| AGH<br>Akademia Górniczo-Hutnicza<br>im. Stanisława Staszica<br>w Krakowie | **Technika cyfrowa** |
|---|---|
| **Temat ćwiczenia**<br>Przerzutniki i rejestry | **Numer ćwiczenia**<br>3 |
| **Wykonawca**<br>Marcin Przewięźlikowski | **Ocena** |

# 1 Cel ćwiczenia

Zapoznanie się z różnymi rodzajami przerzutników i zbudowanie z nich rejestrów SISO, SIPO, PIPO i PISO.

# 2 Przebieg ćwiczenia

## 2.1 Asynchroniczny przerzutnik RS

| S | R | Q(n) (NOR) | Q(n) (NAND) |
|---|---|---|---|
| 0 | 0 | Q(n-1) | zab |
| 1 | 0 | 1 | 0 |
| 0 | 1 | 0 | 1 |
| 1 | 1 | zab | Q(n-1) |

**NOR**

| Q(n-1)\SR | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| **0** | 0 | 0 | x | 1 |
| **1** | 1 | 0 | x | 1 |

Q(n) = S + R'Q(n-1) = (S nor (R nor Q'(n-1)))'
Q'(n) = R + S'Q'(n-1) = (R nor (S nor Q(n-1)))'

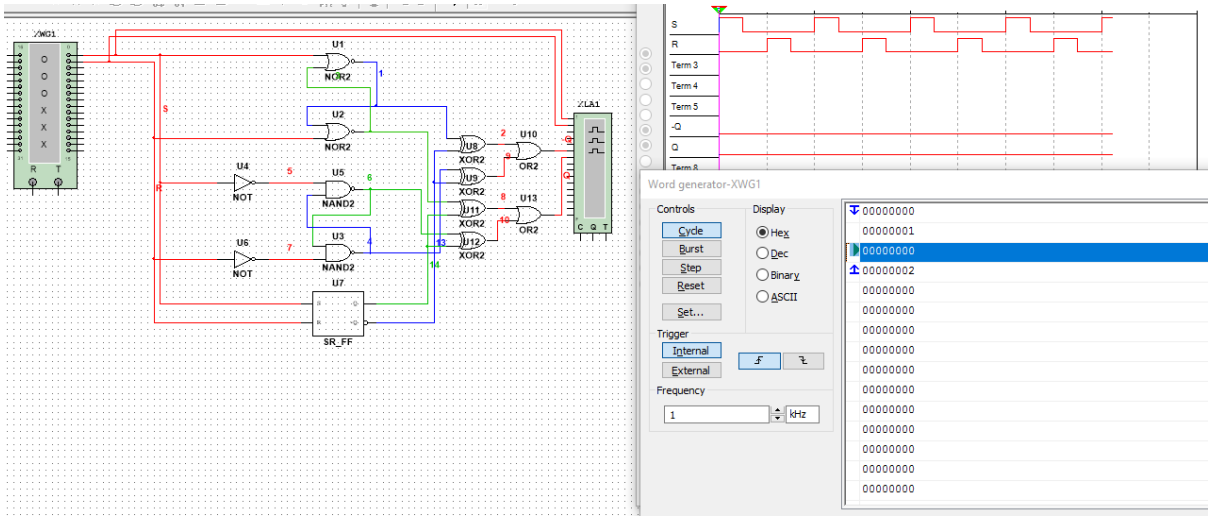
Po zanegowaniu stronami obu równań:
Q(n) = R nor (S nor Q(n-1))
Q'(n) = S nor (R nor Q'(n-1))

**NAND**

| Q(n-1)\SR | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| **0** | x | 1 | 0 | 0 |
| **1** | x | 1 | 1 | 0 |

