

# Technika Cyfrowa - sprawozdanie 3

Łukasz Jezapkowicz

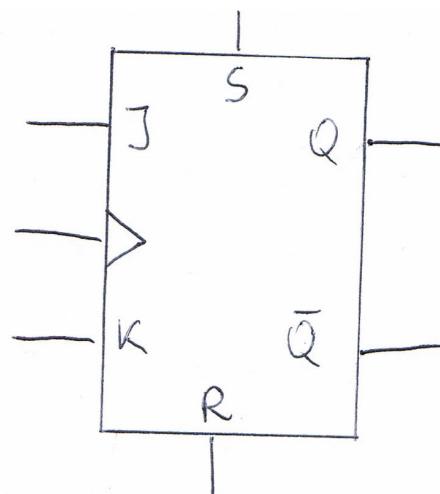
21.04.2020

- 1 Zaprojektować, zrealizować i przetestować dwójkę liczącą w oparciu o przerzutnik  $JK$ . Proszę rozpatrzyć wszystkie możliwe warianty.

## 1.1 Przerzutnik typu *JK*

Na początku warto zaznajomić się z przerzutnikiem typu  $JK$ . Jest to synchroniczny przerzutnik z dwoma wejściami  $J$  i  $K$  oraz dwoma wyjściami  $Q$  oraz  $\bar{Q}$ . Dla wartości logicznych 0 na wejściu stan przerzutnika jest podtrzymyany, dla wartości logicznej 1 na wejściu  $J$  na wyjściu  $Q$  pojawi się stan wysoki, dla wartości logicznej 1 na wejściu  $K$  na wyjściu  $Q$  pojawi się stan niski. Dla wartości logicznych 1 na obydwu wejściach przerzutnik zmienia swój stan na przeciwny. Poniżej tabela prawdy oraz schemat przerzutnika  $JK$ :

J	K	$Q_n$	$Q_{n+1}$
0	0	011	011
0	1	011	0
1	0	011	1
1	1	011	110



Schemat oraz tabela prawdy przerzutnika typu *JK*

## 1.2 Co to licznik, co to dwójką licząca

Licznik to układ cyfrowy, którego zadaniem jest zliczanie wystąpień sygnału zegarowego, najczęściej realizowany na układzie dzielnika częstotliwości. Licznik może liczyć albo w górę, albo w dół oraz może być asynchroniczny lub synchroniczny. Tutaj rozpatrywać będę dzielnik modulo 2 - tzw. dwójkę liczącą. Dzielnik modulo  $n$  zlicza  $n - 1$  impulsów zegarowych, a przy  $n$ -tym impulsie zeruje się. Dla  $n = 2$  licznik będzie przyjmował następujące stany:

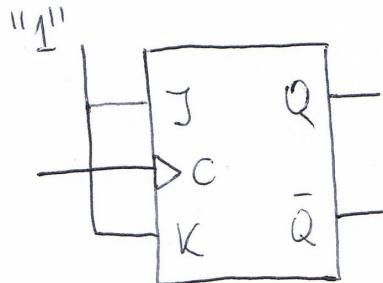
$$0 \rightarrow 1 \rightarrow 0 \rightarrow 1 \rightarrow 0 \rightarrow 1 \rightarrow \dots$$

### 1.3 Projektowanie licznika modulo 2 przy pomocy przerzutnika JK

Chcemy, żeby na wyjściu Q przerutnika wartość zmieniała się co każdy impuls.

Pierwsza możliwość:

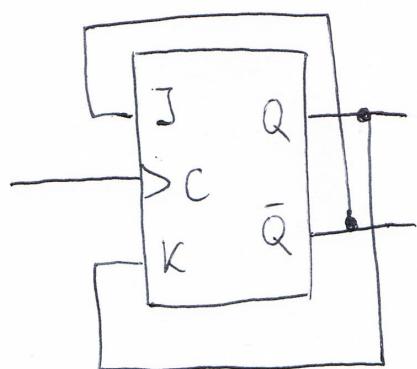
Z tabeli prawdy przerutnika JK łatwo zauważyc, że wystarczy na wejścia J i K podawać wartości logiczne 1 by przerutnik zmieniał wartości na wyjściu co każdy impuls. Schemat:



Druga możliwość:

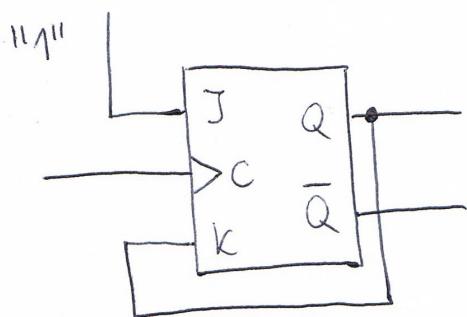
Z tabeli prawdy widać, że gdyby udało nam się na wejście podawać na zmianę sygnały 01 oraz 10 to na wyjściu również sygnał będzie zmieniał swoją wartość co impuls.

Zmiennej sygnał wejściowy możemy wziąć jako wyjścia Q oraz  $\bar{Q}$ . Schemat:



Mozna jeszcze ozymisie wziac kombinacje z dwóch pierwszych wariantów.

