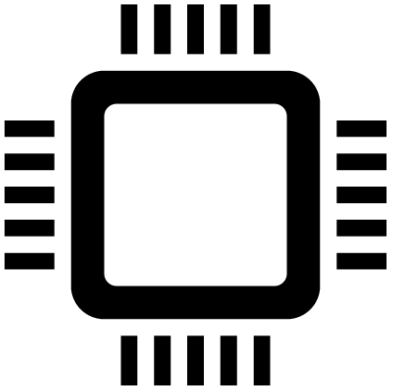
Programowanie struktur cyfrowych



Podstawowe zagadnienia Układy sekwencyjne

dr Aleksander Lamża

Uniwersytet J. Kochanowskiego w Kielcach
Uniwersytet Śląski w Katowicach

aleksander.lamza@us.edu.pl

Układy sekwencyjne

W omawianych poprzednio układach kombinacyjnych cechą charakterystyczną było to, że **stan ich wyjść zależał tylko od stanu wejść**.

W układach sekwencyjnych jest inaczej. Uogólniając, stan wyjść takich układów zależy od stanu wejść (zazwyczaj, choć niekoniecznie) oraz **stanu poprzedniego**, czyli tego, który był przed zmianą.

Tak naprawdę układ sekwencyjny jest połączeniem układu kombinacyjnego i rejestru (pamięci stanu).

Ale po kolei...

Układy sekwencyjne

Zastanówmy się, co by było, gdybyśmy stan z wyjścia bramki OR podali na jej wejście:

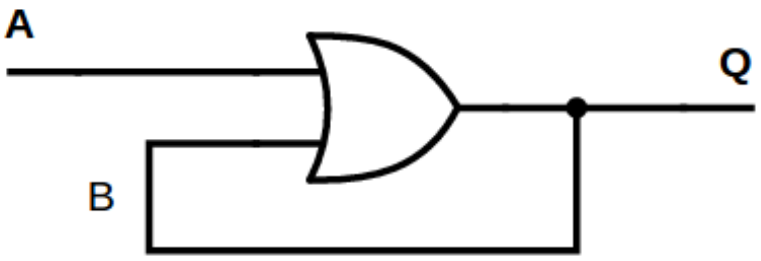


Tabela prawdy bramki OR

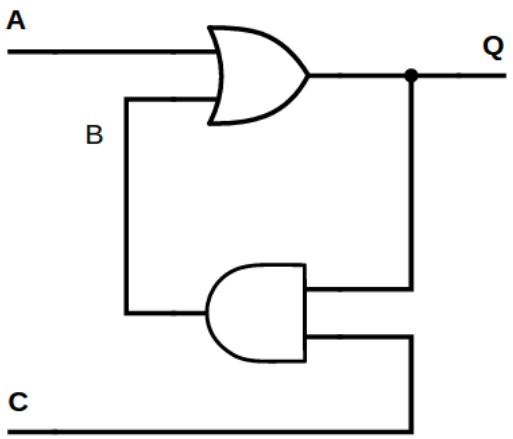
A	B	Y
0	0	**0**
0	1	**1**
1	0	**1**
1	1	**1**

Na początku na obu wejściach jest 0. Po podaniu na A jedynki na wyjściu ustala się stan wysoki (1), który jest kierowany na wejście B. Jak wiadomo, jeżeli którekolwiek wejście bramki OR jest równe 1, wyjście również jest 1. W związku z tym stan wejścia A nie ma teraz wpływu na stan wyjścia, które zawsze jest równe 1.

Można powiedzieć, że „zapamiętaliśmy” jedynkę.
Ale jak ją „zapomnieć”, czyli ustawić zero?

Układy sekwencyjne

Trzeba „przerwać” połączenie Q → B. Możemy użyć bramki AND:



Aby skasować wyjście (ustawić 0), należy na wejście C podać 0.
W celu zachowania podstawowego działania (czyli zapamiętywania 1) wejście C musi być w stanie 1.

Układ działa, ale w praktyce by się nie sprawdził, bo jest zbudowany z dwóch różnych bramek.

Sprawdźmy, czy można to zrobić inaczej.

