# Technika Cyfrowa.
# Ćwiczenie 3.

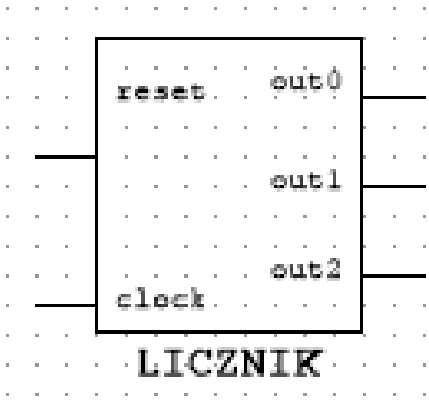
Maciej Pieta Piotr Koproń Jakub Woś
Rafał Piwowar

Marzec 2023

# 1 Zadanie 3a

**Treść zadania.** Bazując na dowolnie wybranych przerzutnikach, zaprojektować, zbudować i przetestować synchroniczny trzybitowy licznik liczący w następujący sposób: 0, 2, 4, 6, 1, 3, 5, 7, 0, 2 , 4 , ... itd. Inaczej mówiąc: licznik najpierw przechodzi po wartościach parzystych, a potem po wartościach nieparzystych, i znowu po parzystych, i tak w kółko.

## 1.1 Ogólna idea rozwiązania




Licznik jest trzybitowy - wartość wyjściowa jest definiowana jako $1 * out_0 + 2 * out_1 + 4 * out_2$. Jako wartości wejściowe przyjmujemy zewnętrzny zegar oraz sygnał resetujący.

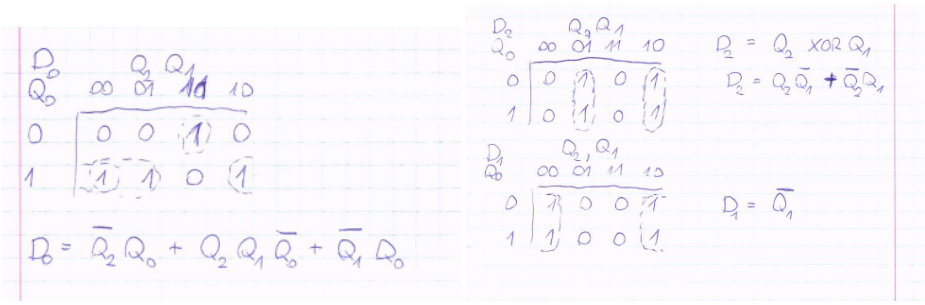
## 1.2 Tabele prawdy

Tabele prawdy informują o kolejnej wartości wysyłanej przez licznik, wyznaczone z wartości aktualnej.

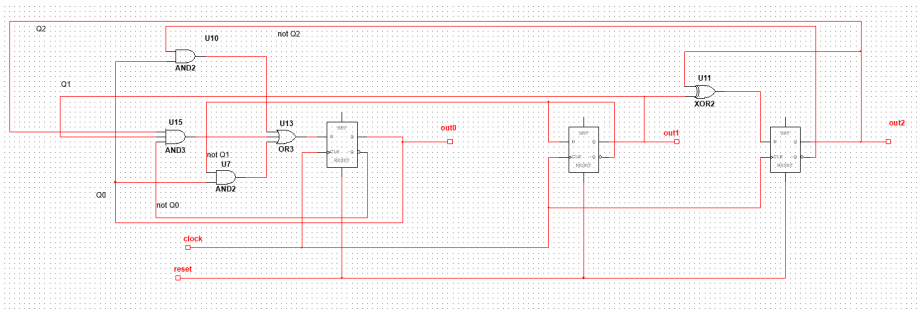
| | A | B | C | D | E | F | G | H | I | J |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | | | OBECNIE | | | | NASTĘPNY | | | |
| 2 | wartość | | Q2 | Q1 | Q0 | | Q2 | Q1 | Q0 | |
| 3 | 0 | | 0 | 0 | 0 | | 0 | 1 | 0 | |
| 4 | 2 | | 0 | 1 | 0 | | 1 | 0 | 0 | |
| 5 | 4 | | 1 | 0 | 0 | | 1 | 1 | 0 | |
| 6 | 6 | | 1 | 1 | 0 | | 0 | 0 | 1 | |
| 7 | 1 | | 0 | 0 | 1 | | 0 | 1 | 1 | |
| 8 | 3 | | 0 | 1 | 1 | | 1 | 0 | 1 | |
| 9 | 5 | | 1 | 0 | 1 | | 1 | 1 | 1 | |
| 10 | 7 | | 1 | 1 | 1 | | 0 | 0 | 0 | |
| 11 | | | | | | | | | | |

