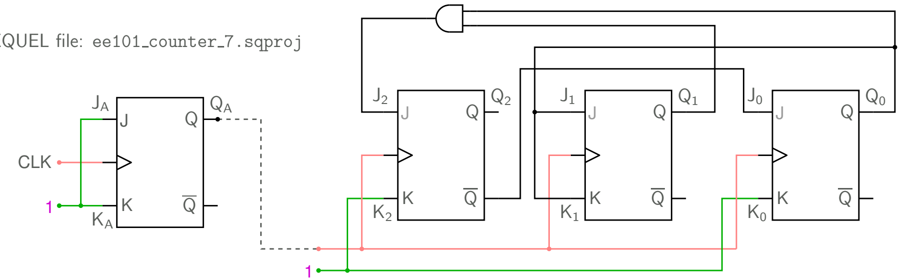
mod–2 counter

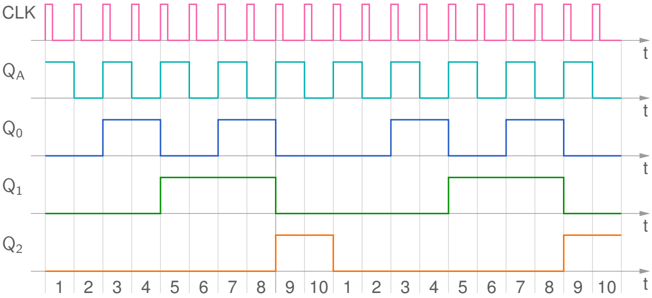


mod–5 counter

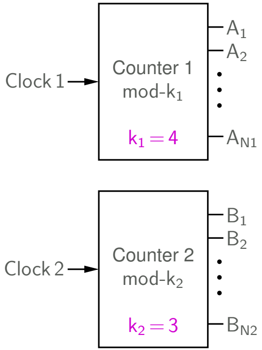
Combination of counters: example

SEQUEL file: `ee101_counter_7.sqproj`

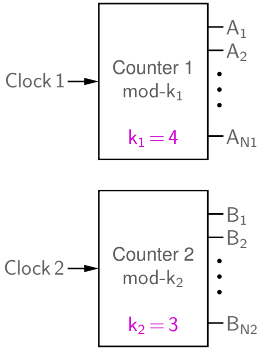


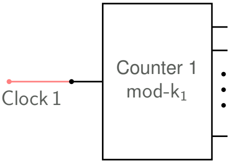


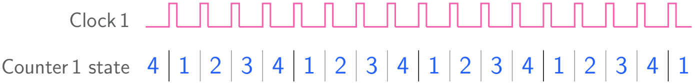
Combination of counters: Approach 2



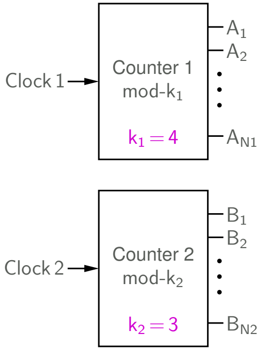
Combination of counters: Approach 2

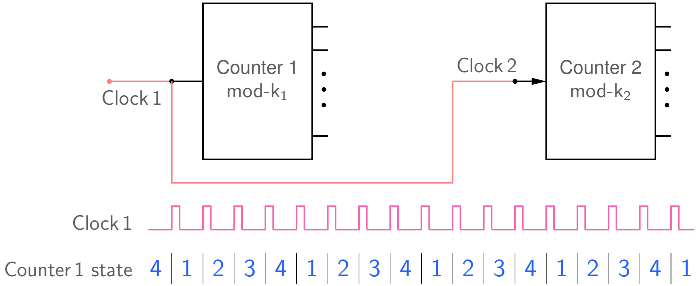




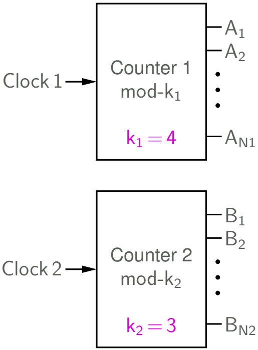


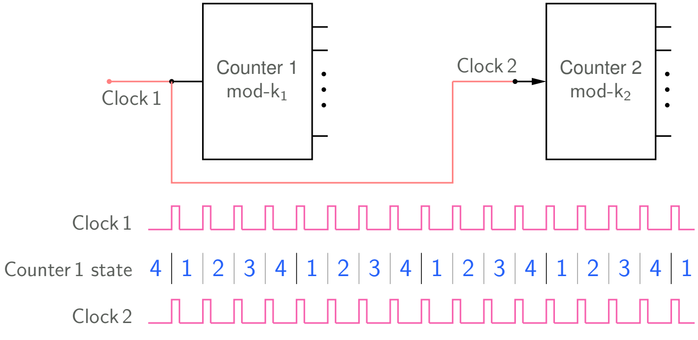
Combination of counters: Approach 2



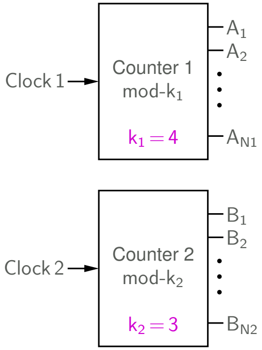


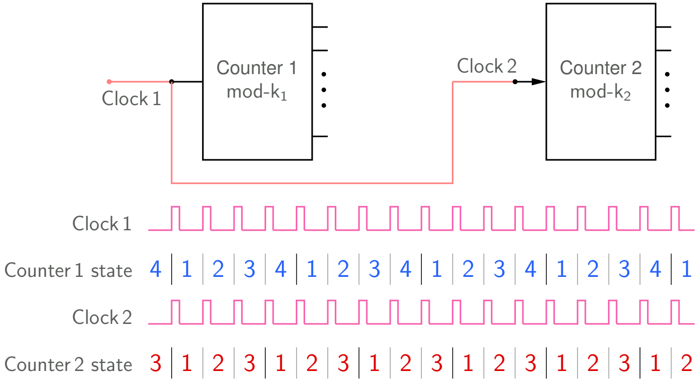
Combination of counters: Approach 2



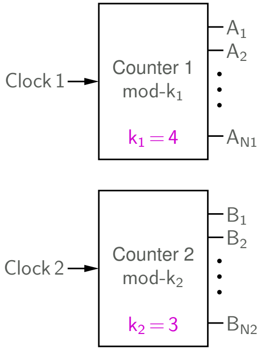


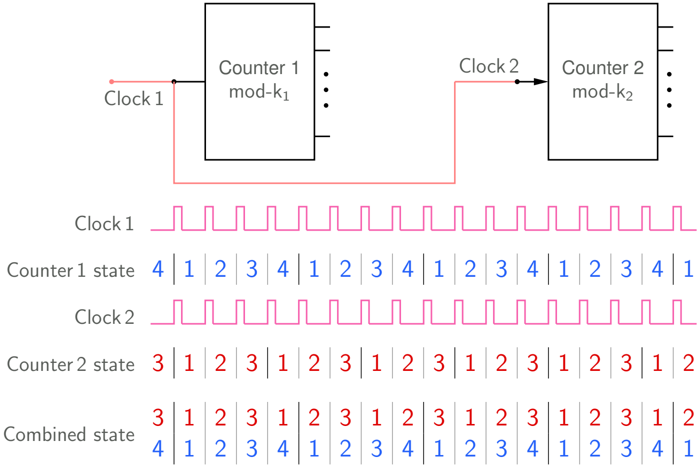
Combination of counters: Approach 2



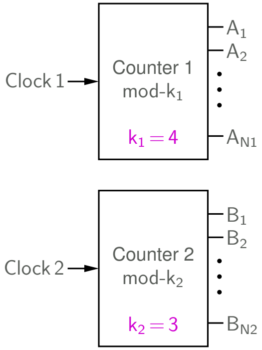


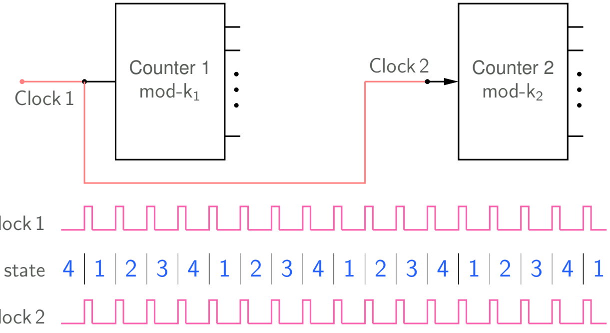
Combination of counters: Approach 2





Combination of counters: Approach 2

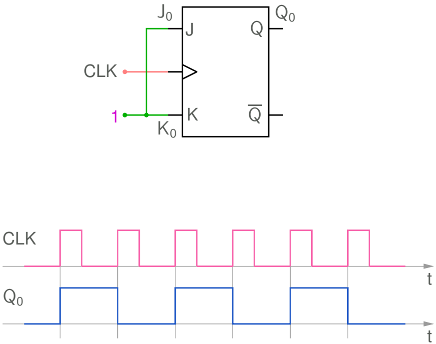