Układy sekwencyjne

Spróbujmy zbudować podstawowy układ zapamiętujący za pomocą bramki NOR, czyli OR z zanegowanym wyjściem:

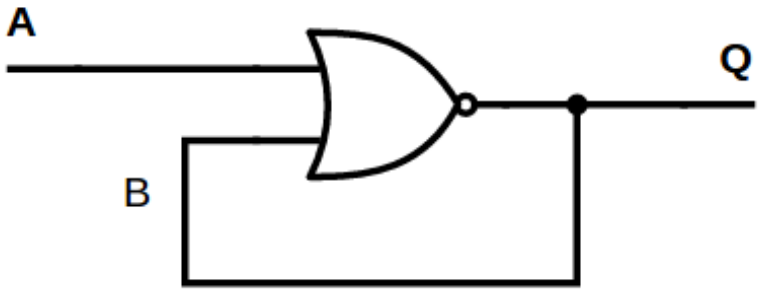


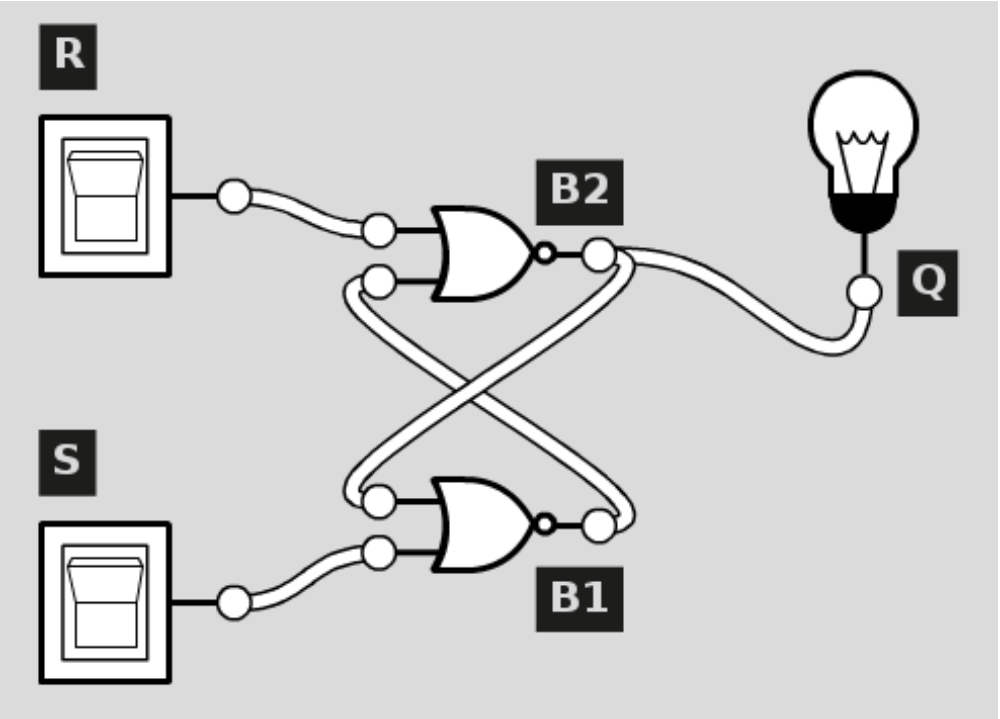
Tabela prawdy bramki NOR

A	B	Y
0	0	**1**
0	1	**0**
1	0	**0**
1	1	**0**

Jak działa układ? Trudno określić stan początkowy, więc rozważmy działanie od podania jedynki na wejście A.
Jeżeli na A jest 1, na wyjściu na pewno jest 0. Ale co będzie, jeżeli wyłączymy wejście (A = 0)? Zero i zero da nam 1 na wyjściu, co z kolei spowoduje zmianę na 0 i tak w kółko... Układ się wzbudził i generuje sygnał 01010101010...
Z całą pewnością można powiedzieć, że nie zapamiętuje stanu.