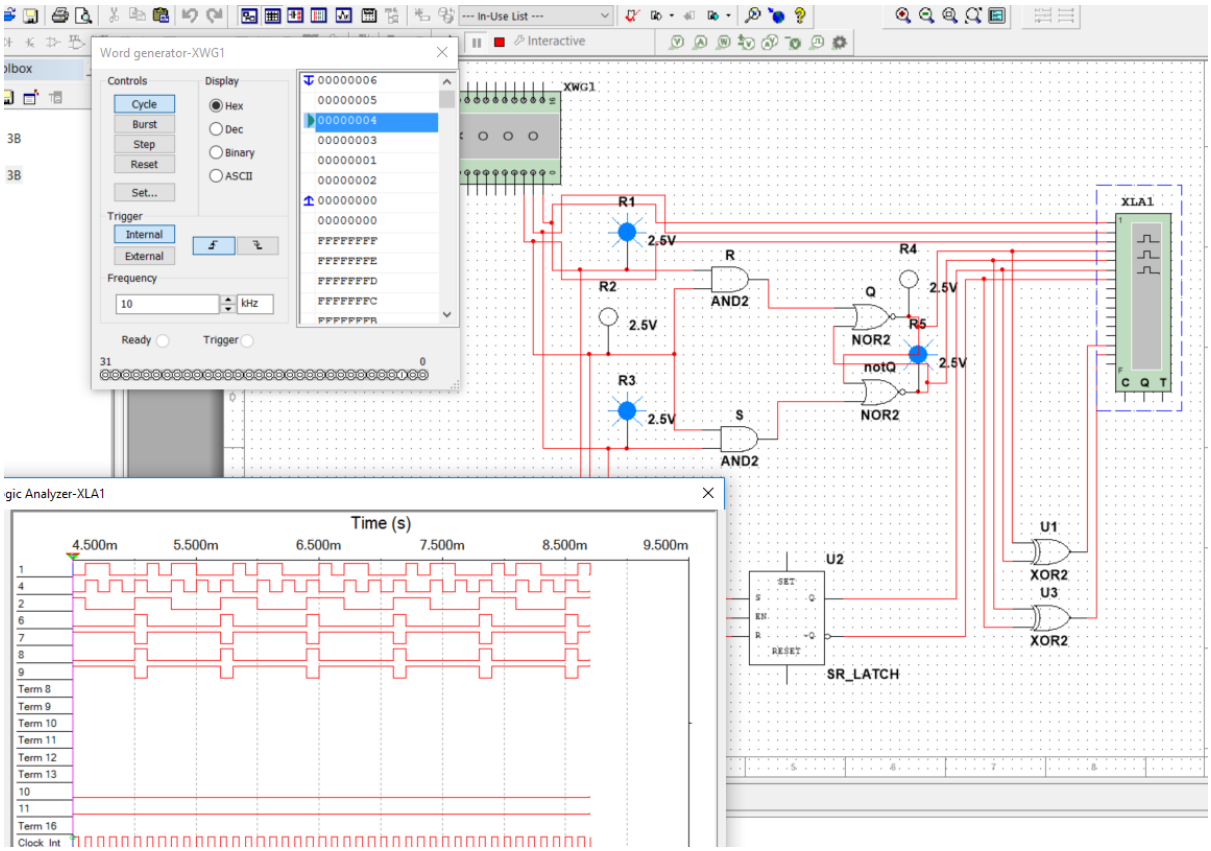
Q(n) = S' + RQ(n-1) = S nand (R nand Q(n-1))
Q'(n) = R' + SQ'(n-1) = R nand (S nand Q'(n-1))

Zauważmy, że możemy przekształcić inaczej:
Q(n) = S' + RQ(n-1) = (S' nor (R' nor Q'(n-1)))'
Q'(n) = R' + SQ'(n-1) = (R' nor (S' nor Q(n-1)))'
Po zanegowaniu stronami:
Q(n) = R' nor (S' nor Q(n-1))
Q'(n) = S' nor (R' nor Q'(n-1))
Forma analogiczna jak dla bramek nor,
jednak z zanegowanymi wejściami.




Widzimy, że przerzutnik działa zgodnie z przewidywaniami. Przerzutnik oparty o bramki NAND działa tak, jakbyśmy zanegowali wejścia przerzutnika opartego o bramki NOR.

