# Laboratorium 1
# Transkoder liczb

Łukasz Kwinta, Kacper Kozubowski, Ida Ciepiela

marzec 2024

## Spis treści

**1 Cel zadania** **2**

**2 Idea rozwiązania** **2**

**3 Układ transkodera liczb pierwszych** **2**
3.1 Black box . . . 2
3.2 Tabela prawdy . . . 3
3.3 Tablice Karnaugh . . . 4
3.3.1 Człon J . . . 4
3.3.2 Człon I . . . 4
3.3.3 Człon H . . . 5
3.3.4 Człon G . . . 5
3.3.5 Człon F . . . 6
3.3.6 Człon E . . . 6
3.4 Implementacja . . . 6
3.4.1 Transkoder J . . . 8
3.4.2 Transkoder I . . . 8
3.4.3 Transkoder H . . . 9
3.4.4 Transkoder G . . . 10
3.4.5 Transkoder F . . . 11
3.4.6 Transkoder E . . . 11

**4 Enkoder BCD** **12**
4.1 Black box . . . 12
4.2 Funkcje logiczne . . . 12
4.3 Tabela prawdy . . . 13
4.4 Implementacja . . . 15

**5 Układ testowy** **15**
5.1 Analiza przebiegu . . . 16

**6 Zastosowania** **16**

# 1 Cel zadania

Celem zadania było zaprojektować, zbudować i przetestować układ kombinacyjny realizujący transkoder czterobitowej liczby naturalnej (wraz z zerem) na sześciobitową liczbę pierwszą, bazując wyłącznie na bramkach NAND.

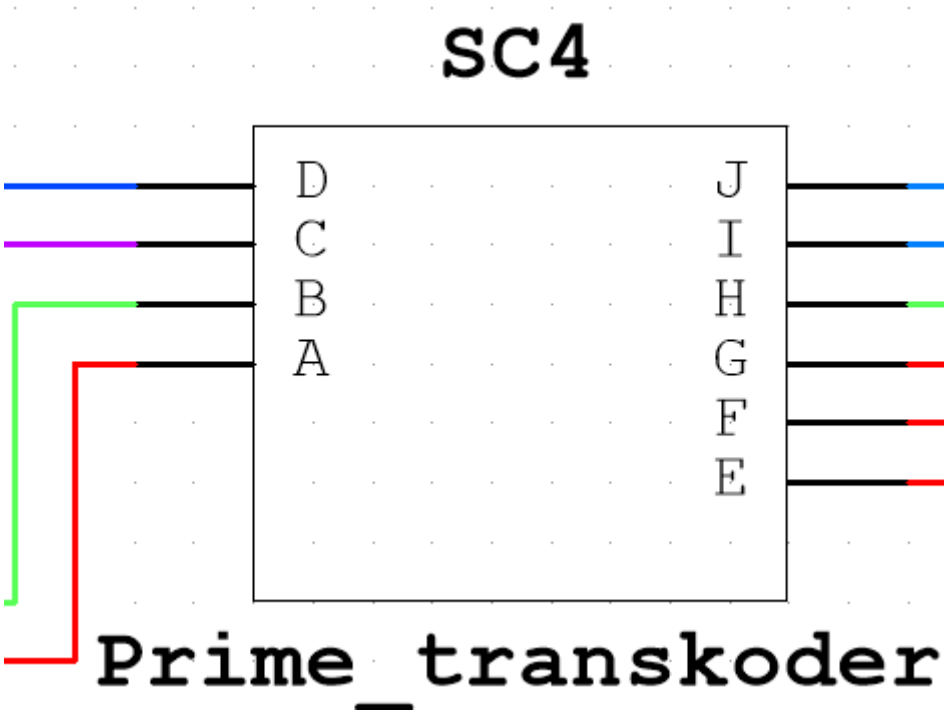
# 2 Idea rozwiązania

Nasze rozwiązanie opiera się na przekształcaniu 4 bitów wejściowych za pomocą transkoderów, generując w rezultacie 6 bitów wyjściowych. Aby uzyskać konkretne kombinacje bitów wyjściowych, zastosowaliśmy funkcje logiczne opracowane z wykorzystaniem tablic Karnaugh.

# 3 Układ transkodera liczb pierwszych

## 3.1 Black box

Pierwszym krokiem w projektowaniu układu jest przedstawienie go jako tzw. "Black Box". Czyli taką czarną skrzynkę dla której określamy tylko wejście i oczekiwany wynik działania dla danego wejścia ale nie wgłębiamy się w implementację.




