Sprawozdanie 1: Bramki i funkcje logiczne

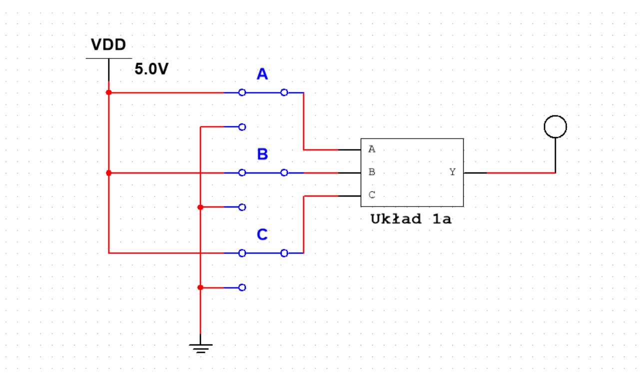
Jakub Szymczak, Damian Tworek,
Maksymilian Sulima, Łukasz Wala
AGH, Wydział Informatyki, Elektroniki i Telekomunikacji
Technika Cyfrowa 2021/2022

Kraków, 13 czerwca 2022

1 Ćwiczenie *1a*

Ideą ćwiczenia jest zaprojetowanie, zbudowanie i przetestowanie układu realizującego funkcję logiczną:

$$\mathbf{Y} = \overline{\mathbf{A}}\mathbf{C} + \mathbf{B}(\mathbf{A} + \mathbf{B})$$



Rysunek 1: Układ 1a

1.1 Rozwiązanie teoretyczne

Rozpocznijmy od przekształcenia funkcji logicznej do postaci zawierającej tylko koniunkcje i zaprzeczenia. Korzystając z prawa rozdzielności koniunkcji względem alternatywy

$$\mathbf{Y} = \overline{\mathbf{A}}\mathbf{C} + \mathbf{B}\mathbf{A} + \mathbf{B}\mathbf{B}$$

Tutaj można zauważyć, że $\mathbf{BB} = \mathbf{B}$, więcej

$$\mathbf{Y} = \overline{\mathbf{A}}\mathbf{C} + \mathbf{BA} + \mathbf{B}$$

Ponownie korzystając z prawa rozdzielności koniunkcji względem alternatywy

$$\mathbf{Y} = \overline{\mathbf{A}}\mathbf{C} + \mathbf{B}(\mathbf{A} + \mathbf{1})$$

Tutaj $\mathbf{A} + \mathbf{1} = \mathbf{1}$, więc

$$\mathbf{Y} = \overline{\mathbf{A}}\mathbf{C} + \mathbf{B}$$

Używając prawa de Morgana

$$\mathbf{Y} = \overline{\overline{\overline{\mathbf{A}}\mathbf{C}}\,\overline{\mathbf{B}}}$$

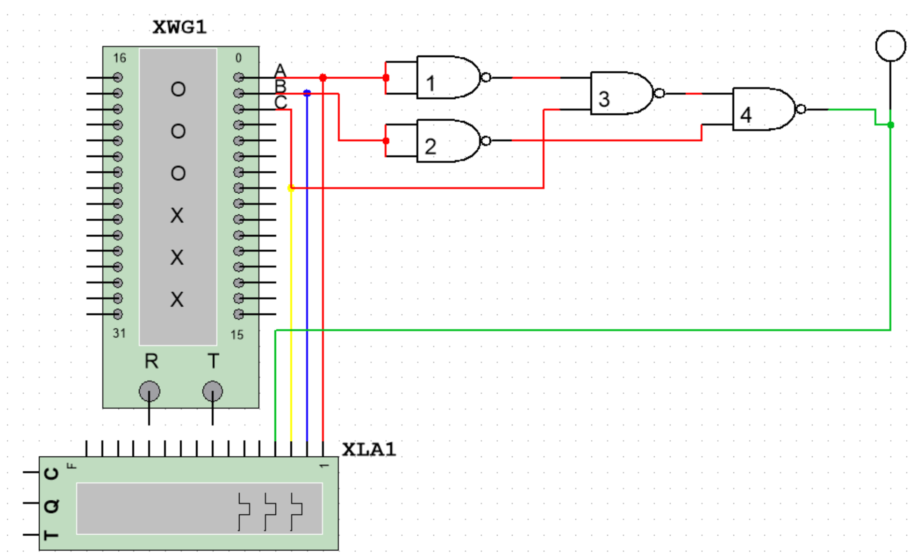
1.2 Implementacja w programie Multisim

Do zbudowania układu wykorzystano 4 bramki NAND, diodę LED, generator słów oraz analizator logiczny. Przedstawiony jest na **rysunku 2**. Na podstawie prawa de Morgana oraz prawa rozdzielności alternatywy względem koniunkcji