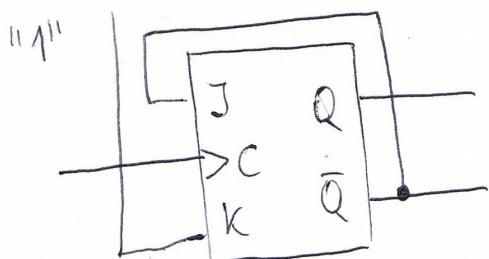
Trećia możliwość



$$Q_n = 0 \Rightarrow K = 0 \Rightarrow JK = 10 \Rightarrow Q_{n+1} = 1$$

$$Q_n = 1 \Rightarrow K = 1 \Rightarrow JK = 11 \Rightarrow Q_{n+1} = 0$$

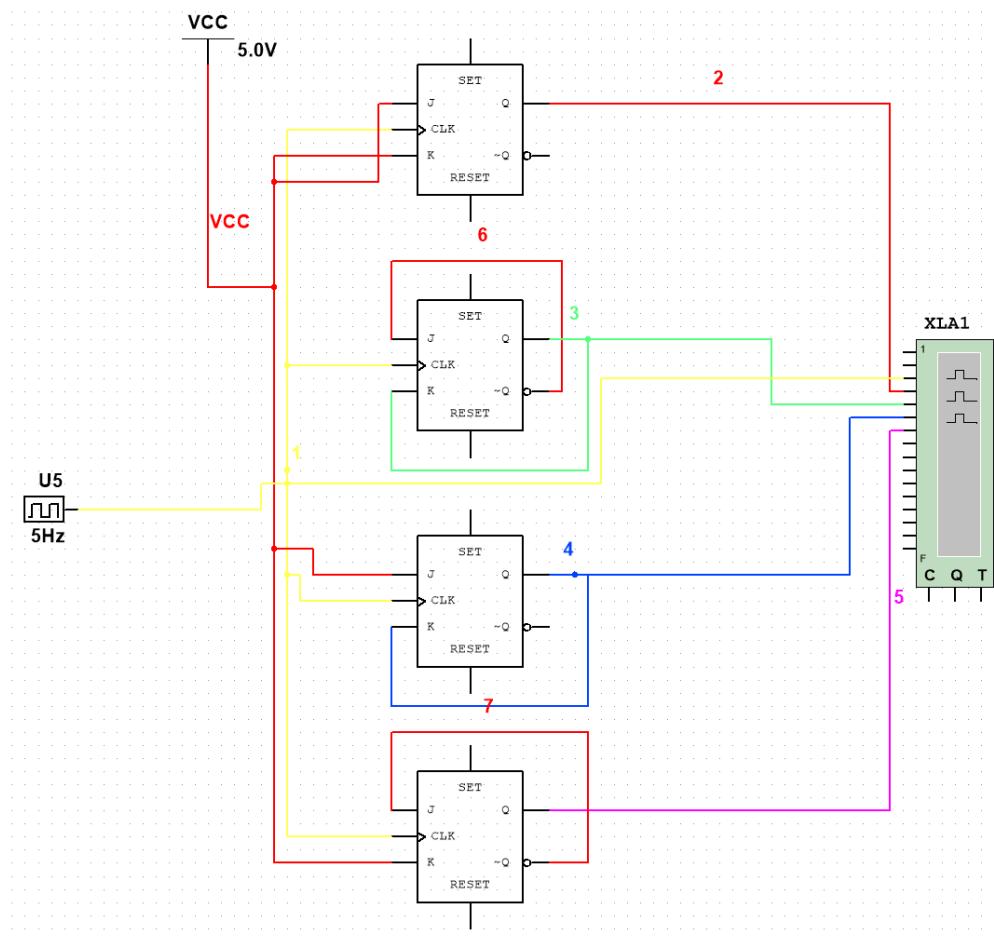
Czwarta możliwość



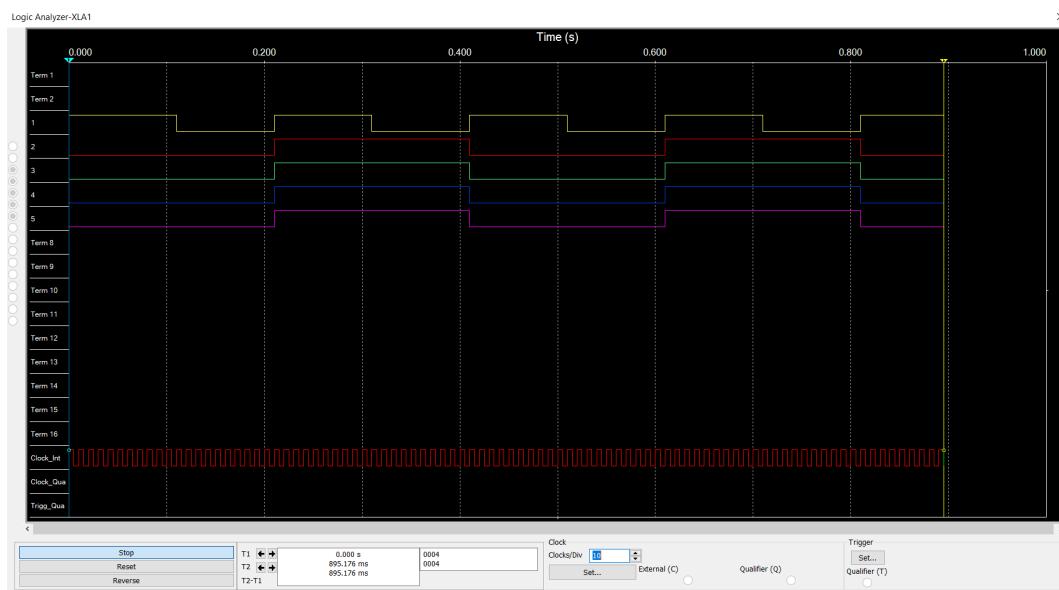
$$Q_n = 0 \Rightarrow J = 1 \Rightarrow JK = 11 \Rightarrow Q_{n+1} = 1$$

$$Q_n = 1 \Rightarrow J = 0 \Rightarrow JK = 01 \Rightarrow Q_{n+1} = 0$$

## 1.4 Budowa układu w Multisimie



Schemat układu dwójkki liczącej w Multisimie



Wyniki Logic Analyzer'a

## 1.5 Wnioski

### 1.5.1 Czy układ działa poprawnie?

Przedstawione przeze mnie rozumowanie pozwoliło mi dojść do poprawnego wyniku końcowego, którym są poprawnie zbudowane układy dwójkii liczącej - dzielnika modulo 2. Przedstawione powyżej zdjęcie Logic Analyzer'a pokazuje, że faktycznie każdy dzielnik zadziałał jak dzielnik częstotliwości przez 2. Każdy z 4 wariantów działa tak samo. Ponieważ założenia zadania zostały spełnione, stwierdzam, iż układ został wykonany poprawnie.

### 1.5.2 Co można było zrobić inaczej?

- Do budowy dwójkii liczącej można było użyć innych przerzutników takich jak przerzutnik typu  $D$  lub  $T$ .

### 1.5.3 Gdzie to można zastosować?

- W budowie większych, bardziej rozbudowanych liczników.
- Stoper, zegar.
- W budowie dzielników modulo  $n$  dla  $n \geq 2$ .
- Sprawdzanie parzystości długości sygnału wejściowego.

- 2 W oparciu o dowolnie wybrany typ dwójkę liczących, zaproponować i zbudować czterobitowy licznik asynchroniczny liczący wstecz. Przetestować i pokazać jego działanie w programie Multisim. Pokazać przebiegi sygnałów na wyjściach poszczególnych dwójkę liczących przy pomocy wielokanałowego oscyloskopu.**

## 2.1 Licznik asynchroniczny

Licznik czterobitowy pozwala na przechowywanie liczby sygnałów w postaci liczby czterobitowej. Pozwala nam zatem przechowywać liczby od 0 do 15. Licznik asynchroniczny cechuje się tym, że każdy sygnał na wejściu zliczającym powoduje inkrementację lub dekrementację przechowywanej liczby.

Nasz licznik posiadać będzie jedno wejście zliczające oraz cztery wyjścia reprezentujące naszą czterobitową liczbę binarną.

## 2.2 Przygotowania

Poniżej widoczna tabela z kolejnymi liczbami, które powinien reprezentować nasz licznik:

Q0	Q1	Q2	Q3	System dziesiętny
1	1	1	1	15
1	1	1	0	14
1	1	0	1	13
1	1	0	0	12
1	0	1	1	11
1	0	1	0	10
1	0	0	1	9
1	0	0	0	8
0	1	1	1	7
0	1	1	0	6
0	1	0	1	5
0	1	0	0	4
0	0	1	1	3
0	0	1	0	2
0	0	0	1	1
0	0	0	0	0

Nietrudno zauważyc, że w każdym z bitów  $Q_0, Q_1, Q_2, Q_3$  występuje pewna zależność dotycząca zmian wartości logicznej. Można zauważyc, że częstotliwość zmian wartości logicznej w każdym kolejnym bicie jest dwa razy większa. Gdybyśmy reprezentowali liczby w odwrotnej kolejności to (od  $Q_3$  do  $Q_0$ ) to częstotliwość malała by 2 razy z każdym krokiem. To daje pomysł wykorzystania dwójkę liczących jako dzielników częstotliwości przez 2 w celu rozwiązania naszego problemu. W tym rozwiązańu posłużę się przerzutnikami typu  $D$  do stworzenia układu.