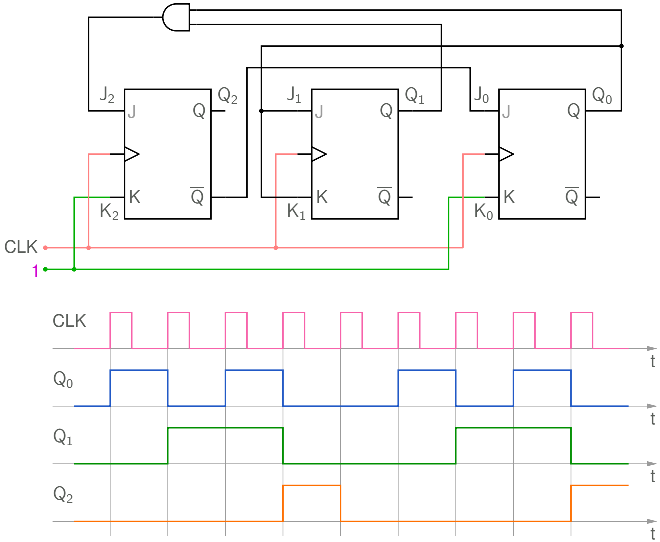
Counter 1 state	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1
Counter 2 state	3	1	2	3	1	2	3	1	2	3	1	2	3	1	2	3	1	2
Combined state	3	1	2	3	1	2	3	1	2	3	1	2	3	1	2	3	1	2
	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1

$\rightarrow$ the combined counter is a mod-k_1k_2 counter.

Combination of counters: example (same as before)

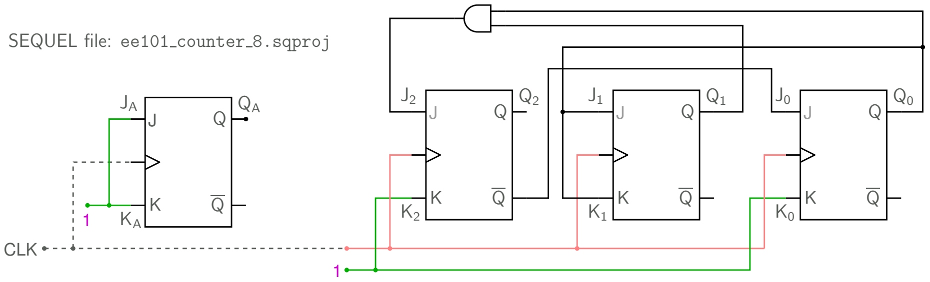


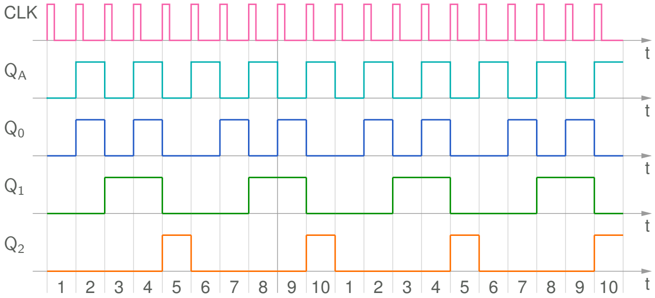
mod–2 counter



mod–5 counter

SEQUEL file: `ee101_counter_8.sqproj`

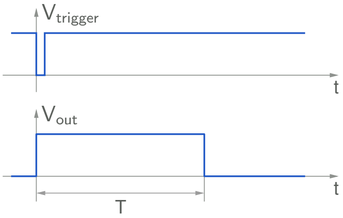




The 555 timer is useful in timer, pulse generation, and oscillator applications. We will look at two common applications.

The 555 timer is useful in timer, pulse generation, and oscillator applications. We will look at two common applications.

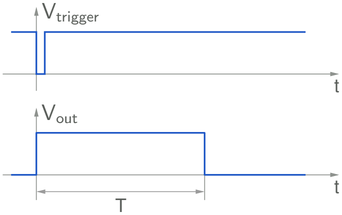
* Monostable multivibrator



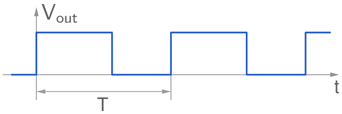
555 timer IC

The 555 timer is useful in timer, pulse generation, and oscillator applications. We will look at two common applications.

