Technika Cyfrowa - sprawozdanie 2

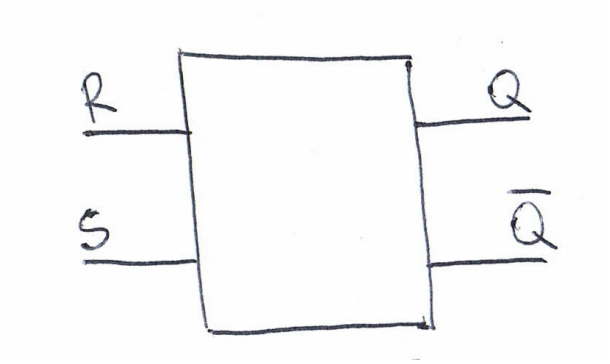
Łukasz Jezapkowicz

30.04.2020

1 Na podstawie dostępnych tabel prawdy, zaprojektować i praktycznie zrealizować asynchroniczny przerzutnik "RS" w oparciu o tylko bramki NOR, po czym przetestować poprawność jego działania w programie Multisim.

1.1 Po co przerzutniki?

W wielu układach cyfrowych niewystarczające są dla nas układy uzależnione jedynie od stanów panujących na jego wejściach. Chcielibyśmy, żeby stan wyjścia układu zależał w pewien sposób od tego co wcześniej działo się w owym układzie. Prostą implementacją takiego pomysłu jest przerzutnik. Przerzutnik to krótko mówiąc układ cyfrowy wyposażony w swego rodzaju pamięć. Przerzutnik zapamiętuje swój stan wewnętrzny, który może się zmieniać pod wpływem odpowiednich stanów na wejściu. W tym rozdziale przyjrzę się bliżej przerzutnikowi RS



Rysunek 1: Przerzutnik RS

1.2 Co to przerzutnik RS?

Żeby pokusić się o zaprojektowanie oraz stworzenie przerzutnika RS potrzebna jest wiedza co właściwie taki przerzutnik powinien robić. Przerzutnik RS to jeden z najprostszych rodzajów przerzutnika. Naszym głównym celem jest zaprojektowanie układu, który będzie posiadał dwa stany wejściowe oraz dwa wyjścia komplementarne - jedno wyjście jest zaprzeczeniem drugiego. Nazwijmy te stany Q oraz $\bar{Q}$ zaś stany wejściowe R oraz S. Stanem aktywnym stanu wejściowego nazywamy sytuację, gdy jest on w stanie logicznym 1, w przeciwnym razie mówimy, że jest on w stanie neutralnym. Chcemy, żeby stan Q był równy 1 jeżeli stan S jest w stanie aktywnym zaś 0 jeżeli stan R jest w stanie aktywnym. W przypadku, gdy obydwa stany są w stanie neutralnym bierzemy stan wewnętrzny przerzutnika Q_{n-1} oraz $\bar{Q}_{n-1}$. Obydwa stany wejściowe w stanie aktywnym to tak zwany stan niedozwolony, w którym nie da się przewidzieć wyjścia układu. Powyższe rozważania można przedstawić w postaci tabeli prawdy:

R	S	Q	$\bar{Q}$
0	1	1	0
1	0	0	1
0	0	Q_{n-1}	$\bar{Q}_{n-1}$
1	1	0	0

Tabela prawdy przerzutnika RS

1.3 Projektowanie przerzutnika RS zbudowanego z bramek NOR

Bramka NOR

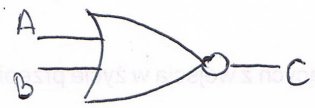
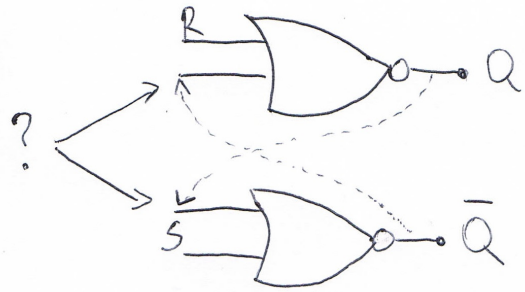


Tabela Prawdy:

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

Stan Q musi ostatecznie wychodzić z pewnej bramki NOR.
Widać, że Q będzie miało stan logiczny 1 jeżeli są dwa stany
niskie na wejściu. Patrząc na tabelę prawdy jednym z nich może być R.

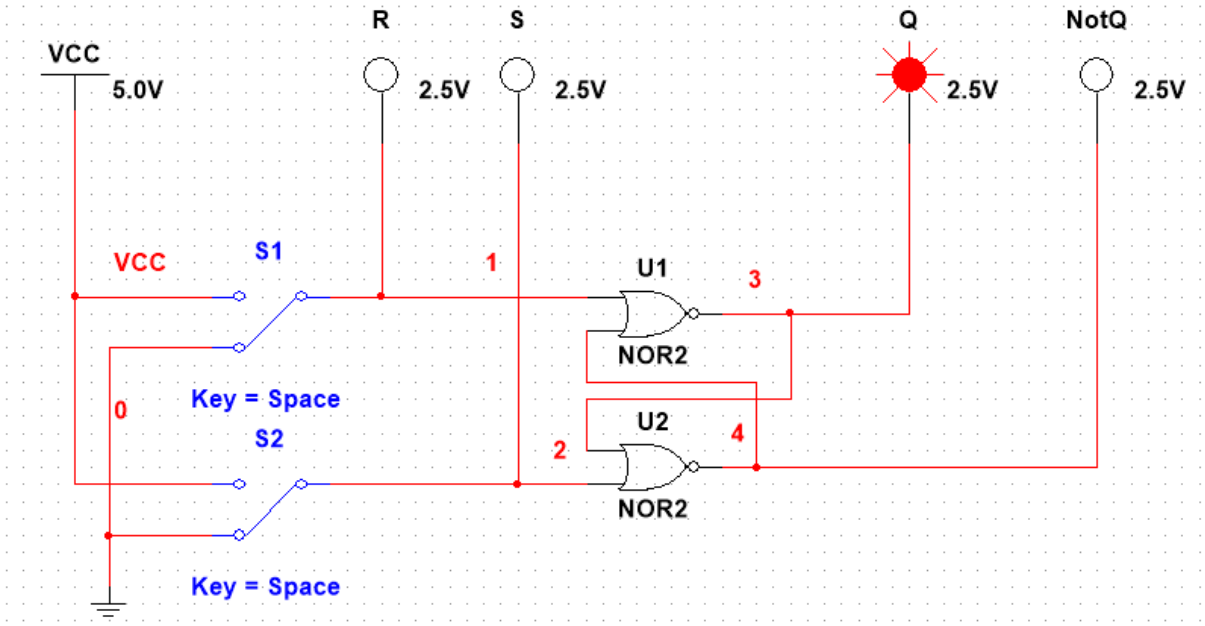


Można zauważyć, że by Q było w stanie 1 brakuje nam
sygnału wejściowego 0. Ponieważ nie możemy negować
R i S jedynym źródłem stanu 0 może być $\bar{Q}$.

Proces szukania odpowiedniego układu cyfrowego

1.4 Budowa schematu w Multisimie

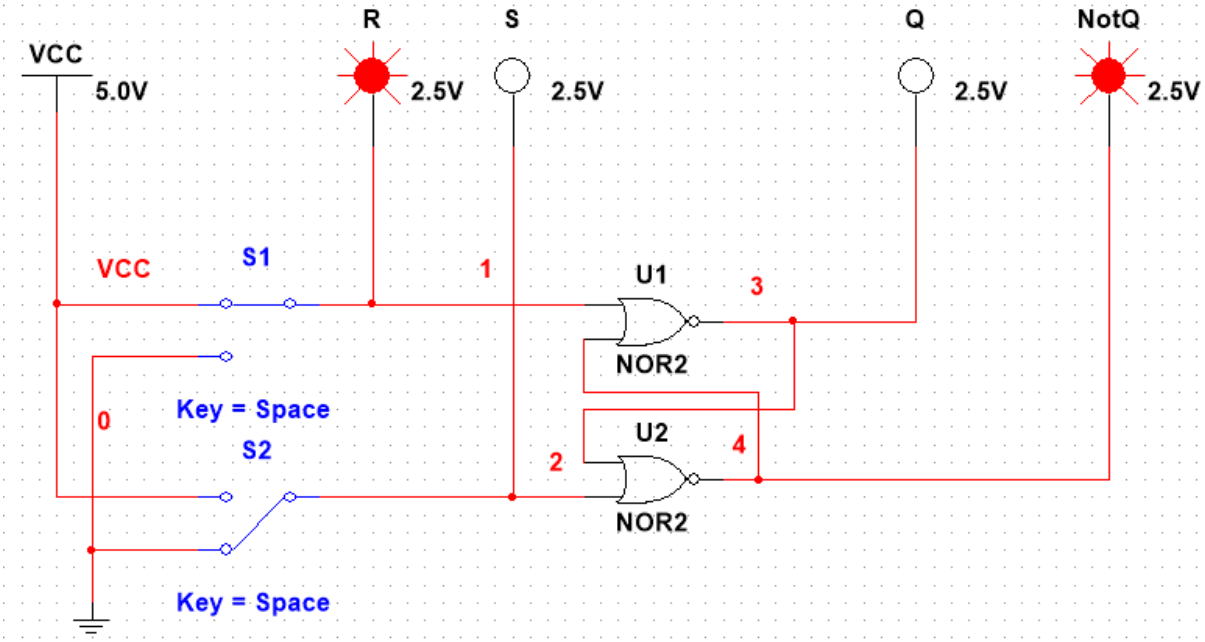
Ze względu na małą ilość stanów oraz przejrzystość wskaźników postanowiłem nie korzystać w tym zadaniu z Word Generator'a oraz Logic Analyzer'a.