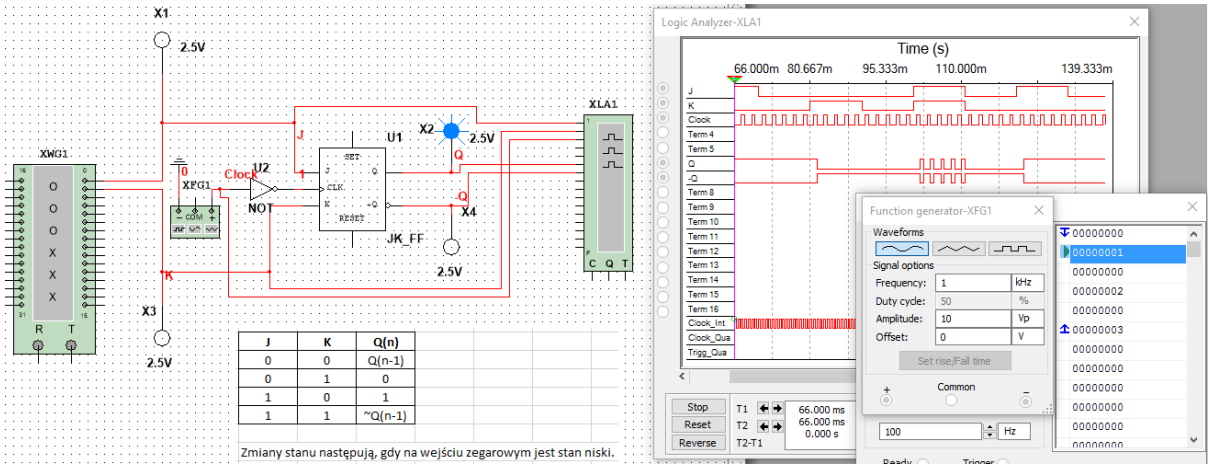
## 2.2 Synchroniczny przerzutnik RS



| C | S | R | Q(n) |
|---|---|---|---|
| 0 | 1 | 1 | Q(n-1) |
| 0 | 0 | 1 | Q(n-1) |
| 0 | 1 | 0 | Q(n-1) |
| 0 | 0 | 0 | zab |
| 1 | 1 | 1 | Q(n-1) |
| 1 | 1 | 0 | 1 |
| 1 | 0 | 1 | 0 |
| 1 | 0 | 0 | Q(n-1) |

Synchroniczny przerzutnik RS zmienia stan swoich wyjść tylko, gdy wejście zegara jest w stanie wysokim
Z analizy logic analyzerem widać, że zmontowany przerzutnik działa poprawnie.
Synchroniczne przerzutniki RS są często stosowane przy taktowaniu sieci cyfrowych - dzięki temu można być pewnym, że ich elementy są ze sobą zsynchronizowane i skoordynowane.

## 2.3 Synchroniczny przerzutnik JK



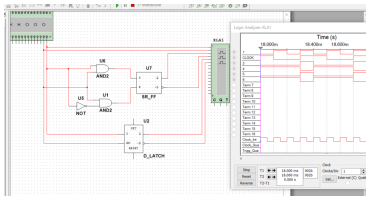

| J | K | Q(n) |
|---|---|---|
| 0 | 0 | Q(n-1) |
| 0 | 1 | 0 |
| 1 | 0 | 1 |
| 1 | 1 | ~Q(n-1) |

Przerzutnik JK ma dwa wejścia:

- jedynkujące
- kasujące

Jest on również synchroniczny. Jego działanie opisuje powyższa tabela prawdy. Jak widać na załączonym schemacie, przetestowanie tej tablicy w Multisimie zakończyło się sukcesem.
Ciekawostka: nazwa przerzutnika pochodzi od inicjałów **Jacka Kilby'yego** - wynalazcy układów scalonych (nie mylić z Jackiem Kirbym).