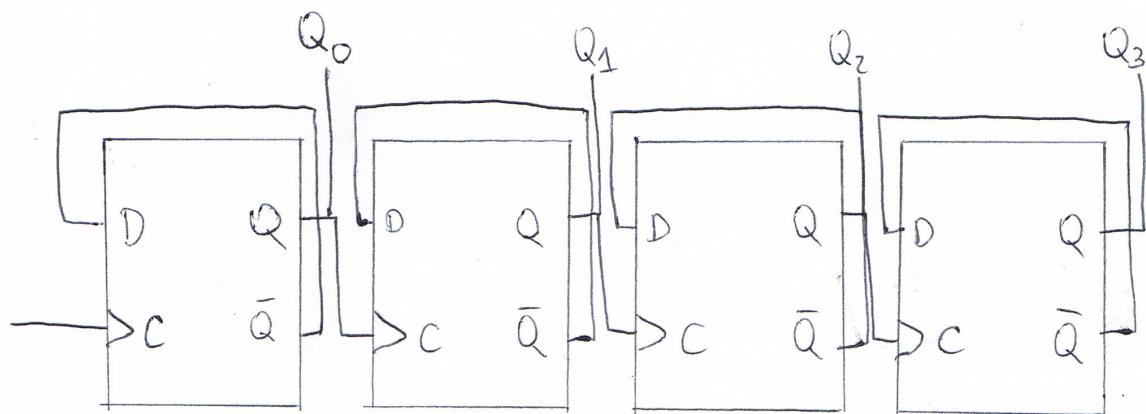
### 2.3 Projektowanie licznika czterobitowego

Nasz licznik przechowywać będzie bity „od tyłu”.

Żeby zmniejszyć częstotliwość dwa razy należy jako sygnał zegarowy w kolejnych przerzutnikach traktować wyjście poprzedniego - pomniejszonego dwa razy.

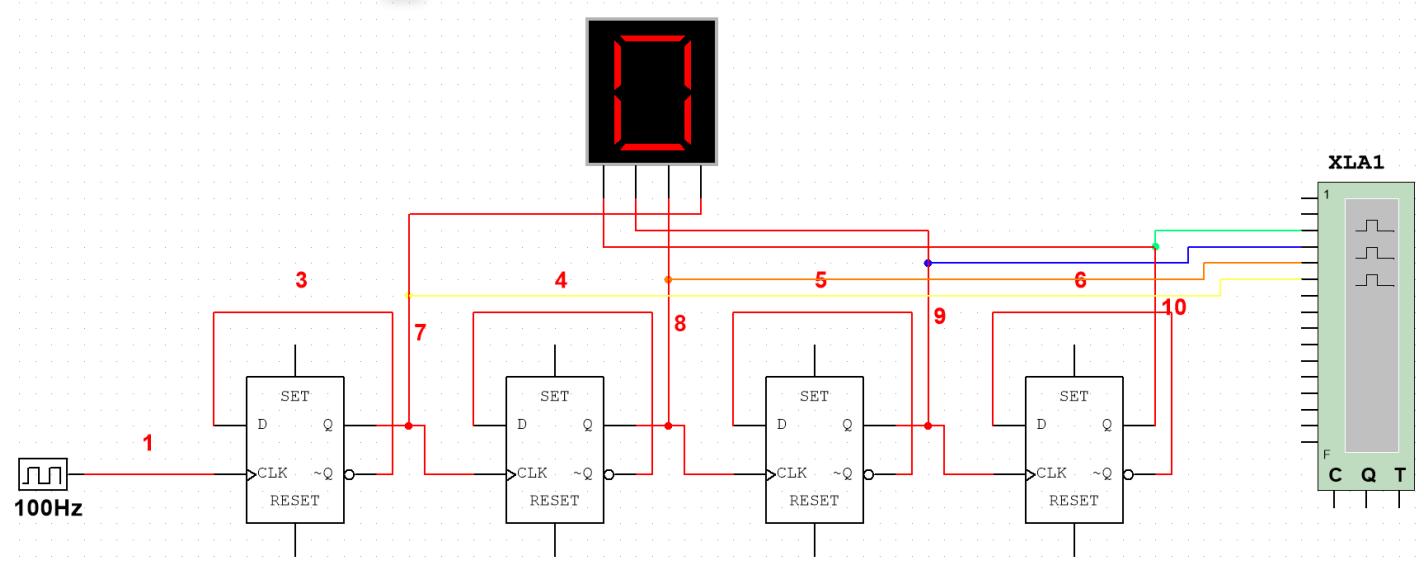
Ponieważ chcemy zmieniać bity to na wejściu  $D$  będziemy podawać wyjście  $\bar{Q}$  przerzutnika.

Nasz układ powinien wyglądać tak:



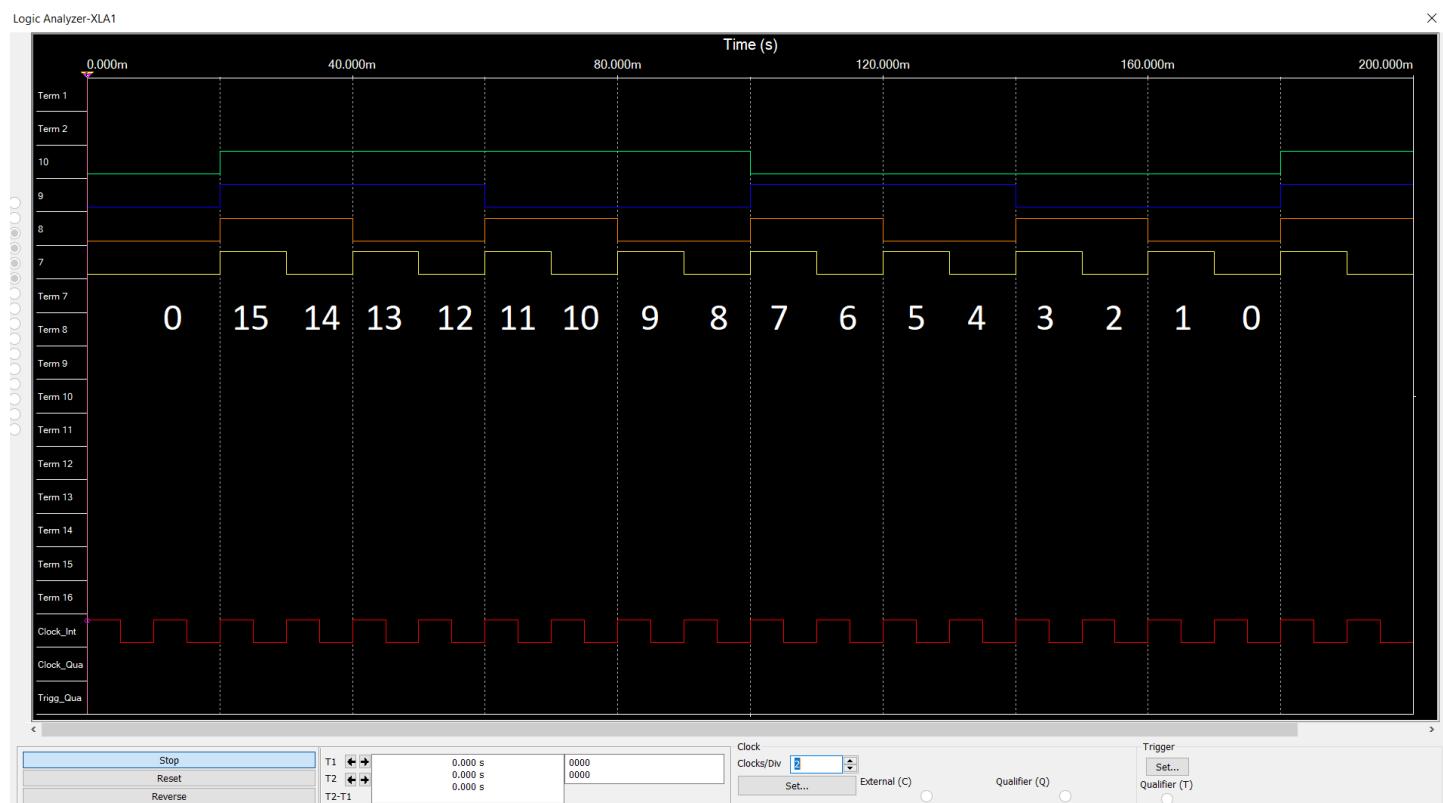
Gdzie przechowywanahaba to  $Q_3 Q_2 Q_1 Q_0$ .