Wejście do układu stanowią 4 piny ABCD kodujące binarnie wejściową liczbę 0-15. Stan wysoki na pinach stanowi logiczną jedynkę (1), a stan niski logiczne zero

(0).

| Numer bitu | 3 | 2 | 1 | 0 |
|---|---|---|---|---|
| Bit | A | B | C | D |
| Mnożnik | $2^3$ | $2^2$ | $2^1$ | $2^0$ |

Wyjście układu stanowi 6 pinów EFGHIJ kodujące pierwsze 16 liczb pierwszych. Tak samo jak na wejściu stan wysoki na pinach stanowi logiczną jedynkę (1), a stan niski logiczne zero (0).

| Numer bitu | 5 | 4 | 3 | 2 | 1 | 0 |
|---|---|---|---|---|---|---|
| Bit | E | F | G | H | I | J |
| Mnożnik | $2^5$ | $2^4$ | $2^3$ | $2^2$ | $2^1$ | $2^0$ |

Układ mapuje wejście na wyjście w następujący sposób - zakładając notację binarną zapisanego mapowania.

| Wejście | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Oczekiwane wyjście | 2 | 3 | 5 | 7 | 11 | 13 | 17 | 19 | 23 | 29 | 31 | 37 | 41 | 43 | 47 | 51 |

## 3.2 Tabela prawdy

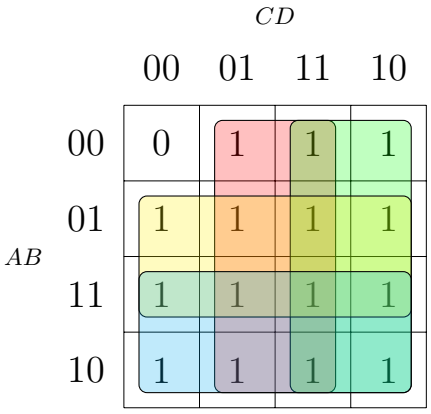
Poniżej zapisaliśmy tabelę prawd dla projektowanego układu:

| Wejście | | | | Wyjście | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| A | B | C | D | E | F | G | H | I | J |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 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 | 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 |
| 0 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| 0 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 1 |
| 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 | 0 | 1 | 1 | 1 | 0 | 1 |
| 1 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 1 |
| 1 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 |
| 1 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 1 |
| 1 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 1 |

## 3.3 Tablice Karnaugh

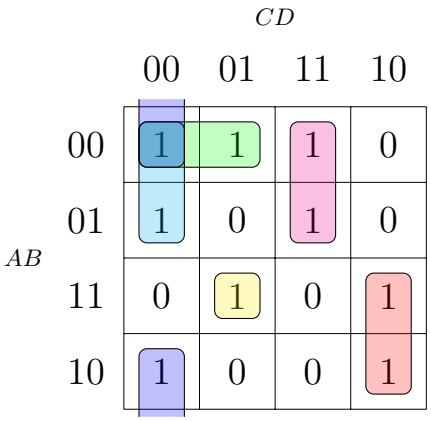
Poniżej przedstawione są tabele Karnaugh których użyliśmy do zbudowania transkoderów poszczególnych bitów

### 3.3.1 Człon J




$$f_{(1)} = A + C + D + B$$

### 3.3.2 Człon I




$$f_{(1)} = AC\overline{D} + \overline{A}CD + \overline{ABC} + \overline{ACD} + \overline{BCD} + AB\overline{C}D$$

### 3.3.3 Człon H

| AB \ CD | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 0 | 0 | 1 | 1 |
| 01 | 0 | 1 | 0 | 0 |
| 11 | 0 | 0 | 1 | 1 |
| 10 | 1 | 1 | 1 | 1 |

$$f_{(1)} = A\overline{B} + AC + \overline{B}C + \overline{A}B\overline{C}D$$

### 3.3.4 Człon G

| AB \ CD | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 0 | 0 | 0 | 0 |
| 01 | 1 | 1 | 0 | 0 |
| 11 | 1 | 1 | 0 | 1 |
| 10 | 0 | 1 | 0 | 1 |

$$f_{(1)} = B\overline{C} + AC\overline{D} + A\overline{C}D$$

### 3.3.5 Człon F

| AB \ CD | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 0 | 0 | 0 | 0 |
| 01 | 0 | 0 | 1 | 1 |
| 11 | 0 | 0 | 1 | 0 |
| 10 | 1 | 1 | 0 | 1 |