## 2.4 Przerzutnik D na podstawie asynchronicznego przerzutnika RS



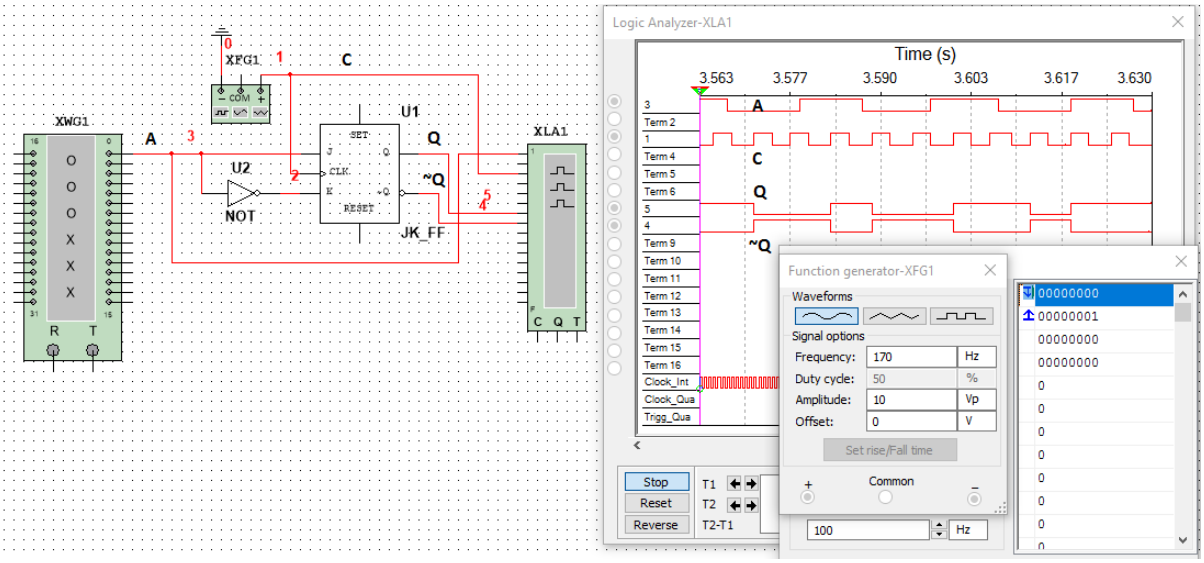
| C | D | Q(n) |
|---|---|---|
| 0 | 0 | Q(n-1) |
| 0 | 1 | Q(n-1) |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

Nazwa przerzutnika pochodzi z j. angielskiego - od słowa *data* lub *delay*. Jest to układ opóźniający - przepisuje sygnał wejściowy, ale tylko przy odpowiednim sygnale zegara. Jest to zatem także układ synchroniczny.
Przerzutnik D znajduje szerokie zastosowanie w budowie rejestrów.

## 2.5 Przerzutnik T na podstawie synchronicznego przerzutnika D

Przerzutnik typu T (eng. *Toggle*), który po podaniu stanu wysokiego na wejście T i opadnięciu sygnału zegarowego zmienia stan wyjścia na przeciwny od dotychczasowego.

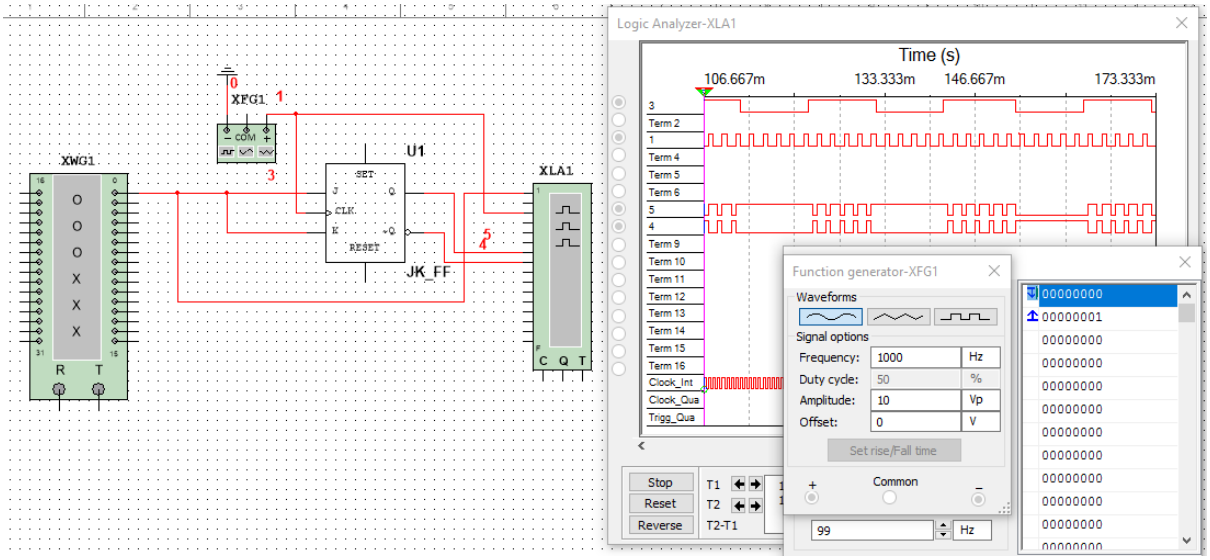
## 2.6 Przerzutnik D na podstawie synchronicznego przerzutnika JK




