Jan Chyczyński
Błażej Nowicki
Bartłomiej Słupik
Przemysław Węglik

Technika cyfrowa
Sprawozdanie 2

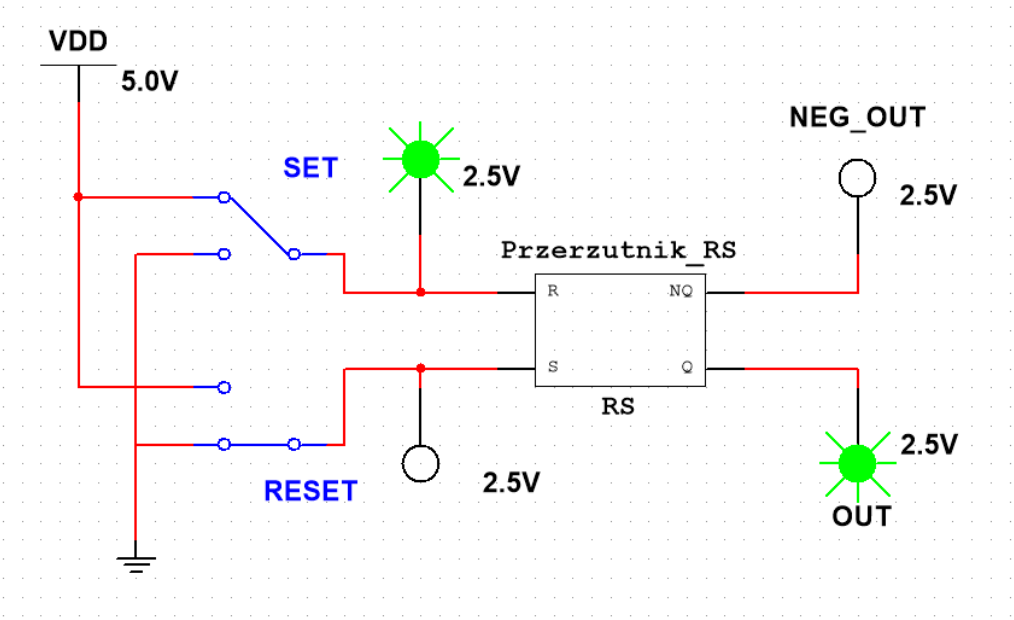
20 kwietnia 2022

1. Zadanie 2a

Zadanie polega na zaprojektowaniu i zbudowaniu asynchronicznego przerzutnika RS przy pomocy dwóch bramek NAND.

1.1. Idea

Układ docelowy powinien posiadać wyjścia Q i $\overline{Q}$, oraz wejścia S i R oraz działać zgodnie z tabelą prawdy przedstawioną poniżej.



Rysunek 1.1: Schemat idei rozwiązania

S	R	Q	Q_+
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	-
1	1	1	-

Tabela 1.1: Tabela prawdy przerzutnika RS

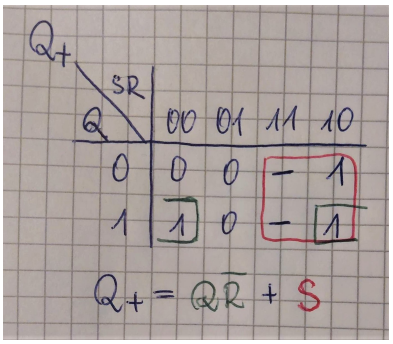
Zauważmy, że w tabeli dla $S = R = 1$ brakuje wartości Q_+ Jest to tzw. stan zakazany, zatem zakładamy, że nie zachodzi i nie rozważamy wyjścia bramki w takim przypadku.