$$f_{(1)} = A\overline{BD} + A\overline{BC} + \overline{A}BC + BCD$$

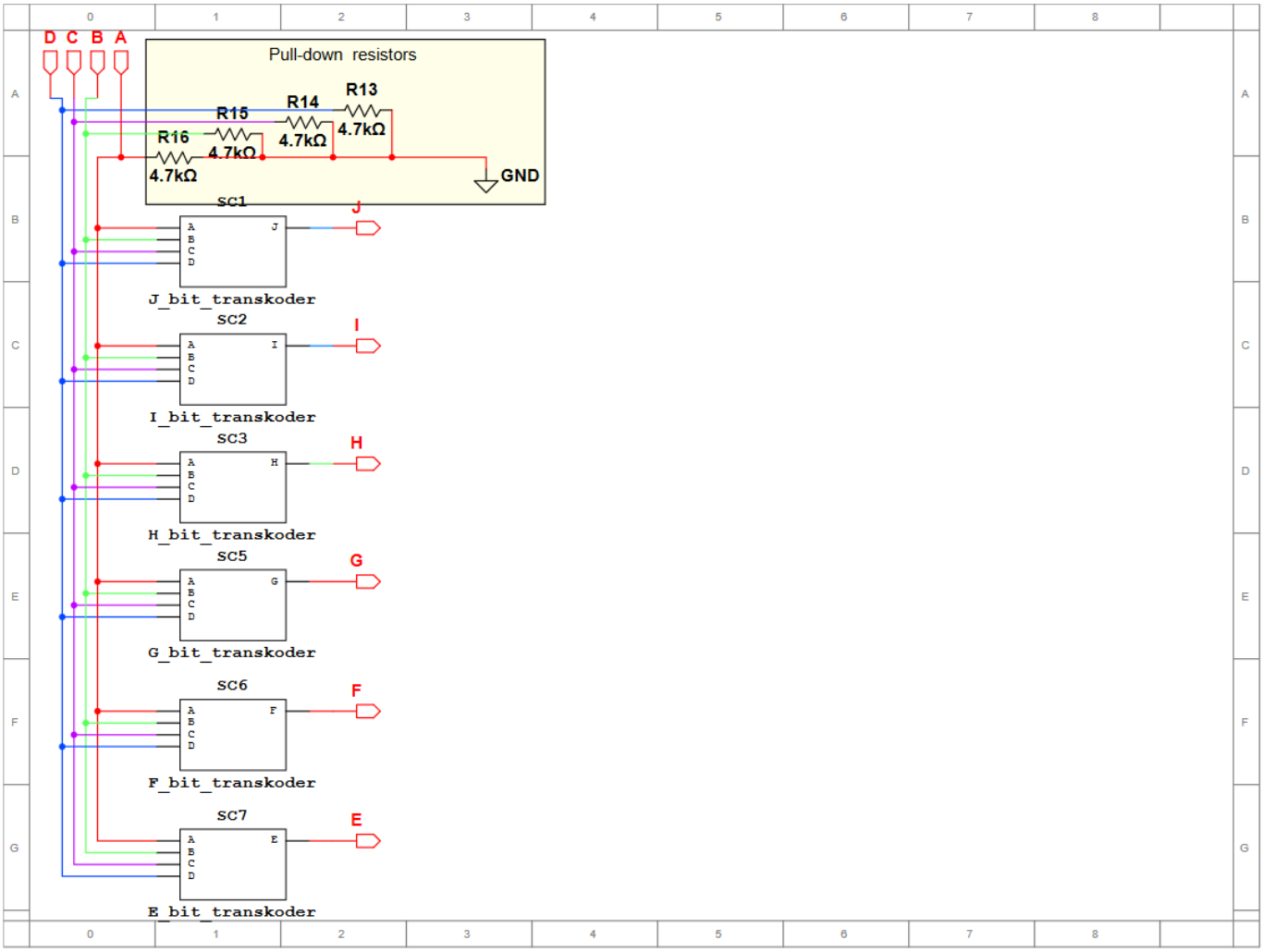
### 3.3.6 Człon E

| AB \ CD | 00 | 01 | 11 | 10 |
|---|---|---|---|---|
| 00 | 0 | 0 | 0 | 0 |
| 01 | 0 | 0 | 0 | 0 |
| 11 | 1 | 1 | 1 | 1 |
| 10 | 0 | 0 | 1 | 0 |

$$f_{(1)} = AB + ACD$$

## 3.4 Implementacja

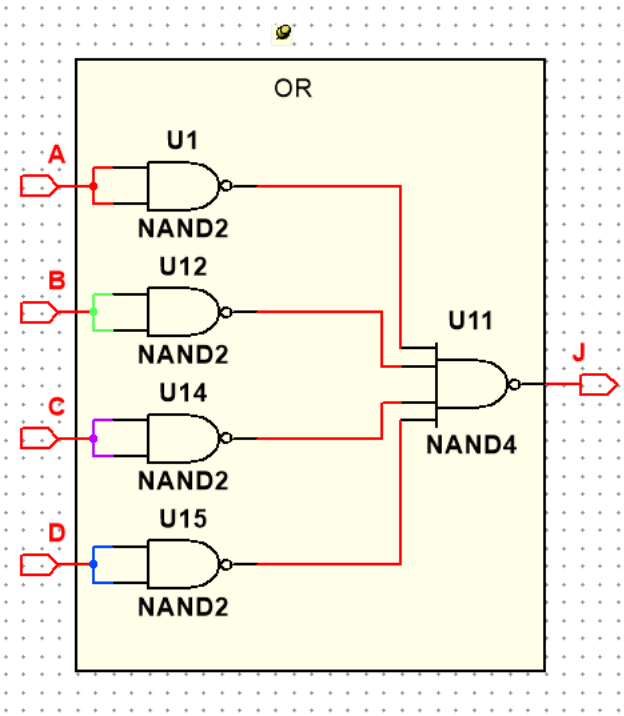
Otrzymane z tablic Karnough funkcje zostały odpowiednio przekształcone tak, aby używały jedynie bramek NAND i następnie zostały one zaimplementowane. Zdecydowaliśmy się na użycie schematów hierarchicznych - subcircuit - dla zwiększenia czytelnosci schematu. Na pierwszym poziomie znajdują się same transkodery pojedynczych bitów - czarne skrzynki które otrzymują całe wejście, a w wyniku dają konkretny bit wyjścia, innymi słowy każda funkcja logiczna została zaimplementowana na oddzielnym schemacie, a następnie zgromadziliśmy je w jeden podschemat stanowiący cały projektowany układ "Prime transkoder"