## 1.3 Tabele Karnaugh

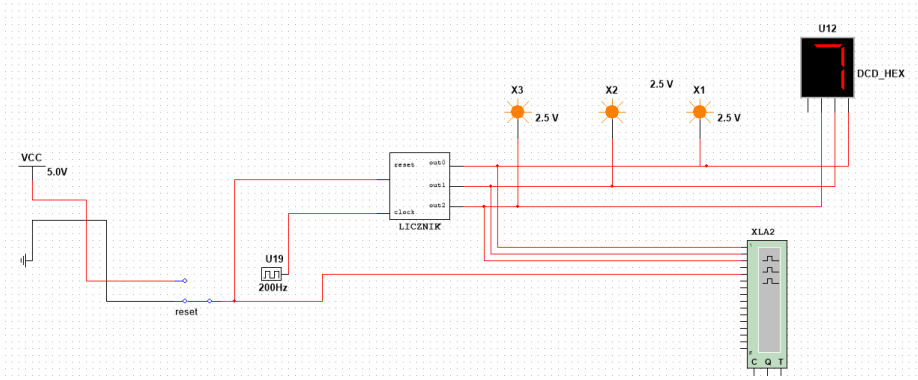
Dokonujemy minimalizacji, w celu wyznaczenia bezpośrednich wzorów na kolejne wartości, oznaczone $D_0, D_1, D_2$.




## 1.4 Schemat układu

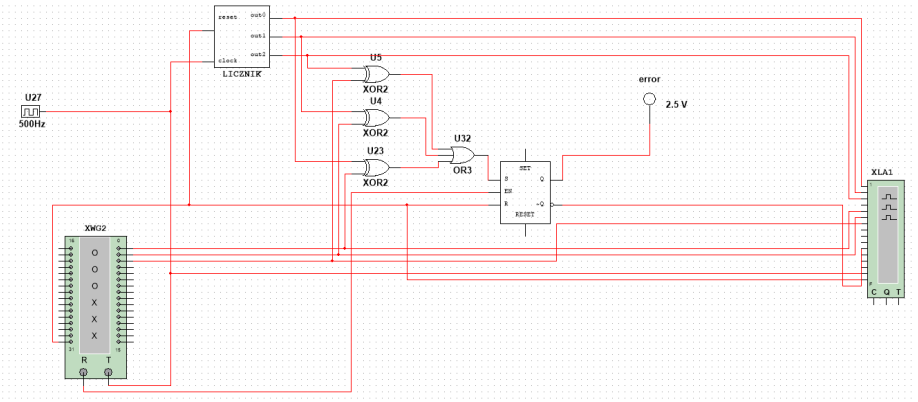



Dodatkowo załączamy układ wizulalizacjy działanie naszego licznika.