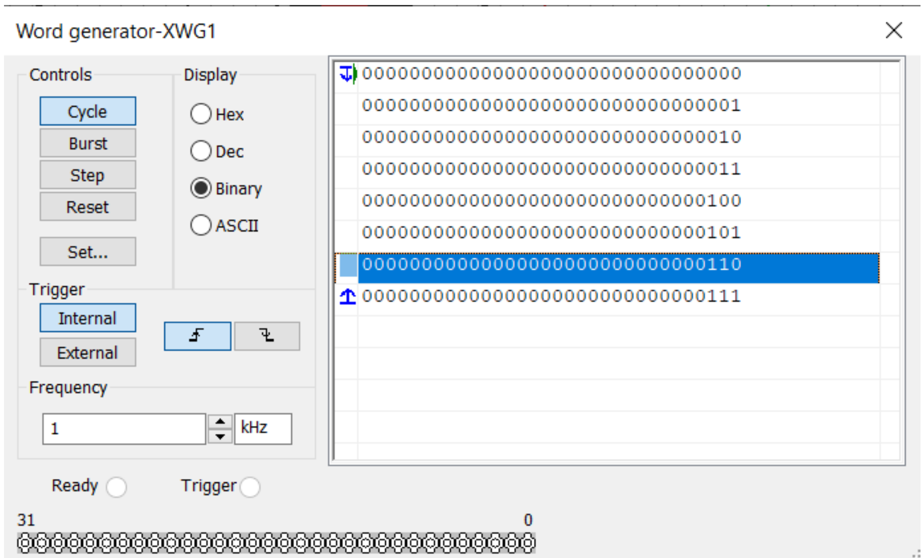
$$\overline{\mathbf{AA}} = \overline{\mathbf{A}} + \overline{\mathbf{A}} = \overline{\mathbf{A}}(\mathbf{1} + \mathbf{1}) = \overline{\mathbf{A}}$$

więc bramka NAND może posłużyć jako NOT, jeżeli do obu wejść podany zostanie ten sam sygnał.

$$\mathbf{Y} = \overline{\overline{\overline{\mathbf{A}}^{\mathbf{1}}\mathbf{C}}^{\mathbf{3}}\,\overline{\mathbf{B}}^{\mathbf{2}}}^{4}$$

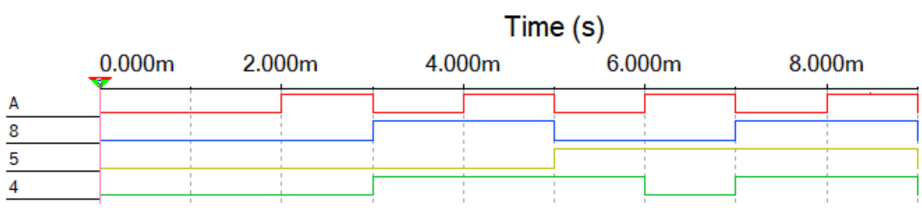


Rysunek 2: Układ logiczny w programie Multisim na podstawie przekształconej funkcji



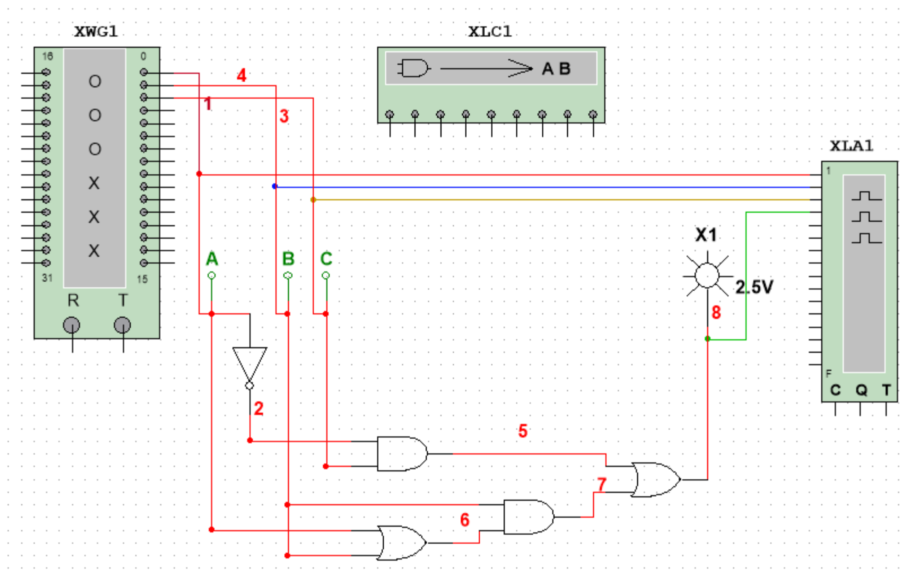
Rysunek 3: Generator słów dla pierwszego układu

Za pomocą analizatora logicznego zbadane zostały przebiegi czasowe badanego układu przedstawione na **rysunku 4** (**A** - czerwony, **B** - niebieski, **C** - żółty, **Y** - zielony).