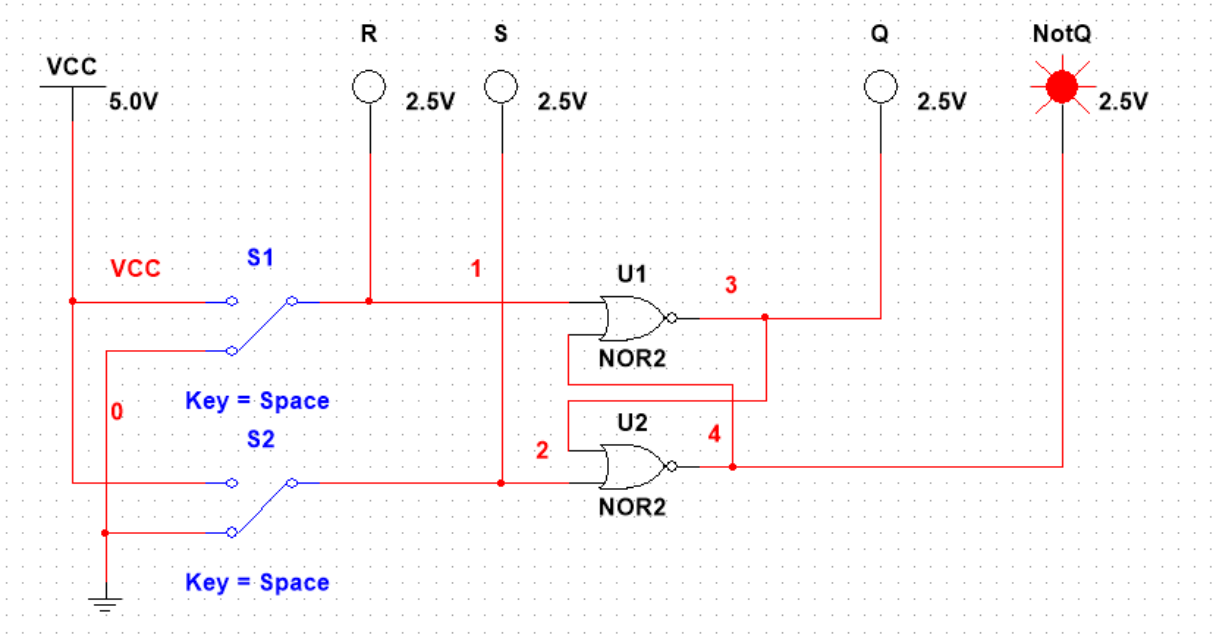
Schemat układu przerzutnika RS w Multisimie - stan 1

1.5 Testowanie układu

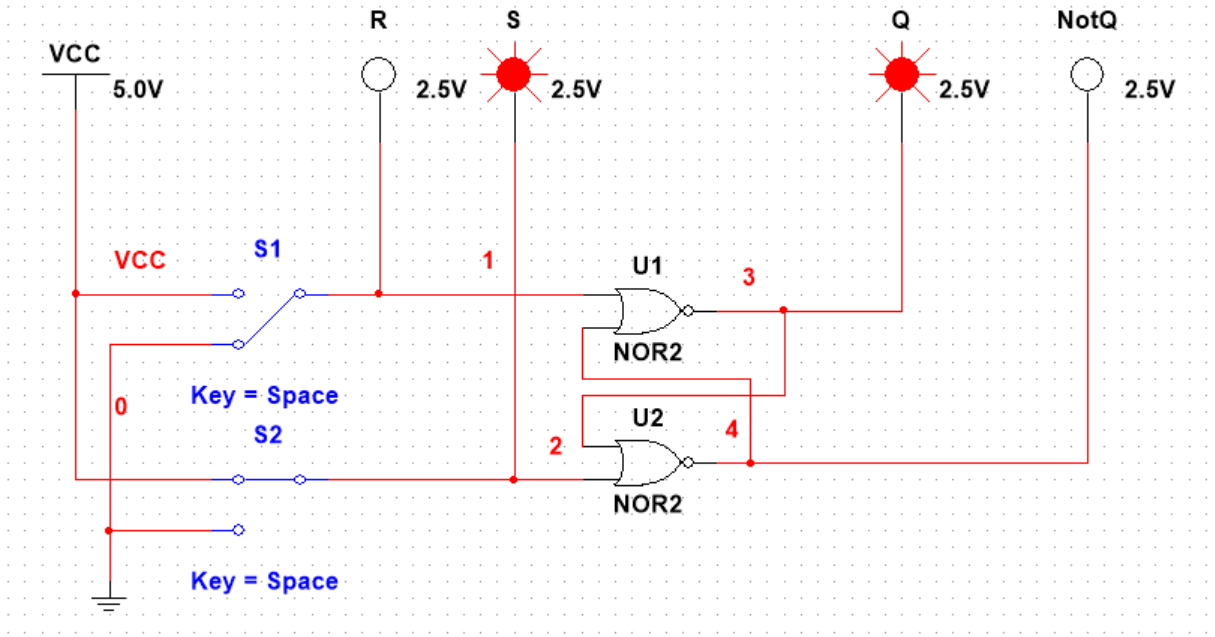
Warto zaznaczyć, że kolejne obrazki to kolejne stany układu.



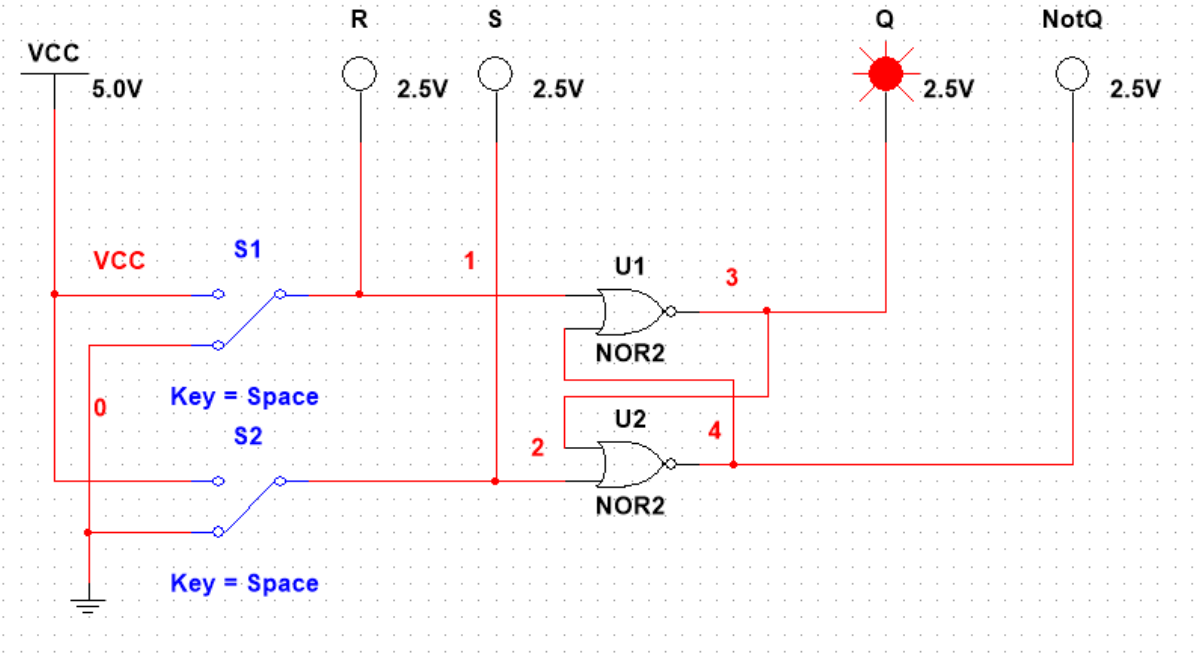
Stan 2


R
S
Q
NotQ
VCC
5.0V
2.5V
2.5V
2.5V
2.5V
VCC
S1
1
U1
3
NOR2
0
Key = Space
S2
U2
2
4
NOR2
Key = Space

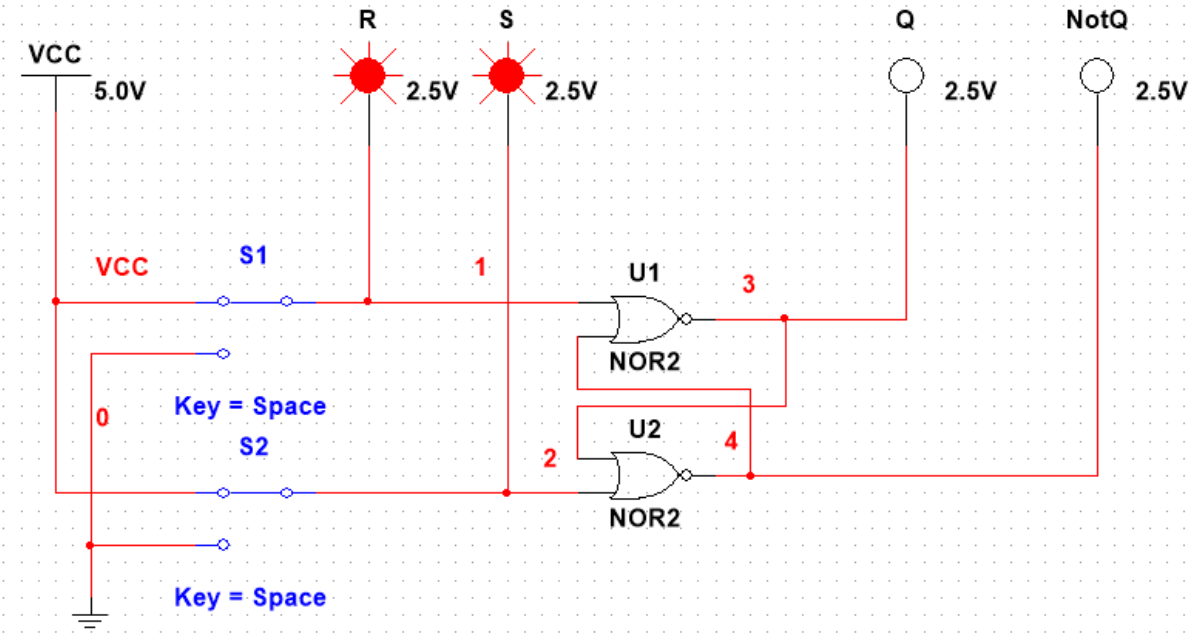
Stan 3


R
S
Q
NotQ
VCC
5.0V
2.5V
2.5V
2.5V
2.5V
VCC
S1
1
U1
3
NOR2
0
Key = Space
S2
U2
2
4
NOR2
Key = Space

Stan 4


R
S
Q
NotQ
VCC
5.0V
2.5V
2.5V
2.5V
2.5V
VCC
S1
1
U1
3
NOR2
0
Key = Space
S2
U2
2
4
NOR2
Key = Space

Stan 5


R
S
Q
NotQ
VCC
5.0V
2.5V
2.5V
2.5V
2.5V
VCC
S1
1
U1
3
NOR2
0
Key = Space
S2
U2
2
4
NOR2
Key = Space

Stan 6

1.6 Wnioski

1.6.1 Czy układ działa poprawnie?

Przedstawione przeze mnie rozumowanie pozwoliło mi dojść do poprawnego wyniku końcowego, którym jest poprawnie zbudowany układ asynchronicznego przerzutnika RS wykorzystujący bramki NOR. Przedstawione powyżej zdjęcia układów pokazują każdą możliwą kombinacje wartości R oraz S oraz to jak zmieniają się między nimi stany wyjściowe. Wartości stanów wyjściowych są zgodne z tym co założyłem na początku, układ jest zatem prawidłowy.

1.6.2 Co można było zrobić inaczej?

- zbudować równoważny układ przy pomocy bramek $NAND$, które zajmują fizycznie mniej miejsca. Cały układ zajmowałby więc mniej miejsca.
- zastąpić switche $S1$ oraz $S2$ przez bardziej pewne rozwiązania takie jak Word Generator.
- zastąpić wskaźniki Q oraz $NotQ$ przez inne rozwiązania takie jak Logic Analyzer
- zastąpić układ jego synchronicznym odpowiednikiem, który radzi sobie z problemem hazardu.

1.6.3 Gdzie to można zastosować?

- budowanie innych przerzutników takich jak przerzutniki typu T czy D.
- dzielniki częstotliwości.
- budowanie rejestrów przesuwających, liczników, układów sekwencyjnych.
- eliminacja zmian sygnału/odbić np. przy naciskaniu klawisza .