| C | D | Q(n) |
|---|---|---|
| 0 | 0 | Q(n-1) |
| 0 | 1 | Q(n-1) |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

Powyższy układ ilustruje budowę przerzutnika D za pomocą przerzutnika JK. Jak widać, inżynierzy elektronicy mają w budowie swoich układów do dyspozycji często całe wachlarze możliwości konstrukcyjnych dla zrealizowania tych samych potrzeb.

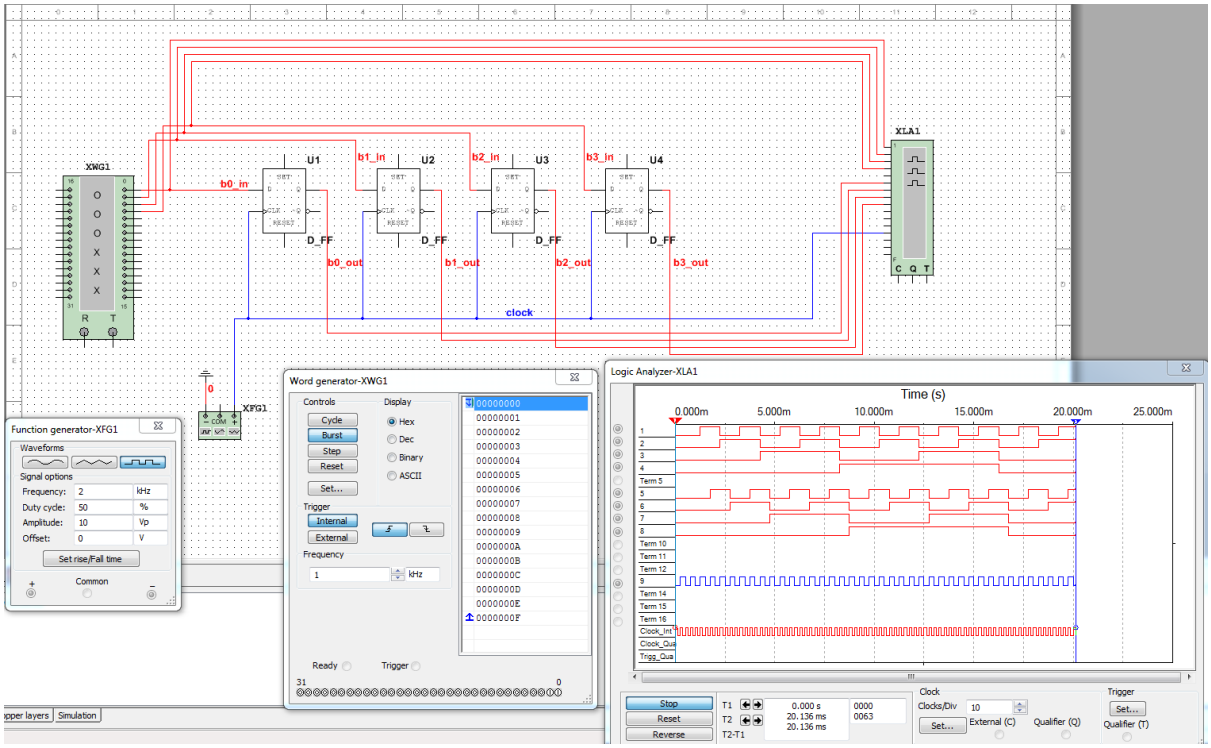
## 2.7 Przerzutnik T na podstawie synchronicznego przerzutnika JK



Powyższy układ ilustruje budowę przerzutnika T za pomocą przerzutnika JK.
Jak widać z powyższego i poprzedniego przykładu, prosty układ JK może być podstawą wielu przerzutników pełniących cały szereg różnych funkcji.