1.2. Rozwiązanie teoretyczne

Szukamy funkcji logicznej określającej stan kolejnej iteracji Q_+ w zależności od stanu poprzedniego:

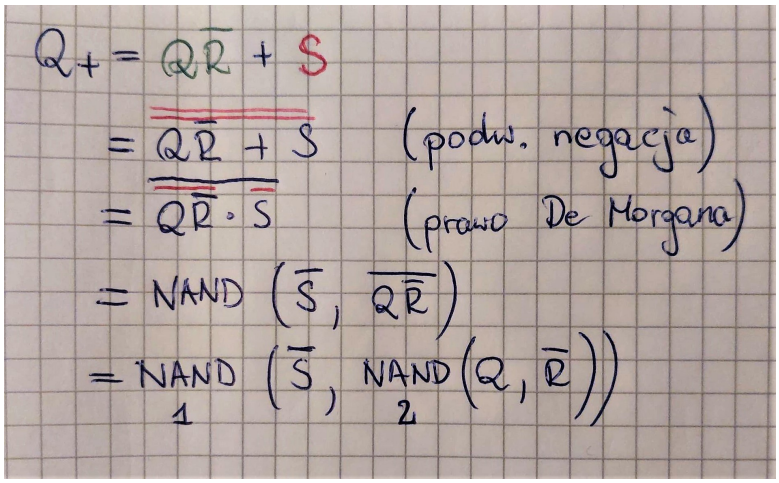
$$Q_+ = Q_+(S, R, Q) \tag{1.1}$$

W celu znalezienia tej funkcji posłużono się tabelą Karnough (pojawiają się w niej stany zabronione, jednak są one niejako "poza dziedziną" funkcji, a więc nie powinny nigdy zajść w prawidłowo użytkowanym przerzutniku):



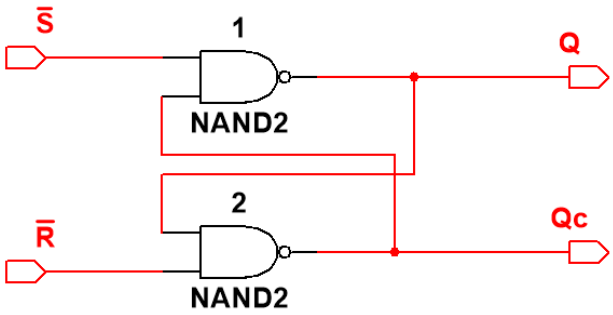
Rysunek 1.2: Tabela Karnough zastosowana w celu znalezienia funkcji logicznej przerzutnika RS

Następnie przekształcono funkcję Q_+, aby zapisać ją przy pomocy funkcji NAND:



Rysunek 1.3: Przekształcenia wzoru funkcji logicznj do postaci złożonej z funkcji NAND

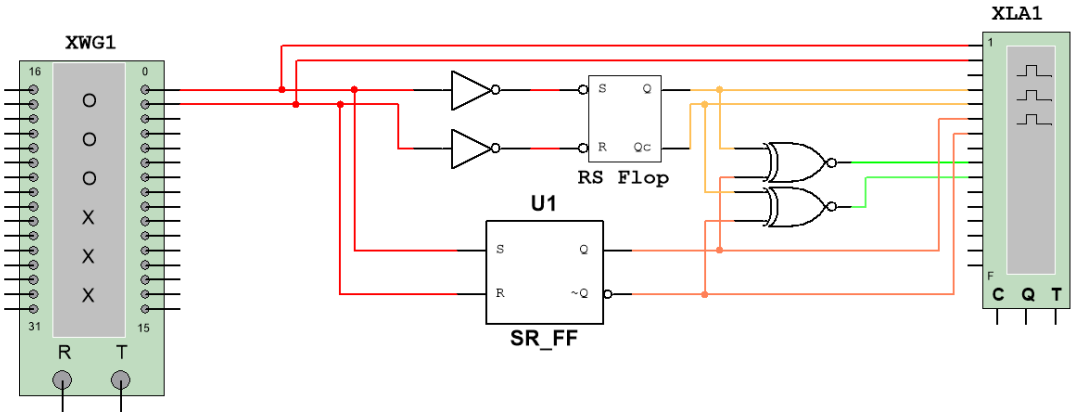
Na podstawie wzoru funkcji sporządzono schemat układu, który został przedstawiony na poniższym rysunku:



Rysunek 1.4: Schemat przerzutnika RS. Wejścia $\overline{S}$ i $\overline{R}$ są aktywne w stanie niskim.