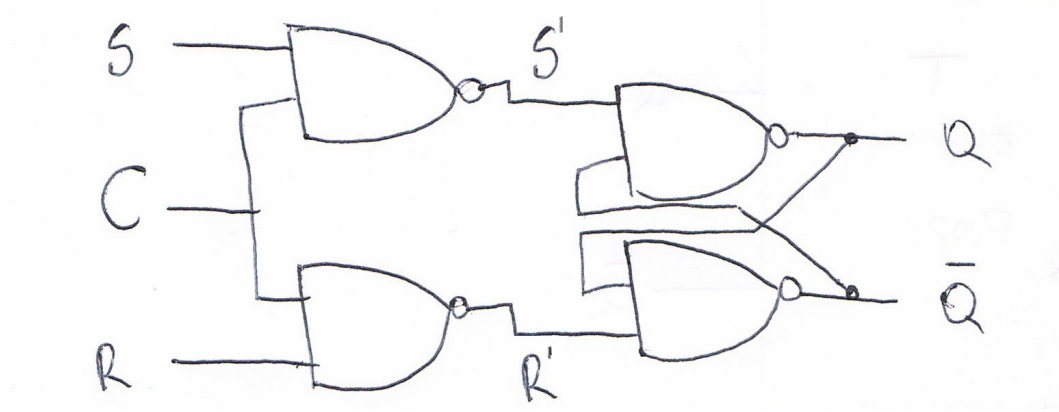
2 Bazując na synchronicznym przerzutniku "RS" i ewentualnie bramkach logicznych, zaprojektować, zbudować i przetestować przerzutnik typu "T".

2.1 Różnica pomiędzy asynchronicznym a synchronicznym przerzutnikiem RS

Synchroniczny przerzutnik RS różni się od wersji asynchronicznej tym, że zmiana stanu wyjść następuje synchronicznie z sygnałem zegarowym C. Pomaga to zlikwidować zakłócenia oraz synchronizować poszczególne elementy układu logicznego. Gdy na wejściu zegarowym pojawi się narastające zbocze sygnałowe to przerzutnik działa jak jego asynchroniczny odpowiednik. W przeciwnym razie stan przerzutnika się nie zmieni.

Wejście					Wyjście		Stan wyjścia
C	S	R	S'	R'	Q_{n+1}	$\overline{Q_{n+1}}$	
0	X	X	1	1	Q_n	$\overline{Q_n}$	Bez zmian
1	0	0	1	1			
1	0	1	1	0	0	1	Reset
1	1	0	0	1	1	0	Set
1	1	1	0	0	1	1	Stan niedozwolony

Tabela prawdy synchronicznego przerzutnika RS



Schemat układu logicznego synchronicznego przerzutnika RS zbudowanego z bramek NAND

2.2 Co to przerzutnik T?

Przerzutnik typu T to prosty rodzaj przerzutnika, który po podaniu wartości logicznej 0 (stan neutralny) na wejście zachowuje bieżący stan zaś po podaniu wartości logicznej 1 zmienia stan wyjść na przeciwny względem wcześniejszego stanu. W celu osiągnięcia takiego rezultatu używa on również wyzwalania zboczem sygnału zegarowego C. Przerzutnik ten można opisać tabelą prawdy oraz schematem.

T	Q_{n-1}	Q_n
0	0	0
0	1	1
1	0	1
1	1	0

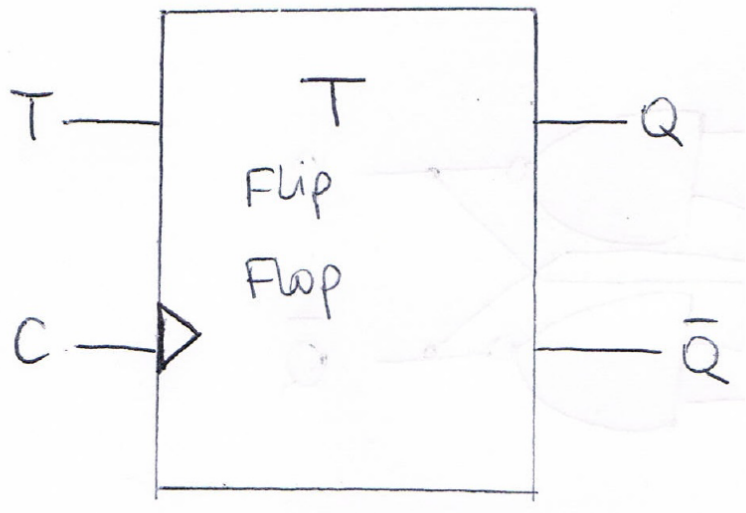
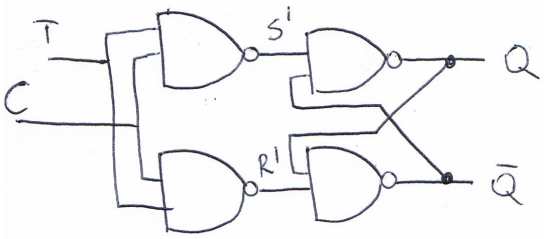


Tabela prawdy przerzutnika typu T oraz schemat układu zbudowanego z bramek NAND

2.3 Projektowanie przerzutnika typu T zbudowanego z synchronicznego przerzutnika RS

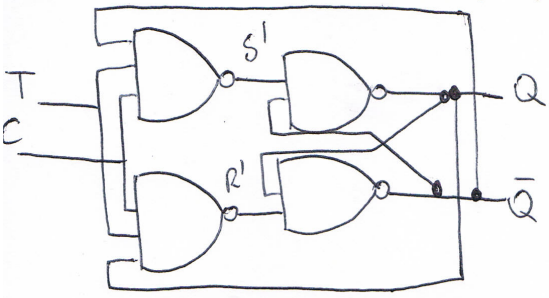


Schemat przerzutnika RS, gdy zastąpimy sygnały R i S przez T.

C	T	S'	R'	Q_{n+1}	$\overline{Q_{n+1}}$
0	X	1	1	Q_n	$\overline{Q_n}$
1	0	1	1		
1	1	0	0	1	1

Jak widać nie jest to wystarczający zabieg.

Dla C i T równego 1 S' oraz R' nie mogę wymóc 0 w tym samym momencie. By jedno z nich było różne od drugiego należy dotrzeć sygnały na wejście (różne!) Można dotrzeć stan wewnętrzny (Q i $\bar{Q}$ różne ∇).
Żeby Q zmieniło się z 1 na 0 to S' nie może być równe 0 → Q dotrzeczony do R a $\bar{Q}$ do S.



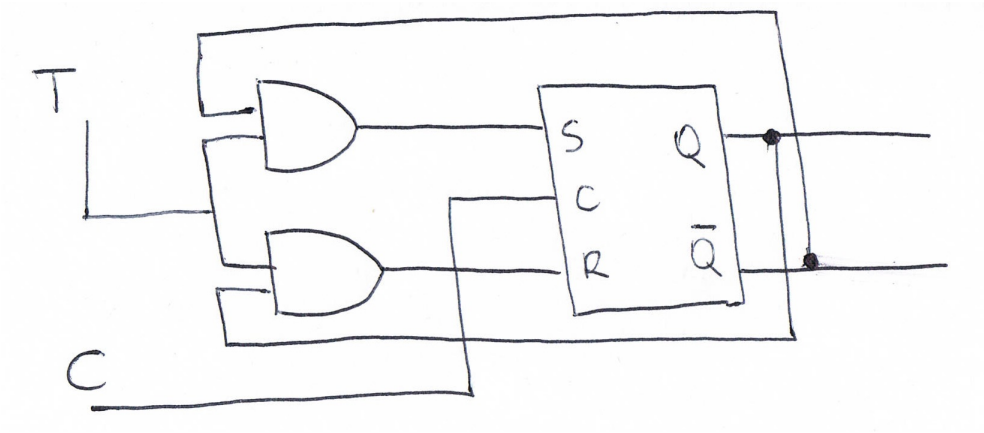
Sprawdzamy poprawność poprzez tabele prawdy

C	T	Q_n	$\overline{Q_n}$	S'	R'	Q_{n+1}	$\overline{Q_{n+1}}$	
0	X	X	X	1	1	Q_n	$\overline{Q_n}$	Bez zmian
1	0	X	X	1	1			
1	1	0	1	0	1	1	0	zmiana
1	1	1	0	1	0	0	1	zmiana

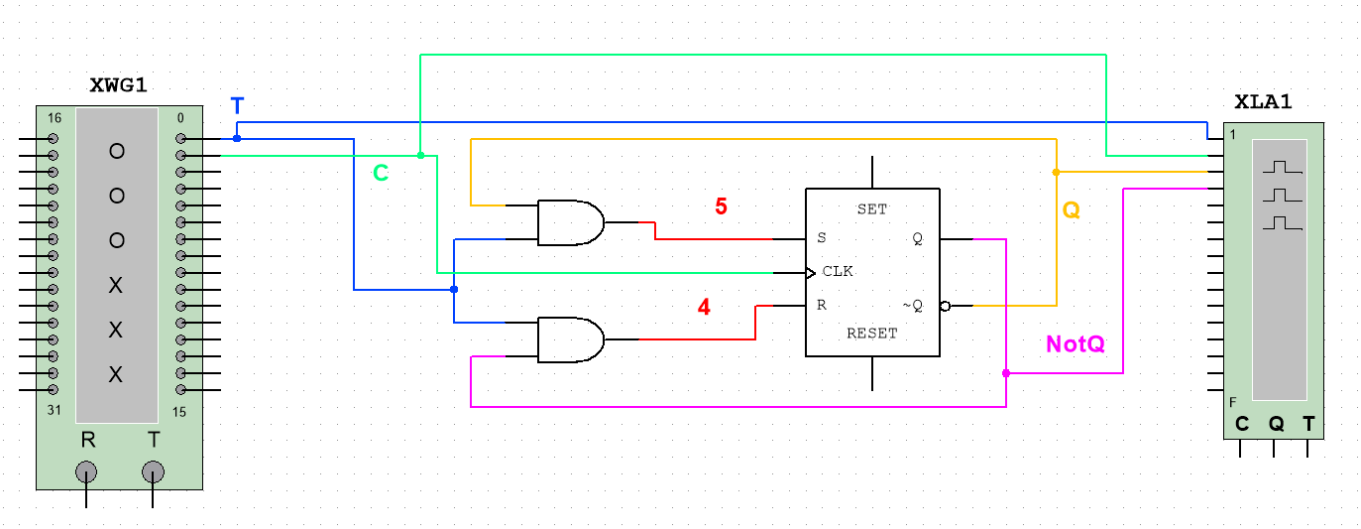
Jest OK ∇ :)

2.4 Uproszczenie układu

Zaprezentowany powyżej schemat pozwolił mi lepiej zrozumieć działanie przerzutnika typu T oraz łatwiej dojść do rozwiązania. Schemat ten można jednak uprościć używając gotowego synchronicznego przerzutnika RS oraz dwóch bramek AND. Układ widoczny poniżej:

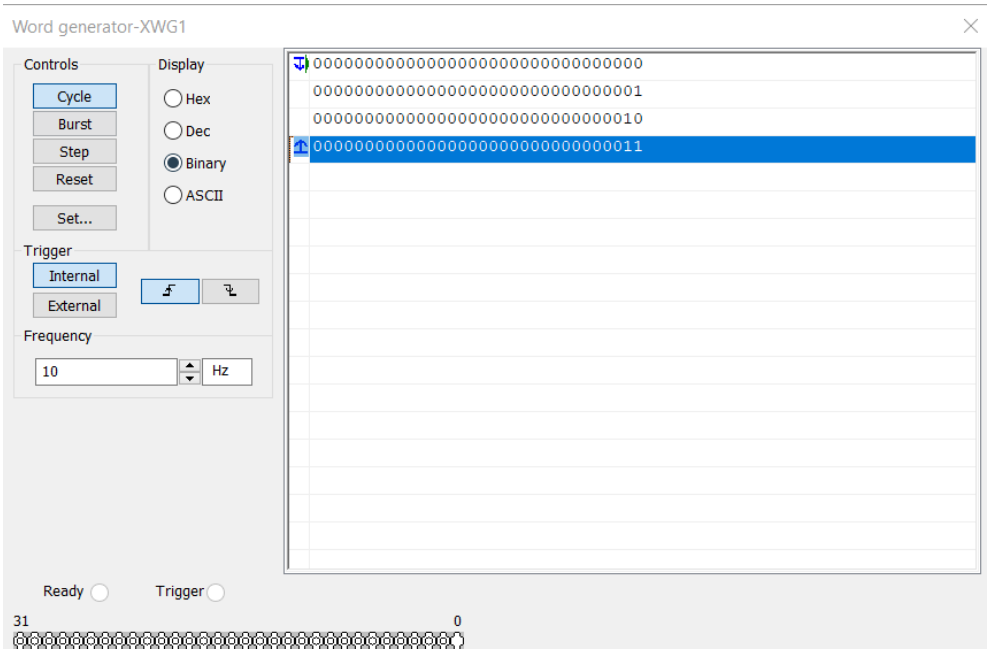


2.5 Budowa schematu w Multisimie

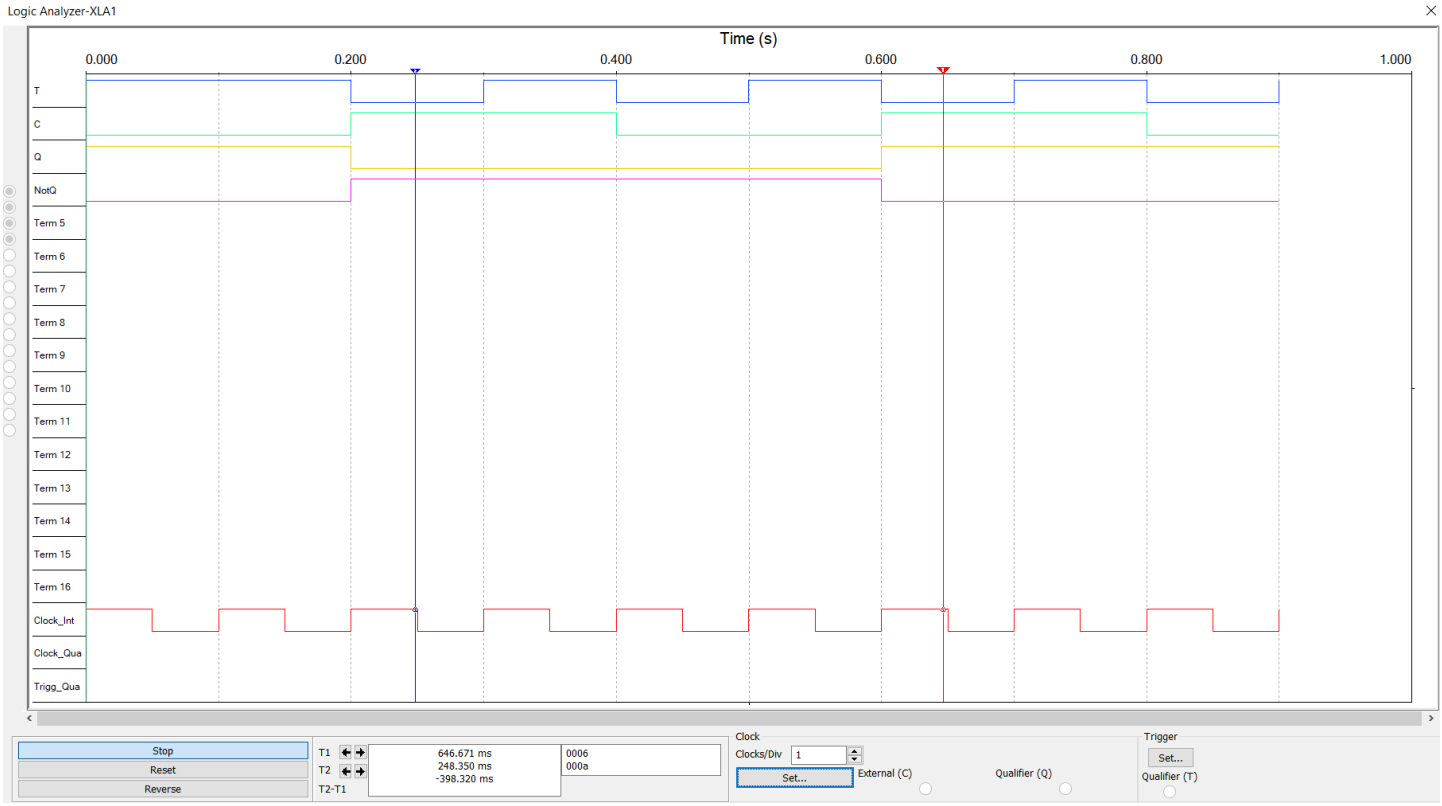


Przerzutnik typu T

2.6 Testowanie układu



Ustawienia Word Generator'a



Wyniki Logic Analyzer'a

Widać, że wyjście Q oraz $\bar{Q}$ zmieniają wartości logiczne dla narastającego zbocza sygnału zegarowego C (zielony kolor) oraz T o wartości logicznej 1. Układ działa zatem poprawnie.

2.7 Wnioski

2.7.1 Czy układ działa poprawnie?

Przedstawione przeze mnie rozumowanie pozwoliło mi dojść do poprawnego wyniku końcowego, którym jest poprawnie zbudowany układ przerzutnika typu T bazujący na synchronicznym przerzutniku RS. Przedstawione powyżej zdjęcie Logic Analyzer'a pokazuje, że przerzutnik przełącza się tylko dla wartości logicznej T równej 1 oraz przy narastającym zboczu. Wartości stanów wyjściowych są zgodne z tym co założyłem na początku, układ jest zatem prawidłowy.

2.7.2 Co można było zrobić inaczej?

- zastąpić układ synchronicznego przerzutnika RS układem zbudowanym z bramek NOR oraz AND.
- zbudować układ przerzutnika T nie używając gotowego przerzutnika RS.
- porównać wyniki Logic Analyzer'a z wynikami dla gotowego przerzutnika typu T.

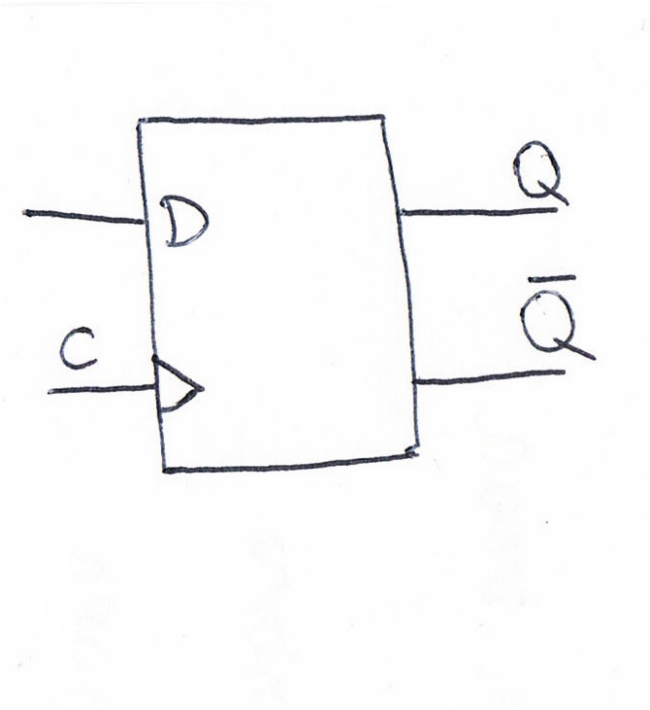
2.7.3 Gdzie to można zastosować?

- układ dzielenia częstotliwości przez 2.
- budowa liczników binarnych, synchronicznych.
- budowa rejestrów przesuwających.
- budowa sumatorów binarnych.

3 Korzystając z wybranych przerzutników zbudować odpowiednie rejestry oraz na ich podstawie zaproponować, zbudować i przetestować własne układy szeregowego nadajnika i odbiornika dowolnej czterobitowej liczby binarnej. Następnie w programie Multisim proszę przetestować i pokazać działanie transmisji danych pomiędzy zbudowanym nadajnikiem i odbiornikiem.

3.1 Przerzutnik typu D

W celu zbudowania nadajnika oraz odbiornika liczb czterobitowych trzeba zbudować odpowiednie rejestry zawierające odpowiednie przerzutniki. W naszym przypadku potrzebny jest przerzutnik, który na wejściu otrzymuje pewien bit i przepisuje go na wyjście Q. Ważne jest również, żeby przerzutnik ten był wyzwalany zboczem sygnału zegarowego C. Okazuje się, że idealnym kandydatem dla tego opisu jest przerzutnik typu D, który spełnia opisane powyżej warunki. Poniżej widoczna tabela prawdy oraz schemat przerzutnika typu D.