## 2.4 Budowa układu w Multisimie



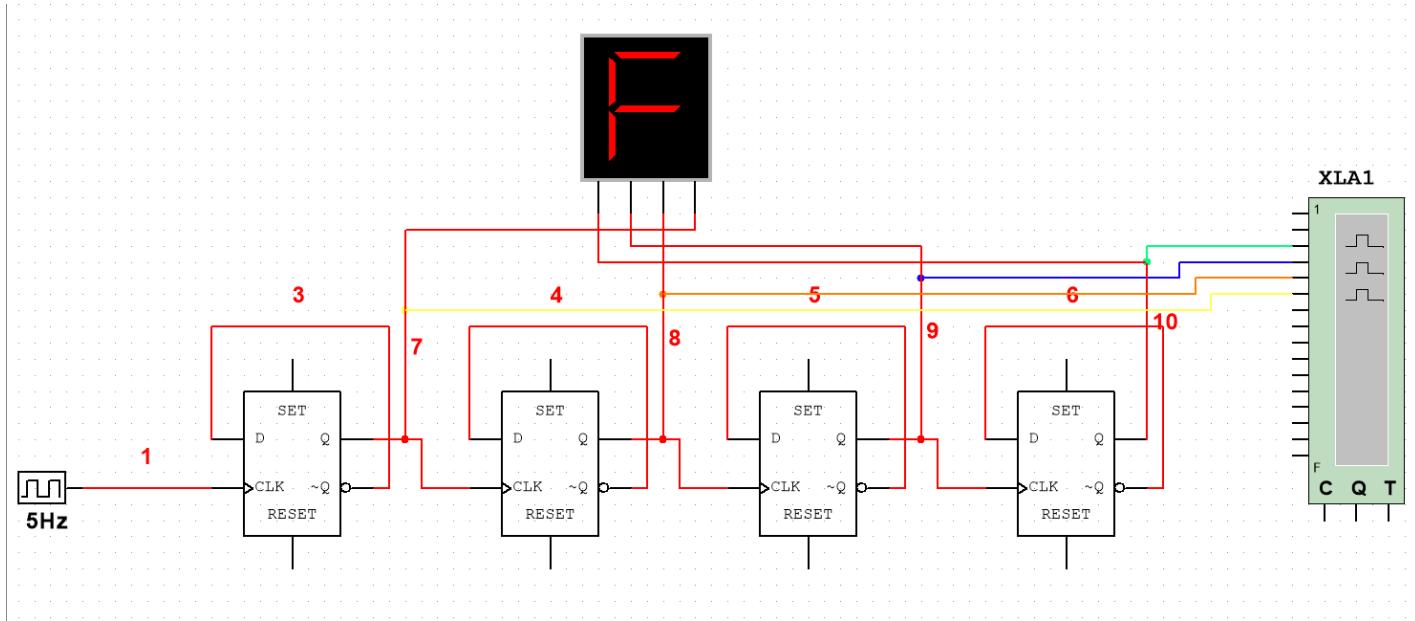
Schemat układu licznika czterobitowego (krok 1)

Zamiast oscyloskopu o czterech kanałach użyłem Logic Analyzer'a, który wydaje mi się dawać bardziej przejrzyste wyniki.

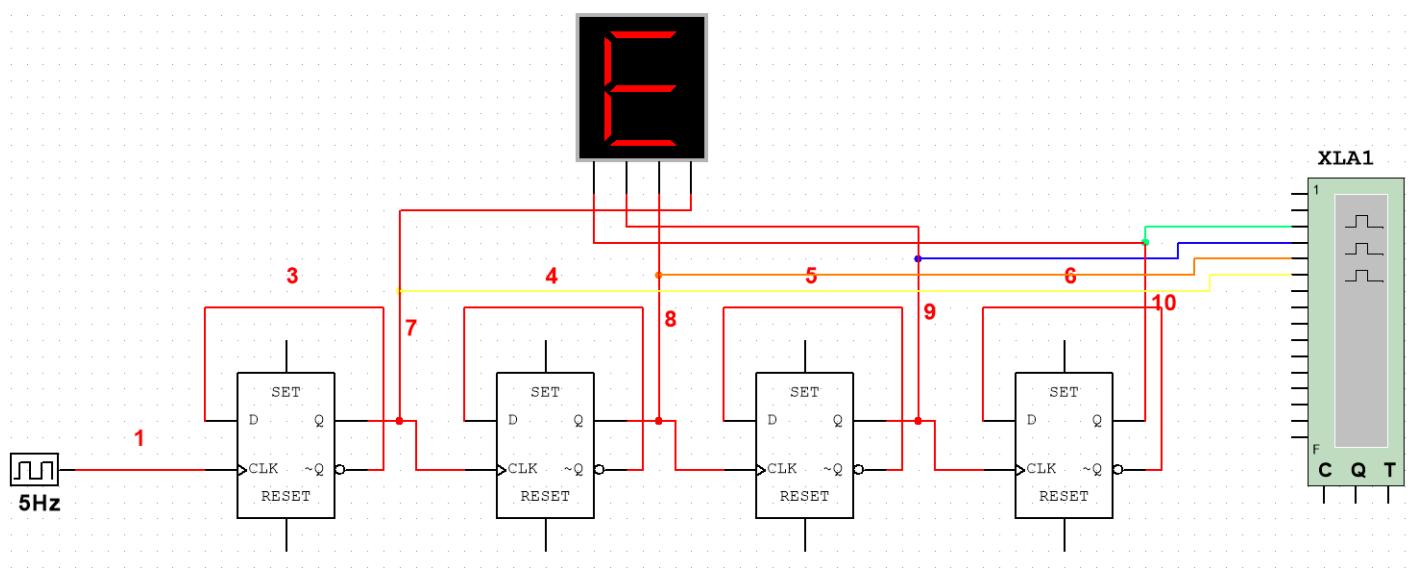


Wyniki Logic Analyzer'a

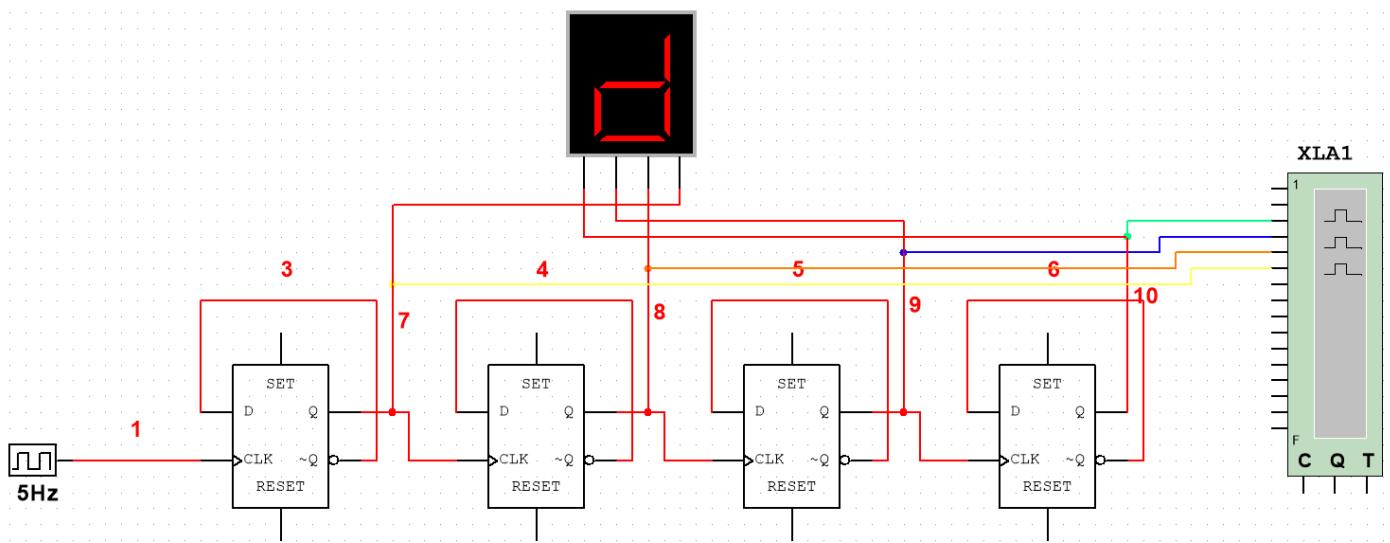
Kolejne bity symbolizowane są od góry do dołu, począwszy od bitu najbardziej znaczącego.