Układy sekwencyjne

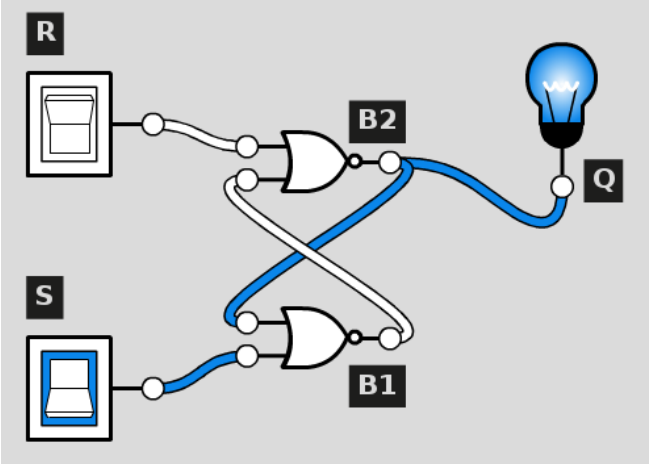
Dodajmy drugą bramkę NOR i połączmy je w taki fantazyjny sposób:



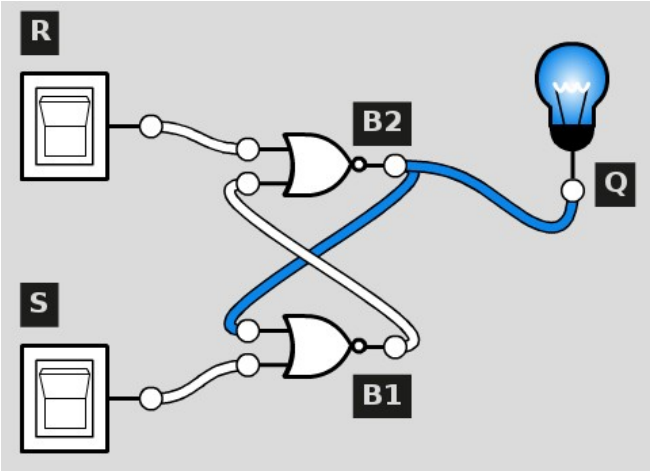
Układ ma dwa wejścia: **R** i **S** oraz wyjście **Q**.
Jak to działa?

Układy sekwencyjne

Jeżeli na wejście S podamy 1, na wyjściu bramki B1 pojawi się 0 (bez względu na stan drugiego wejście bramki). To z kolei spowoduje włączenie wyjścia bramki B2 (jeżeli wejście R jest równe 0).

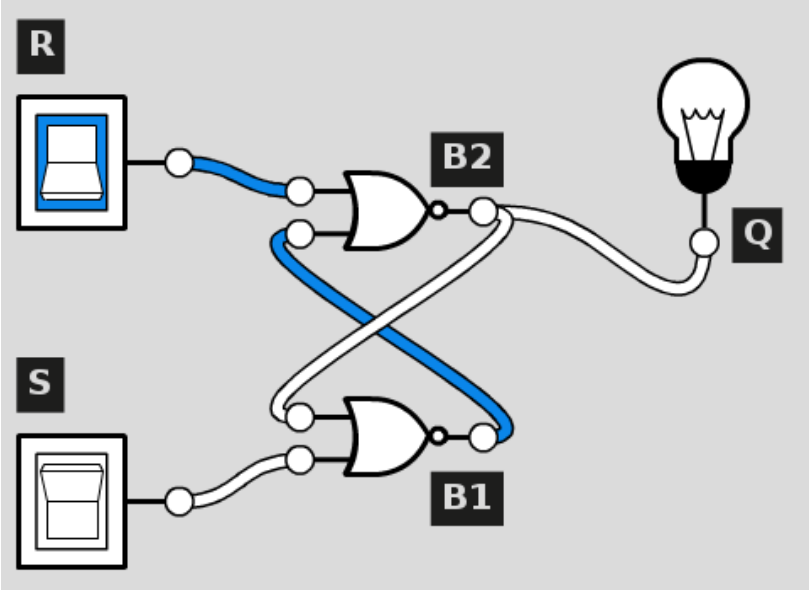


Stan wyjścia Q jest podawany na drugie wejście bramki B1. Co to oznacza? Jeśli wyłączymy wejście S, bramka B1 i tak będzie miała 0 na wyjściu:



Układy sekwencyjne

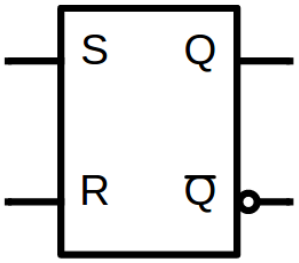
Aby wyłączyć wyjście Q, trzeba podać 1 na wejście R:



Powoduje to ustalenie się stanu 0 na wyjściu B2, co prowadzi do włączenia bramki B1. Stan z jej wyjścia jest przenoszony na drugie wejście bramki B2, dzięki czemu wyłączenie jest „podtrzymywane”, tak jak poprzednio stan włączenia.