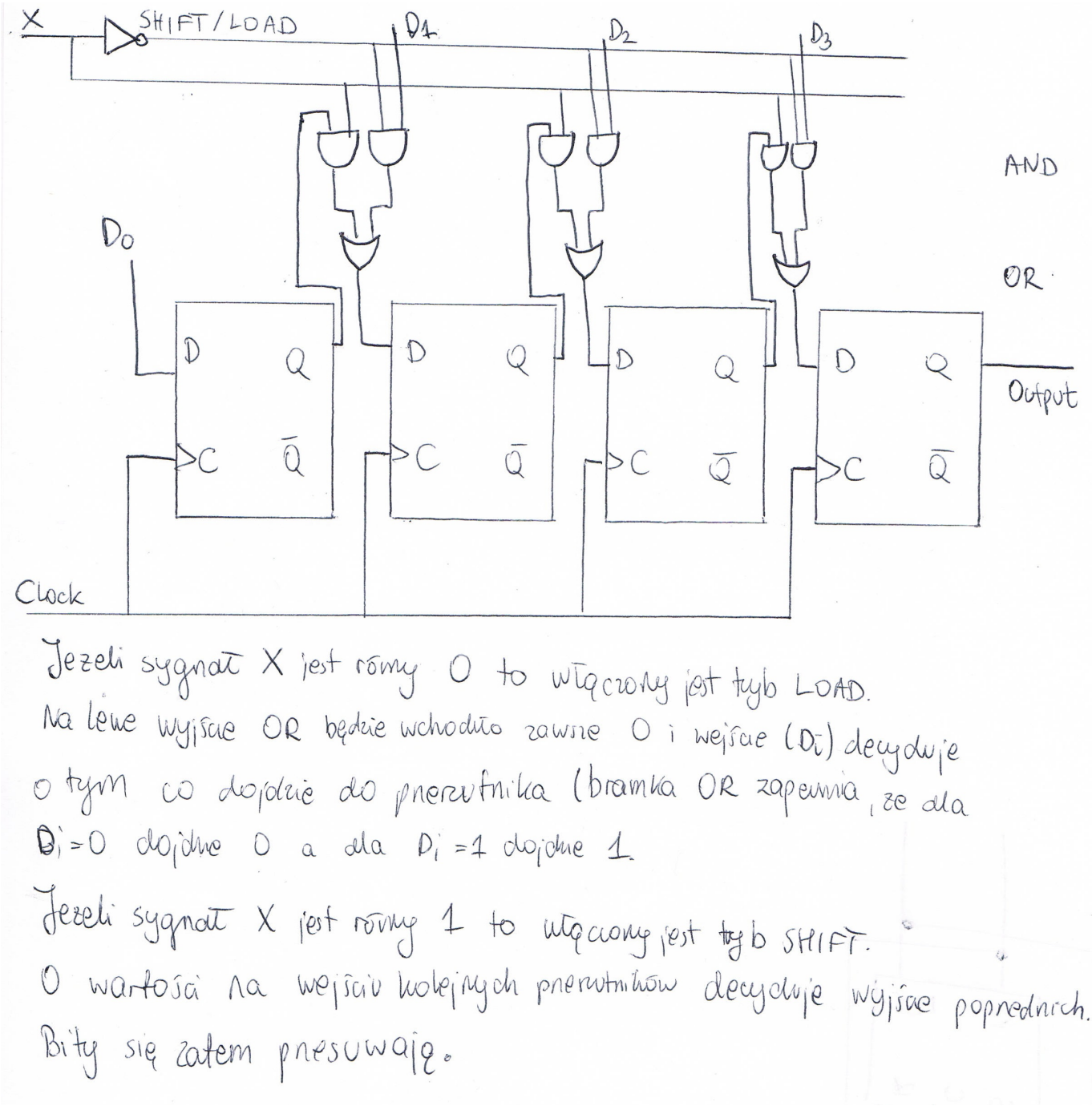
D	Q_n	Q_{n+1}
0	0	0
0	1	0
1	0	1
1	1	1

Schemat oraz tabela prawdy przerzutnika typu D

3.2 Rejestr przesuwający

W celu zapisania liczby czterobitowych musimy zaprojektować rejestr, który będzie mógł przechowywać informację o wartości logicznej tych czterech bitów. W naszym układzie użyjemy rejestru przesuwającego, który zbudowany jest z przerzutników połączonych w taki sposób, iż każdy impuls

zegarowy (wyzwalanie zboczem) przemieszcza informacje o bitach do kolejnych przerzutników. W naszym przypadku takimi przerzutnikami będą zaprezentowane wcześniej przerzutniki typu D. Musimy jeszcze zdecydować w jaki sposób chcemy wprowadzać i wyprowadzać dane. W naszym przypadku dane będziemy wprowadzać równolegle a wyprowadzać szeregowo. Jest to zatem rejestr $PISO$. Musimy jeszcze w jakiś sposób wprowadzić możliwość decydowania czy przerzutniki mają odczytywać wejście czy przesuwać bity. Posłużymy się w tym celu odpowiednimi bramkami logicznymi. Dla czterech bitów schemat naszego rejestru wygląda następująco:

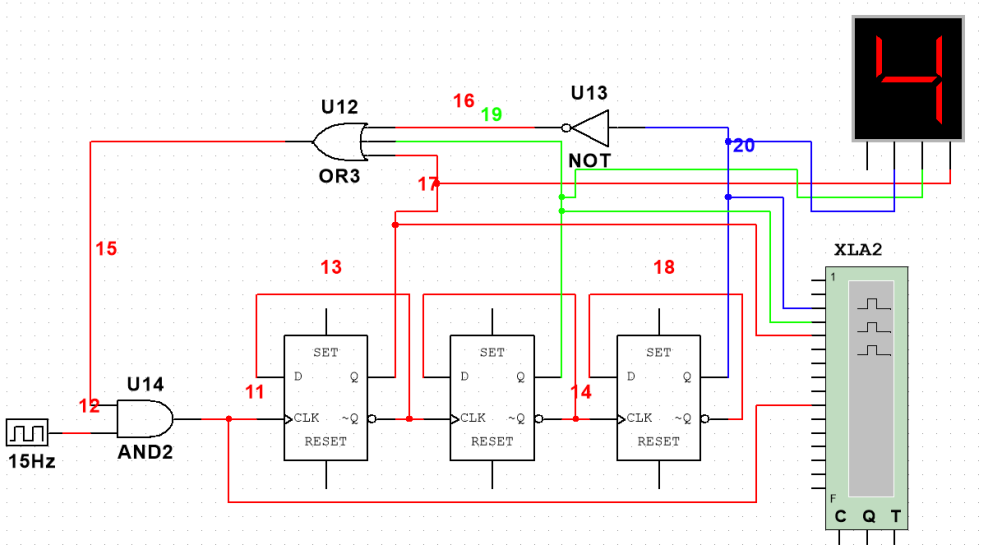


Schemat nadajnika dla 4 bitów, dane wychodzą z układu szeregowo

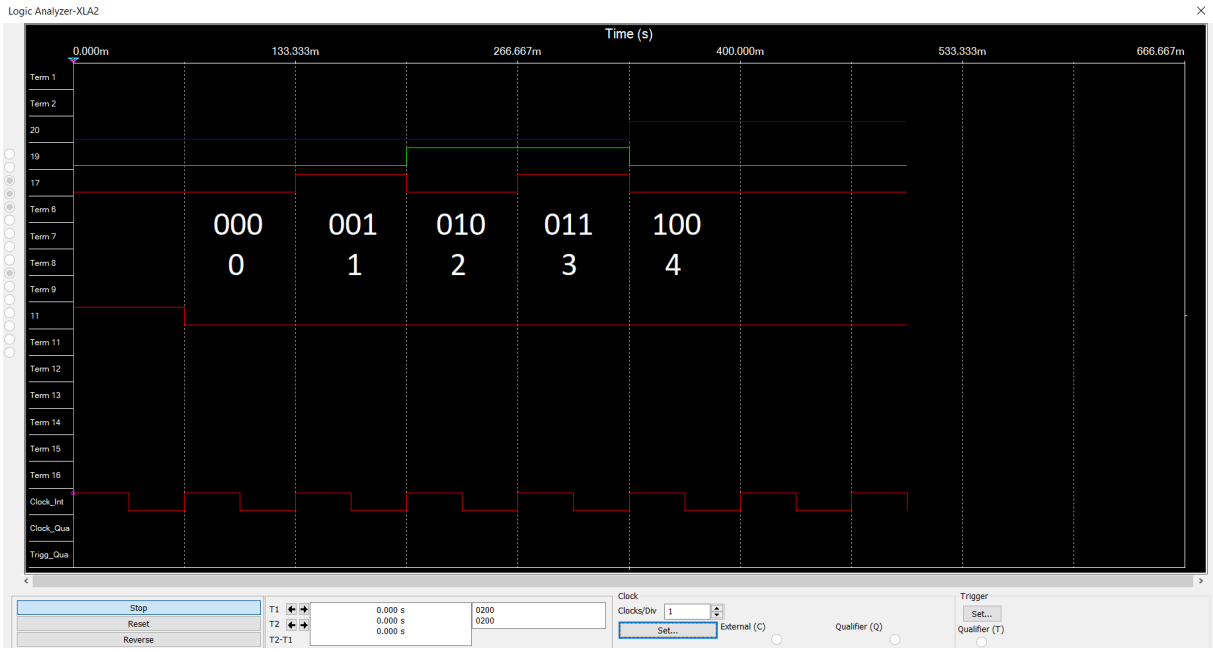
Wszystkie przerzutniki operowane są tym samym sygnałem zegarowym C.

3.3 Zatrzymujący się licznik modulo 5

W celu zapewnienia jedynie 4 taktów zegara w naszym nadajniku posłużyłem się licznikiem modulo 5. Problem budowy licznika modulo 5 został przeze mnie omówiony w sprawozdaniu 3 w zadaniu nr 4. Poniższy asynchroniczny licznik zawiera trzy dwójki liczące zbudowane z przerzutników typu D. Układ blokuje się w momencie, gdy licznik dojdzie do liczby 4 (w zapisie binarnym 100) poprzez odpowiednie połączenie bramek NOT,$OR3$ oraz $AND2$. Taka budowa sprawia, że przez bramkę $AND2$ przejdzie sygnał o wartości logicznej 1 dokładnie 4 razy. Wystarczy więc to miejsce w układzie potraktować jako źródło 4 impulsów, które potrzebujemy w naszym układzie nadajnika.