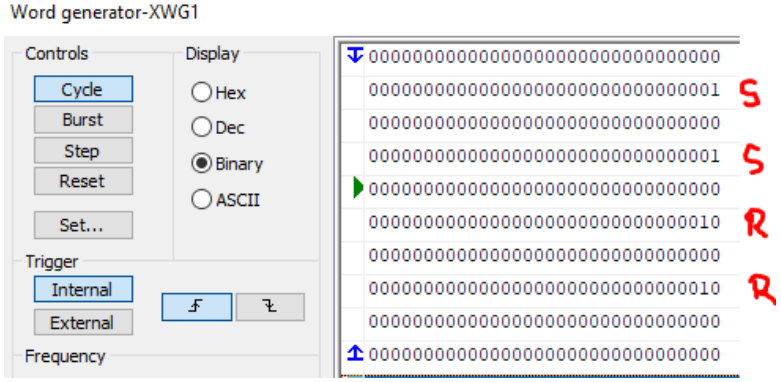
1.3. Testy w programie Multisim

Aby przetestować zaprojektowany układ, zbudowano następujący układ testowy w programie Multisim, który porównuje działanie układu z rzeczywistym przerzutnikiem RS:



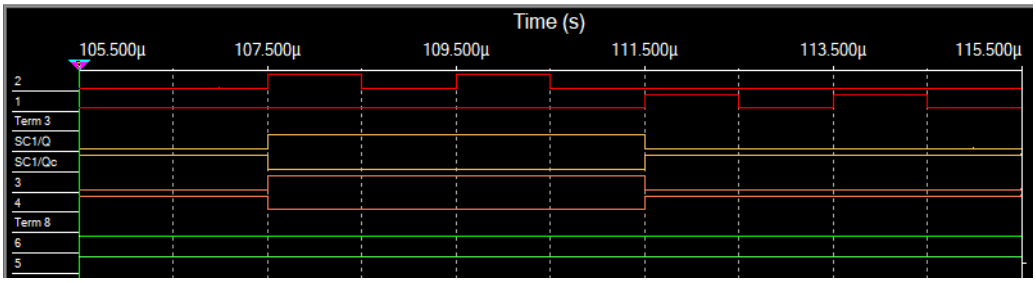
Rysunek 1.5: Schemat układu testowego.

Generator słów bitowych ustawiono na następującą sekwencję, która testuje przejścia między wszystkimi możliwymi stanami:



Rysunek 1.6: Sekwencja słów bitowych na wejściu układu testowego.

Na analizatorze stanów logicznych zaobserwowano następujące dane:



Rysunek 1.7: Sekwencja słów bitowych na wejściu układu testowego.

Ciągły stan wysoki na wyjściach bramek XNOR sprawdzających równoważność układów dowodzi poprawności działania przerzutnika.