Przerzutnik RS

Omówiony układ jest nazywany **przerzutnikiem RS** i jest podstawowym elementem pamiętającym stan.



Ma dwa wejścia: S (ustawiające) i R (kasujące) oraz przeważnie dwa wyjścia: Q i zanegowane Q.

Tabelę prawdy można przedstawić następująco:

S	R	Q_t
0	0	Q_{t-1}
0	1	**0**
1	0	**1**
1	1	**?**

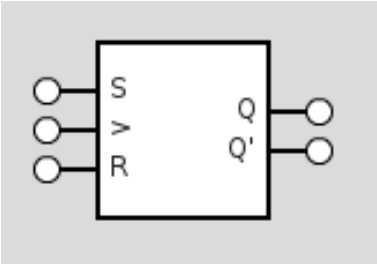
Stan Q_{t-1} oznacza stan poprzedni. Znak zapytania oznacza stan nieustalony (i przy okazji niedozwolony, bo może powodować wzbudzanie się układu).

Układy sekwencyjne (a)synchroniczne

Przerzutnik, który analizowaliśmy, jest układem **asynchronicznym**. Dlaczego? Dlatego, że zmiana stanu jego wyjść następuje od razu po zmianie stanu wejść.

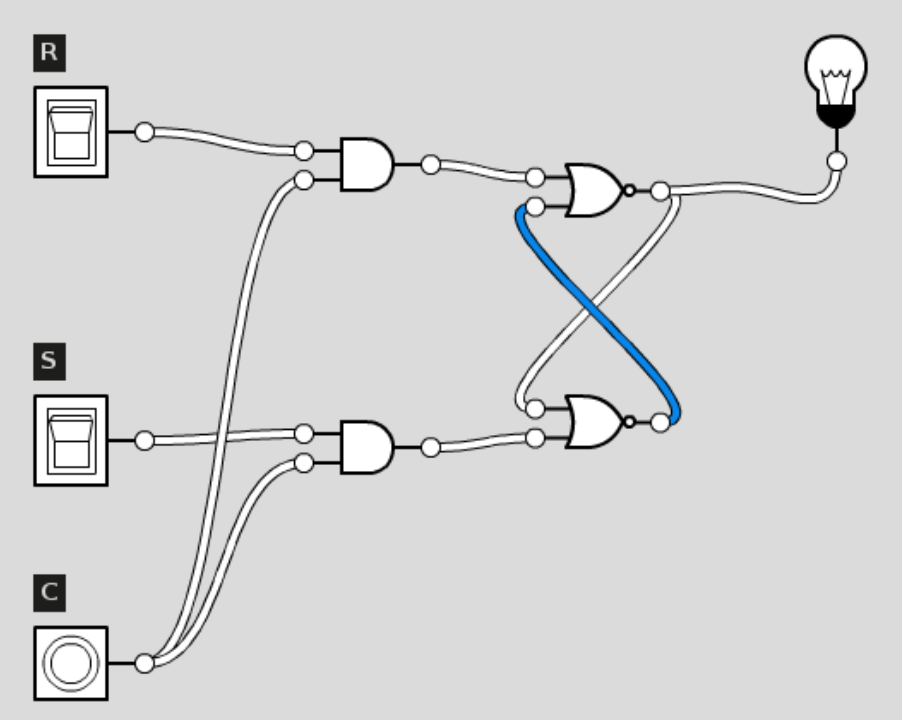
Przeciwieństwem układów asynchronicznych są układy **synchroniczne**, w których stan wyjść zmienia się w takt sygnału zegarowego.

Zastanówmy się, jak będzie wyglądała realizacja synchronicznego przerzutnika RS.



Układy sekwencyjne (a)synchroniczne

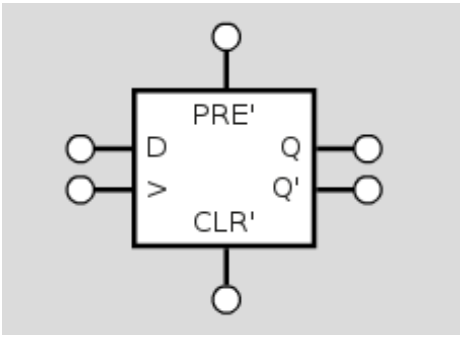
W najprostszej realizacji wystarczy „zablokować” wejścia S i R bramkami AND, które „przepuszczają” sygnały przy aktywnym stanie wejścia zegarowego.



