Technika Cyfrowa
Ćwiczenie 1

Kacper Bieniasz
Aleksander Buszek
Maciej Nowakowski
Albert Pęciak

03.04.2024

1 Temat ćwiczenia

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

Układ taki powinien zatem zamieniać kolejne liczby:
0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15

na odpowiednie kolejne liczby pierwsze:
2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53

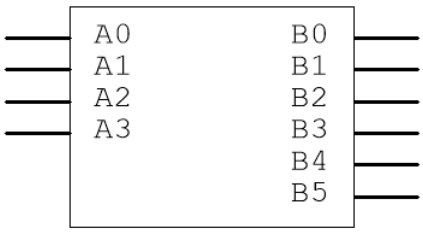


Figure 1: A_i - bity liczby wejściowej; B_j - bity liczby wyjściowej

Projekt będzie podzielony na sześć podukładów, z których każdy będzie wyznaczał kolejne wartości bitów wyjściowych.

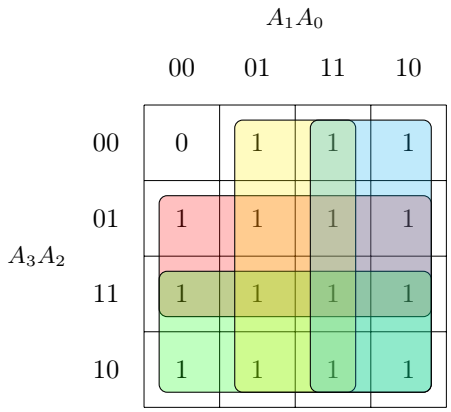
2 Tablice Karnough

Aby zminimalizować liczbę potrzebnych funkcji skorzystaliśmy z tablic Karnough. Pozwalają one w stosunkowy łatwy sposób uprościć formę funkcji logicznych zapewniając poprawność. Jednak na początku stworzyliśmy tabele dla wejść (A_3, A_2, A_1, A_0) oraz wyjść $(B_5, B_4, B_3, B_2, B_1, B_0)$. Mówi ona jakie wartości muszą przyjmować bity na wejściu, aby popranwie transkodować daną liczbę na liczbę pierwszą.

Decimal	A_3	A_2	A_1	A_0	B_5	B_4	B_3	B_2	B_1	B_0	Decimal
0	0	0	0	0	0	0	0	0	1	0	2
1	0	0	1	1	0	0	0	0	1	1	3
2	0	0	1	0	0	0	0	1	0	1	5
3	0	0	1	1	0	0	0	1	1	1	7
4	0	1	0	0	0	0	1	0	1	1	11
5	0	1	0	1	0	0	1	1	0	1	13
6	0	1	1	0	0	1	0	0	0	1	17
7	0	1	1	1	0	1	0	0	1	1	19
8	1	0	0	0	0	1	0	1	1	1	23
9	1	0	0	1	0	1	1	1	0	1	29
10	1	0	1	0	0	1	1	1	1	1	31
11	1	0	1	1	1	0	0	1	0	1	37
12	1	1	0	0	1	0	1	0	0	1	41
13	1	1	0	0	1	0	1	0	1	1	43
14	1	1	1	0	1	0	1	1	1	1	47
15	1	1	1	1	1	1	0	1	0	1	53

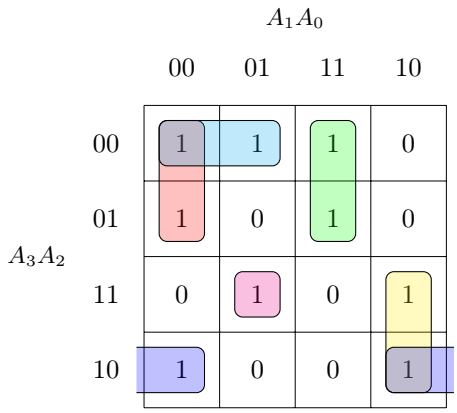
Następnie dla każdego bitu wyjścia tworzymy tablice Karnough'a. W tworzeniu siatek pamiętamy o kodzie Gray'a, aby sąsiednie bity różniły się między sobą o jeden bit.