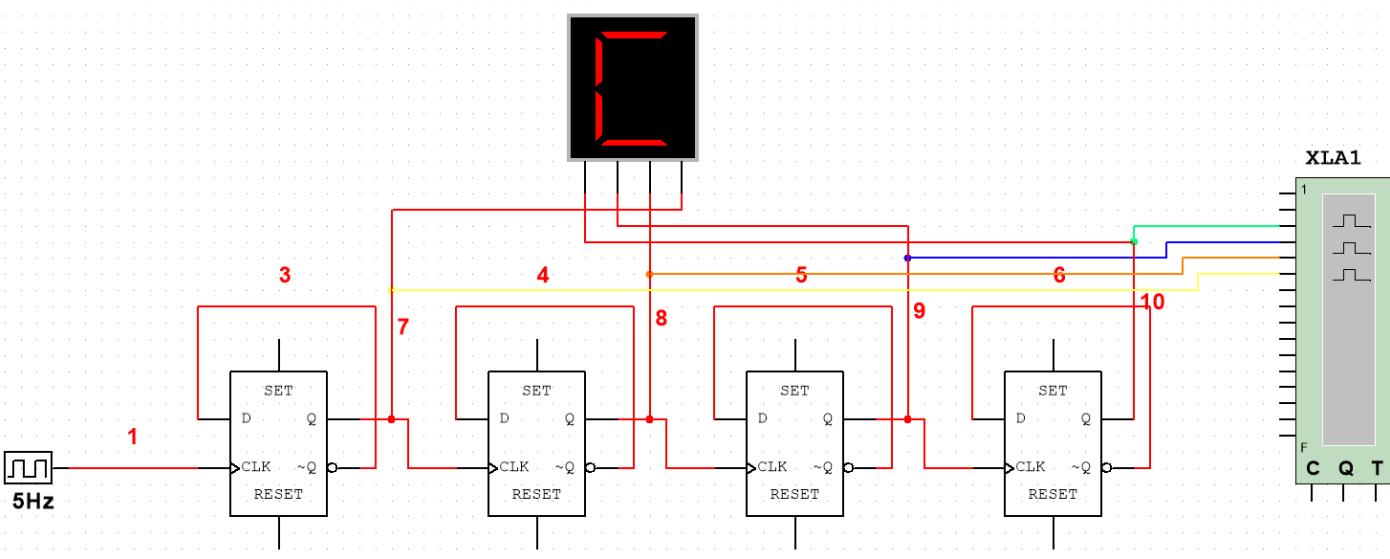
Schemat układu licznika czterobitowego (krok 2)



Schemat układu licznika czterobitowego (krok 3)



Schemat układu licznika czterobitowego (krok 4)



Schemat układu licznika czterobitowego (krok 5)

## 2.5 Wnioski

### 2.5.1 Czy układ działa poprawnie?

Przedstawione przeze mnie rozumowanie pozwoliło mi dojść do poprawnego wyniku końcowego, którym jest poprawnie zbudowany czterobitowy licznik asynchroniczny liczący wstecz. Przedstawione powyżej zdjęcia układu oraz Logic Analyzer'a pokazują, że faktycznie każdy bit na wyjściu zmieniał się z dwukrotnie większą częstotliwością co pozwoliło nam stworzyć licznik liczący wstecz poprzez odwrócenie kolejności bitów. Widać również, że licznik faktycznie przechowuje liczby od 15 do 0. Wszystkie założenia zostały spełnione zatem stwierdzam, iż układ został wykonany poprawnie.

### 2.5.2 Co można było zrobić inaczej?

- Do budowy licznika można było użyć innych przerzutników takich jak przerzutnik typu *JK* czy *T*.
- Zbudować czterobitowy licznik synchroniczny, w którym nie pojawiłoby się zjawisko hazardu spowodowane przełączaniem stanów przez każdy kolejny przerzutnik.
- Zbudować czterobitowy licznik asynchroniczny liczący do przodu. Wystarczy by kolejne przerzutniki jako sygnał zegarowy dostawały nie sygnał  $Q$  poprzedniego przerzutnika tylko sygnał  $\bar{Q}$
- Zastosować oscyloskop zamiast Logic Analyzer'a.
- Dodać możliwość zresetowania.

### 2.5.3 Gdzie to można zastosować?

- Minutniki kuchenne, alarmy, stopery, licznik kilometrów w samochodzie.
- Układ liczący wstecz nietrudno zaadaptować do układu liczącego wprzód.
- Dzielnik częstotliwości dla kolejnych potęg dwójkii.
- Licznik kroków np. w grze lub zegarku.
- W jakimkolwiek liczniku, który jest ograniczony z góry pewną liczbą.

### 3 Bazując na przerzutnikach $D$ , zaprojektować, zbudować i przetestować synchroniczny licznik modulo 8.

#### 3.1 Przygotowania

Do budowy licznika modulo 8 potrzebować będziemy conajmniej 3 przerzutników typu  $D$  (bo  $2^3 = 8$ ). Ponieważ nie da się na pierwszy rzut oka znaleźć zależności pomiędzy przerzutnikami pozwalającej zbudować taki licznik to posłużymy się tablicami Karnaugh'a. Zliczamy kolejno od 0 do 7 więc nasze zadanie możemy opisać pokazaną poniżej tablicą przejść ( $Q_0, Q_1, Q_2$  to kolejne bity wyjścia zaś  $D_0, D_1, D_2$  to wejścia kolejnych przerzutników):

$Q_2$	$Q_1$	$Q_0$		$Q_2'$	$Q_1'$	$Q_0'$
0	0	0	→	0	0	1
0	0	1	→	0	1	0
0	1	0	→	0	1	1
0	1	1	→	1	0	0
1	0	0	→	1	0	1
1	0	1	→	1	1	0
1	1	0	→	1	1	1
1	1	1	→	0	0	0

Tablica przejść pomiędzy stanami

Ponieważ przerzutnik typu  $D$  przepisuje stan wejścia  $D$  na stan wyjścia  $Q$  to tablica wzbudzeń wejść  $D_0, D_1, D_2$  wygląda następująco:

$D_2$	$D_1$	$D_0$
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1
0	0	0

Tablica wzbudzeń wejść przerzutników

Musimy teraz odnaleźć wzory na wejścia kolejnych przerzutników w zależności od wyjść  $Q_0, Q_1, Q_2$ . W tym celu użyjemy metody Karnaugh'a do minimalizowania funkcji logicznych:

$\cancel{Q_1 Q_0}$	00	01	11	10	
$\cancel{Q_2}$	0	1	0	0	1
	1	1	0	0	1

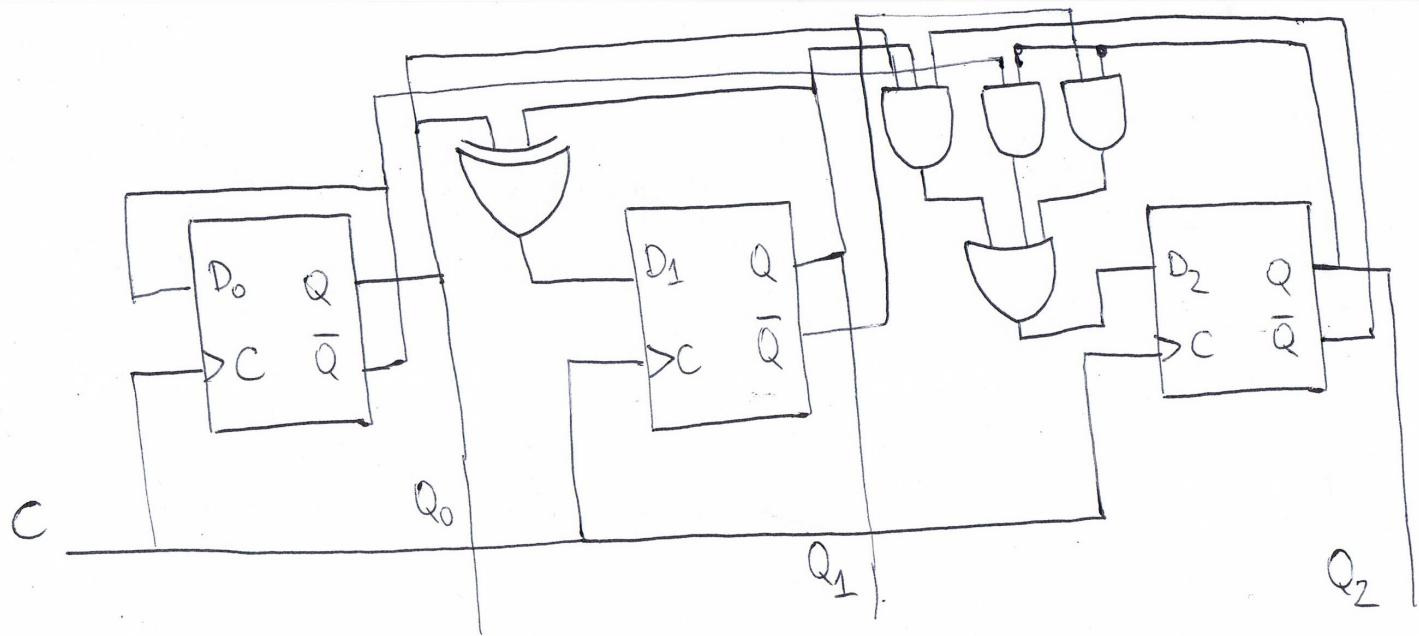
$$D_0 = \overline{Q_0}$$

$\cancel{Q_1 Q_0}$	00	01	11	10	
$\cancel{Q_2}$	0	0	1	0	1
	1	0	1	0	1

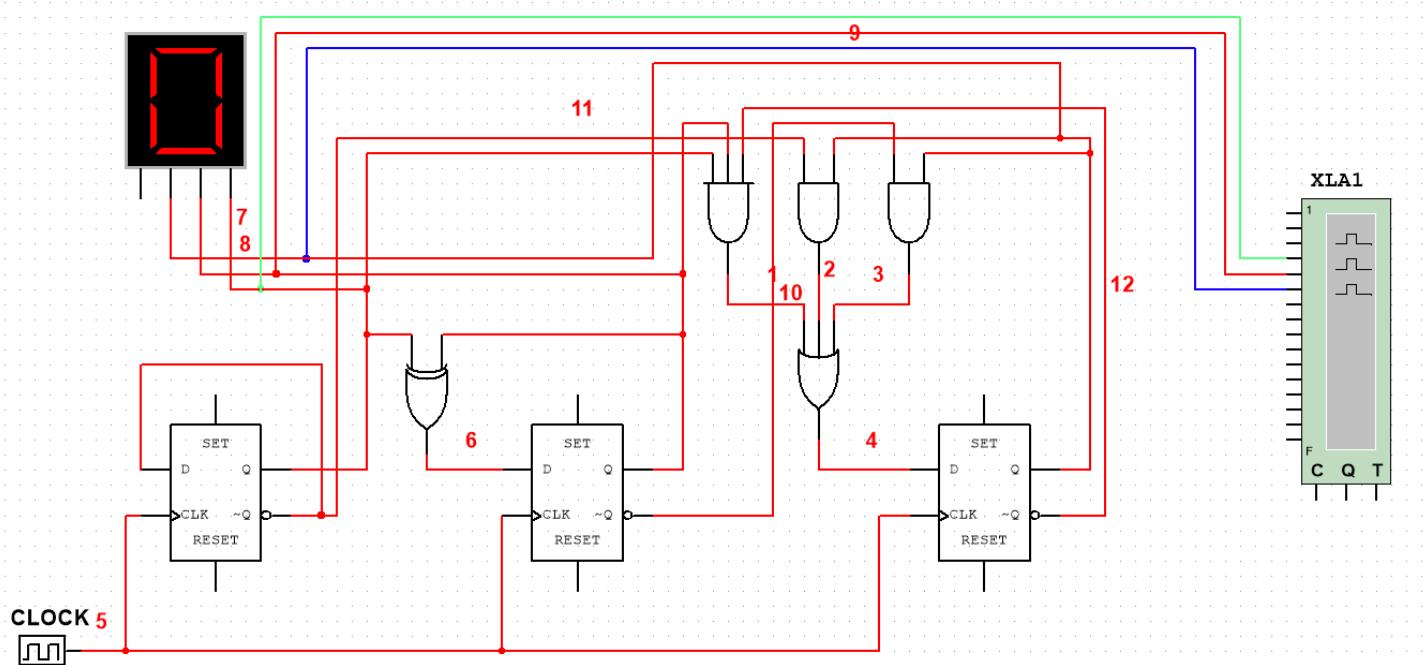
$$D_1 = Q_0 \overline{Q_1} + \overline{Q_0} Q_1 = Q_0 \text{ xor } Q_1$$