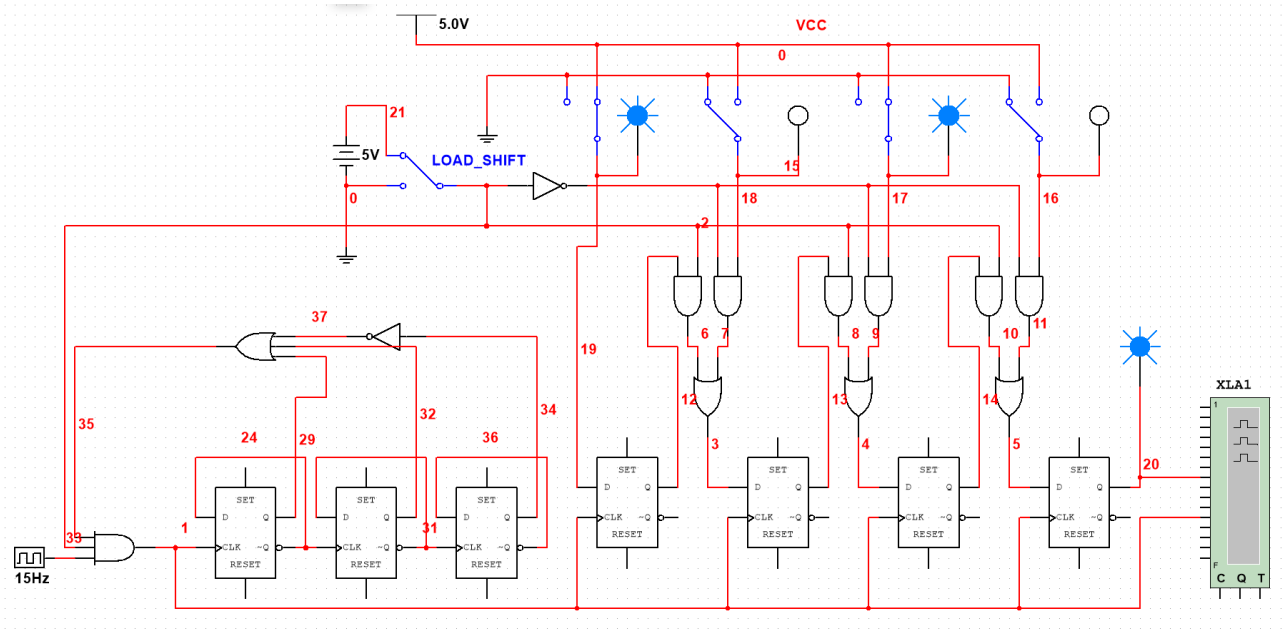


Zatrzymujący się licznik modulo 5



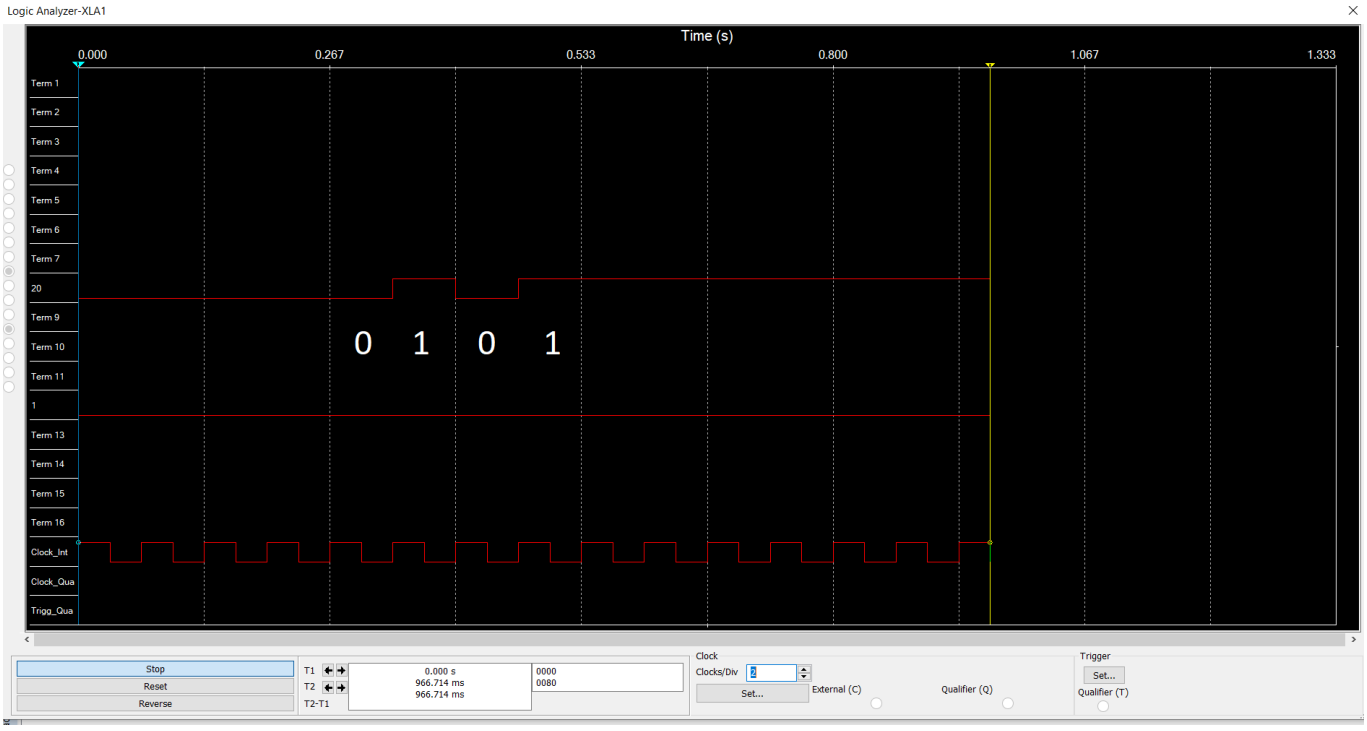
Jak widać, układ blokuje się, gdy otrzymuje wartość 4. Od tego momentu wartość na liczniku się nie zmienia.

3.4 Budowa nadajnika w Multisimie oraz przykład działania



Nadajnik liczb czterobitowych

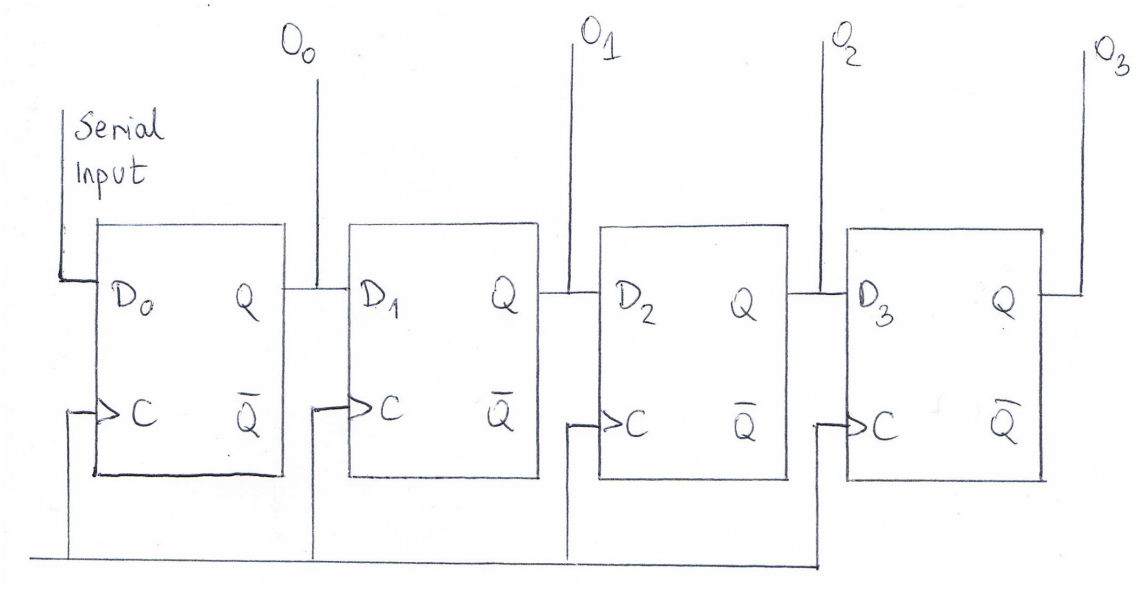
W naszym liczniku zmieniłem bramkę $AND2$ na bramkę $AND3$, dzięki której 4 impulsy zostaną wysłane dopiero w momencie włączenia opcji $SHIFT$ (czyli wartości logicznej 1 na przełączniku)



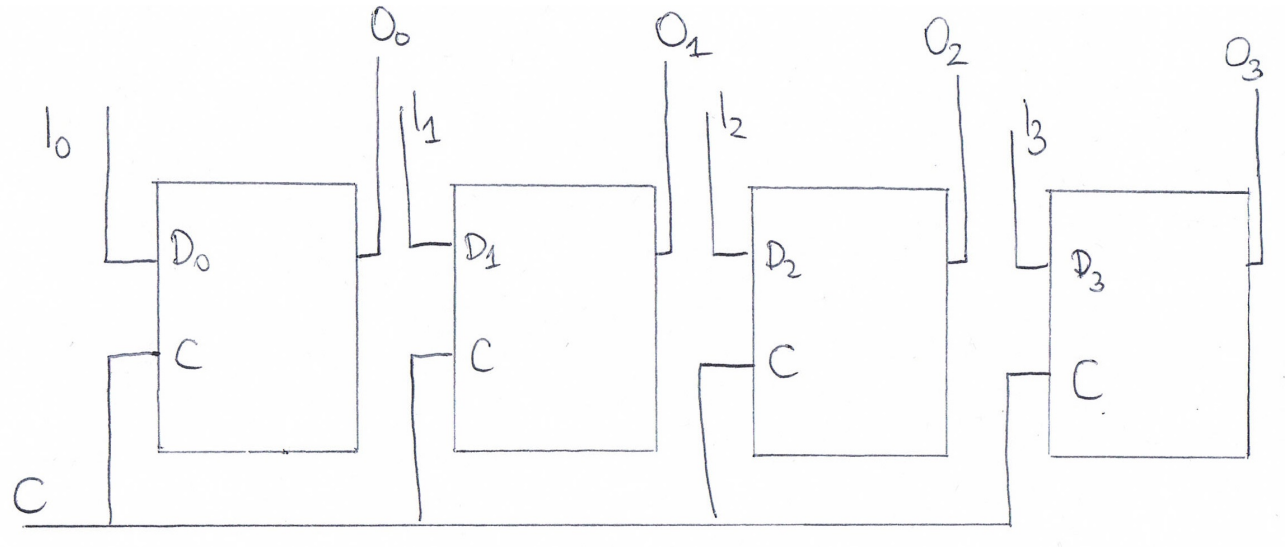
Przykład nadania kodu czterobitowego 1010. Bity wysyłane są szeregowo począwszy od bitu najmniej znaczącego. Po wysłaniu ostatniego znaku wartość jest zablokowana i stale równa 1.

3.5 Odbiornik liczb czterobitowych

Skoro mamy już nadajnik liczb czterobitowych to powinniśmy teraz stworzyć ich odbiornik. Będzie to również odpowiedni rejestr przesuwający, lecz tym razem dane będziemy wprowadzać szeregowo (ponieważ tak wyprowadzamy w nadajniku). Wyprowadzać dane będziemy równolegle. Jest to zatem rejestr $SIPO$. Nasz nadajnik połączymy również z rejestrem $PIPO$, którego zadaniem będzie "zatrzasnąć" wartości po skończonej transmisji. Taki zabieg sprawi, że na wyjściu nie będzie stanów nieustalonych w czasie nadawania. Poniżej schematy rejestrów $SIPO$ oraz $PIPO$.

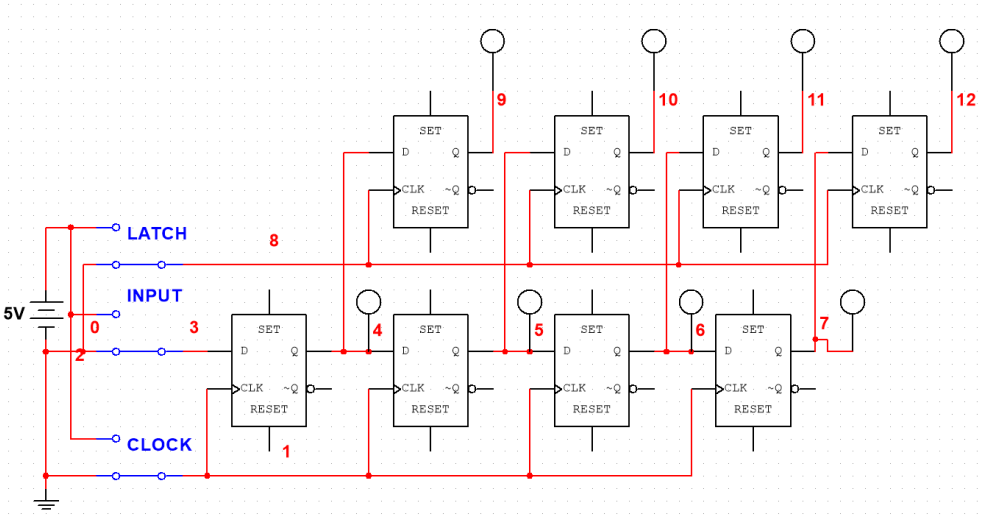


Rejestr $SIPO$. Dane $I_0I_1I_2I_3$ wprowadzane są szeregowo począwszy od bitu najmniej znaczącego. Każdy impuls zegara przesuwa nam bity o 1 pozycję w prawo. Po 4 impulsach nasza liczba przechowywana jest jako $O_0O_1O_2O_3$



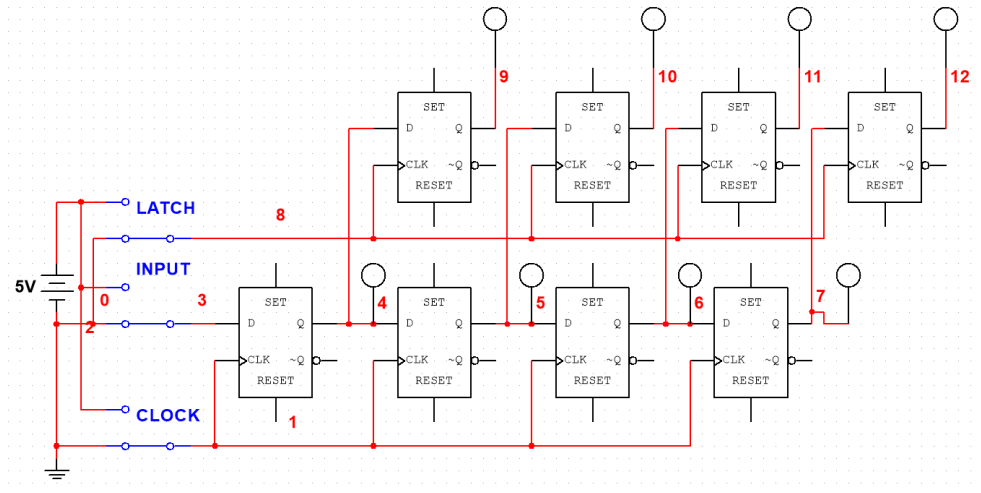
Rejestr $PIPO$. Dane $I_0I_1I_2I_3$ wprowadzane są równolegle do każdego przerzutnika. Kolejne wyjścia przerzutników tworzą nam liczbę wynikową $Q_0Q_1Q_2Q_3$. Rejestr jest synchroniczny, zasilany jest wspólnym sygnałem zegarowym. Wystarczy więc jeden impuls by odebrać liczbę.