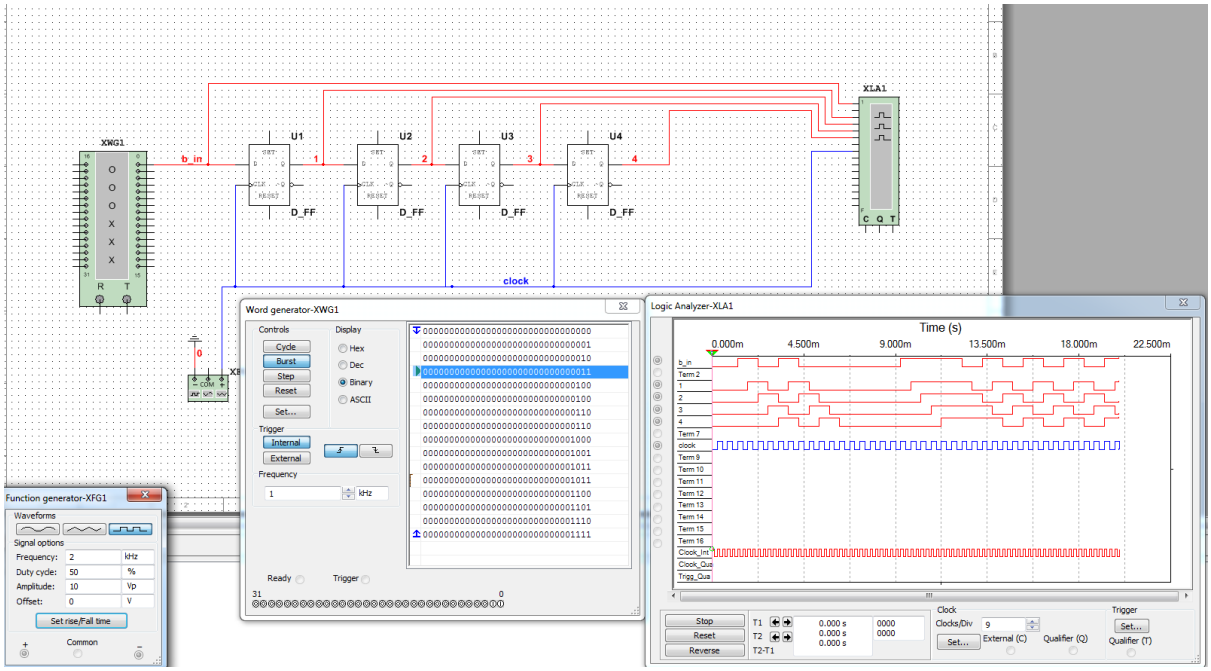
## 2.8 Rejestry na podstawie synchronicznych przerzutników D