W praktyce większość układów sekwencyjnych zmienia stan w wyniku zbocza sygnału zegarowego, czyli zmiany 0→1 lub 1→0.

Przerzutnik D

Podstawowym przerzutnikiem synchronicznym z prawdziwego zdarzenia jest przerzutnik D.

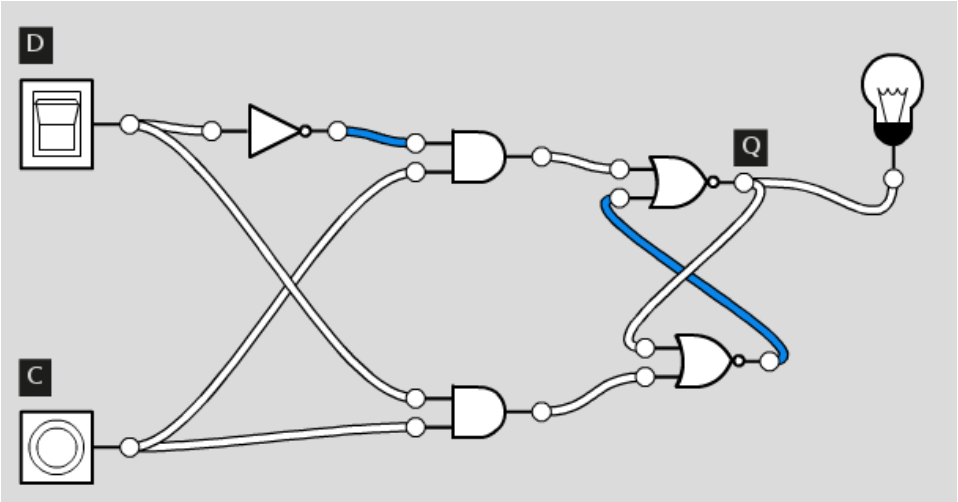


Stan z wejścia danych (D) jest przenoszony na wyjście przy zboczu narastającym (0→1) lub opadającym (1→0).

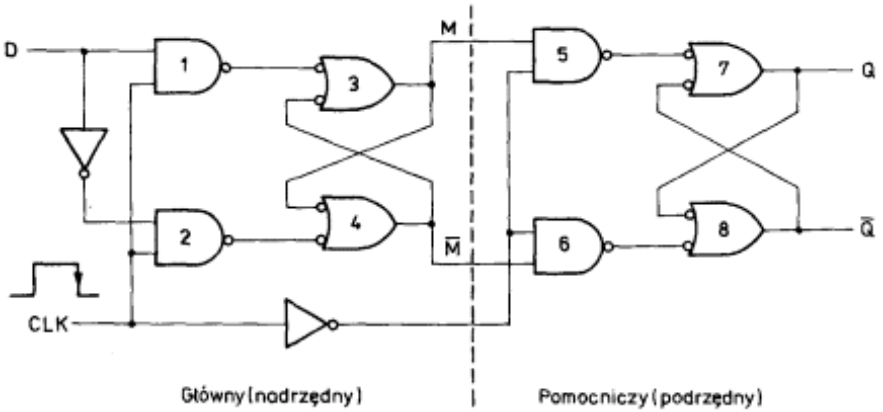
C	D	Q_{t+1}
0/1	X	Q_t
⌟	0	**0**
⌟	1	**1**

Przerzutnik D

W najprostszej realizacji przerzutnik D można zrobić z przerzutnika RS:



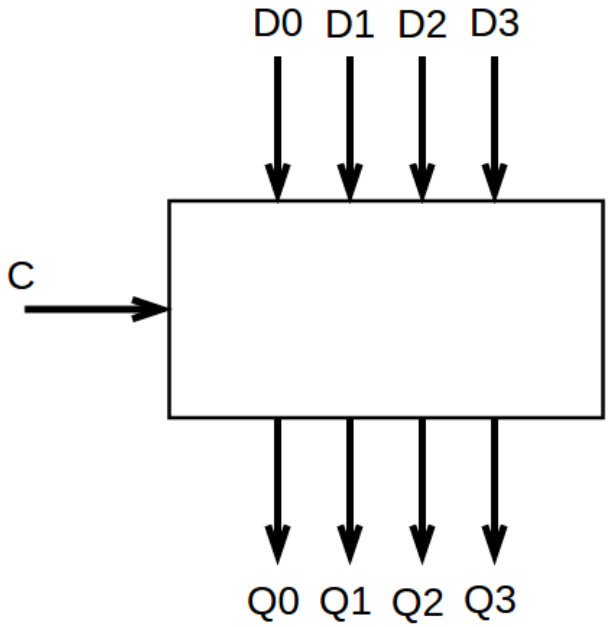
Aby stan był ustawiany zboczem, trzeba zastosować dwustopniowe wyzwalanie, np.:



Rejestry

Przerzutnik D jest elementem składowym wielu innych, bardziej złożonych, układów synchronicznych, np. **rejestrów**.

Podstawowym typem rejestru jest rejestr z równoległym wejściem i równoległym wyjściem:

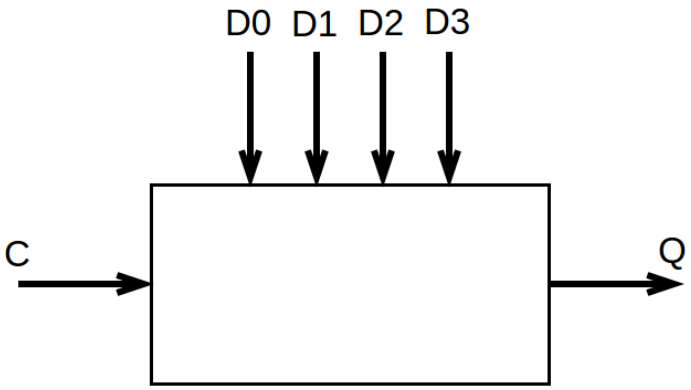


Układ ten służy do przechowywania wielobitowych słów danych.

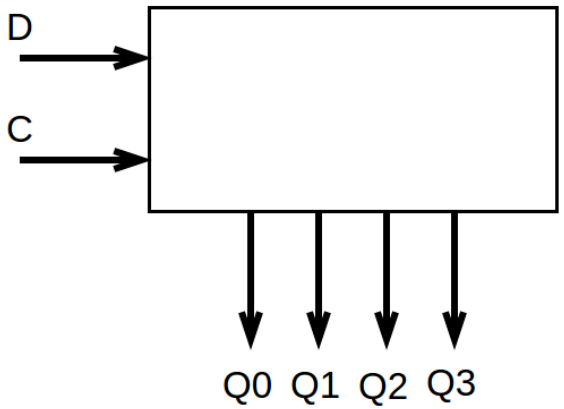
Rejestry

Niekiedy dane w postaci równoległej trzeba zamienić na szeregową (i na odwrót).

Wtedy przydają się rejestry równoległo-szeregowe:



i szeregowo-równoległe:



Licznik

Teraz zastanówmy się, jak moglibyśmy zrealizować układ liczący w górę lub w dół, czyli **licznik**.

Zacznijmy od przyjrzenia się kolejnym wartościom 0 – 7 w postaci binarnej:

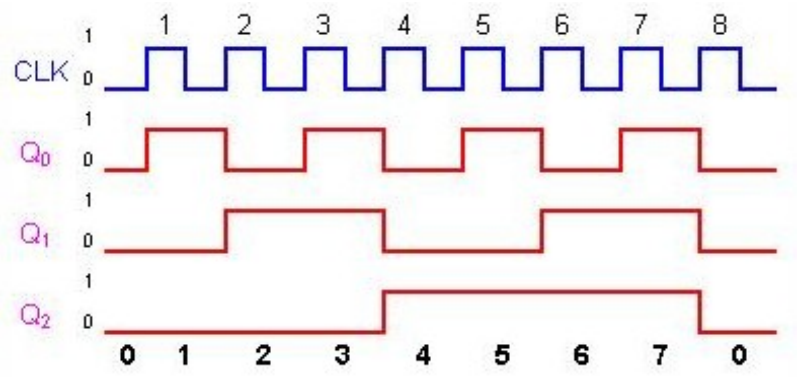
liczba	binarnie
0	000
1	001
2	010
3	011
4	100
5	101
6	110
7	111

Spróbujcie znaleźć pewną prawidłowość zmian każdego bitu.