3.6 Budowa odbiornika w Multisimie oraz przykład działania

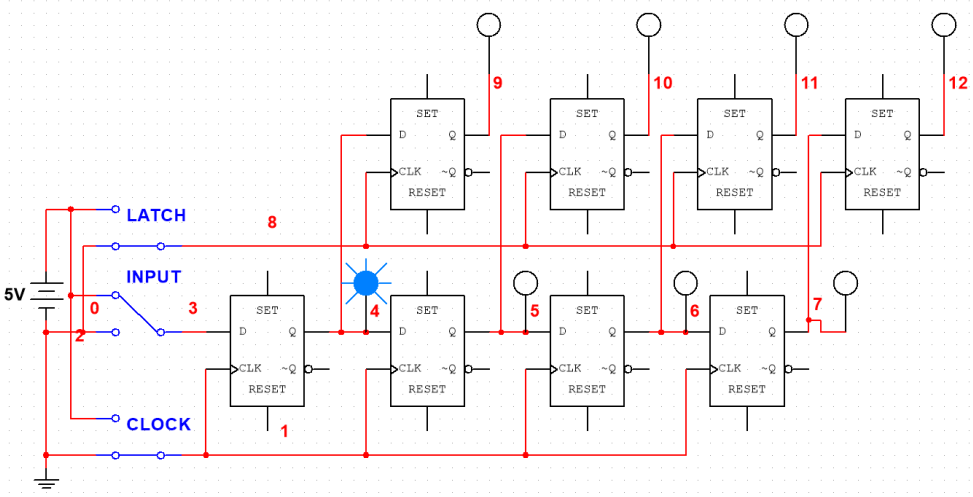


Odbiornik liczb czterobitowych (krok 1)

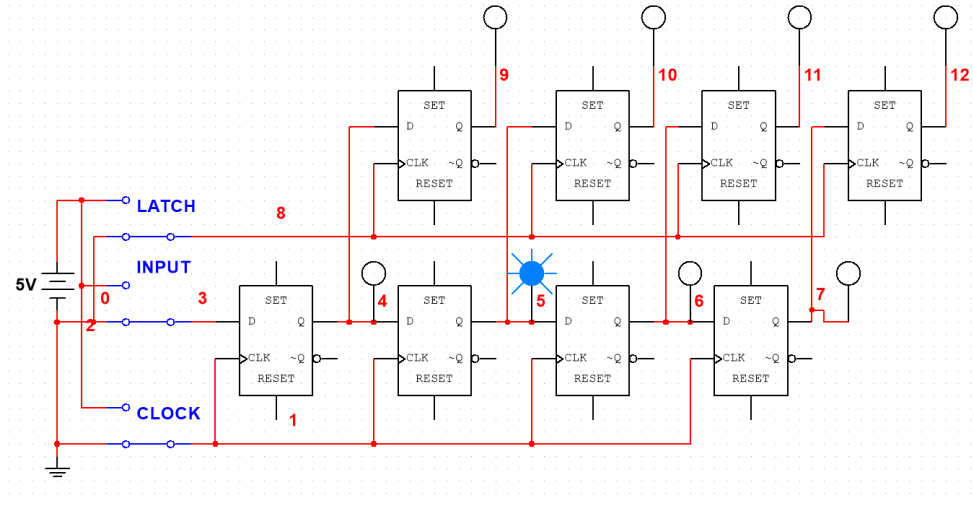
Nasz odbiornik ma charakter mocno dydaktyczny, ponieważ wszystko na nim działa na podstawie przełączników. Gdy połączymy go z odbiornikiem to pozbędziemy się tych przełączników. Załóżmy, że chcemy odkodować cztery bity 1010 (tak jak nadaliśmy). Poniżej krok po kroku:



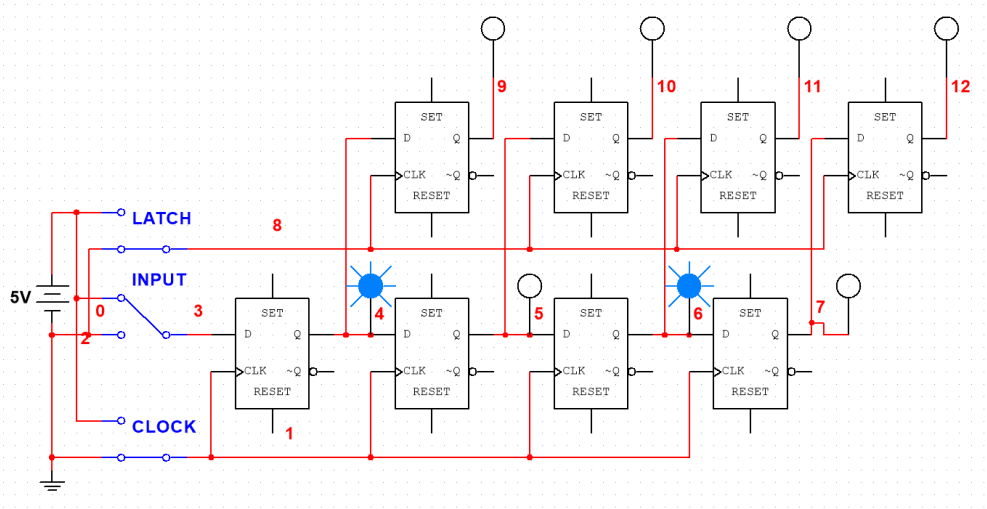
Odbiornik liczb czterobitowych (krok 2), po odebraniu bitu 0



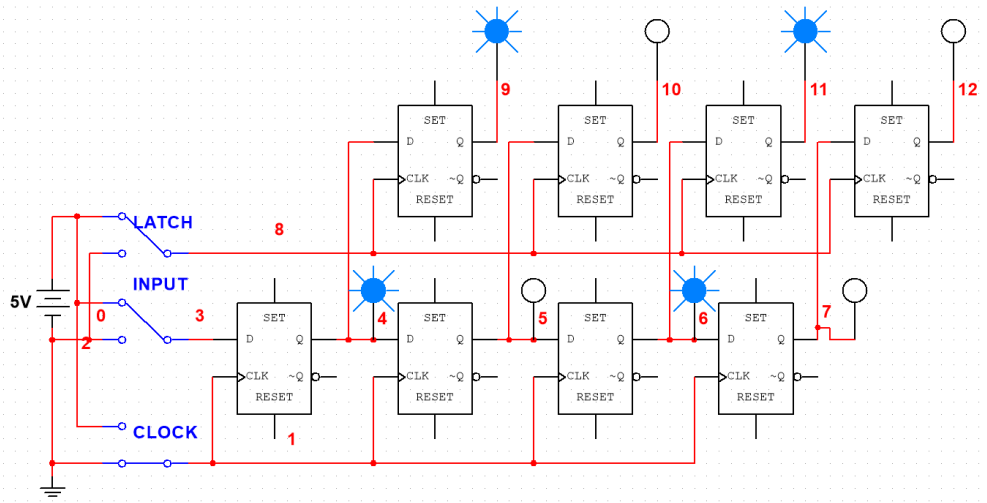
Odbiornik liczb czterobitowych (krok 3), po odebraniu bitu 1



Odbiornik liczb czterobitowych (krok 4), po odebraniu bitu 0



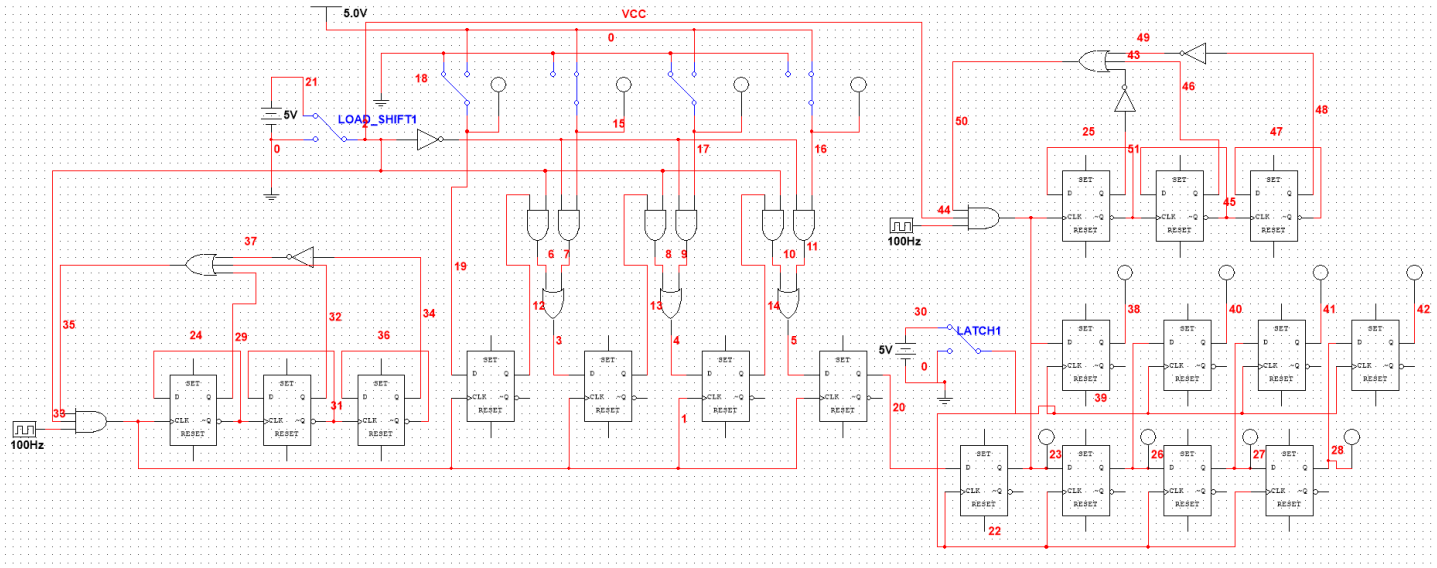
Odbiornik liczb czterobitowych (krok 5), po odebraniu bitu 1