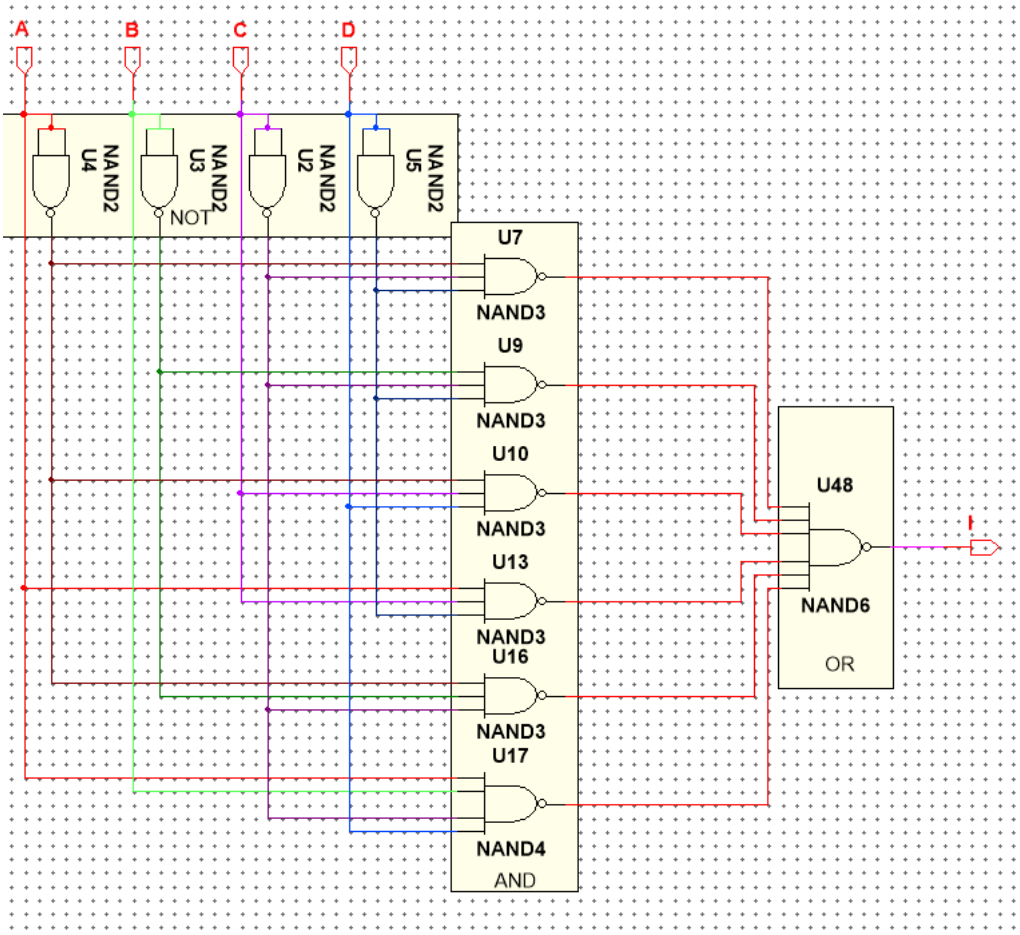

Użyliśmy rezystorów pull-down na wejściu układu, które ściągają niepodpięty sygnał do logicznego zera aby zapewnić poprawne funkcjonowanie układu nawet gdy wejście nie jest podpięte w całości.