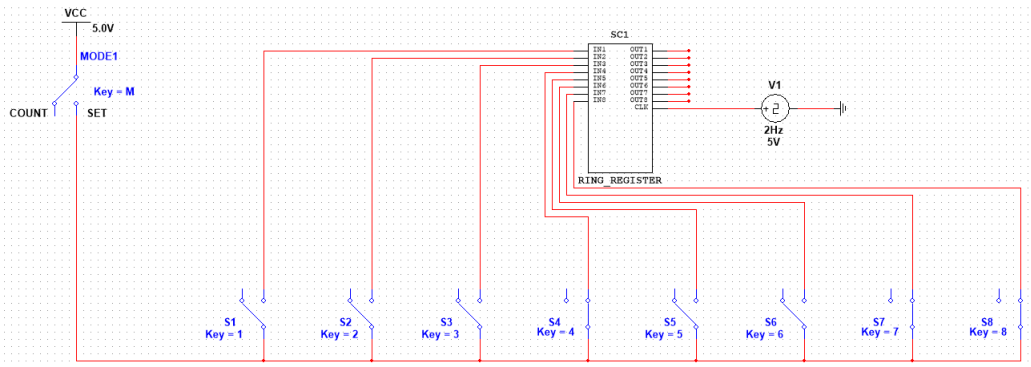
1.4. Wnioski

1. Przerzutnik RS jest prostym układem, który sprawdza się, gdy mamy pewność, że układ sterujący nim nie poda na wejściu stanu zabronionego $S = R = 1$. Jeśli przerzutnikiem steruje wprost użytkownik, lepiej zastosować przerzutnik JK, który jest odporny na niekontrolowane zachowanie.
2. Synchronicznej wersji przerzutnika RS (takiej która zmienia stan tylko podczas taktu zegara) można użyć do budowy rejestrów przesuwnych.
3. Przerzutnik RS można zastosować w celu uniknięcia "efektu skakania" przycisków i przełączników mechanicznych.
4. Wszelkiego rodzaju detektory wartości krytycznej (zachodzenia danego zjawiska). Możemy podpiąć przerzutnik RS do detektora i jeśli zajdzie oczekiwane zjawisko, przerzutnik zostanie włącznony (SET. Wtedy nawet po dłuższym czasie będziemy w stanie stwierdzić czy zaszło dane zjawisko/warunek. Przykład: alarm przeciwpożarowy - przy wykryciu dymu, włączany jest przerzutnik RS, który uruchamia syreny, bez możliwości automatycznego resetu. Zresetowanie jest możliwe tylko poprzez wciśniecie guzika (RESET), nie przez czujnik, który jest podłączony do wejścia SET.

2. Zadanie 2b

Korzystając z wybranych przerzutników, proszę zbudować rejestr pierścieniowy. Rejestr ten powinien realizować dwie podstawowe funkcje wybierane przy pomocy pojedynczego przełącznika: ładowanie danych do rejestru i krążenie danych w rejestrze.

2.1. Idea



Rysunek 2.1: Idea rozwiązania.

Przełącznik MODE pozwala nam zmieniać tryb pracy na tryb ładowania lub krążenia danych. Przełączniki S1 do S8 pozwalają na ustawienie wartości poszczególnych bitów rejestru kiedy przełącznik MODE jest w trybie SET. Nasz rejestr będzie rejestr beðzie rejestrem PIPO (parallel in, parallel out). Co oznacza, że może równolegle przejmować wejście i zwracać wyjście dla wszystkich bitów przerzutnika.

2.2. Wstęp teoretyczny

Rejestr pierścieniowy to rejestr przesuwny, gdzie wyjście ostatniego przerzutnika podłączone jest do wejścia przerzutnika pierwszego. W ten sposób wartości w rejestrze mogą w ciągły sposób w nim krążyć. Do budowy rejestru użyliśmy 8 przerzutników D w taki sposób, że wyjście poprzedniego połączone jest w wejściem kolejnego (jak w rejestrze przesuwnym).

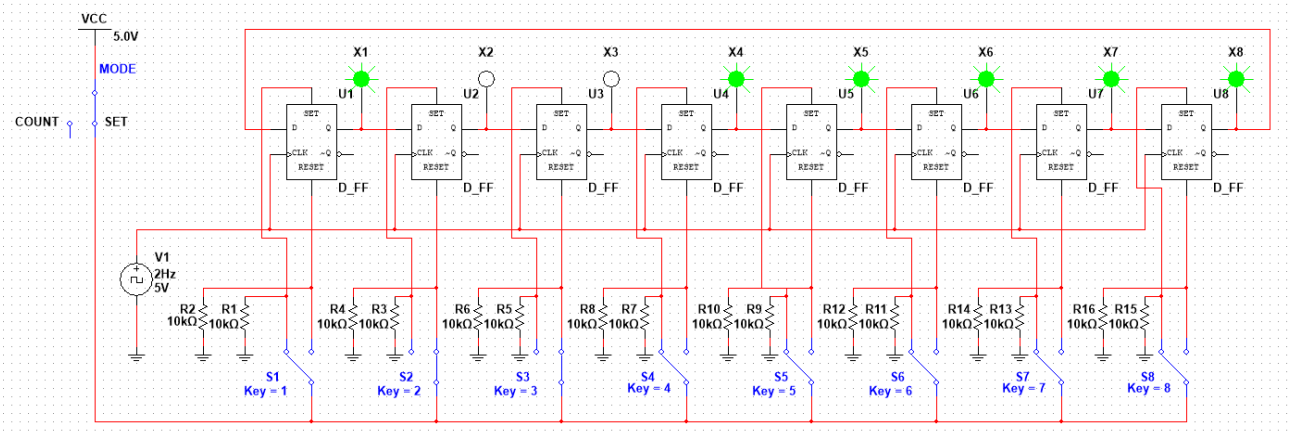
Na przykład, jeśli załadujemy do rejestru (w trybie SET) liczbę 10011111, to po przełączeniu w tryb COUNT wartości rejestru po kolejnych 8 taktach zegara przedstawia poniższa tabela.