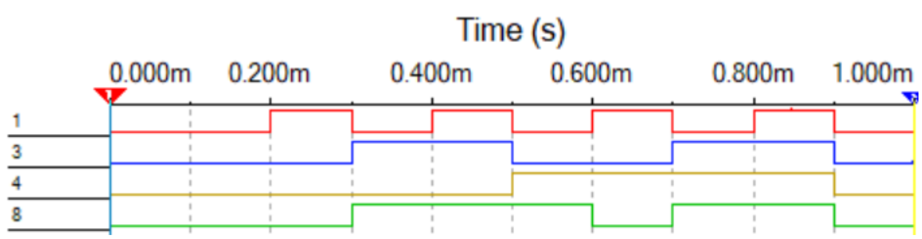


Rysunek 4: Wykres przebiegu czasowego badanego układu

Ponizej na **rysunkach 5 i 6** automatycznie wygenerowany układ na podstawie funkcji przed przekształceniami, wyniki są zgodne dla obu wariantów, co potwierdza poprawność funkcji po przekształceniu.

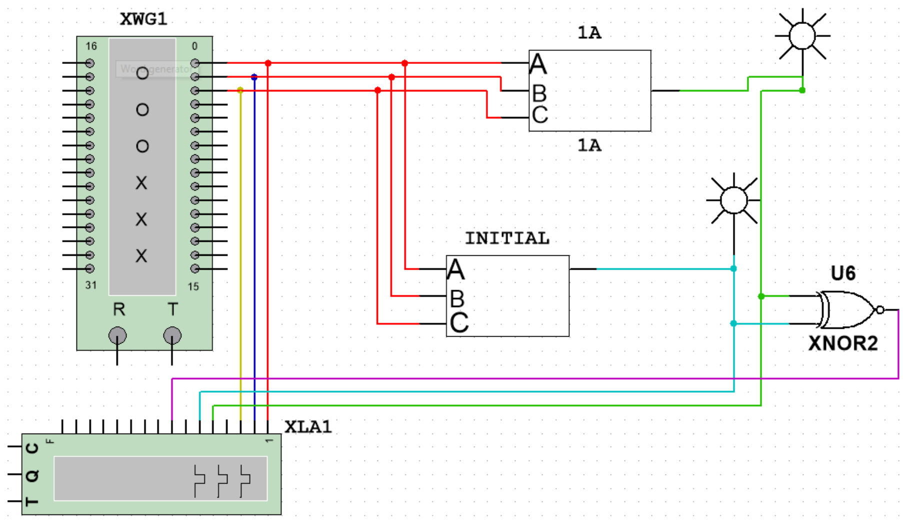


Rysunek 5: Układ logiczny wygenerowany przez Multisim na podstawie funkcji przed przekształceniami

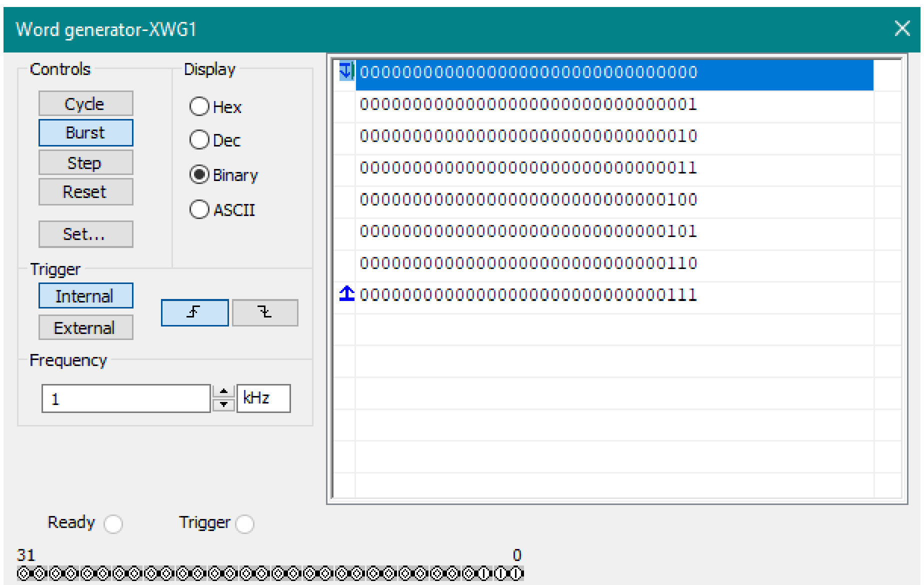


Rysunek 6: Wykres przebiegu czasowego badanego układu (przed przekształceniami)