Licznik

Prawidłowość, o której wspomniałem, wygląda tak:

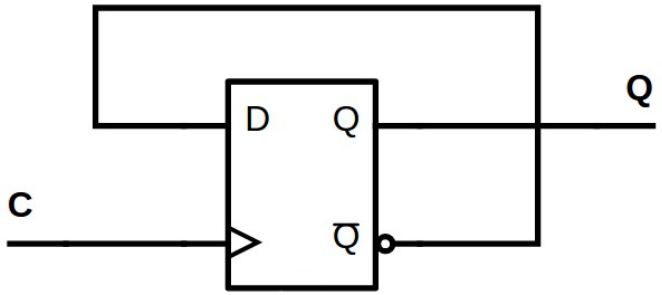
- najmłodszy bit zmienia stan na przeciwny za każdym razem,
- kolejny bit zmienia się co 2 razy,
- najstarszy bit zmienia się co 4 razy.



Przerzutnik przełączający

Przydałby nam się element pamiętający stan, ale przy okazji zmieniający ten stan na przeciwny.

Zastanówmy się, czy da się go zrobić z przerzutnika D.



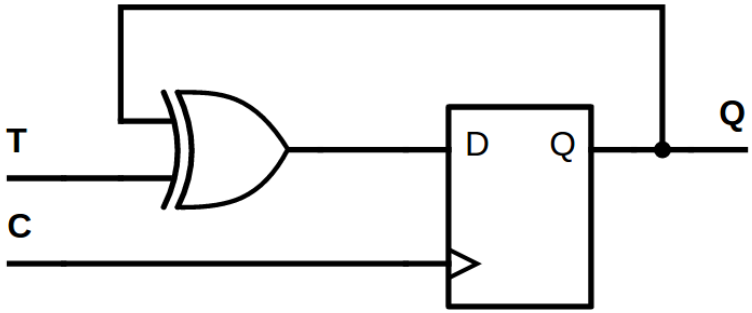
Jeżeli na wejście D podamy sygnał z wyjścia $\overline{Q}$ (zanegowane Q), stan wyjścia Q będzie się zmieniał na przeciwny za każdym zboczem na wejściu zegarowym C.

Przerzutnik T

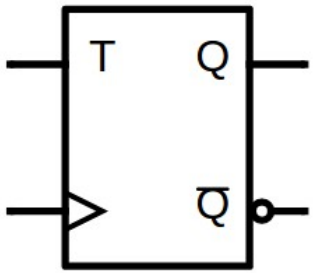
Aby móc sterować przełączaniem stanu przerzutnika, wprowadzimy bramkę XOR, która działa jak sterowana negacja.

Jeżeli wejście T = 1, na wyjściu bramki pojawi się zanegowana wartość Q.

Jeżeli T = 0, na wyjściu bramki jest stan Q.



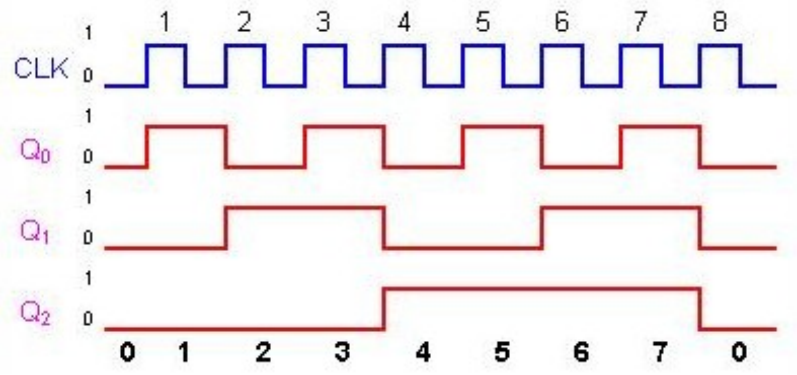
Tak więc wejście T (od „toggle”) służy do sterowania przełączaniem.
Przerzutnik działający w ten sposób jest nazywany **przerzutnikiem T**.