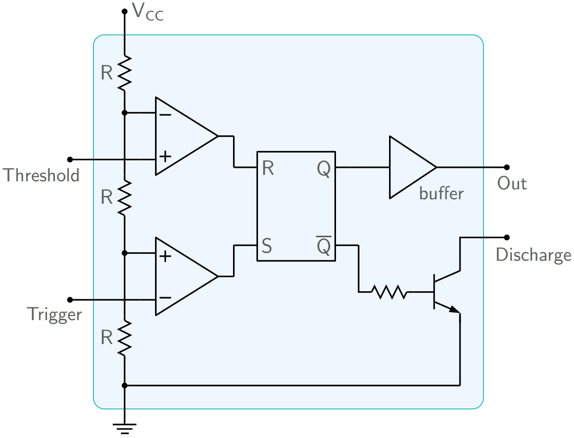
* Monostable multivibrator



* Astable multivibrator

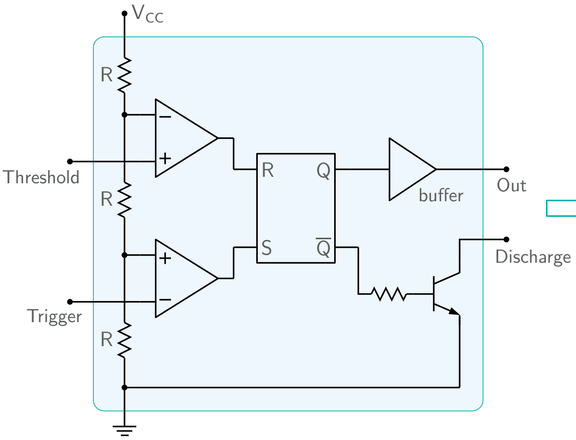


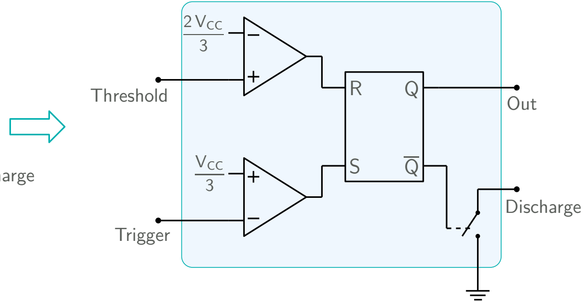
555 timer



R	S	Q	$\overline{Q}$
1	0	0	1
0	1	1	0
0	0	previous	

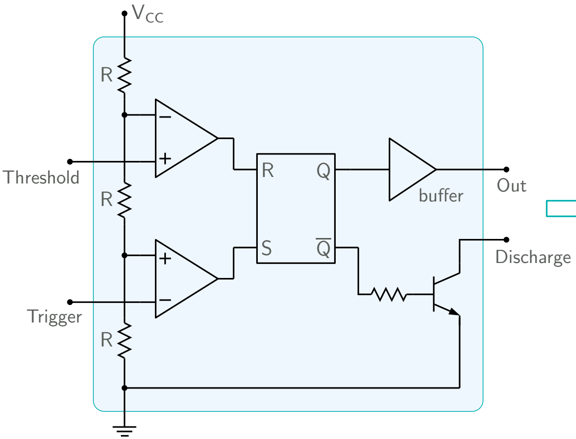
555 timer

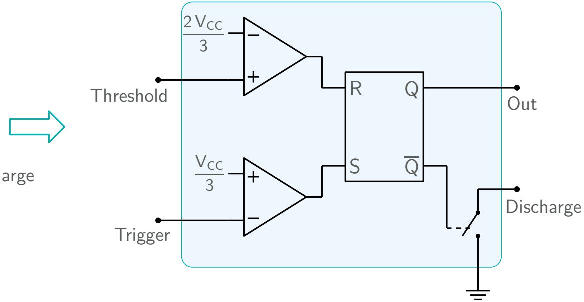




R	S	Q	$\overline{Q}$
1	0	0	1
0	1	1	0
0	0	previous	

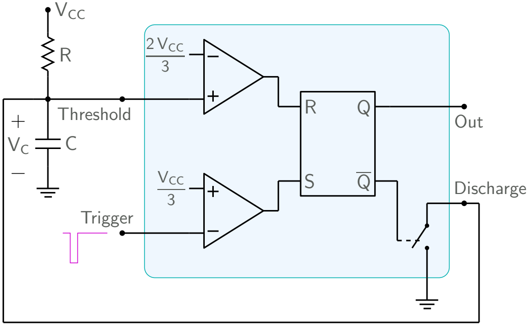
555 timer



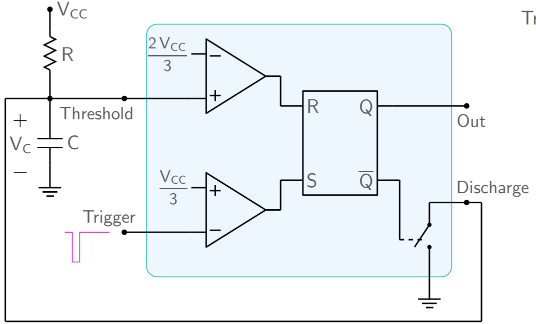


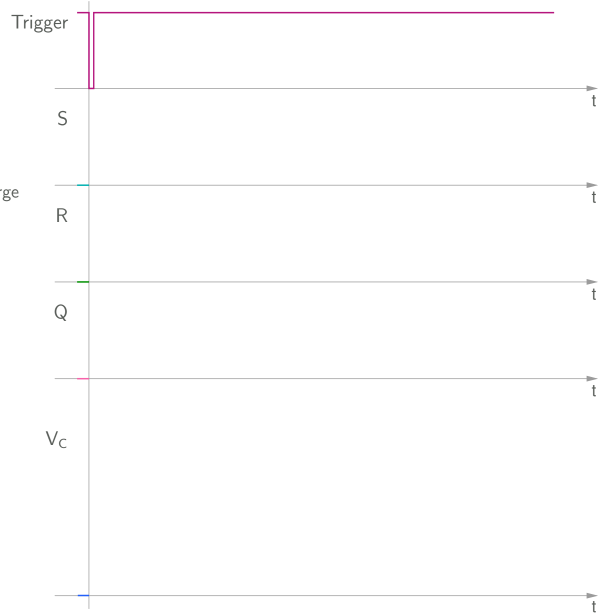
R	S	Q	$\overline{Q}$
1	0	0	1
0	1	1	0
0	0	previous	