Odbiornik liczb czterobitowych (krok 6), po zatrzaśnięciu wartości w rejestrze $PIPO$

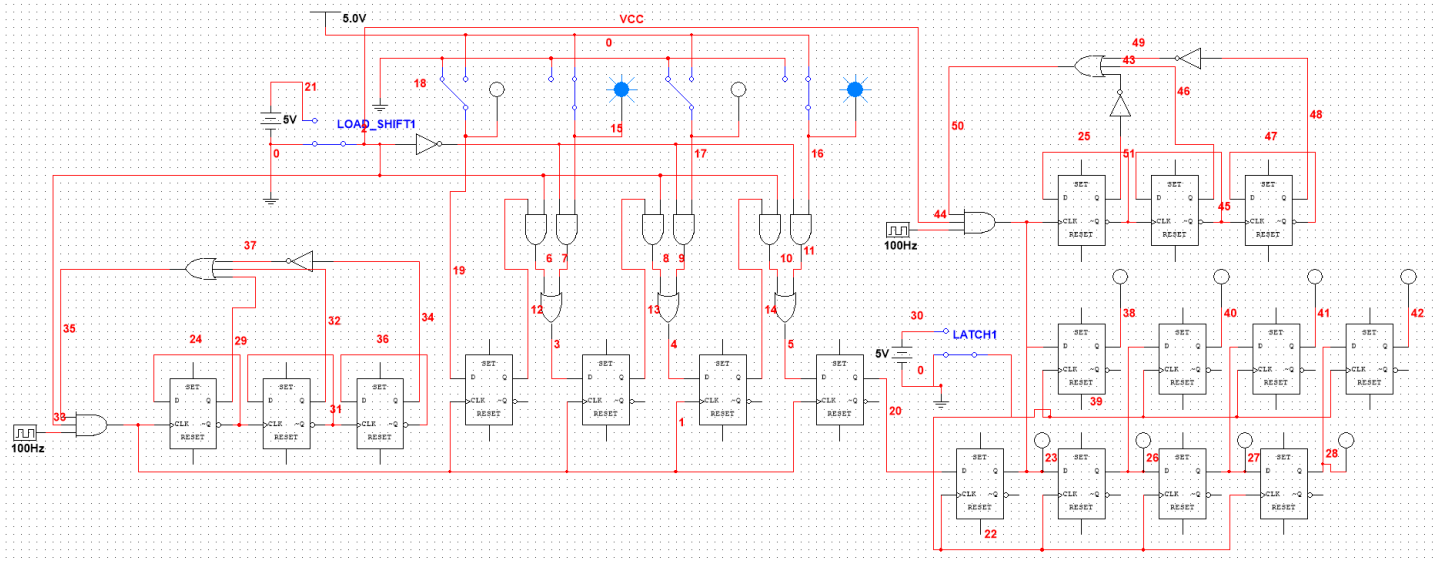
3.7 Połączenie nadajnika z odbiornikiem w Multisimie oraz przykład działania

Mając nasze układy łatwo połączyć je we wspólny układ nadajnika-odbiornika. Na wejście odbiornika należy przekierować wyjście nadajnika. Tym razem w odbiorniku potrzebujemy nie 4 a 5 impulsów, ponieważ dopiero po pierwszym impulsie nadajnika mamy co odbierać. Zegar dla odbiornika zrobiłem symetrycznie jak dla nadajnika tylko tym razem blokuje się na wartości 5 a nie 4. Dzięki temu mamy o jeden impuls więcej. Poniżej układ nadajnika-odbiornika:

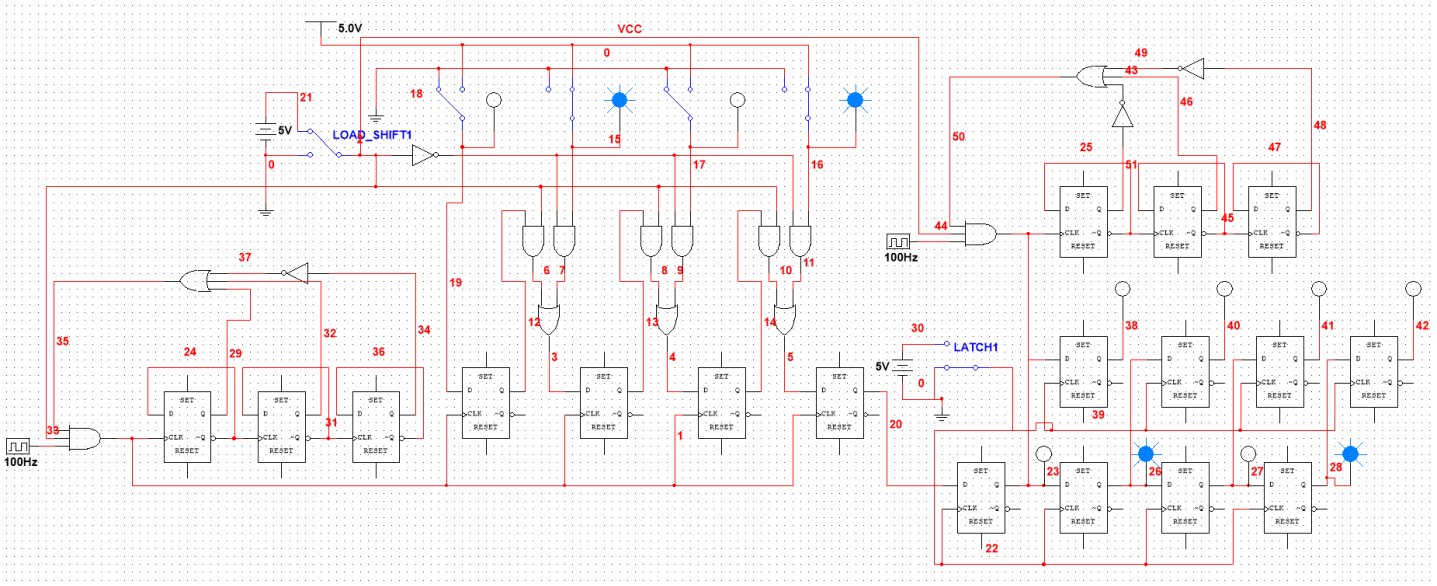


Nadajnik-odbiornik liczb czterobitowych (krok 1)

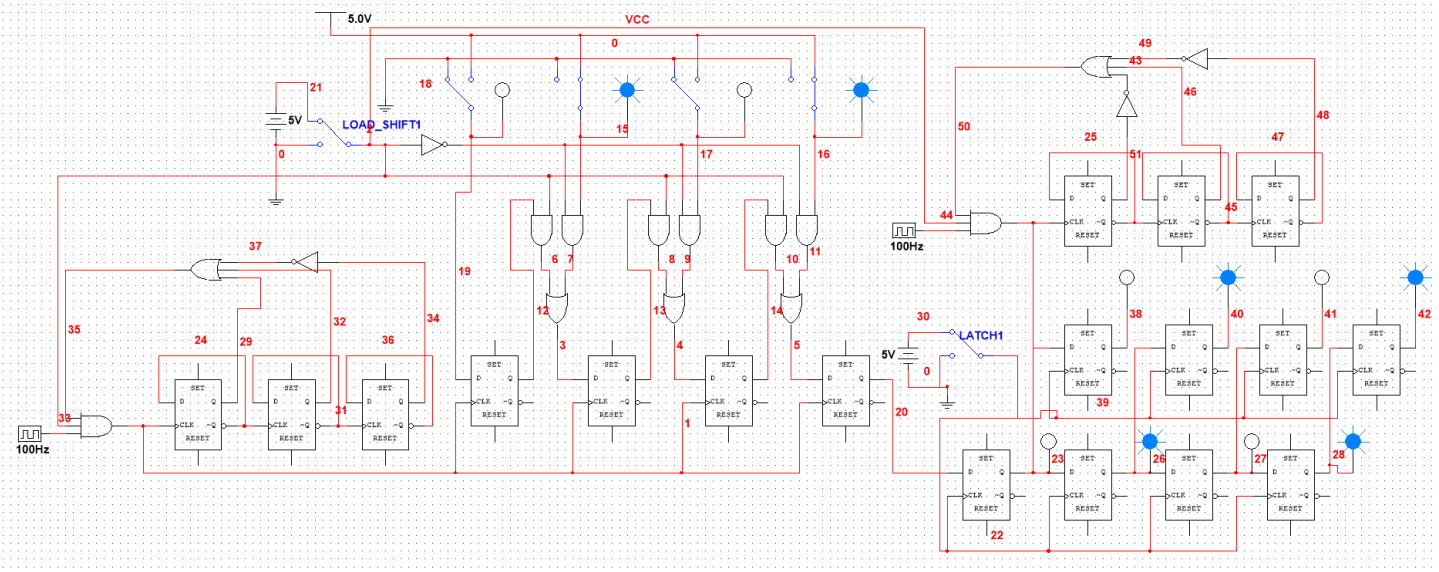
Załóżmy, że chcemy zakodować oraz odkodować cztery bity 1011. Zakładam poprawność działania każdego z rejestrów z osobna, ponieważ pokazałem to wcześniej. Poniżej krok po kroku:



Nadajnik-odbiornik liczb czterobitowych (krok 2), przed nadaniem czterech bitów 1010



Nadajnik-odbiornik liczb czterobitowych (krok 3), po nadaniu i odebraniu czterech bitów 1010



Nadajnik-odbiornik liczb czterobitowych (krok 4), po zatrzaśnięciu odebranego wyniku

3.8 Wnioski

3.8.1 Czy układ działa poprawnie?

To zadanie okazało się zdecydowanie najtrudniejszym i najbardziej kształcącym z wszystkich. Żeby je wykonać musiałem najpierw wykonać sprawozdanie 3, żeby powołać się na liczniki modulo n. Przedstawione przeze mnie rozumowanie pozwoliło mi dojść do poprawnego wyniku końcowego, którym jest poprawnie zbudowany układ nadajnika oraz odbiornika liczb czterobitowych. Przedstawione powyżej zdjęcia układów pokazują przykładowe nadawanie oraz odbieranie liczb oraz ich poprawność dzięki widocznym wskaźnikom. Po wielu testach stwierdzam, iż układ wykonany jest poprawnie.

3.8.2 Co można było zrobić inaczej?

- zbudować równoważny układ przy pomocy innych przerzutników takich jak T czy JK.
- sprawdzić działanie dla większej ilości danych poprzez Word Generator.
- zastosować inne kombinacje sposobu wejścia/wyjścia danych w poszczególnych rejestrach ($PISO$,$PIPO$,$SIPO$,$SISO$)
- zastosować wyzwalanie zboczem opadającym sygnału zegarowego
- stworzyć subcircuity dla większej czytelności układu
- zastosować jeden zegar dla nadajnika i odbiornika, zegar odbiornika musiałby w pewien sposób opóźniać wysłane impulsy
- zastosować Logic Analyzer w celu innego spojrzenia na wyniki

3.8.3 Gdzie to można zastosować?

- zamiana danych szeregowych na równoległe
- zamiana danych równoległych na szeregowe
- generator pseudoprzypadkowych sekwencji bitów
- liczniki pierścieniowe oraz Johnsona
- przechowywanie danych, transfer danych