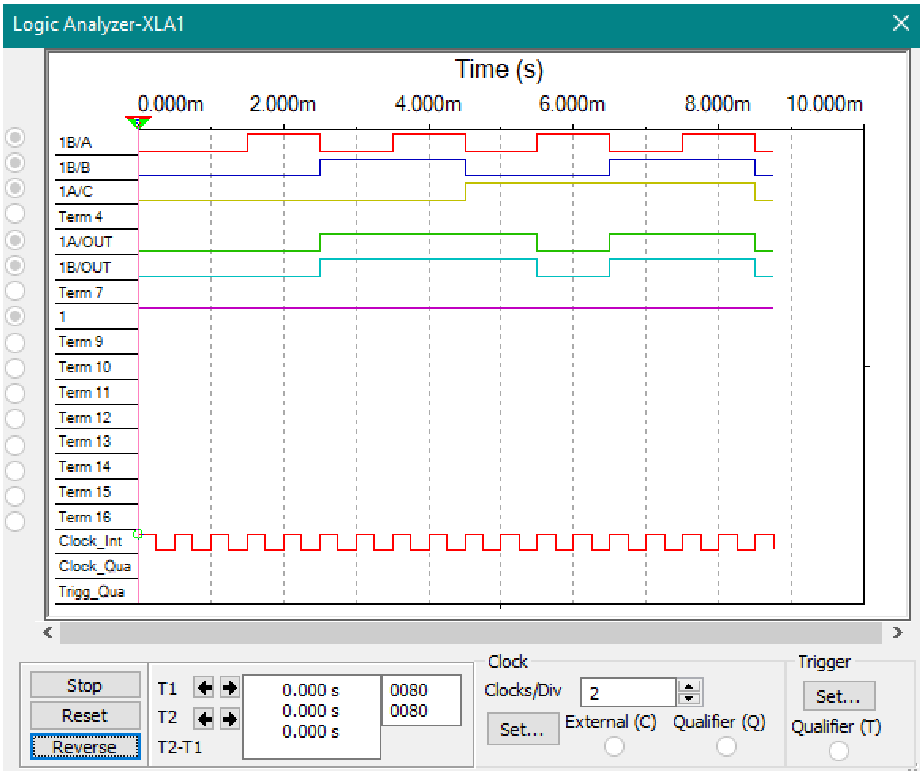
Poniżej (**na rysunku 7**) znajduję się automatyczny układ testujący sprawdzający poprawność działania układu



Rysunek 7: Automatyczny układ testujący dla układu 1a



Rysunek 8: Generator słów dla układu testującego



Rysunek 9: Analiza dla układu testującego

1.3 Wnioski

- W rzeczywistości ceny i dostępność różnych bramek różnią się, dlatego w tym przypadku opłacalna była zmiana na bramki NAND ze względu na ich niską cenę.
- Przekształcając funkcję zgodnie z prawami logiki zmniejszyliśmy liczbę elementów potrzebnych do stworzenia układu (z 3 elementów (bramki OR, AND, OR) do jednego elementu z 4 bramkami NAND).
- Układ może być zastosowany jako asynchroniczny przerzutnik "RS". Jeżeli podstawimy:

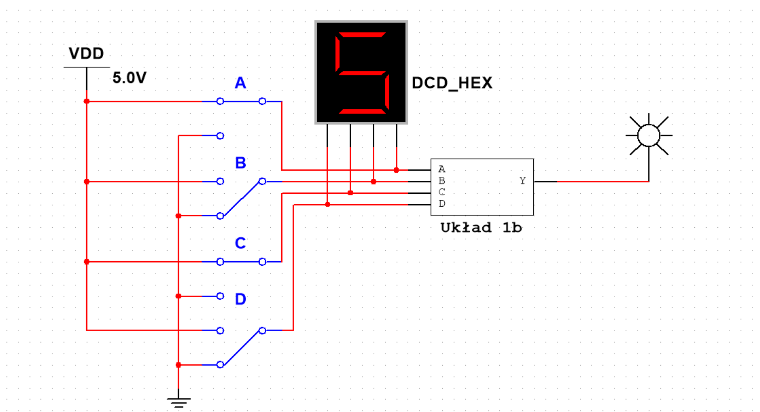
  $$S := B, \quad R := A, \quad Q_{n-1} := C, \quad Q_n = Y$$

  Wówczas otrzymujemy funkcję charakteryzującą stan kolejny stan przerzutnika.

  $$Q_n = S + \overline{R}Q_{n-1}$$

2 Ćwiczenie *1b*

Ideą ćwiczenia jest zaprojektowanie, zbudowanie i przetestowanie układu, który sprawdza, czy liczba w postaci czterobitowego słowa jest pierwsza.