Jak widzimy po 8 taktach zegara (czyli liczbie równej długości rejestru) stan rejestru jest taki sam jak na początku - odbył się pełen cykl.

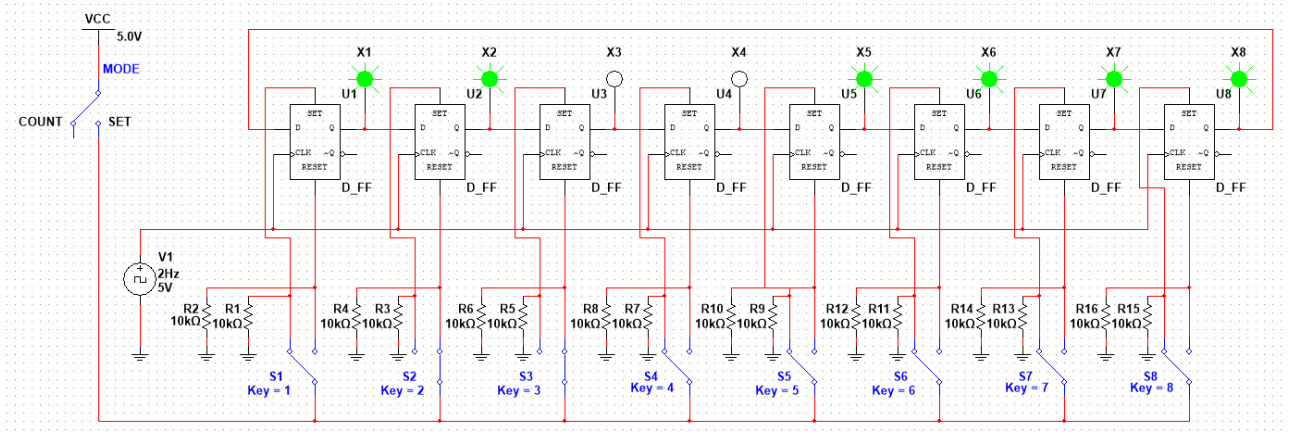
Czas	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8
0	1	0	0	1	1	1	1	1
1	1	1	0	0	1	1	1	1
2	1	1	1	0	0	1	1	1
3	1	1	1	1	0	0	1	1
4	1	1	1	1	1	0	0	1
5	1	1	1	1	1	1	0	0
6	0	1	1	1	1	1	1	0
7	0	0	1	1	1	1	1	1
8	1	0	0	1	1	1	1	1

Tabela 2.1: Stan bitów Q1 do Q8 rejestru pierścieniowego podczas przebiegu w czasie

2.3. Realizacja w Multisim



Rysunek 2.2: Układ w trybie SET, z przełącznikami ustawiajacymi wartość rejestru na 10011111



Rysunek 2.3: Układ w trybie COUNT po jednym takcie zegara

2.4. Testy

TODO zrobić subcircuit comparator i dodać obrazek z comparatorem, wyglądem word generatora i outputem z sygnałów

2.5. Wnioski

1. Używamy przerzutników D, które posiadają opcje ustawiania i resetowania. Bez tego, korzystając z "czystego" przerzutnika D nie moglibyśmy zrealizować funkcji ładowania danych do rejestru.
2. Nasz rejestr jest tzw. rejestrem PIPO (parallel in, parallel out), ale istnieją także wersje z wejściem i wyjściem szeregowym SIPO, PISO i SISO. Każdy z nich znajduje zastosowania w nieco innych przypadkach, u nas konieczne było zastosowanie wejścia równoległego do realizacji funkcji ładowania, a równoległe wyjście pozwala na efektywne testowanie.
3. Rejestru pierścieniowego można użyć do konstrukcji lampek choinkowych, które cyklicznie zmieniają się zgodnie z bitami krążącymi w rejestrze. Przesunięcia rejestru tworzą iluzję animacji na choince.

2.6. Podusmowanie