$\cancel{Q_1 Q_0}$	00	01	11	10	
$\cancel{Q_2}$	0	0	0	1	0
	1	1	1	0	1

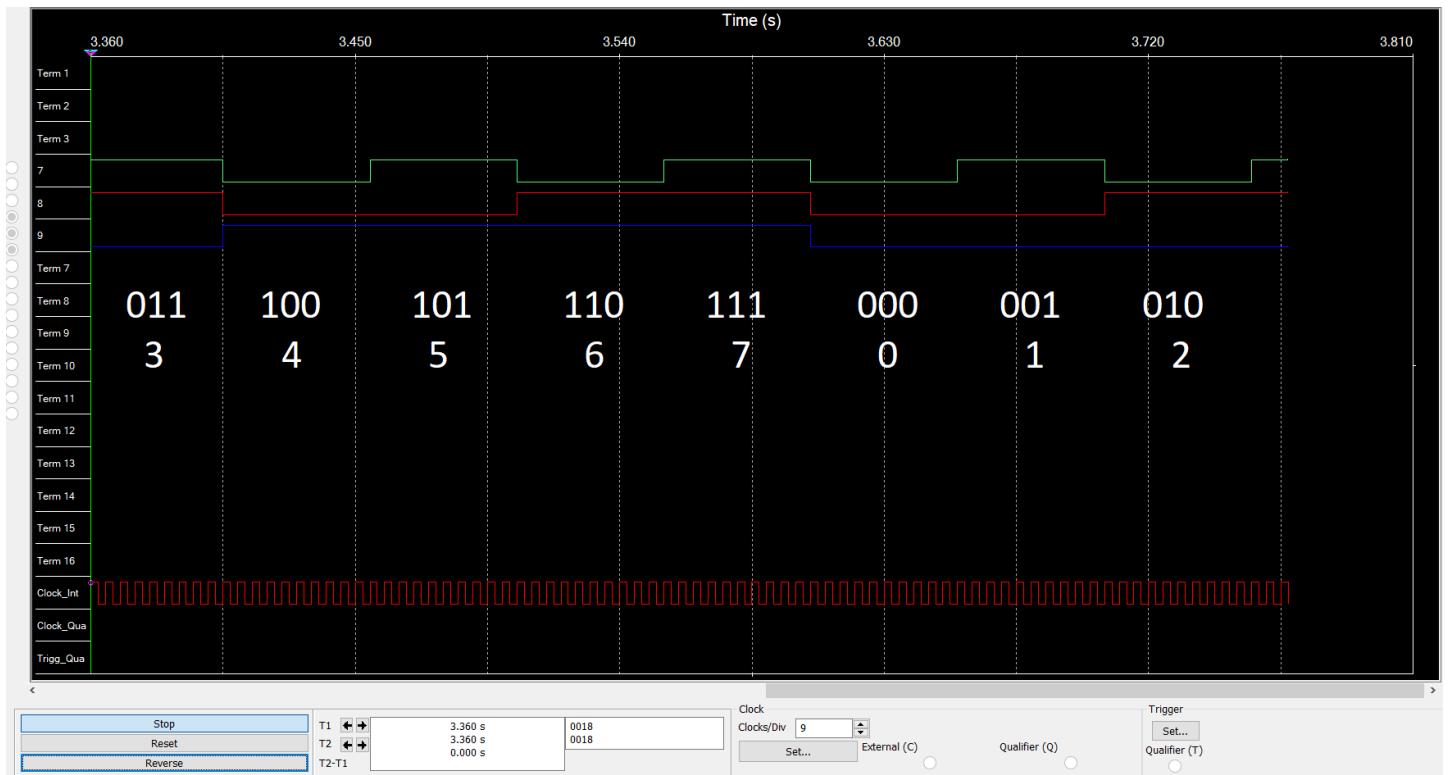
$$D_2 = \overline{Q_0} Q_2 + \overline{Q_1} Q_2 + Q_0 Q_1 \overline{Q_2}$$



### 3.2 Budowa układu w Multisimie



Schemat układu licznika modulo 8



Wyniki Logic Analyzer'a wraz z naniesionymi liczbami przechowywanymi w liczniku  
Bity ułożone są od dołu do góry od najbardziej znaczącego bitu

### 3.3 Wnioski

#### 3.3.1 Czy układ działa poprawnie?

Przedstawione przeze mnie rozumowanie pozwoliło mi dojść do poprawnego wyniku końcowego, którym jest poprawnie zbudowany licznik modulo 8. Przedstawione powyżej zdjęcia układu oraz Logic Analyzer'a pokazują, że faktycznie licznik przechowuje kolejne liczby modulo 8 (od 0 do 7). Wszystkie założenia zostały spełnione zatem stwierdzam, iż układ został wykonany poprawnie.

#### 3.3.2 Co można było zrobić inaczej?

- Do budowy licznika można było użyć innych przerzutników takich jak przerzutnik typu *JK*.
- Porównać wyniki z gotowym licznikiem modulo 8
- Zbudować licznik asynchroniczny modulo 8, który nie wymagałby bramek logicznych
- Zbudować licznik o większej ilości bramek (bez minimalizacji metodą Karnaugh'a)
- Zastosować produkt sum dla metody Karnaugh'a
- Dodać możliwość zresetowania
- Dodać więcej zdjęć z testowania układu

#### 3.3.3 Gdzie to można zastosować?

- Liczniki towarów np. w fabrykach
- Licznik kolorów w myszce komputerowej
- Numery trybów np. w pralce albo mikrofalówce
- Licznik rozkazów w mikrokontrolerze
- W protokołach szeregowego przesyłania danych, jako odmierzacz czasu przy generacji sygnału zegarowego

## 4 Bazując na dowolnym liczniku czterobitowym zliczającym wprzód, zrealizować licznik modulo 6.

### 4.1 Przygotowania

Każdy licznik czterobitowy potrafi przechowywać liczby od 0 do 15. Nas jednak interesują jedynie liczby od 0 do 5 (modulo 6). Moglibyśmy zatem użyć licznika wprzód, który resetuje się dla wartości 6. Bezpieczniej jest jednak przyjąć, że licznik może mieć początkową wartość z przedziału 0 – 15 i resetować licznik dla każdej wartości większej bądź równej 6. Tabela prawdy dla resetowania licznika wygląda następująco:

Q3	Q2	Q1	Q0	Reset
0	0	0	0	0
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	0
0	1	0	1	0
0	1	1	0	1
0	1	1	1	1
1	0	0	0	1
1	0	0	1	1
1	0	1	0	1
1	0	1	1	1
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	1

Tabela prawdy dla resetowania licznika

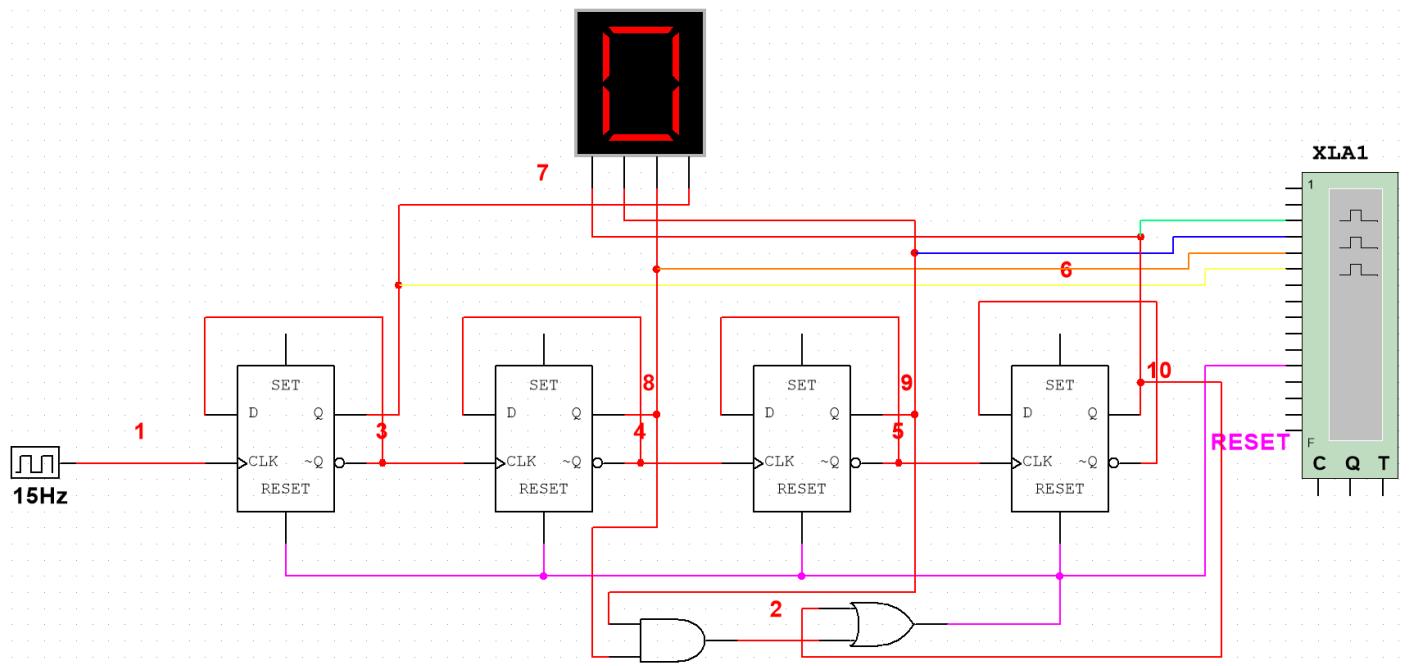
Wzór logiczny na resetowanie licznika możemy znaleźć używając metody Karnaugh'a