### 2.8.1 Rejestr PIPO



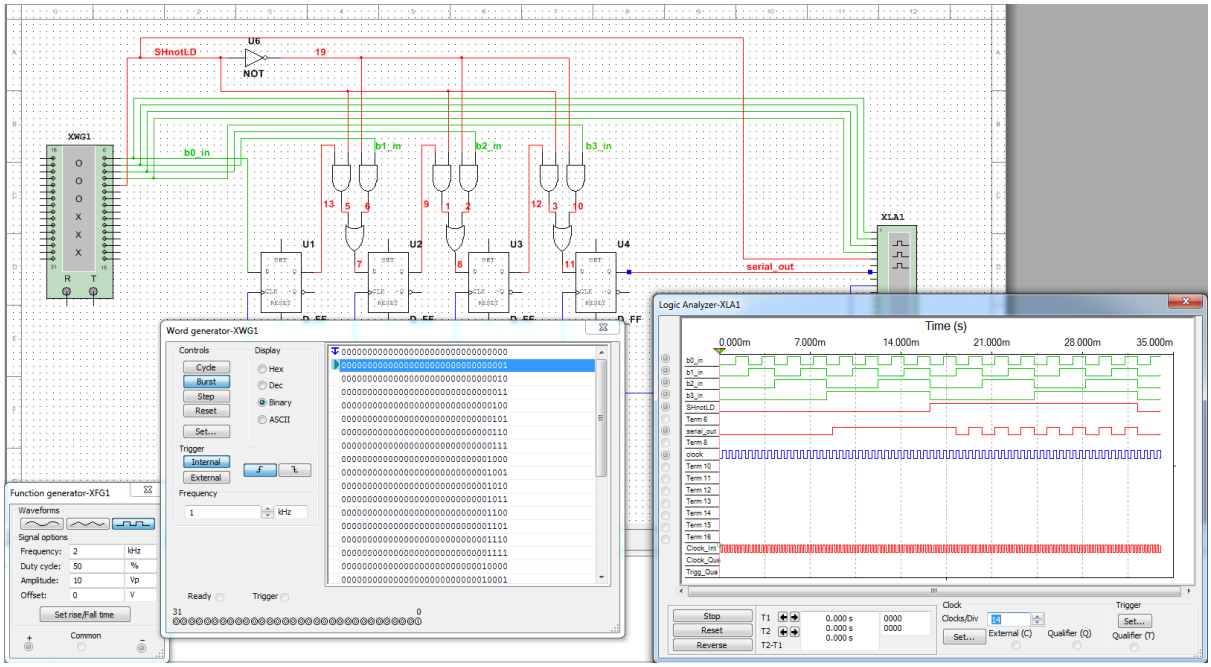

**PIPO - Parallel-In Parallel-Out** - równoległe wejścia i równoległe wyjścia, czyli wejście i wyjście rejestru jest "szyną" 4 bitową - w każdym z przerzutników sygnał zmieniany jest tylko, gdy dany jest odpowiedni sygnał z zegara.