555 monostable multivibrator

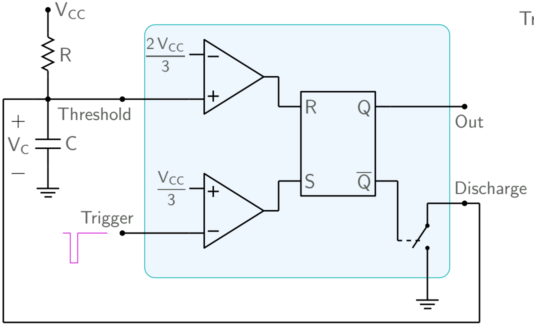


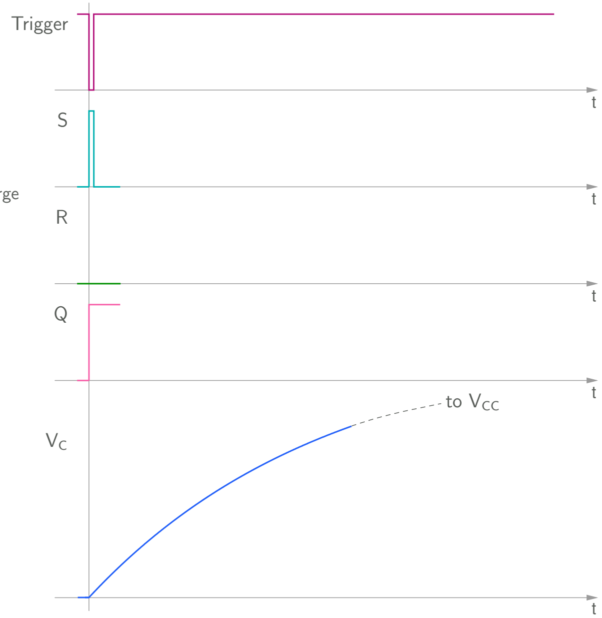
555 monostable multivibrator



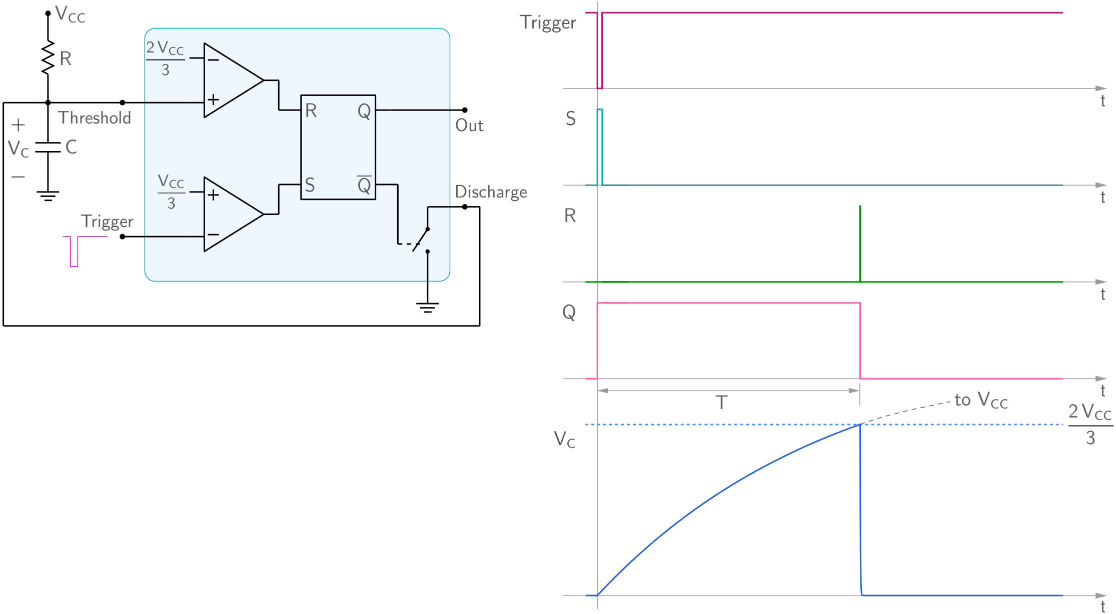


555 monostable multivibrator

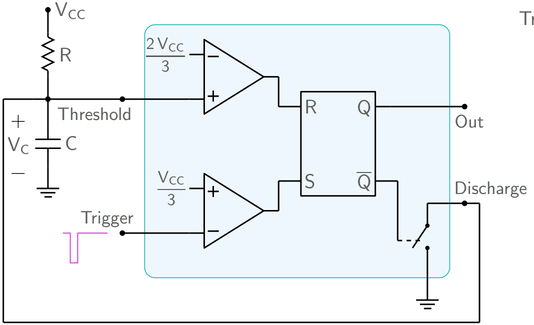


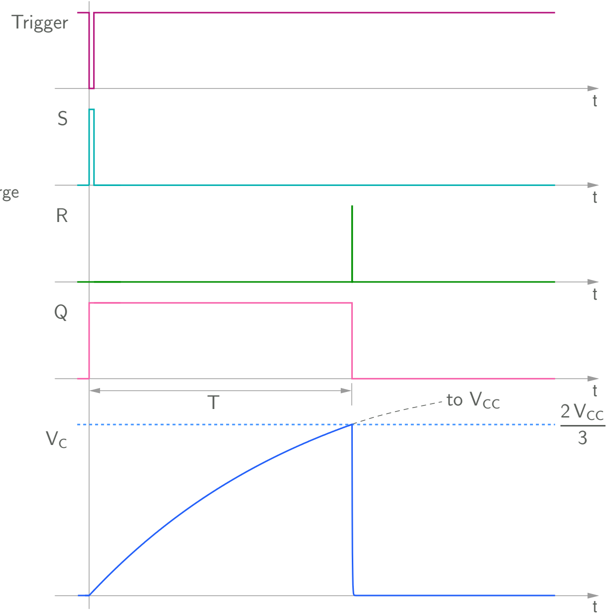


555 monostable multivibrator



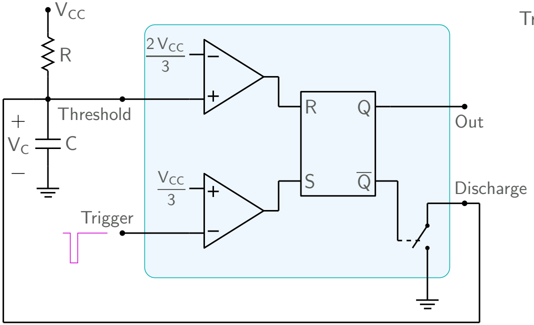
555 monostable multivibrator

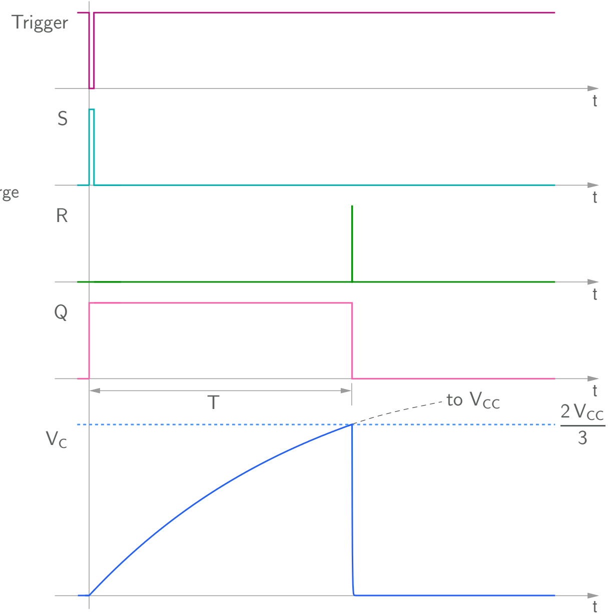




$$V_C(t) = V_{CC}\left(1 - e^{-t/\tau}\right)$$

555 monostable multivibrator



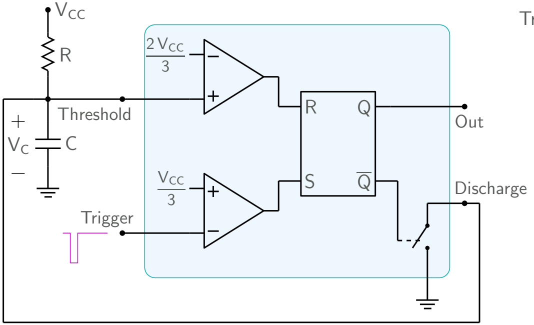


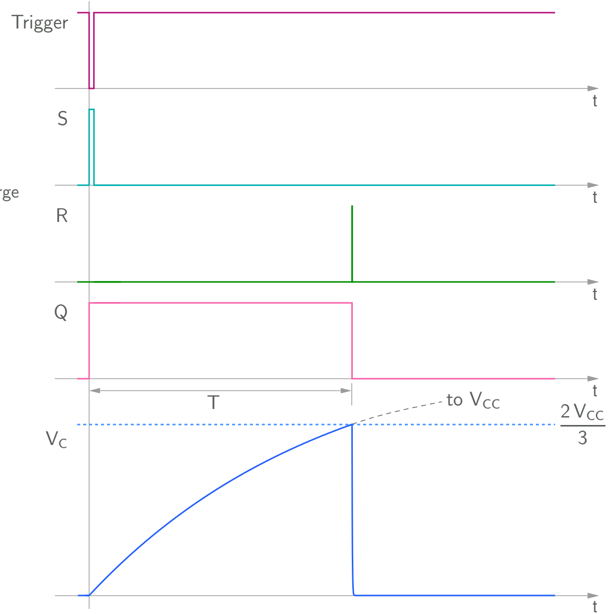
$$V_C(t) = V_{CC}\left(1 - e^{-t/\tau}\right)$$

$$\rightarrow \frac{2\,V_{CC}}{3} = V_{CC}\left(1 - e^{-T/\tau}\right)$$