<del><math>Q_1 Q_0</math></del>	00	01	11	10
$Q_3 Q_2$	00	0	0	0
01	0	0	1	1
11	1	1	1	1
10	1	1	1	1

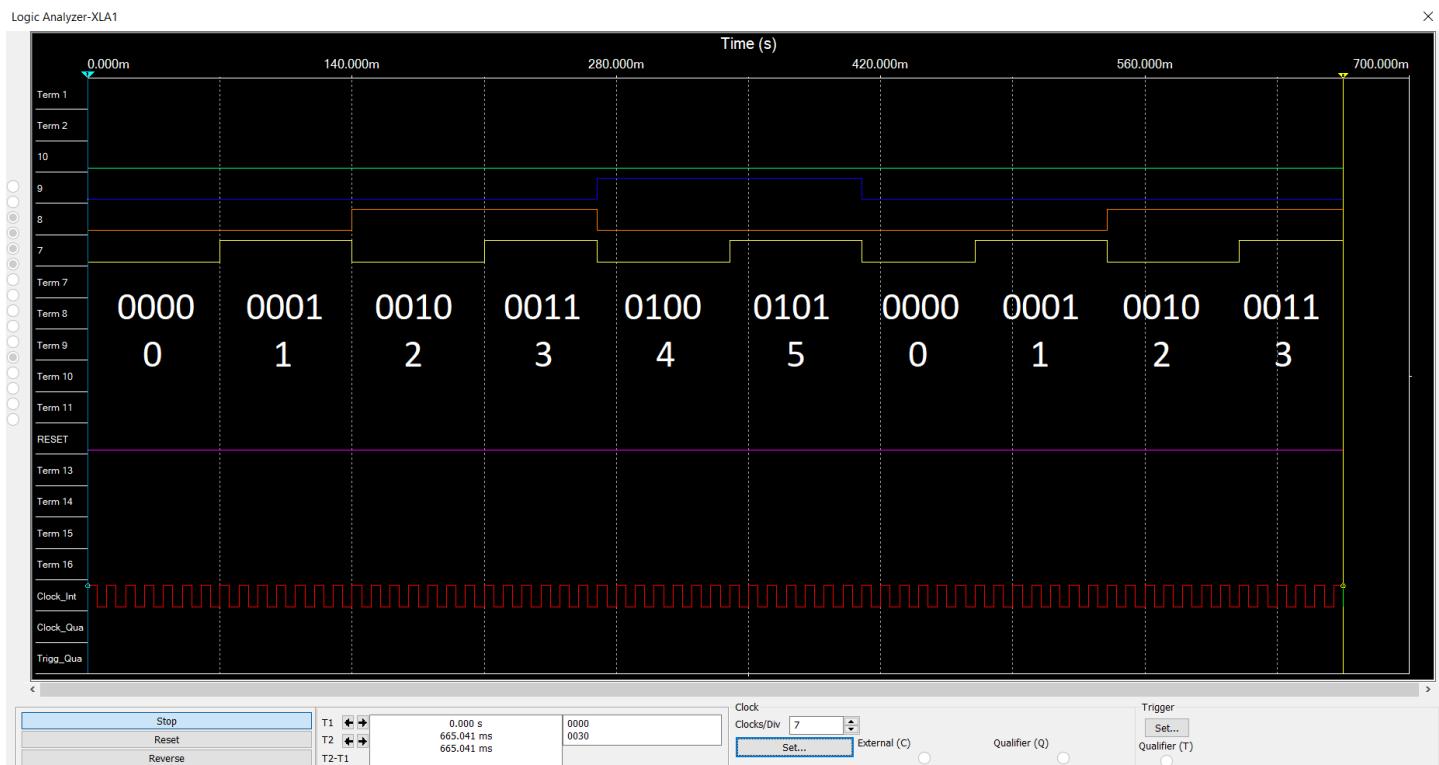
$$\text{Reset} = Q_3 + Q_1 Q_2$$

Jako czterobitowy licznik wprzód możemy wykorzystać nasz układ z zadania 2 wraz z jednym z wniosków dotyczącym zamiany go na licznik liczący wprzód. Wystarczy zmienić odpowiednie wejścia sygnału zegarowego w kolejnych przerzutnikach - zamiast  $Q$  powinien do nich wchodzić sygnał  $\bar{Q}$  poprzedniego przerzutnika. Musimy również dodać możliwość resetowania układu zgodnie z wyznaczoną przez nas funkcją logiczną powyżej.

## 4.2 Budowa układu w Multisimie



Schemat układu licznika modulo 6



Wyniki Logic Analyzer'a wraz z naniesionymi liczbami przechowywanymi w liczniku  
Bity ułożone są od góry do dołu od najbardziej znaczącego bitu

## 4.3 Wnioski

### 4.3.1 Czy układ działa poprawnie?

Przedstawione przeze mnie rozumowanie pozwoliło mi dojść do poprawnego wyniku końcowego, którym jest poprawnie zbudowany licznik modulo 6 przy pomocy czterobitowego licznika wprzód. Jest to oczywiście licznik asynchroniczny. Przedstawione powyżej zdjęcia układu oraz Logic Analyzer'a pokazują, że faktycznie licznik przechowuje kolejne liczby modulo 6 (od 0 do 5). Wszystkie założenia zostały spełnione zatem stwierdzam, iż układ został wykonany poprawnie.

### 4.3.2 Co można było zrobić inaczej?

- Zastosować licznik trzybitowy - byłoby bardziej oszczędnie i wydajnie
- Do budowy licznika można było użyć licznika czterobitowego zbudowanego z innych przetwarzających np. *JK* lub *T*
- Do budowy licznika można było użyć licznika czterobitowego liczącego wstecz
- Licznik modulo 6 mógł być liczący wstecz
- Porównać wyniki z gotowym licznikiem modulo 6
- Zbudować synchroniczny licznik modulo 6
- Zbudować licznik o większej ilości bramek (bez minimalizacji metodą Karnaugh'a)
- Zastosować produkt sum dla metody Karnaugh'a
- Dodać więcej zdjęć z testowania układu

### 4.3.3 Gdzie to można zastosować?

- Zastosowane tutaj rozwiązanie pozwala nam na stworzenie dowolnego licznika modulo  $n$  poprzez resetowanie licznika dla odpowiednich wartości. Dla licznika modulo  $n$  należy zastosować licznik  $m$ -bitowy, gdzie  $m = \lceil \log_2 n \rceil$
- Pomiar czasu, stopery, zegarki
- Dzielniki częstotliwości
- Licznik kroków np. w grze lub zegarku.
- W jakimkolwiek liczniku, który jest ograniczony z góry pewną liczbą.