### 2.8.2 Rejestr SIPO



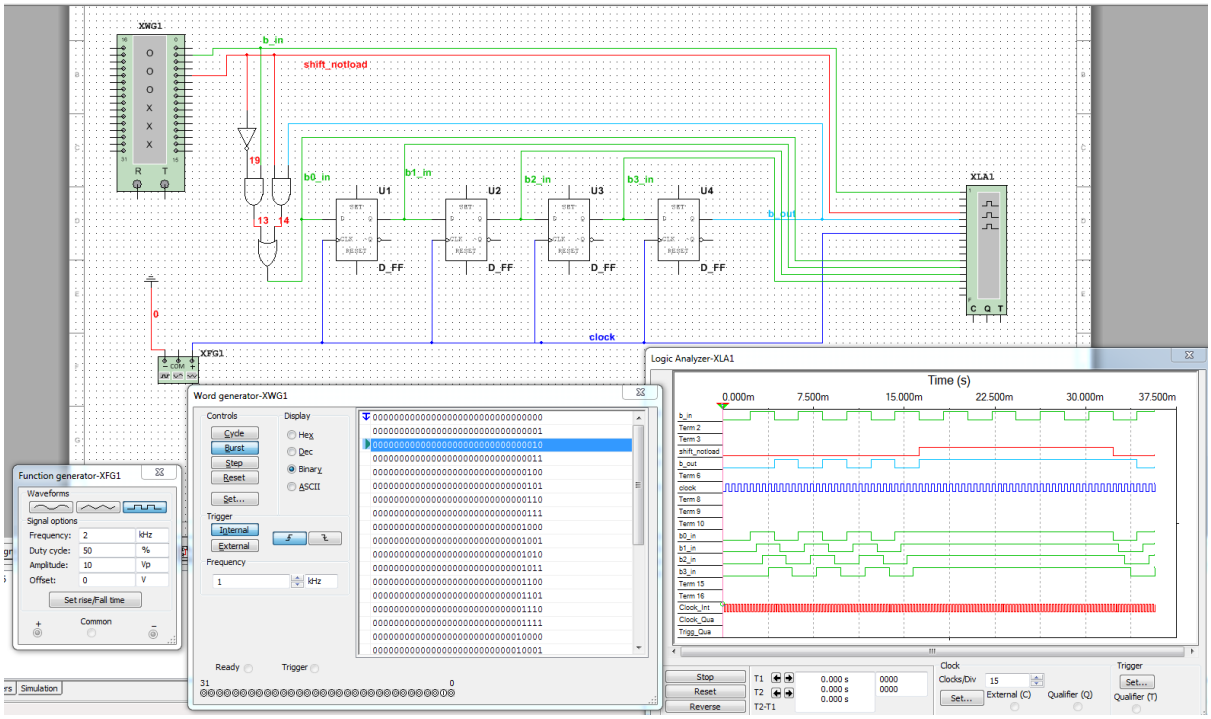

**SIPO - Serial-In Parallel-Out** - szeregowe wejście, równoległe wyjście - na pojedynczym wejściu podajemy ciąg bitów, a na wyjściu dostajemy 4 równoległe bity. Wejście każdego kolejnego przerzutnika D podłączamy do wyjścia poprzedniego, wejście pierwszego z nich podłączamy do źródła danych. Wyjście każdego z przerzutników to kolejne bity wyjścia rejestru. Z każdym kolejnym cyklem zegara bity przesuwają się w prawo. Bit najbardziej po prawej jest tracony, najbardziej po lewej jest odczytywany z wejścia szeregowego.

### 2.8.3 Rejestr PISO




**PISO - Parallel-In Serial-Out** - na wejściu podajemy wiele sygnałów (równolegle), zaś na wyjściu otrzymujemy sygnał szeregowy. Potrzebna jest możliwość wybrania - czy przesuwamy liczbę w rejestrze w prawo, czy też wczytujemy nową liczbę do rejestru wieloma wejściami. Osiąga się to następującą metodą:
Korzystamy z elementu o nazwie multiplekser - na wejściu dostaje on sterujący sygnał jednobitowy i dwa sygnały jednobitowe (źródła), między którymi będzie przełączał. W zależności od wartości sygnału sterującego, na wyjściu multipleksera pojawia się albo sygnał z jednego źródła, albo z drugiego. Na funkcji - jeżeli oznaczymy sygnał sterujący przez C, wejścia przez A, B, to multiplekser jednobitowy realizuje funkcję Y = AC + B( C), czyli jeżeli C==1, na wyjściu jest A, jeżeli C==0, na wyjściu jest B. W tym wypadku jednym ze źródeł jest wyjście poprzedniego rejestru (bo chcemy móc przesuwać bity w rejestrze), a drugim jest wejście danych (czyli miejsce, skąd chcemy wpisać dane do rejestru). Do każdego wejścia równoległego oprócz pierwszego dodajemy taki multiplekser. Sygnałem sterującym jest SHnotLD - jeżeli sygnał sterujący jest wysoki to wykonujemy SHift - przesunięcie, a jeżeli jest niski, to wykonujemy LoaD - załadowanie nowej liczby do rejestru.

### 2.8.4 Rejestr SISO




**SISO - Serial-In Serial-Out** - ten rejestr szeregowo dostaje dane i szeregowo ma je zwrócić. Ponieważ na wejściu dostajemy cały czas jakiś sygnał, to chcemy zapobiec utracie to, co w rejestrze jest. Dlatego tutaj też stosujemy multiplekser korzystający z sygnału SHnotLD, tym razem jednak używamy go, żeby "zapętlić" wyjście szeregowe z wejściem. Wtedy sygnałem SHnotLD, wybieramy czy przesuwamy dalej dane w prawo gubiąc stare (dla sygnału 1 SHnotLD), czy też zapętlamy wyjście z wejściem, czyli efektywnie to co było w rejestrze, będzie się w nim zapętlać.

# 3 Wnioski

Dzięki temu ćwiczeniu dowiedzieliśmy się, jak przerzutniki mogą posłużyć ku wyjściu na kolejny poziom abstrakcji - do zbudowania rejestrów, skąd już tylko niewielki krok do asemblera i języków wysokiego poziomu.