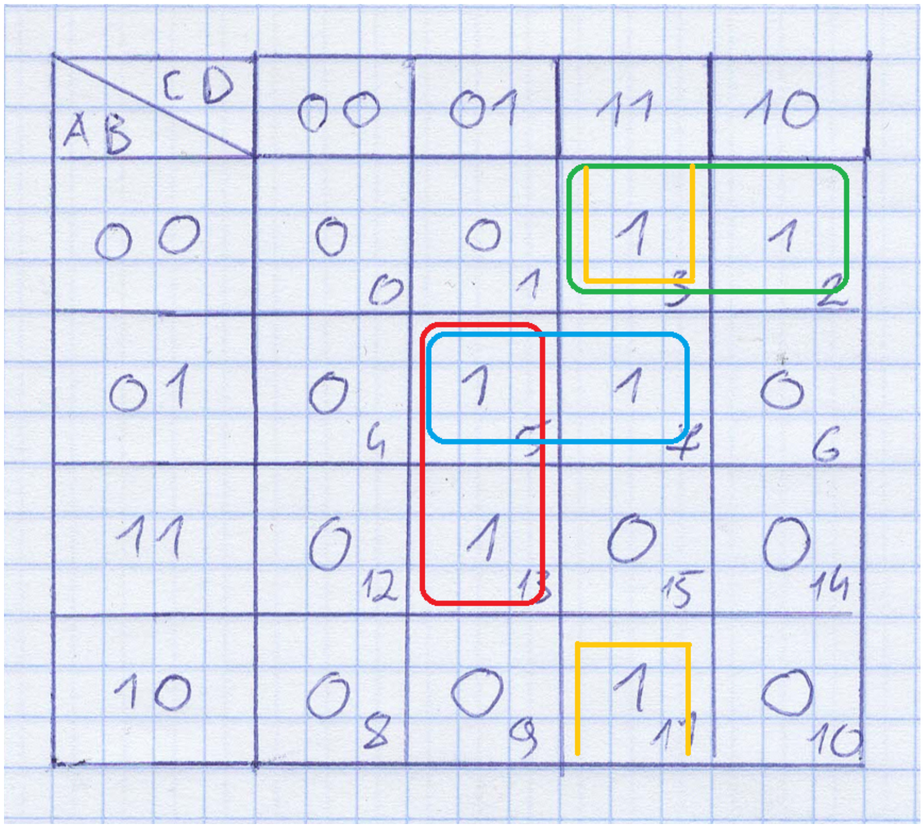
Rysunek 10: Układ 1b

2.1 Rozwiązanie teoretyczne

Do zaprojektowania układu użyta została funkcja

$$f(A,B,C,D) = \sum m_i, i \in \{2,3,5,7,11,13\}$$

Gdzie m_i odpowiada iloczynowi logicznemu każdego z argumentów funkcji lub jego zaprzeczenia odpowiadającemu zapisowi na czterech bitach liczby i. Do jej uproszczenia posłuży tablica Karnough przedstawiona na **rysunku 11**.



Rysunek 11: Tabela Karnough dla jedynek

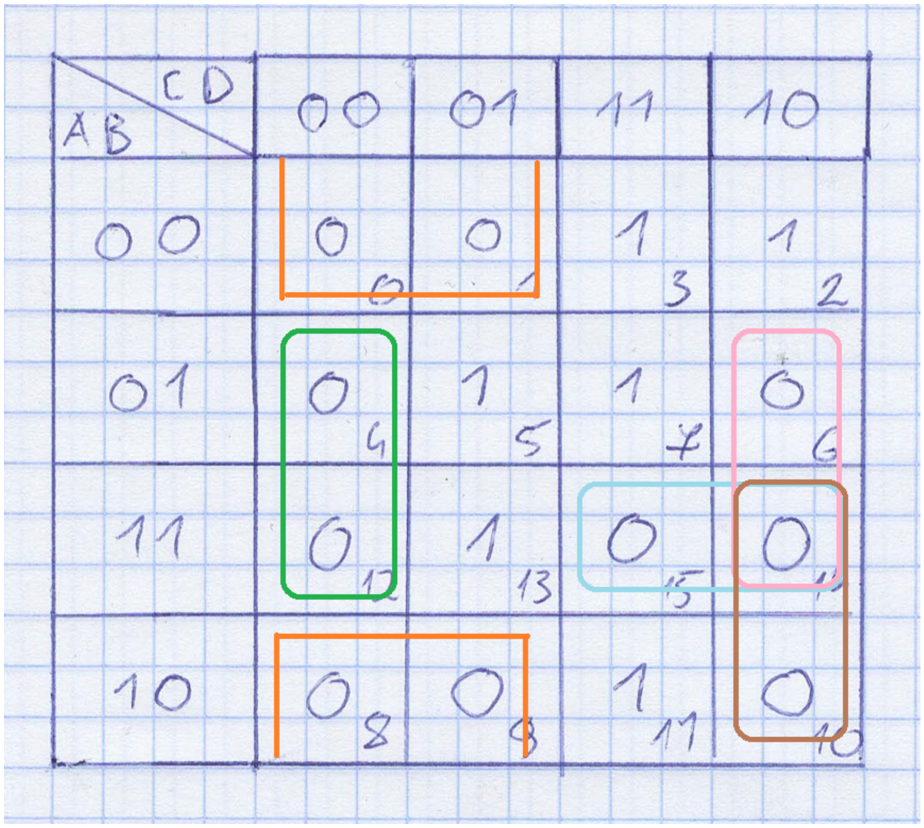
Funkcja po uproszczeniu wygląda następująco

$$f(A, B, C, D) = \overline{A}BD + \overline{A}\,\overline{B}C + B\overline{C}D + \overline{B}CD$$

Korzystając z rozdzielności koniunkcji względem alternatywy

$$f(A, B, C, D) = \overline{B}C(\overline{A} + D) + BD(\overline{A} + \overline{C})$$

Poniżej tablica Karnough z wykorzystanie zaznaczania zer (**rysunek 12**).