M. B. Patil, IIT Bombay

555 monostable multivibrator



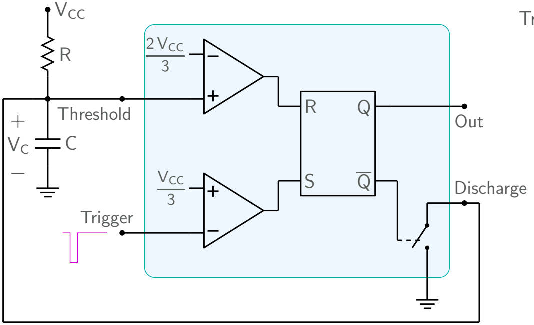


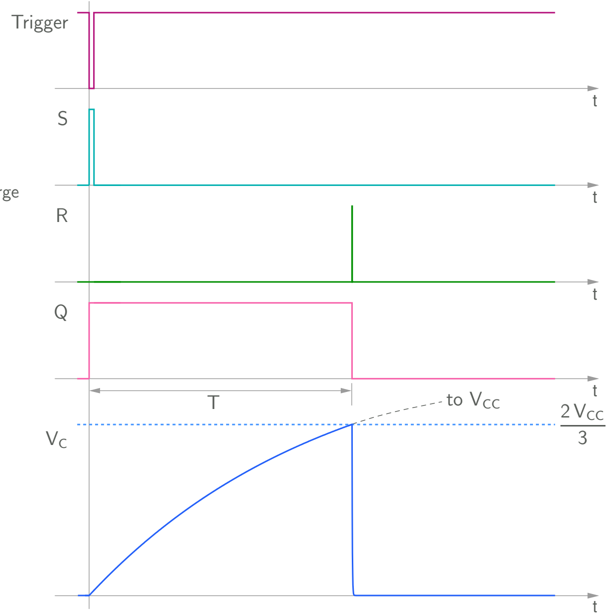
$$V_C(t) = V_{CC}\left(1 - e^{-t/\tau}\right)$$

$$\rightarrow \frac{2\,V_{CC}}{3} = V_{CC}\left(1 - e^{-T/\tau}\right)$$

$$\rightarrow e^{-T/\tau} = \frac{1}{3} \rightarrow \boxed{T = \tau\,\log 3 \approx 1.1\,\tau}$$

555 monostable multivibrator





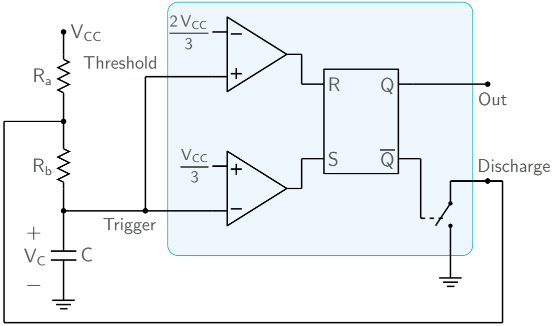
$$V_C(t) = V_{CC}\left(1 - e^{-t/\tau}\right)$$

$$\rightarrow \frac{2\,V_{CC}}{3} = V_{CC}\left(1 - e^{-T/\tau}\right)$$

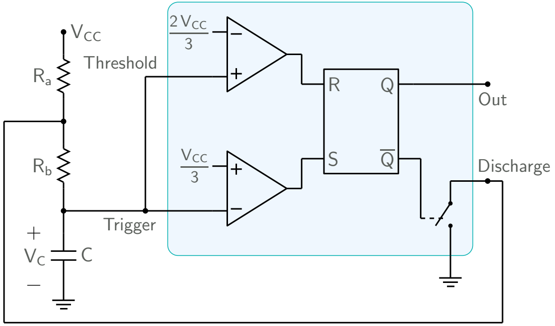
$$\rightarrow e^{-T/\tau} = \frac{1}{3} \rightarrow \boxed{T = \tau\,\log 3 \approx 1.1\,\tau}$$