2.1 Bit B_0



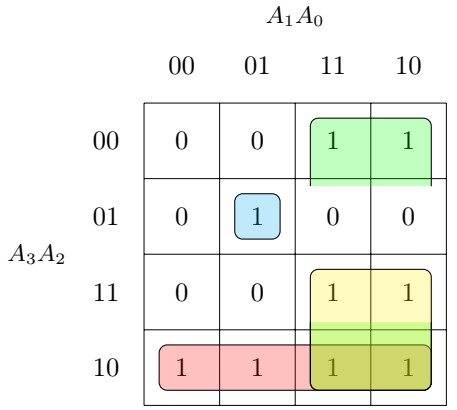
Z czerwonego otrzymujemy: A_0
Z zielonego otrzymujemy: A_1
Z żółtego otrzymujemy: A_2
Z niebieskiego otrzymujemy: A_3
Ostateczny wynik: $A_3 + A_2 + A_1 + A_0$

2.2 Bit B_1



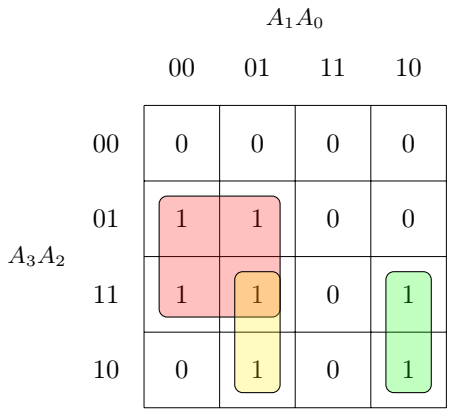
Z czerownego otrzymujemy: $\overline{A_3} \cdot \overline{A_1} \cdot \overline{A_0}$
Z zielonego otrzymujemy: $\overline{A_3} \cdot A_1 \cdot A_0$
Z żółtego otrzymujemy: $A_3 \cdot A_1 \cdot \overline{A_0}$
Z niebieskiego otrzymujemy: $\overline{A_3} \cdot \overline{A_2} \cdot \overline{A_1}$
Z fiolotowego otrzymujemy: $A_3 \cdot \overline{A_2} \cdot \overline{A_0}$
Z różowego otrzymujemy: $A_3 \cdot A_2 \cdot \overline{A_1} \cdot A_0$
Ostateczny wynik: $(\overline{A_3} \cdot \overline{A_1} \cdot \overline{A_0}) + (\overline{A_3} \cdot A_1 \cdot A_0) + (A_3 \cdot A_1 \cdot \overline{A_0}) + (\overline{A_3} \cdot \overline{A_2} \cdot \overline{A_1}) + (A_3 \cdot \overline{A_2} \cdot \overline{A_0}) + (A_3 \cdot A_2 \cdot \overline{A_1} \cdot A_0)$

2.3 Bit B_2



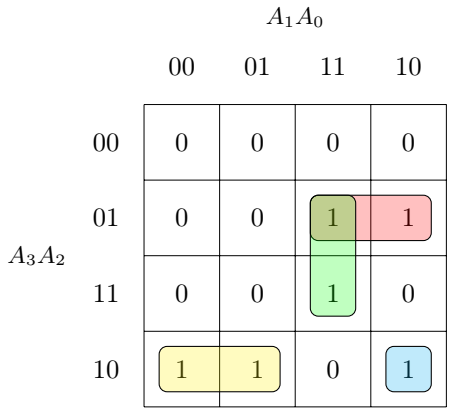
Z czerwonego otrzymujemy: $A_3 \cdot \overline{A_2}$
Z zielonego otrzymujemy: $\overline{A_2} \cdot A_1$
Z żółtego otrzymujemy: $A_3 \cdot A_1$
Z niebieskiego otrzymujemy: $\overline{A_3} \cdot A_2 \cdot \overline{A_1} \cdot A_0$
Ostateczny wynik: $(A_3 \cdot \overline{A_2}) + (\overline{A_2} \cdot A_1) + (A_3 \cdot A_1) + (\overline{A_3} \cdot A_2 \cdot \overline{A_1} \cdot A_0)$