Poniżej implementacje poszczególnych funkcji logicznych zgodnie z funkcjami otrzymanymi z tablic Karnaugh:

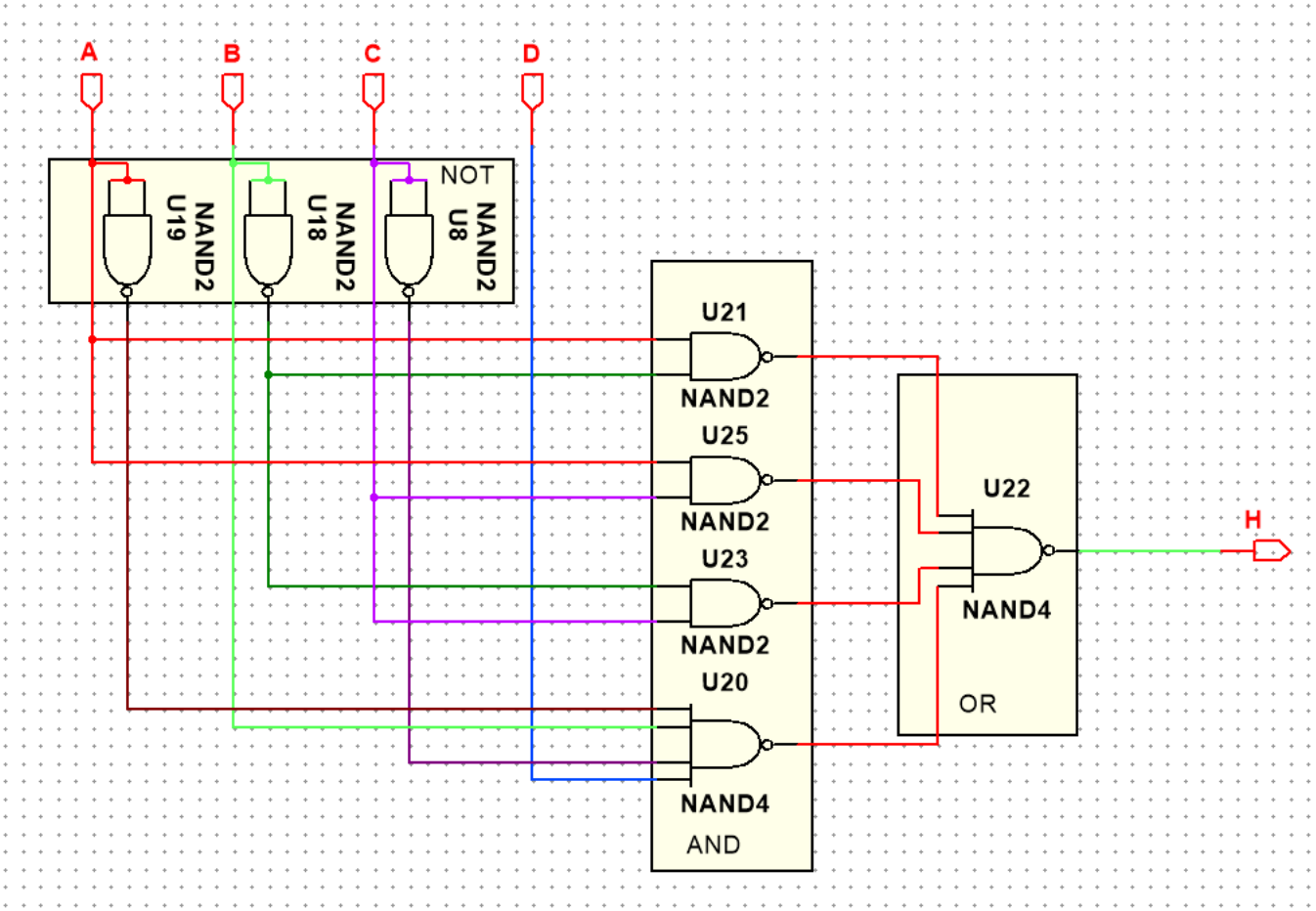
### 3.4.1 Transkoder J



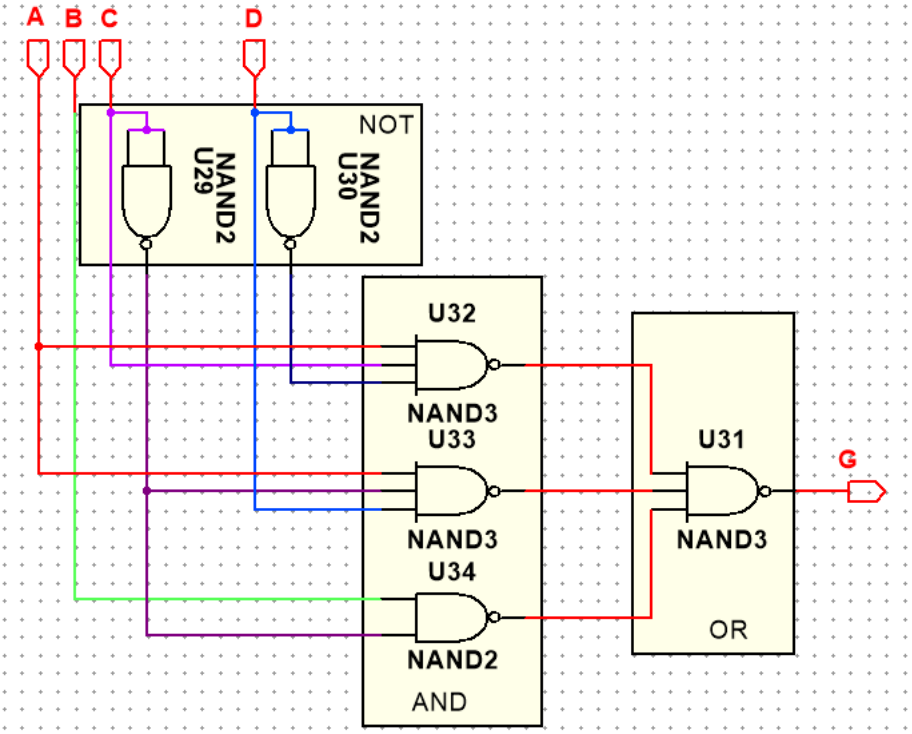

### 3.4.2 Transkoder I



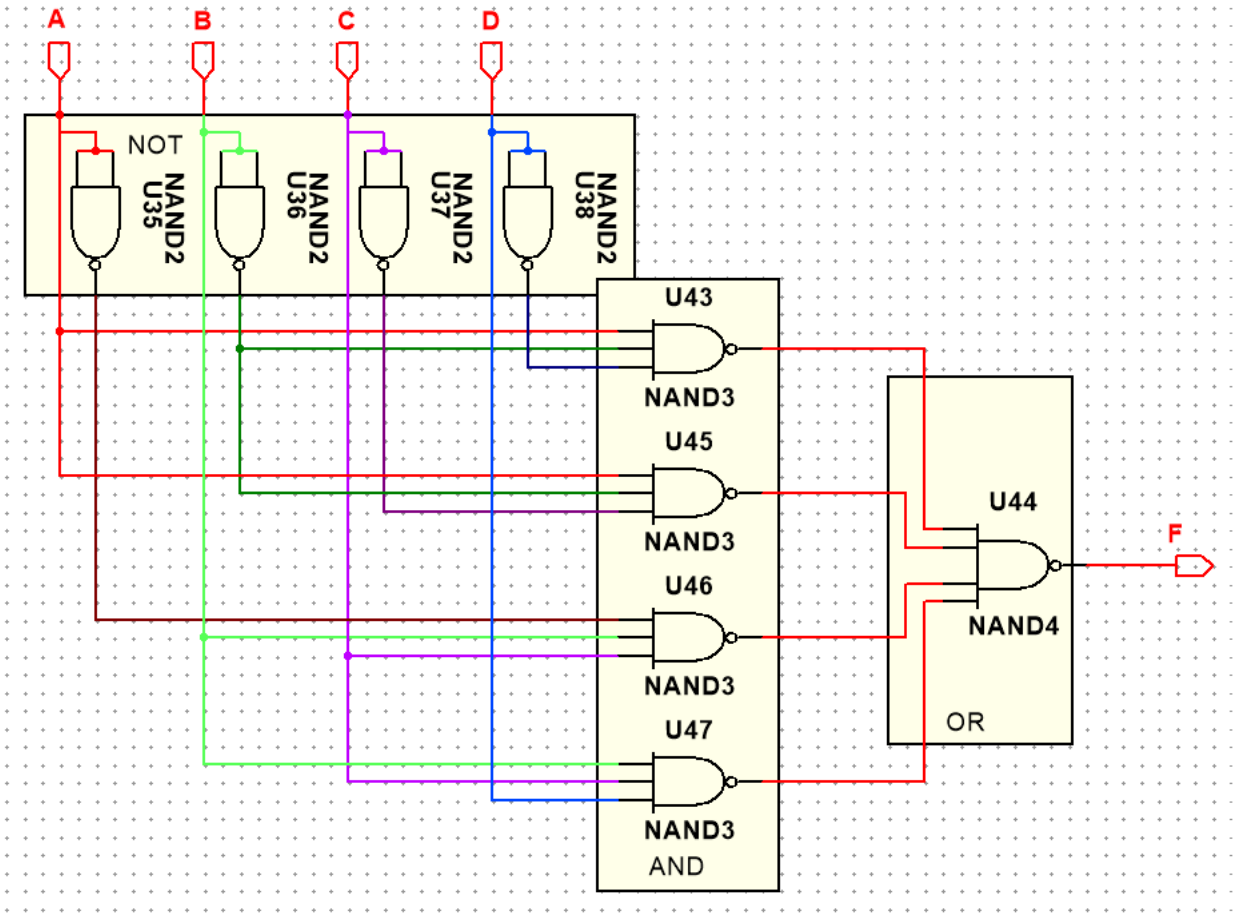
### 3.4.3 Transkoder H



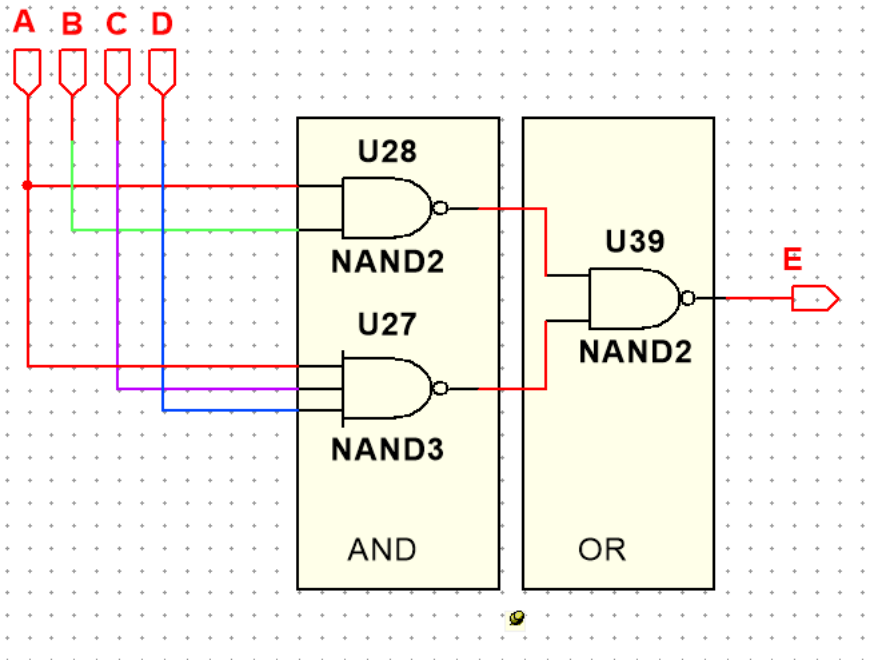
### 3.4.4 Transkoder G



### 3.4.5 Transkoder F




### 3.4.6 Transkoder E



# 4 Enkoder BCD

Aby wyświetlać liczby dwucyfrowe w poprawny sposób zdecydowaliśmy się na zaimplementowanie enkodera BCD (Binary Coded Decimal). Jest to format w którym liczba dziesiętna jest kodowana cyfra po cyfrze przez 4 bitowe liczby binarne. Pozwala to na łatwe wyświetlanie wielocyfrowych liczb. W programie Multisim nie udało nam się znaleźć gotowego układu z większą niż 4 bity liczbą wejść, dlatego sami go zaimplementowaliśmy. Jako, że nie jest to bezpośredni cel ćwiczenia, nie rozpisywaliśmy ręcznie tablic Karnaugh - dla 6 zmiennych wejściowych jest to bardziej złożona operacja, lecz skorzystaliśmy z gotowego kalkulatora w internecie, aby wygenerować funkcję logiczną. Tabelę prawd wygenerowaliśmy samemu algorytmicznie, a następnie zaznaczyliśmy odpowiednie bity w kalkulatorze online w ten sposób otrzymując funkcje logiczne.

## 4.1 Black box

## 4.2 Funkcje logiczne

## 4.3 Tabela prawdy

Poniżej zapisaliśmy tabelę prawd dla projektowanego układu:

| Wejście | | | | | | Wyjście | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| A | B | C | D | E | F | G | H | I | J | K | L | M | N |
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 |
| 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 |
| 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 |
| 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 |
| 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 |
| 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |
| 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 |
| 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 |
| 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 1 |
| 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 |
| 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 |
| 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 |
| 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 |
| 0 | 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 |
| 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 |
| 0 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 1 |
| 0 | 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 |
| 0 | 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 |

cd.