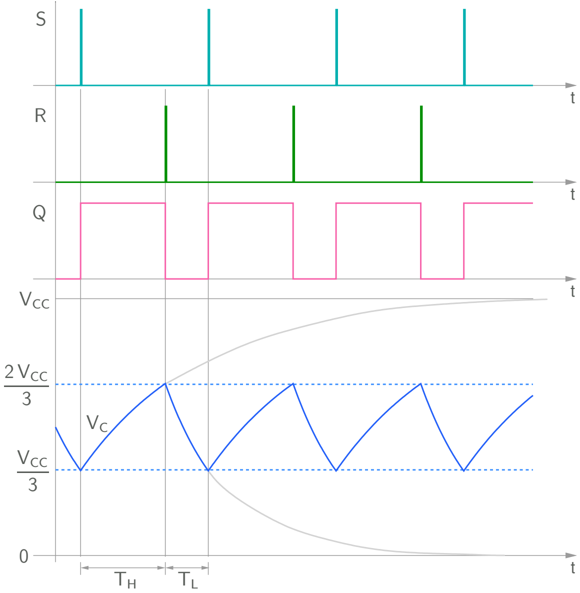
SEQUEL file: `ic555_mono_1.sqproj`

555 astable multivibrator



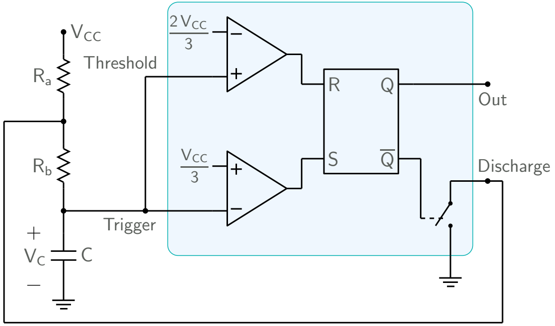
555 astable multivibrator

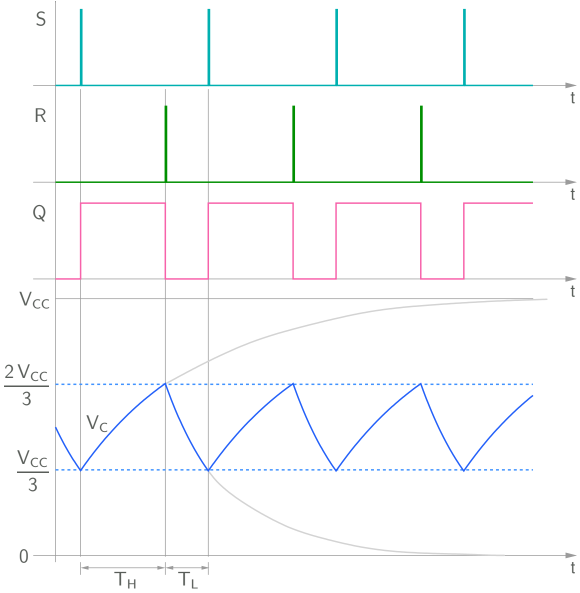




M. B. Patil, IIT Bombay

555 astable multivibrator

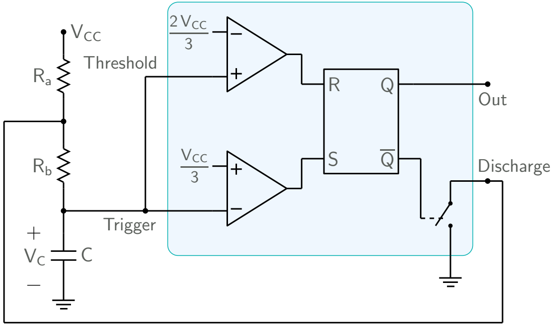


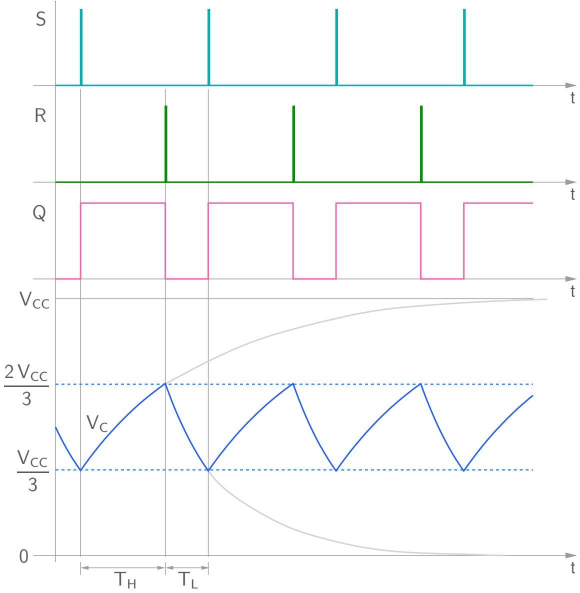


Charging:

$$V_C(0) = \frac{V_{CC}}{3}, \;\; V_C(\infty) = V_{CC}.$$

555 astable multivibrator





Charging:

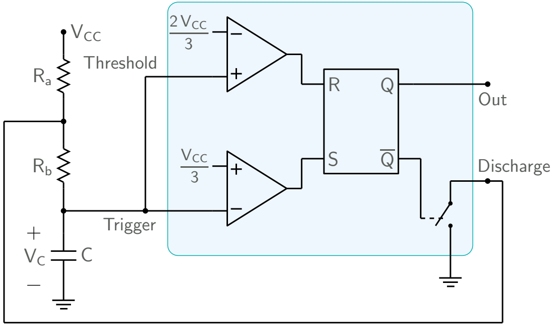
$$V_C(0) = \frac{V_{CC}}{3}, \;\; V_C(\infty) = V_{CC}.$$

Let $V_C(t) = A\,e^{-t/\tau_1} + B$

$$\rightarrow B = V_{CC}, \;\; A = -\frac{2\,V_{CC}}{3}$$

M. B. Patil, IIT Bombay

555 astable multivibrator



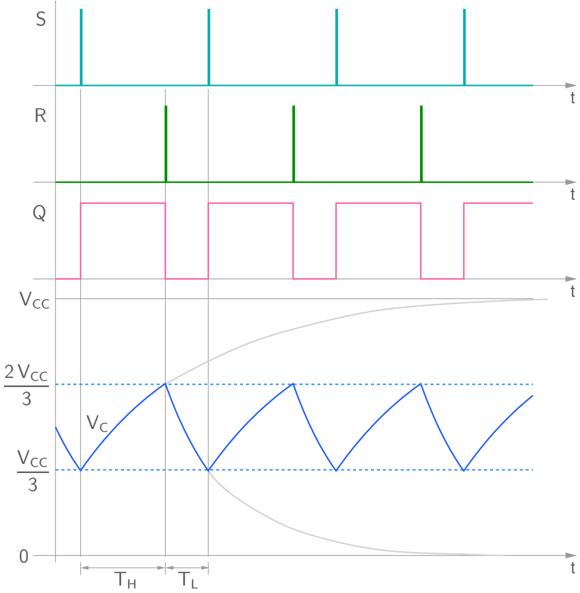
Charging:

$$V_C(0) = \frac{V_{CC}}{3},\ \ V_C(\infty) = V_{CC}.$$

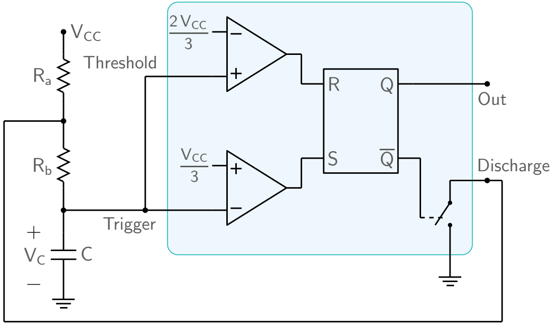
Let $V_C(t) = A\,e^{-t/\tau_1} + B$

$$\rightarrow B = V_{CC},\ \ A = -\frac{2\,V_{CC}}{3}$$

$$\frac{2\,V_{CC}}{3} = -\frac{2\,V_{CC}}{3}\,e^{-T_H/\tau_1} + V_{CC}$$



555 astable multivibrator



Charging:

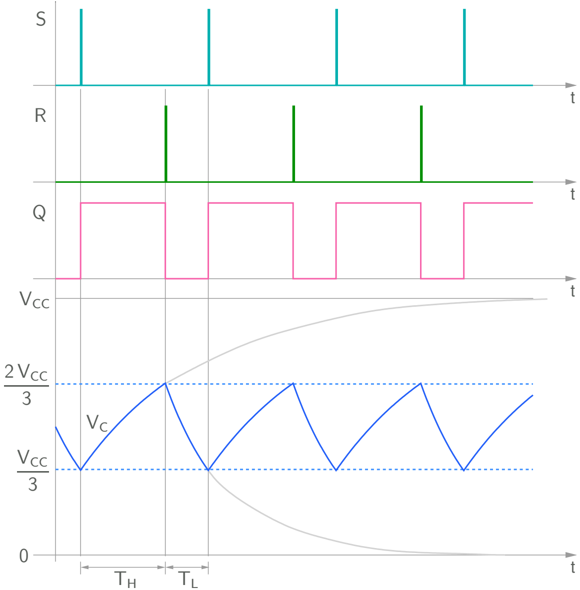
$$V_C(0) = \frac{V_{CC}}{3},\;\; V_C(\infty) = V_{CC}.$$

Let $V_C(t) = A\,e^{-t/\tau_1} + B$

$$\rightarrow B = V_{CC},\;\; A = -\frac{2\,V_{CC}}{3}$$

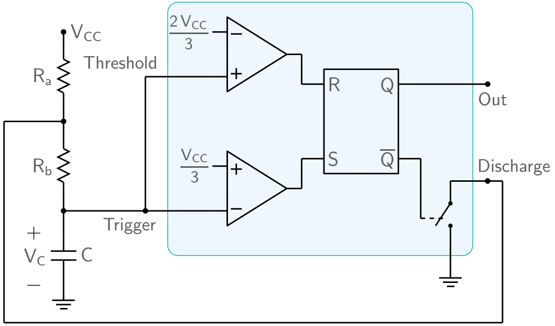
$$\frac{2\,V_{CC}}{3} = -\frac{2\,V_{CC}}{3}\,e^{-T_H/\tau_1} + V_{CC}$$

$\rightarrow T_H = \tau_1 \log 2$, with $\tau_1 = (R_a + R_b)\,C$.

