Laboratorium 1
Transkoder liczb

Łukasz Kwinta, Kacper Kozubowski, Ida Ciepiela

marzec 2024

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.

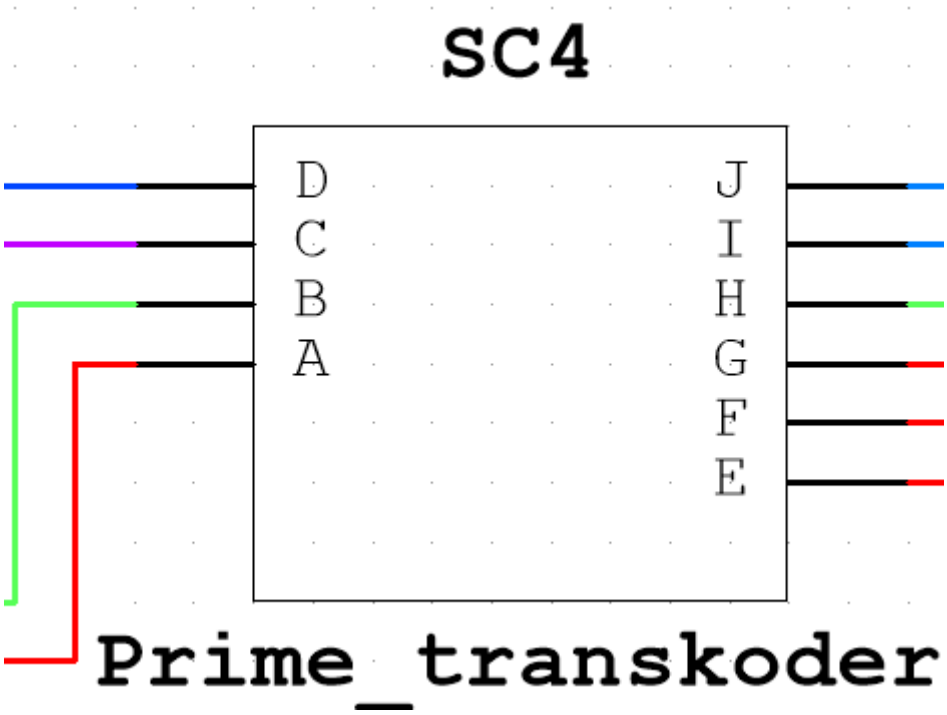
2 Idea rozwiązania

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

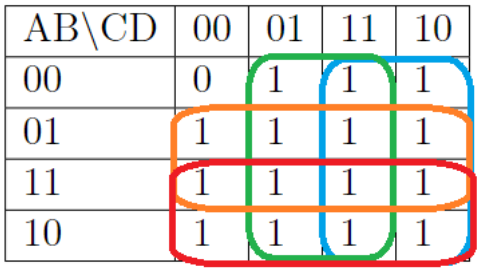
3.3 Tablice Karnaugh

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

3.3.1 Człon J

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

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

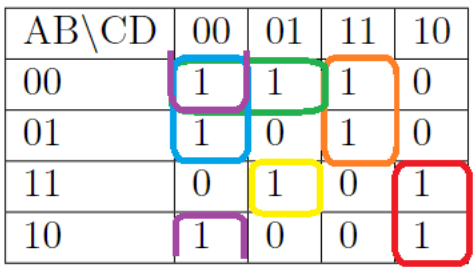


$$f_{(1)} = A + C + D + B$$

3.3.2 Człon I

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

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



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

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