2.4 Bit B_3



Z czerownego otrzymujemy: $A_2 \cdot \overline{A_1}$
Z zielonego otrzymujemy: $A_3 \cdot A_1 \cdot \overline{A_0}$
Z żółtego otrzymujemy: $A_3 \cdot \overline{A_1} \cdot A_0$
Ostateczny wynik: $(A_2 \cdot \overline{A_1}) + (A_3 \cdot A_1 \cdot \overline{A_0}) + (A_3 \cdot \overline{A_1} \cdot A_0)$

2.5 Bit B_4



Z czerwonego otrzymujemy: $\overline{A_3} \cdot A_2 \cdot A_1$
Z zielonego otrzymujemy: $A_2 \cdot A_1 \cdot A_0$
Z żółtego otrzymujemy: $A_3 \cdot \overline{A_2} \cdot \overline{A_1}$
Z niebieskiego otrzymuujemy: $A_3 \cdot \overline{A_2} \cdot \overline{A_0}$
Ostateczny wynik: $(\overline{A_3} \cdot A_2 \cdot A_1) + (A_3 \cdot \overline{A_2} \cdot \overline{A_1}) + (A_2 \cdot A_1 \cdot A_0) + (A_3 \cdot \overline{A_2} \cdot \overline{A_0})$

2.6 Bit B_5

A_3A_2 \ A_1A_0	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	1	1	1	1
10	0	0	1	0

Z czerwonego otrzymujemy: $A_3 \cdot A_2$
Z zielonego otzrymujemy: $A_3 \cdot A_1 \cdot A_0$
Ostateczny wynik: $(A_3 \cdot A_2) + (A_3 \cdot A_1 \cdot A_0)$

3 Sprwadzanie wzorów do postaci NAND

3.1 Bit B_0

$$B_0 = A_3 + A_2 + A_1 + A_0 =$$
$$\overline{\overline{A_3 + A_2 + A_1 + A_0}} =$$
$$\overline{\overline{\boldsymbol{A_3}} \cdot \overline{\boldsymbol{A_2}} \cdot \overline{\boldsymbol{A_1}} \cdot \overline{\boldsymbol{A_0}}}$$

3.2 Bit B_1

$$B_1 = (\overline{A_3} \cdot \overline{A_1} \cdot \overline{A_0}) + (\overline{A_3} \cdot A_1 \cdot A_0) + (A_3 \cdot A_1 \cdot \overline{A_0}) + (\overline{A_3} \cdot \overline{A_2} \cdot \overline{A_1}) +$$
$$(\ A_3 \cdot \overline{A_2} \cdot \overline{A_0}) + (\overline{A_3} \cdot \overline{A_2} \cdot A_0) + (A_3 \cdot A_2 \cdot \overline{A_1} \cdot A_0) =$$

$$\overline{\overline{(\overline{A_3} \cdot \overline{A_1} \cdot \overline{A_0}) + (\overline{A_3} \cdot A_1 \cdot A_0) + (A_3 \cdot A_1 \cdot \overline{A_0}) + (\overline{A_3} \cdot \overline{A_2} \cdot \overline{A_1})} +$$
$$\overline{\overline{(\ A_3 \cdot \overline{A_2} \cdot \overline{A_0}) + (\overline{A_3} \cdot \overline{A_2} \cdot A_0) + (A_3 \cdot A_2 \cdot \overline{A_1} \cdot A_0)}} =$$