Rysunek 12: Tabela Karnough dla zer

Po uproszczeniu na podstawie tablicy Karnough z wykorzystaniem zer funkcja wygląda następująco

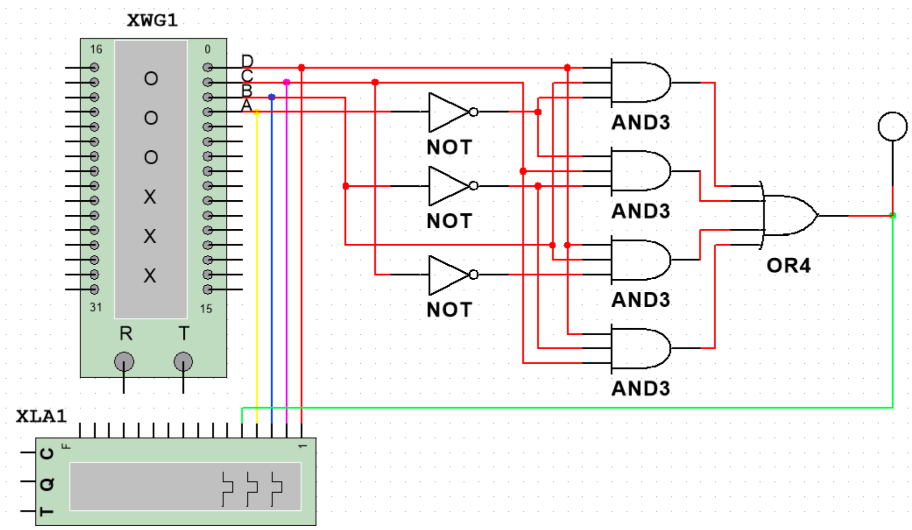
$$f(A, B, C, D) = (C + B)(\overline{C} + D + \overline{B})(\overline{A} + \overline{B} + \overline{C})(\overline{C} + D + \overline{A})(C + D + \overline{B})$$

Ze względu na jej większy poziom skomplikowania wybraliśmy funkcję przekształconą na podstawie tablicy z wykorzystanie zaznaczania jedynek.

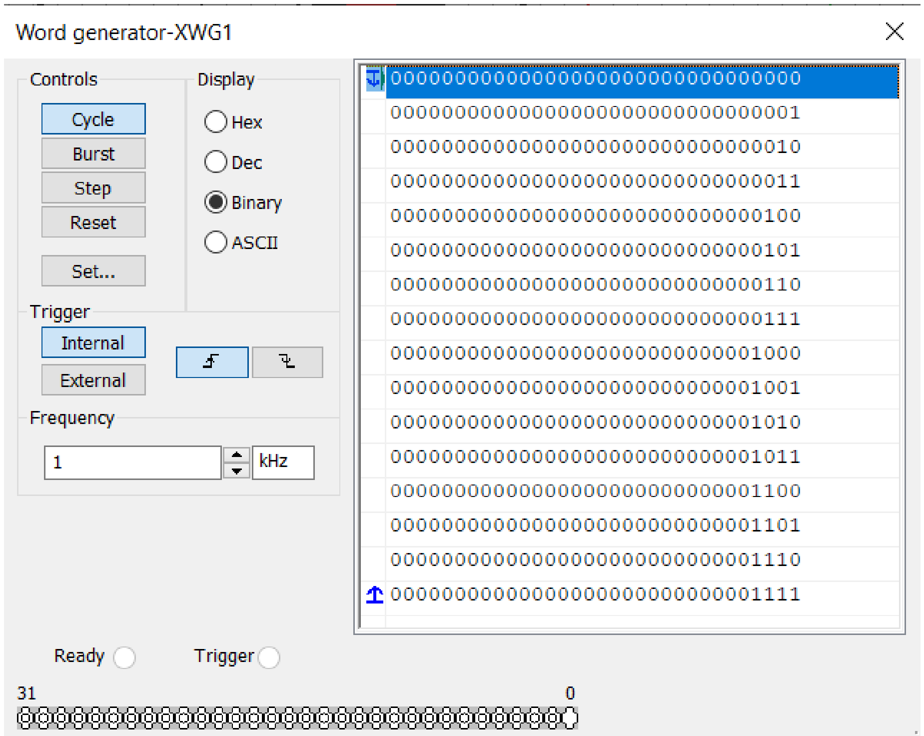
2.2 Implementacja w programie Multisim

Układ (**na rysunku 13**) skonstruowany został na podstawie równania przekształconego z użyciem tablicy Karnough dla jedynek. Alternatywny, bazowany na tym samym równaniu, ale po zastosowaniu zasady rozdzielności koniunkcji względem alternatywy, używa tyle samo, ale innych rodzajów bramek (3 bramek NOT, 3 bramek OR dwuargumentowych, 2 bramek AND trzyargumentowych). Poniżej znadjują się obie wersje układu, odczyty generatora słów dotyczą obu układów.

$$Y = \overline{A}BD + \overline{A}\,\overline{B}C + B\overline{C}D + \overline{B}CD$$