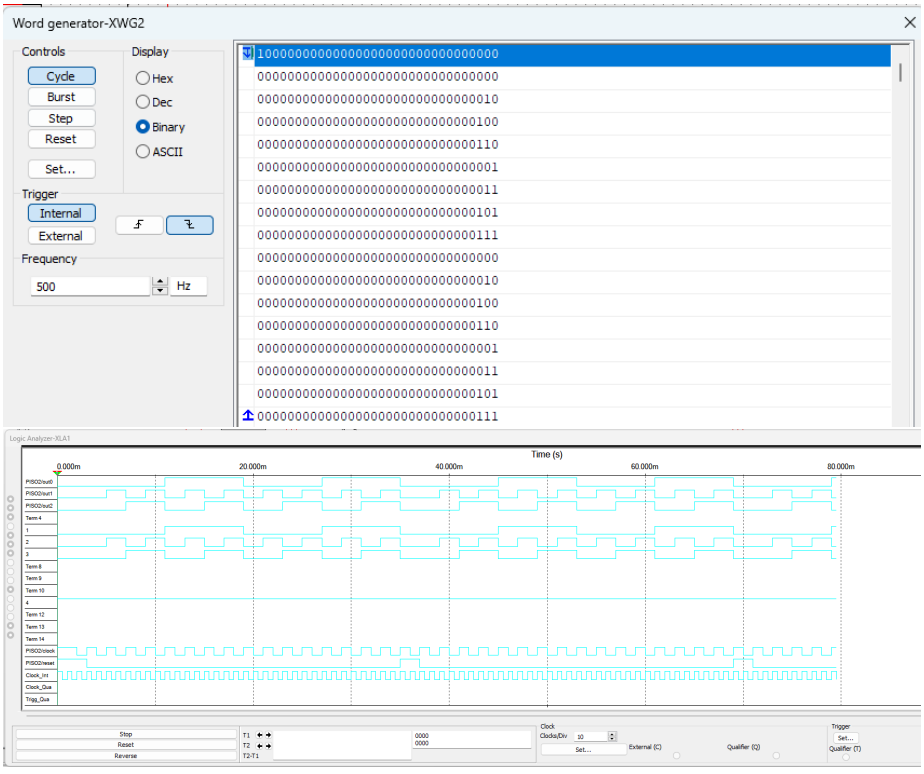


## 1.5 Układ testujący

Przygotowaliśmy automatyczny model testujący. Lampka błędu pozostanie zgaszona tylko jeżeli we wszystkich sytuacjach licznik zachowa się zgodnie z oczekiwaniami.




Ustawienia generatora słów i wyniki analizatora logicznego.



Wartość oznaczona "4" stale ja 1 oznacza że lampka błędu nie świeci, czyli program działa.

### 1.6 Wnioski

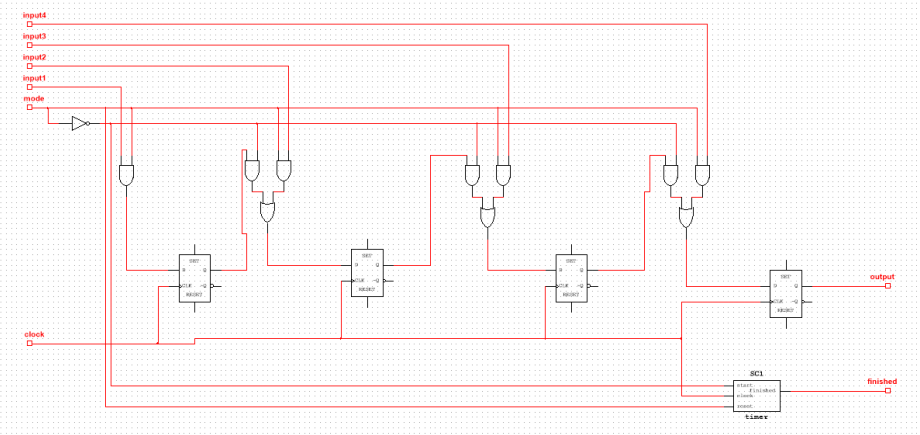
**Alternatywne rozwiązania**

**Zastosowania** Licznik może być wykorzystany do synchronizacji sygnalizacji świetlnej interskrzyżowaniowo, w celu zapewnienia optymalnych warunków jazdy dla kierowców jadących zgodnie z ogarniczeniami prędkości. (Jak jedzie przepisowo, to cały czas będzie miał zielone, jak nie - to i tak będzie musiał hamować na czerwonym). RYSUNEK DO TEGO.