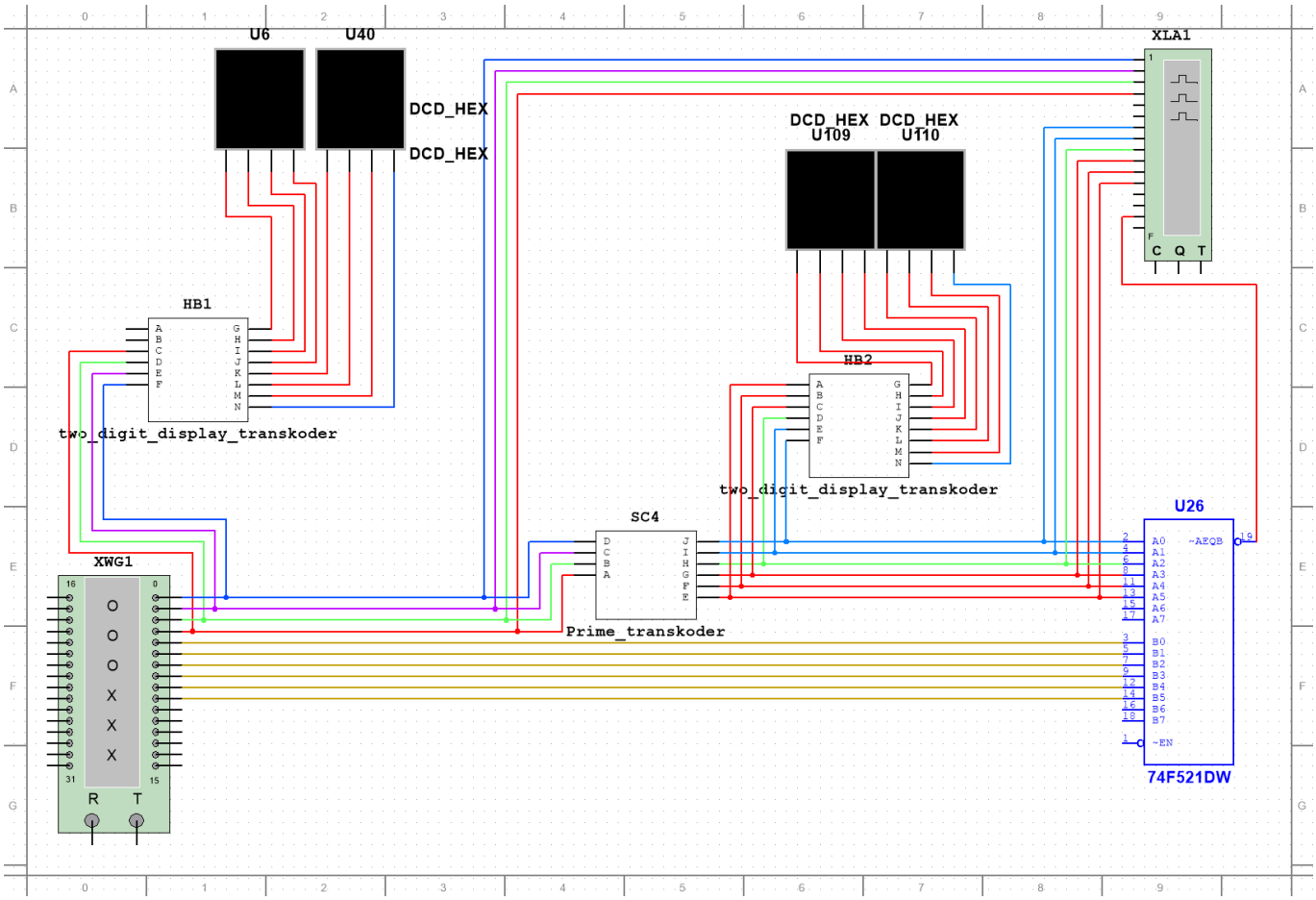
| Wejście | | | | | | Wyjście | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| A | B | C | D | E | F | G | H | I | J | K | L | M | N |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 |
| 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 0 |
| 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 0 |
| 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 0 |
| 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 |
| 1 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| 1 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 |
| 1 | 0 | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 1 |
| 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 |
| 1 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 |
| 1 | 1 | 0 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 1 |
| 1 | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 | 1 | 1 |
| 1 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 0 |
| 1 | 1 | 0 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 |
| 1 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 0 |
| 1 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 1 |
| 1 | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 |
| 1 | 1 | 1 | 0 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 1 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 0 |
| 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | 0 | 1 | 1 |

## 4.4 Implementacja

# 5 Układ testowy

Aby przetestować projektowany element zestawiliśmy układ zawierający:

- Word Generator - generuje 4 bity wejściowe dla naszego układu, oraz 6 bitów oczekiwanego wyjścia - zakodowane liczby pierwsze
- Comparator - porównuje wyjście układu z zakodowanym wyjściem z generatora sygnałów
- Logic Analyzer - pozwala na wyświetlenie przebiegów wejścia i wyjścia układu, aby sprawdzić poprawność jego działania
- Wyświetlacze HEX - wyświetlacze 7 segmentowe z enkoderem pozwalające wyświetlić liczby 0-15 jako cyfry szesnastkowe
- Prime transkoder - nasz zaprojektowany układ transkodujący liczby 0-15 na liczby pierwsze
- two digit display transkoder - nasz autorski enkoder BCD, pozwalający lepiej zwizualizować działanie naszego układu



### 5.1 Analiza przebiegu

## 6 Zastosowania

Poniżej wymieniamy przykładowe zastosowania zaprojektowanego układu:

- Transkoder generujący liczby pierwsze na podstawie prostej liczby może być zastosowany w urządzeniach szyfrujących. Dużo łatwiej jest wygenerować (pseudo)-losową liczbę z przedziału 0-15 i ją przekształcić na liczbę pierwszą za pomocą takiego transkodera niż wybierać losową liczbę pierwszą. Wiele algorytmów szyfrujących czy funkcji hashujących bazuje na liczbach pierwszych więc taki układ miałby tam zastosowanie.
- Innym zastosowaniem układów z rodziny transkoderów - lecz nie koniecznie tego konkretnego - może być przekształcenie kodu jakiegoś błędu reprezentowanego w systemie przez liczby 0-15 na jakiś inny kod błędu, który np. nadaje się do wyświetlenia użytkownikowi - bo mówi coś więcej o istocie problemu.