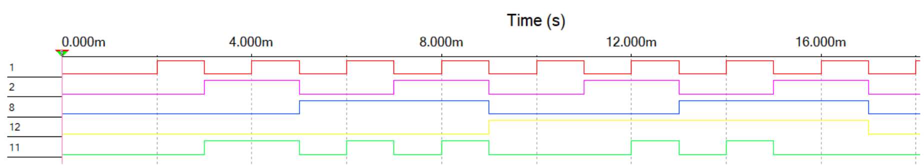


Rysunek 13: Układ logiczny w programie Multisim do ćwiczenia 2



Rysunek 14: Generator słów dla układu z ćwiczenia 2

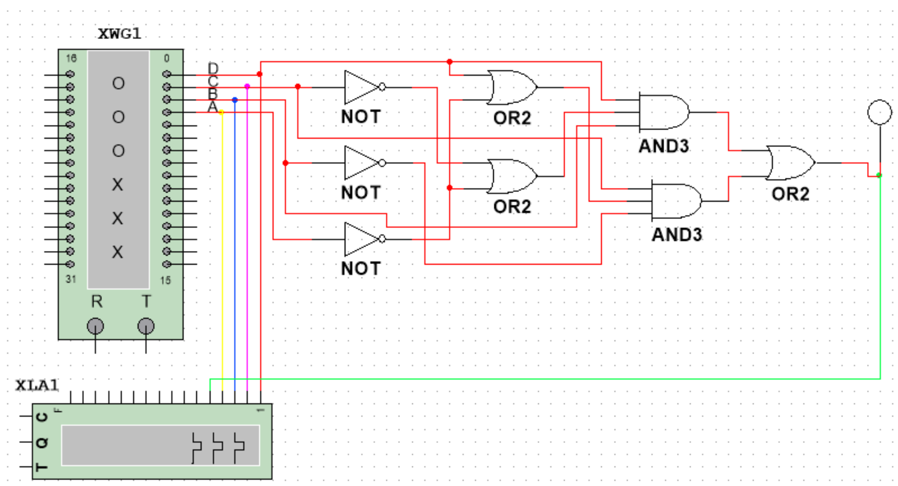
Analogicznie jak w przypadku ćwiczenia pierwszego, przeprowadzona została analiza za pomocą analizarota logicznego. Jego wyniki zostały przedstawione **rysunku 15** (**A** - żółty, **B** - niebieski, **C** - różowy, **D** - czerwony, **wynik** - zielony). Układ dziala poprawnie.



Rysunek 15: Wykres przebiegu czasowego badanego układu

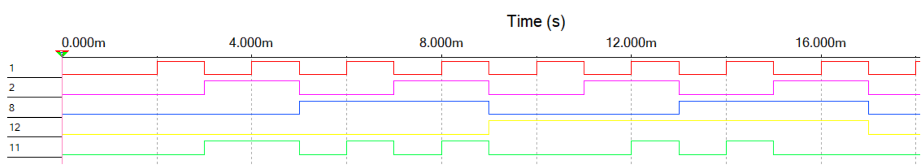
Poniżej alternatywny układ dla równania po przekształceniu

$$f(A,B,C,D) = \overline{B}C(\overline{A}+D) + BD(\overline{A}+\overline{C})$$



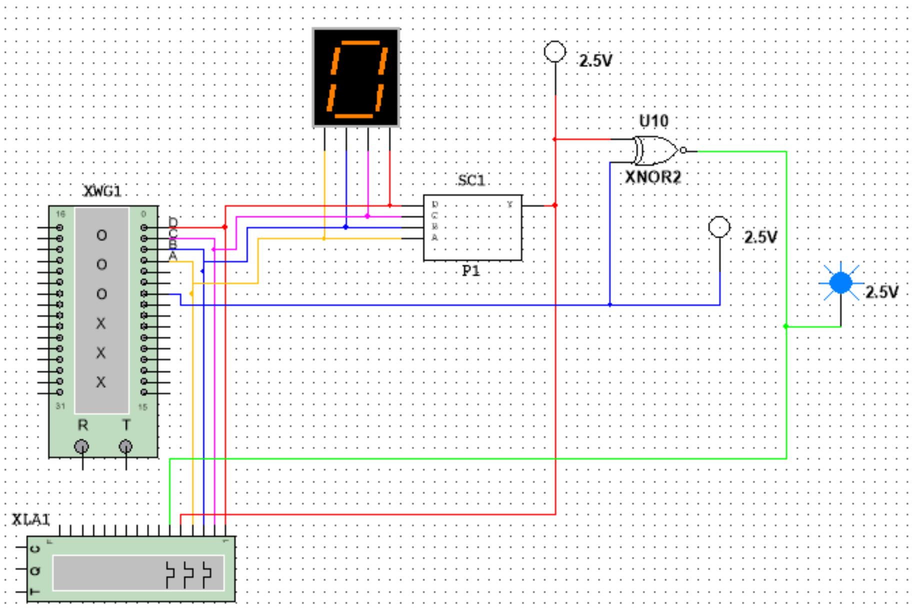
Rysunek 16: Alternatywny układ do ćwiczenia 2