## 2 Zadanie 3b

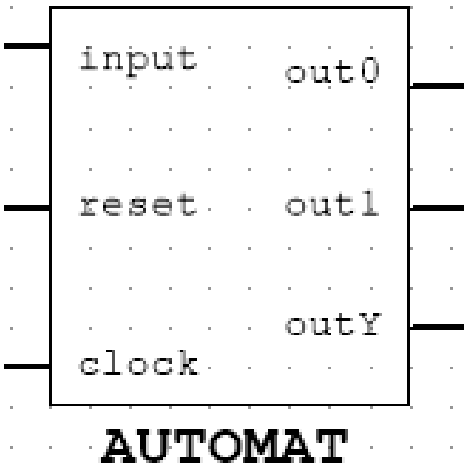
**Treść zadania** Bazując na przerzutnikach "D", zaprojektować, zbudować i przetestować automat realizujący detekcję wprowadzanej na jego wejście czterobitowej wartości. Automat powinien rozpoznawać liczbę binarną: "1101". Jako źródło wprowadzanej wartości proszę użyć układu zbudowanego w ramach ćw.2b.

**Układ zbudowany w ramach ćwiczenia 2b** Zgodnie z komentarzem zwrotnym do ćwiczenia 2b, zbudowany przez nas układ PISO wymagał pewnych poprawek. Dla klaryfikacji, załączamy tutaj poprawioną wersję.

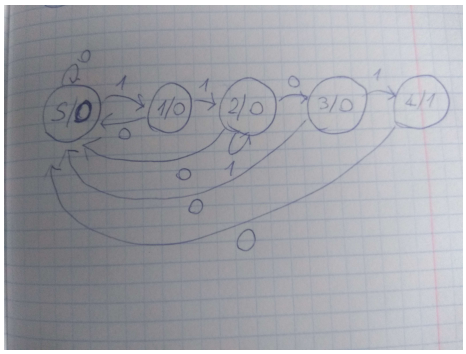



### 2.1 Ogólna idea rozwiązania

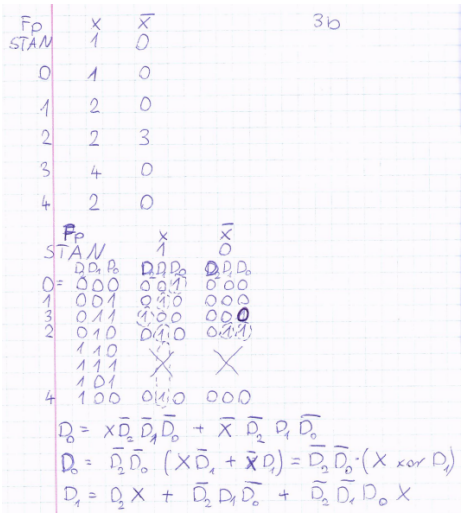
FIXME: Add Wejścia/Wyjścia.



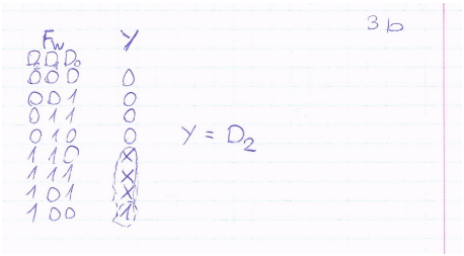

Implementujemy następujący automat:



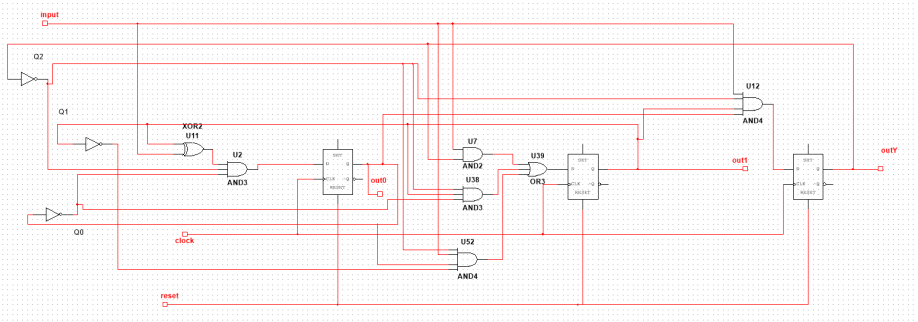
## 2.2 Funkcje przejścia stanu



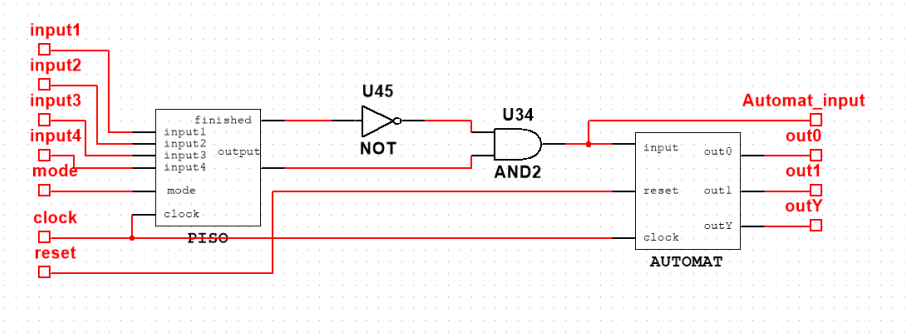

## 2.3 Funkcja wyjścia




## 2.4 Schemat układu

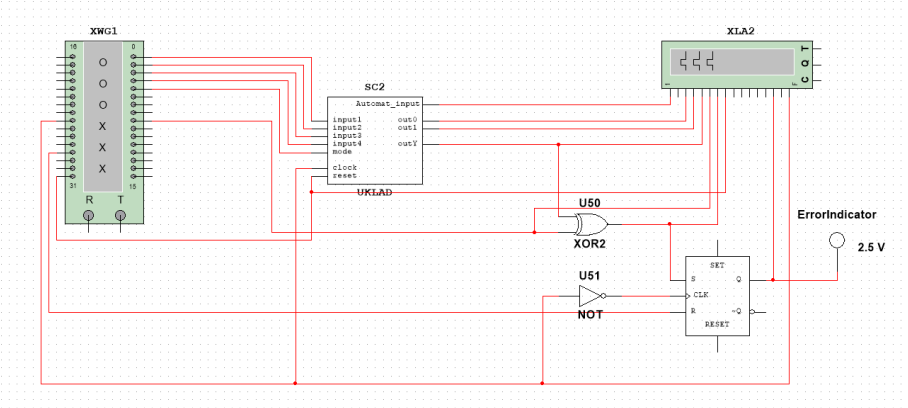



W połączeniu z układem PISO:




## 2.5 Układ testujący

Przygotowaliśmy automatyczny model testujący. Lampka błędu pozostanie zgaszona tylko jeżeli we wszystkich sytuacjach licznik zachowa się zgodnie z oczekiwaniami.




Załączamy kod wykorzystany do utworzenia konfiguracji generatora słów.

```
output = []

def toBinary(val):
    res = []
    while val != 0:
        res.insert(0, val % 2)
        val = val // 2

    while len(res) < 4:
        res.insert(0, 0)

    return res
```

```
def hex4(val):
    res = hex(val)[2:]
    while len(res) < 4:
        res = "0" + res
    return res

def addOutput(data):
    output.append(data[:1] + '1' + data[2:])
    output.append(data[:1] + '0' + data[2:])

output.append("Data:\n")
addOutput("10000000\n")
for i in range(16):
    addOutput("80000000\n")
    addOutput('0000001' + hex(i)[2:] + '\n')
    binary = toBinary(i)
    for k in range(4):
        addOutput('00000000' + '\n')
    addOutput('00000' + str(int(i == 13)) + '00' + '\n')

output.append("Initial:" + '\n')
output.append("0000" + '\n')
output.append("Final:" + '\n')
output.append(hex4(224) + '\n')

with open('output.dp', 'w') as outputFile:
    outputFile.writelines(output)
```

Rezultaty z analizatora logicznego. Linia "20" stale na 1 oznacza brak błędu.

## 2.6 Wnioski

**Alternatywne rozwiązania**

**Zastosowania**