$$\overline{\overline{(\overline{\boldsymbol{A_3}} \cdot \overline{\boldsymbol{A_1}} \cdot \overline{\boldsymbol{A_0}})} \cdot \overline{(\overline{\boldsymbol{A_3}} \cdot \boldsymbol{A_1} \cdot \boldsymbol{A_0})} \cdot \overline{(\boldsymbol{A_3} \cdot \boldsymbol{A_1} \cdot \overline{\boldsymbol{A_0}})} \cdot \overline{(\overline{\boldsymbol{A_3}} \cdot \overline{\boldsymbol{A_2}} \cdot \overline{\boldsymbol{A_1}})} \cdot}$$
$$\overline{\overline{(\ \boldsymbol{A_3} \cdot \overline{\boldsymbol{A_2}} \cdot \overline{\boldsymbol{A_0}})} \cdot \overline{(\overline{\boldsymbol{A_3}} \cdot \overline{\boldsymbol{A_2}} \cdot \boldsymbol{A_0})} \cdot \overline{(\boldsymbol{A_3} \cdot \boldsymbol{A_2} \cdot \overline{\boldsymbol{A_1}} \cdot \boldsymbol{A_0})}}$$

3.3 Bit B_2

$$B_2 = (A_3 \cdot \overline{A_2}) + (\overline{A_2} \cdot A_1) + (A_3 \cdot A_1) + (\overline{A_3} \cdot A_2 \cdot \overline{A_1} \cdot A_0) =$$
$$\overline{\overline{(A_3 \cdot \overline{A_2}) + (\overline{A_2} \cdot A_1) + (A_3 \cdot A_1) + (\overline{A_3} \cdot A_2 \cdot \overline{A_1} \cdot A_0)}} =$$
$$\overline{\overline{(\boldsymbol{A_3} \cdot \overline{\boldsymbol{A_2}})} \cdot \overline{(\overline{\boldsymbol{A_2}} \cdot \boldsymbol{A_1})} \cdot \overline{(\boldsymbol{A_3} \cdot \boldsymbol{A_1})} \cdot \overline{(\overline{\boldsymbol{A_3}} \cdot \boldsymbol{A_2} \cdot \overline{\boldsymbol{A_1}} \cdot \boldsymbol{A_0})}}$$

3.4 Bit B_3

$$B_3 = (A_2 \cdot \overline{A_1}) + (A_3 \cdot A_1 \cdot \overline{A_0}) + (A_3 \cdot \overline{A_1} \cdot A_0) =$$

$$\overline{\overline{(A_2 \cdot \overline{A_1}) + (A_3 \cdot A_1 \cdot \overline{A_0}) + (A_3 \cdot \overline{A_1} \cdot A_0)}} =$$

$$\overline{\overline{(\boldsymbol{A_2 \cdot \overline{A_1}})} \cdot \overline{(\boldsymbol{A_3 \cdot A_1 \cdot \overline{A_0}})} \cdot \overline{(\boldsymbol{A_3 \cdot \overline{A_1} \cdot A_0})}}$$

3.5 Bit B_4

$$B_4 = (\overline{A_3} \cdot A_2 \cdot A_1) + (A_3 \cdot \overline{A_2} \cdot \overline{A_1}) + (A_2 \cdot A_1 \cdot A_0) + (A_3 \cdot \overline{A_2} \cdot \overline{A_0}) =$$

$$\overline{\overline{(\overline{A_3} \cdot A_2 \cdot A_1) + (A_3 \cdot \overline{A_2} \cdot \overline{A_1}) + (A_2 \cdot A_1 \cdot A_0) + (A_3 \cdot \overline{A_2} \cdot \overline{A_0})}} =$$

$$\overline{\overline{(\boldsymbol{\overline{A_3} \cdot A_2 \cdot A_1})} \cdot \overline{(\boldsymbol{A_3 \cdot \overline{A_2} \cdot \overline{A_1}})} \cdot \overline{(\boldsymbol{A_2 \cdot A_1 \cdot A_0})} \cdot \overline{(\boldsymbol{A_3 \cdot \overline{A_2} \cdot \overline{A_0}})}}$$