Oraz wykres jego analizatora (generator słów identyczny jak w przypadku pierwszej części układu)

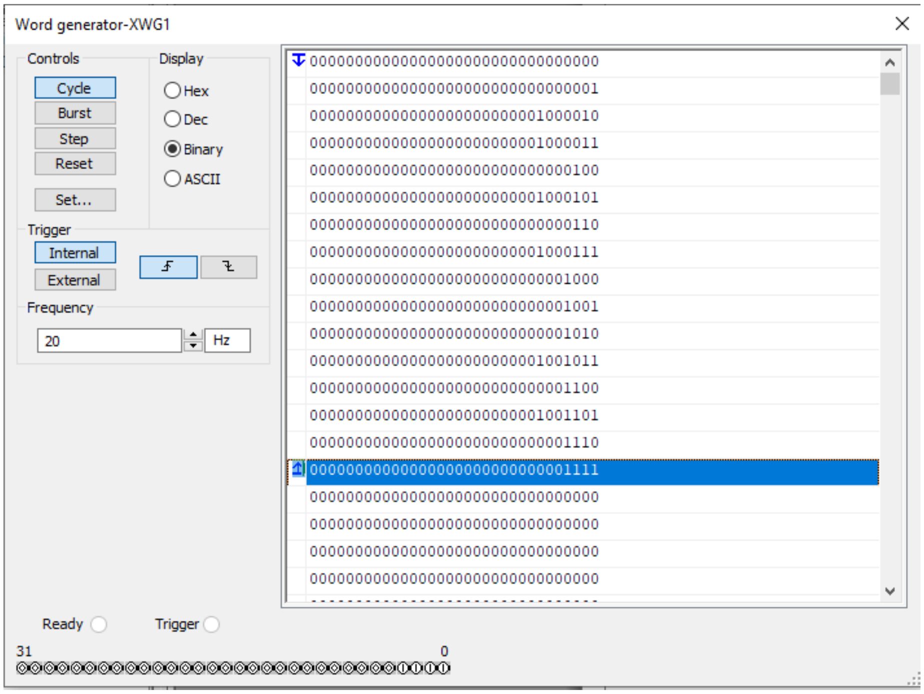


Rysunek 17: Wykres przebiegu czasowego alternatywnego układu

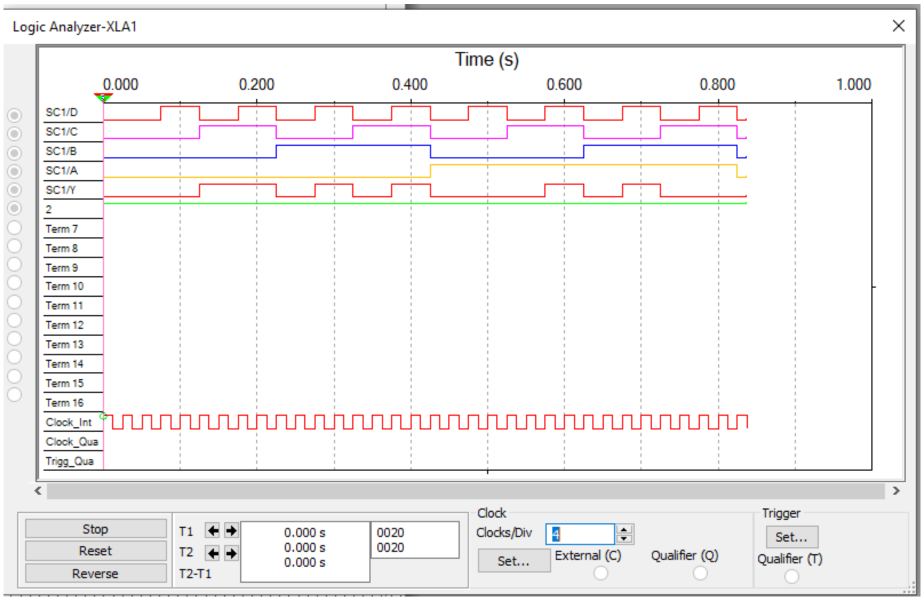
Poniżej (**na rysunku 18**) znajduję się automatyczny układ testujący sprawdzający poprawność działania układu



Rysunek 18: Automatyczny układ testujący dla układu 1b



Rysunek 19: Generator słów dla automatycznego układu testującego



Rysunek 20: Analizator dla automatycznego układu testującego

2.3 Wnioski

- W rzeczywistości ceny i dostępność różnych bramek różnią się, dlatego opłaca się, a nawet trzeba zmieniać wejściowe funkcje logiczne na równoważne, by móc zastosować tańsze bramki.
- Metoda Karnough jest dobrym sposobem na generowanie minimalnej (lub bardzo uproszczonej) funkcji logicznej, którą potem łatwiej wyrazić za pomocą układu cyfrowego.
- Układ może posłużyć do generowania liczb pierwszych używanych w algorytmie szyfrowania RSA.