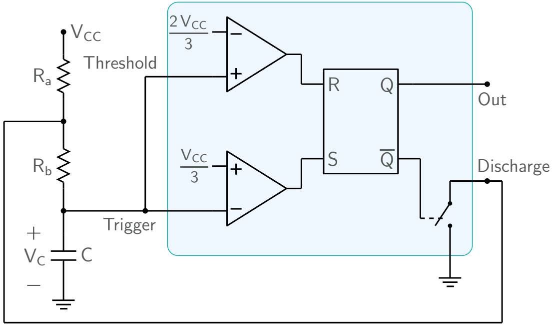


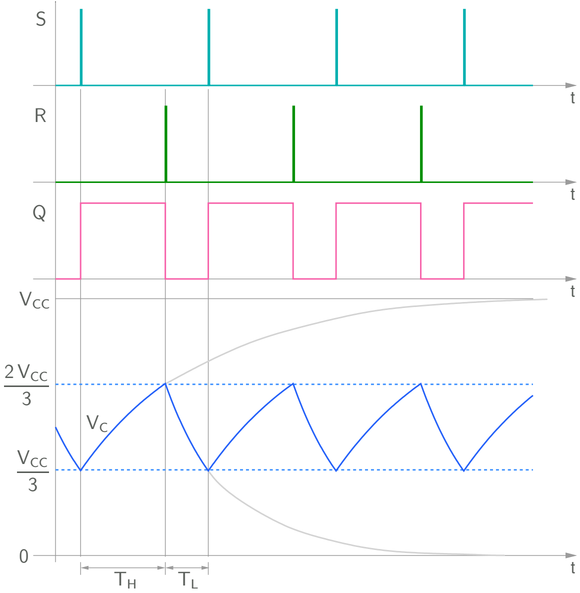
M. B. Patil, IIT Bombay

555 astable multivibrator

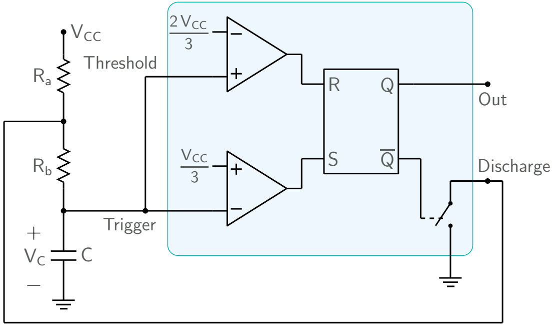


555 astable multivibrator



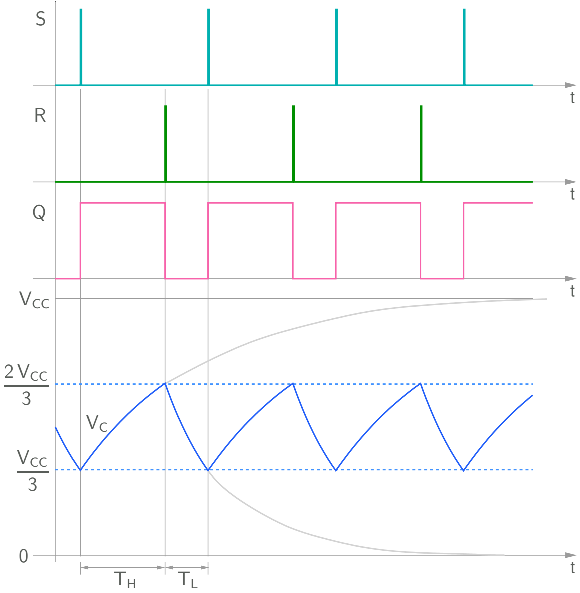


555 astable multivibrator

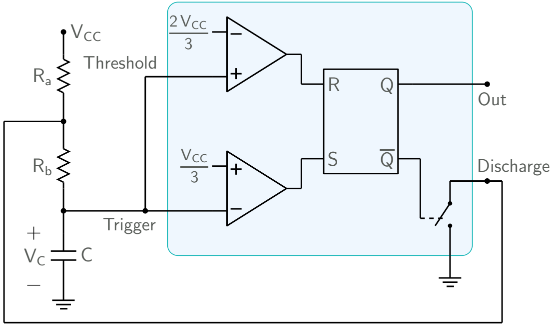


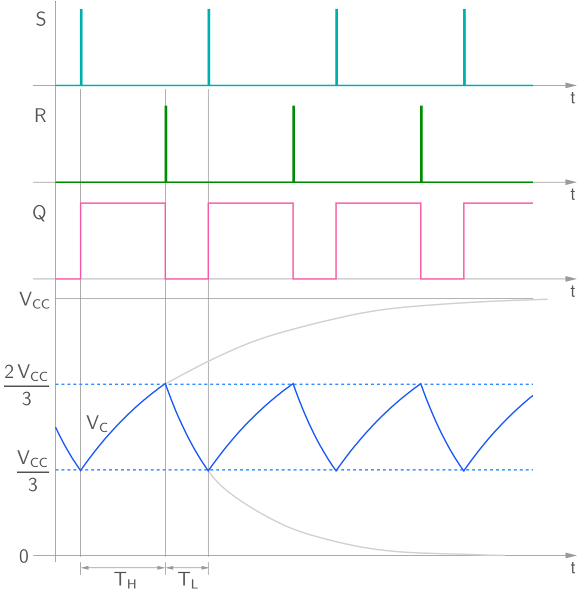
Discharging: $V_C(0) = \dfrac{2\,V_{CC}}{3}$, $V_C(\infty) = 0.$

$\rightarrow V_C(t) = \dfrac{2\,V_{CC}}{3}\, e^{-t/\tau_2}$



555 astable multivibrator



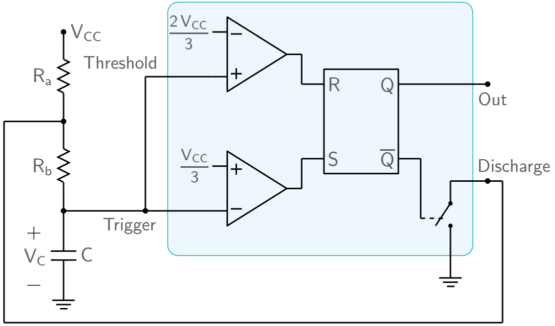


Discharging: $V_C(0) = \dfrac{2\,V_{CC}}{3}$, $V_C(\infty) = 0.$

$$\rightarrow V_C(t) = \frac{2\,V_{CC}}{3}\, e^{-t/\tau_2}$$

$$\frac{V_{CC}}{3} = \frac{2\,V_{CC}}{3}\, e^{-T_L/\tau_2}$$

555 astable multivibrator

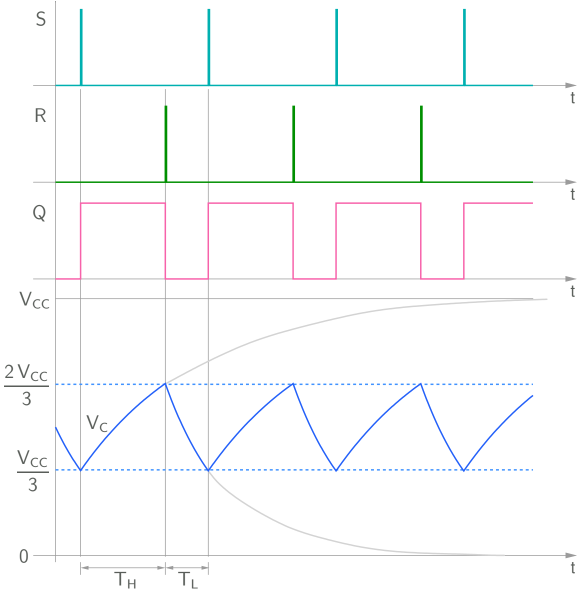


Discharging: $V_C(0) = \dfrac{2\,V_{CC}}{3}$, $V_C(\infty) = 0$.

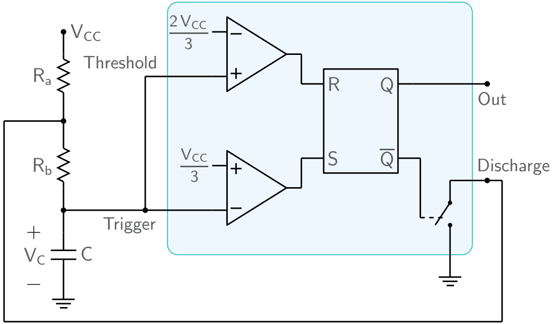
$\rightarrow V_C(t) = \dfrac{2\,V_{CC}}{3}\, e^{-t/\tau_2}$

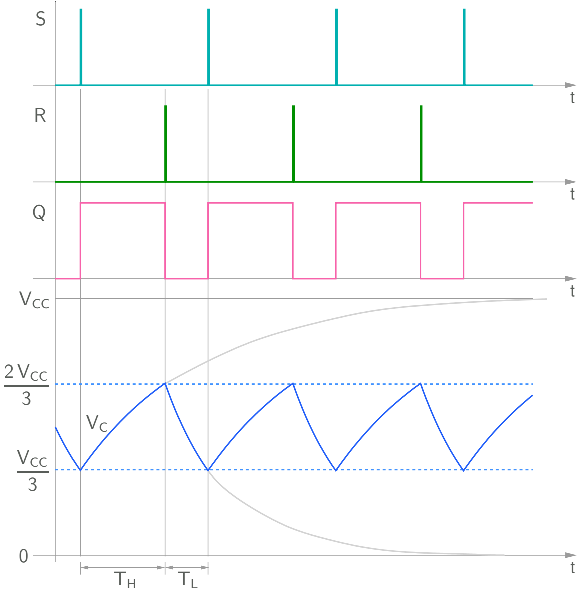
$\dfrac{V_{CC}}{3} = \dfrac{2\,V_{CC}}{3}\, e^{-T_L/\tau_2}$

$\rightarrow T_L = \tau_2 \log 2$, with $\tau_2 = R_b C$.



555 astable multivibrator





Discharging: $V_C(0) = \dfrac{2\,V_{CC}}{3}$, $V_C(\infty) = 0$.

$$\rightarrow V_C(t) = \frac{2\,V_{CC}}{3}\, e^{-t/\tau_2}$$

$$\frac{V_{CC}}{3} = \frac{2\,V_{CC}}{3}\, e^{-T_L/\tau_2}$$

$\rightarrow T_L = \tau_2 \log 2$, with $\tau_2 = R_b C$.

SEQUEL file: `ic555_astable_1.sqproj`