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

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

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

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

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

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

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

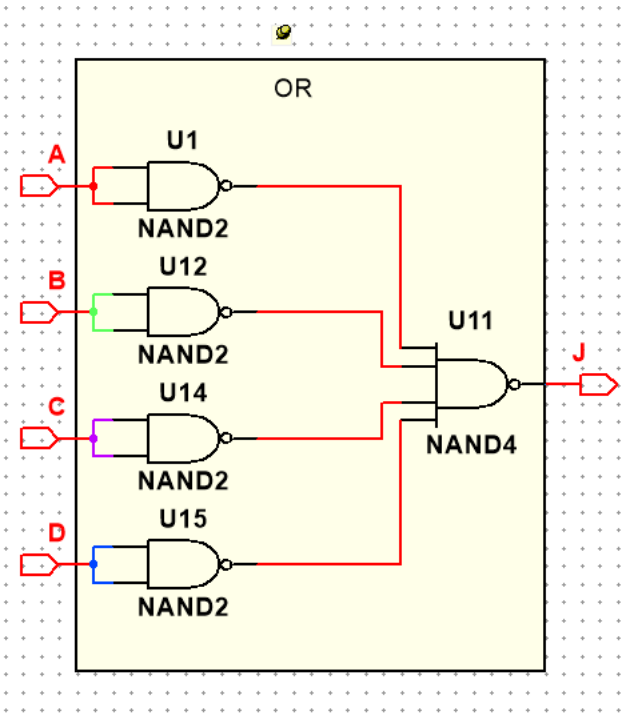
AB\CD	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	1	1	1	1
10	0	0	1	0

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

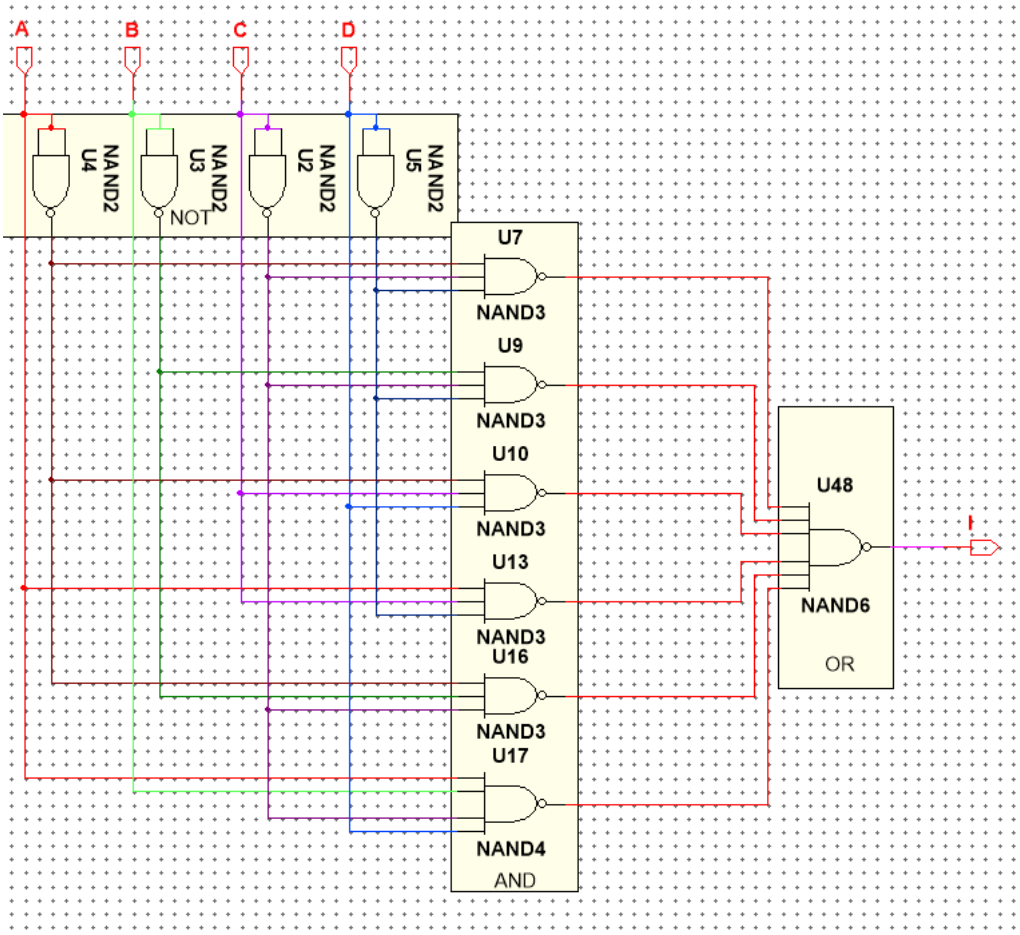
3.4 Implementacja

Otrzymane z tablic Karnough funkcje zostały odpowiednio przekształcone tak, aby używały jedynie bramek NAND i następnie zostały one zaimplementowane.