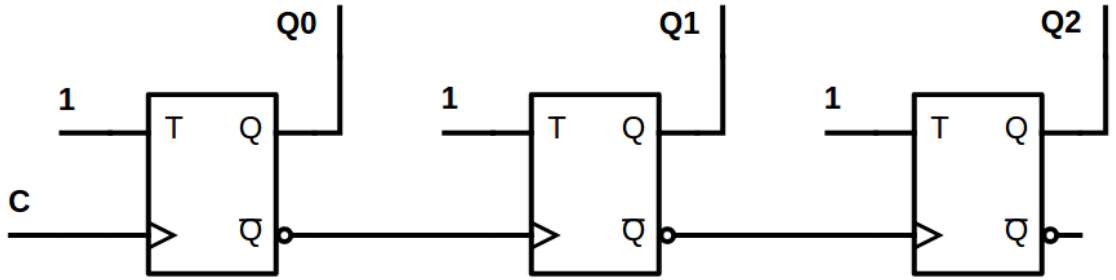
Licznika ciąg dalszy

Mamy już element, z którego możemy zbudować licznik.

Jeszcze raz rzućmy okiem na przebiegi wyjść licznika:



Wygląda na to, że wejście zegarowe kolejnego przerzutnika musi być połączone z wyjściem poprzedniego, np. tak:

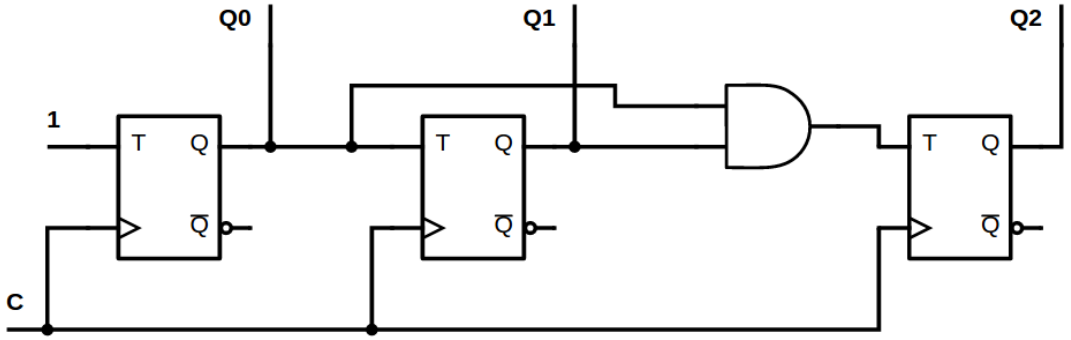


Jest to tzw. **licznik szeregowy** lub **asynchroniczny**.

Licznika ciąg dalszy

Wadą licznika szeregowego jest to, że sygnał zegarowy jest podawany z przerzutnika na przerzutnik, więc do ostatniego dociera z pewnym opóźnieniem. Przy wysokich częstotliwościach zliczanych impulsów może dochodzić do nieprawidłowości.

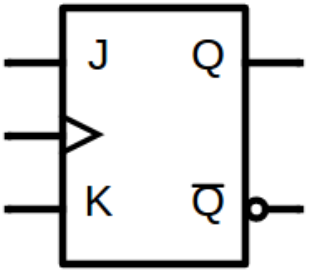
Alternatywą dla takich liczników są **liczniki równoległe** (**synchroniczne**), w których sygnał zegarowy jest podawany równocześnie na wszystkie przerzutniki:



Zwróćcie uwagę, że w tym typie liczników sposób liczenia jest wyznaczany przez odpowiednie wysterowanie wejść danych przerzutników (w tym przypadku T).

Przerzutnik JK

W praktycznych realizacjach wykorzystuje się bardziej uniwersalne przerzutniki JK:



Jeśli wejścia J i K są w stanie niskim, przerzutnik podtrzymuje poprzedni stan. Ustawienie wejścia J powoduje ustawienie wyjścia, natomiast ustawienie wejścia K powoduje wyzerowanie wyjścia przerzutnika.

Jeśli wejścia J i K są w stanie wysokim, przerzutnik zmieni stan wyjścia na przeciwny.

J	K	Q_{t+1}
0	0	Q_t
0	1	**0**
1	0	**1**
1	1	$\overline{Q}_t$

Zadania

Zróbcie **jedno, dwa lub trzy zadania**. Wszystkie układy mają być przetestowane w logic.ly.

1. Narysuj schemat 4-bitowego rejestru szeregowo-równoległego.

2. Zaprojektuj asynchroniczny 3-bitowy licznik liczący w dół.

3. Zaprojektuj synchroniczny 3-bitowy licznik liczący w dół.