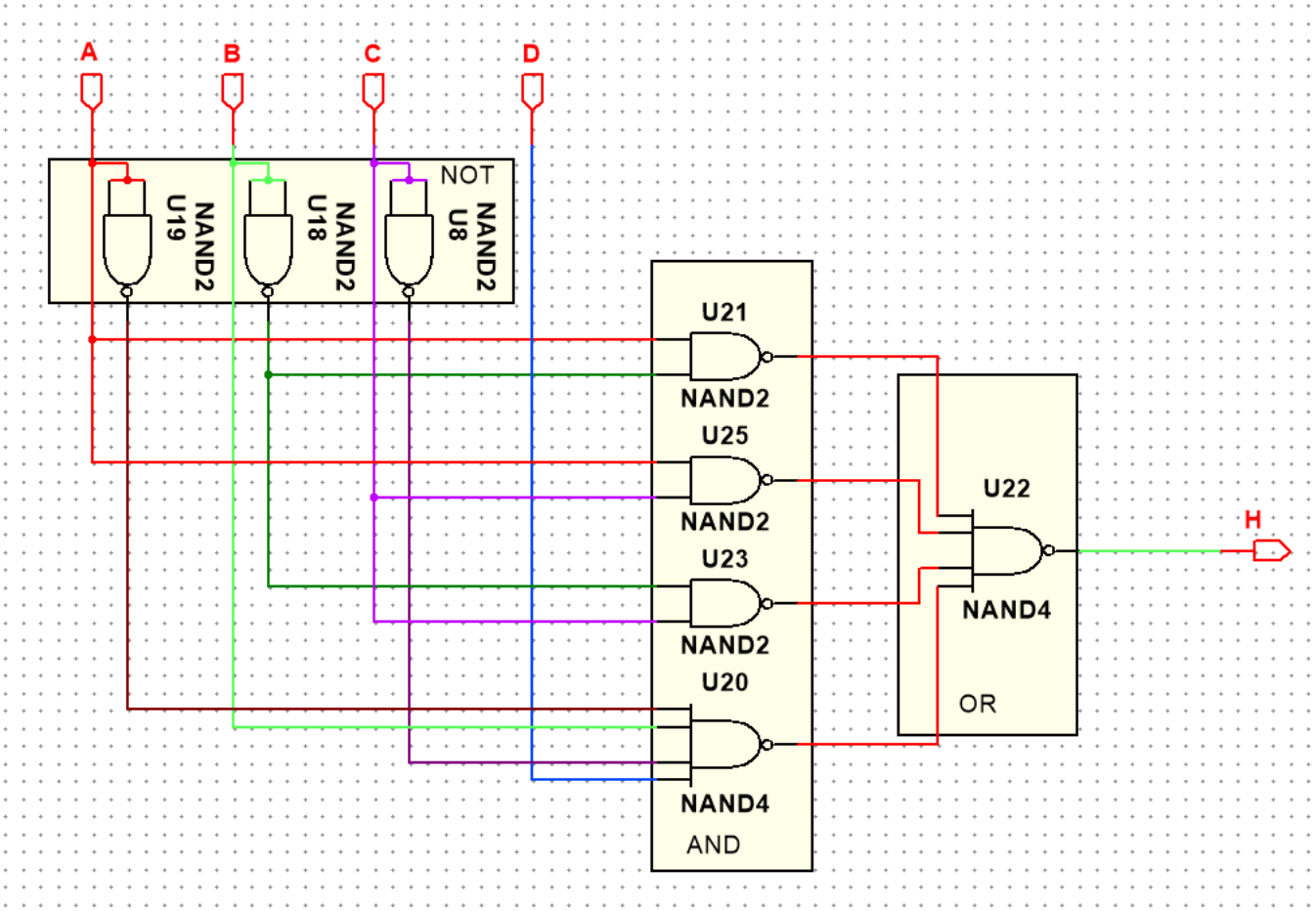
3.4.1 transkoder J



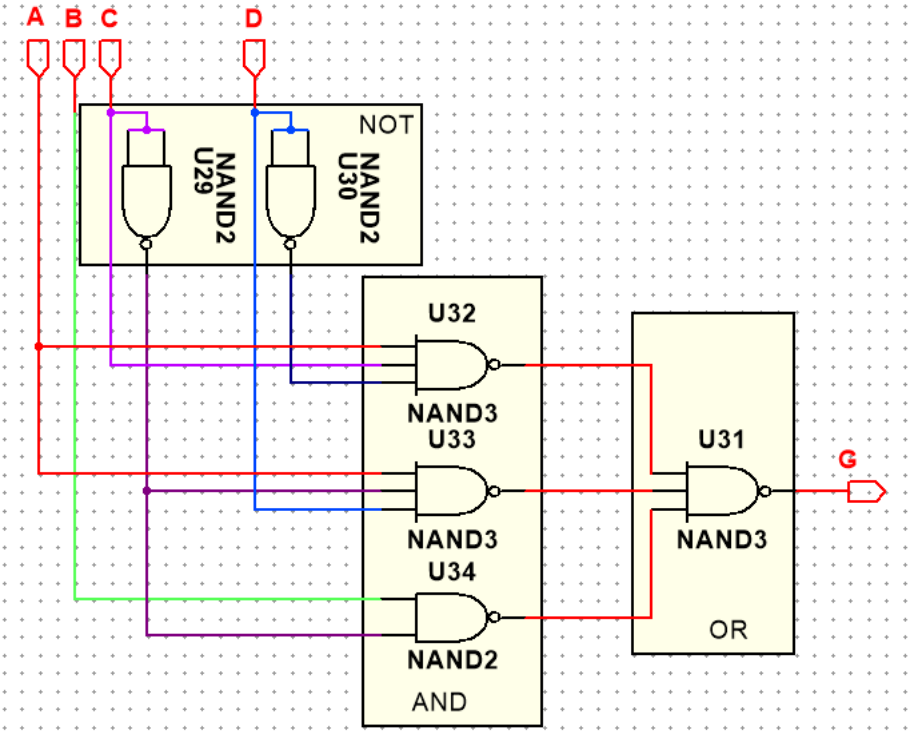
3.4.2 transkoder I



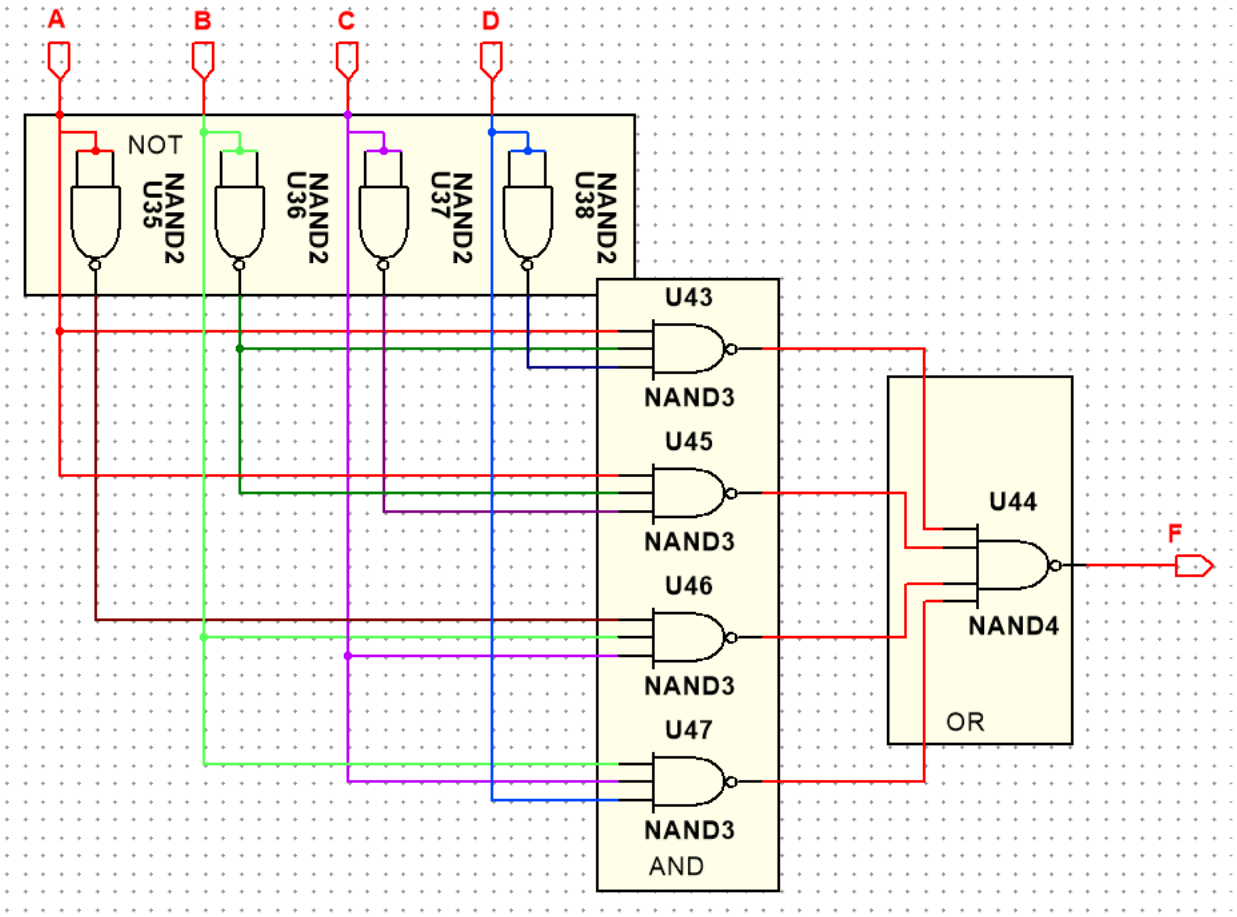
3.4.3 transkoder M



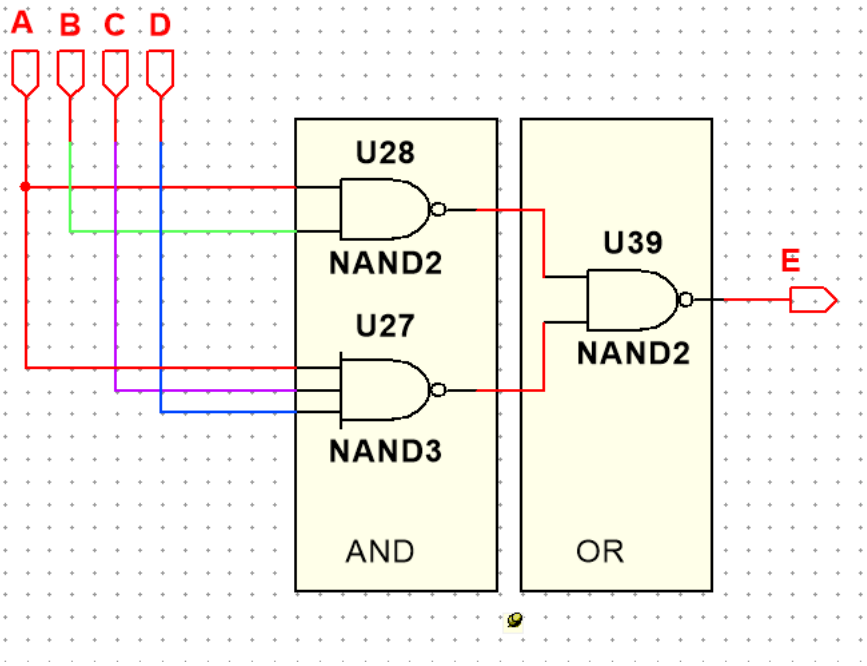
3.4.4 transkoder G



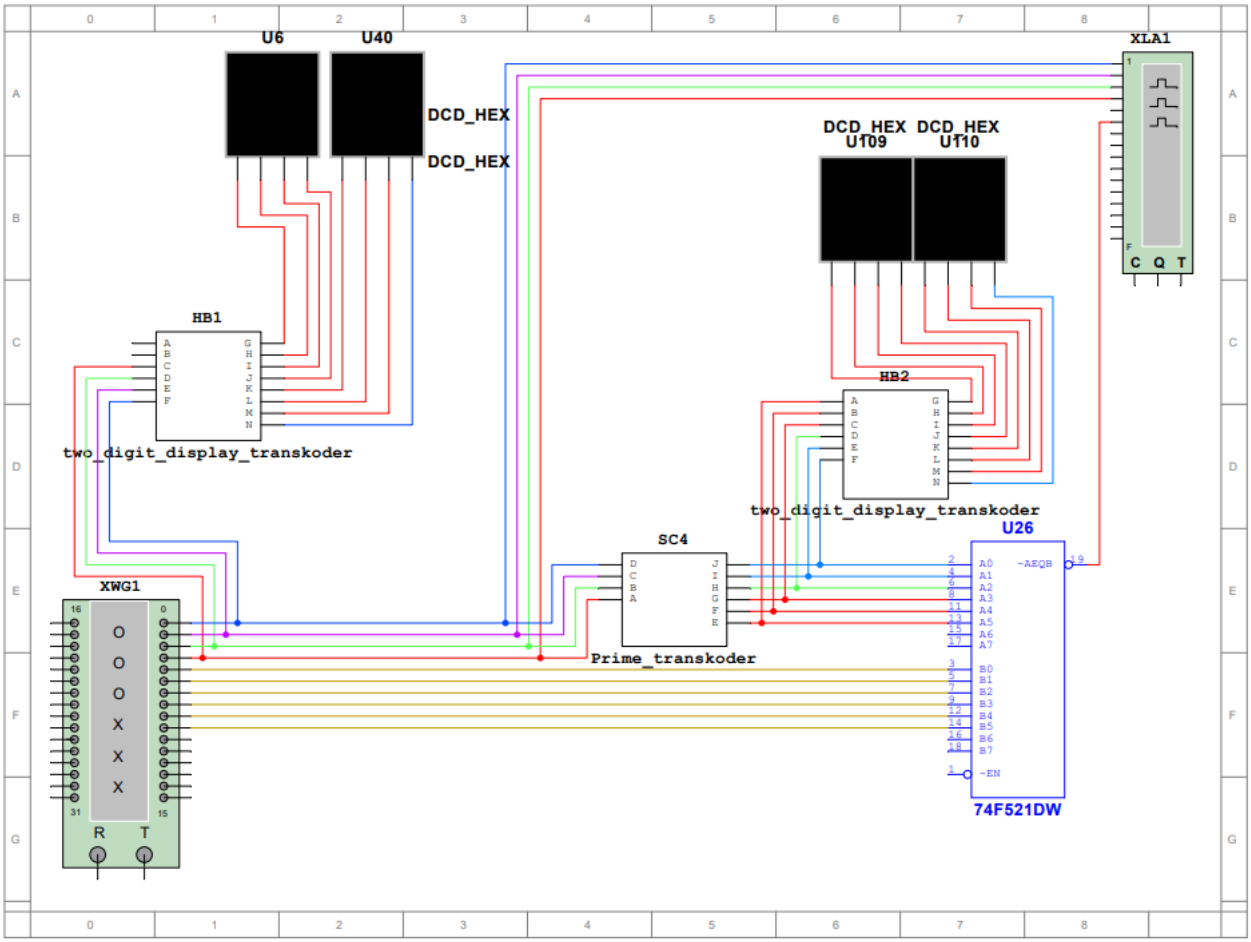
3.4.5 transkoder F



3.4.6 transkoder E



Ostatecznie układ prezentuje się w następujący sposób:



Oprócz przedstawionego wyżej transkodera nazwanego na schemacie Prime transkoder w układzie znajdują się także Word Generator służący do generowania wszystkich sygnałów wejsciowych i Logic Analyzer, który monitoruje sygnały wyjściowe.

4 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.