3.6 Bit B_5

$$B_5 = (A_3 \cdot A_2) + (A_3 \cdot A_1 \cdot A_0) =$$

$$\overline{\overline{(A_3 \cdot A_2) + (A_3 \cdot A_1 \cdot A_0)}} =$$

$$\overline{\overline{(\boldsymbol{A_3 \cdot A_2})} \cdot \overline{(\boldsymbol{A_3 \cdot A_1 \cdot A_0})}}$$

4 Implementacja układu

Układ składa się z 6 podobwodów. Jeśli do podobwodu przekazywane jest zaprzeczenie bitu A_i to znaczy, że poziom wyżej przy pomocy bramki NAND uzyskano zaprzeczenie.

4.1 Implementacja tanskodera bitu B_0

Pierwszy podukład wygląda następująco:

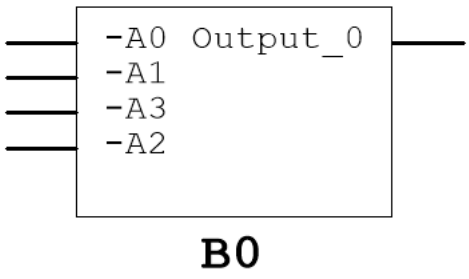


Figure 2: Układ reprezentowany jest jak powyżej; Wejście na cztery bity $\overline{A_0}, \overline{A_1}, \overline{A_2}, \overline{A_3}$; Wyjście wskazujące wartość bitu B_0

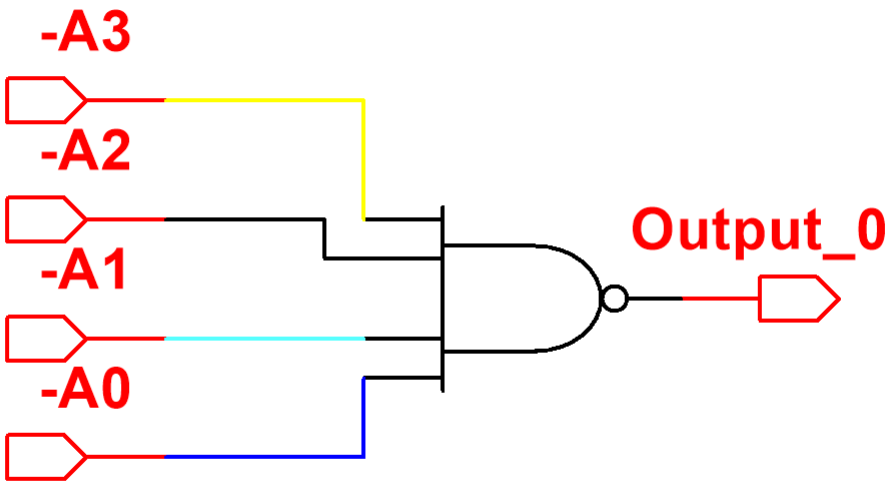


Figure 3: implementacja podobwodu B0

4.2 Implementacja tanskodera bitu B_1

Drugi podukład wygląda następująco:

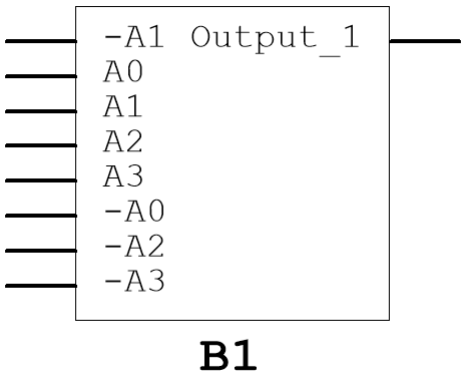


Figure 4: Układ reprezentowany jest jak powyżej; Wejście na osiem bitów : A_i dla i $\in$ {0,1,2,3} oraz ich zaprzeczenia; Wyjście wskazujące wartość bitu B_1

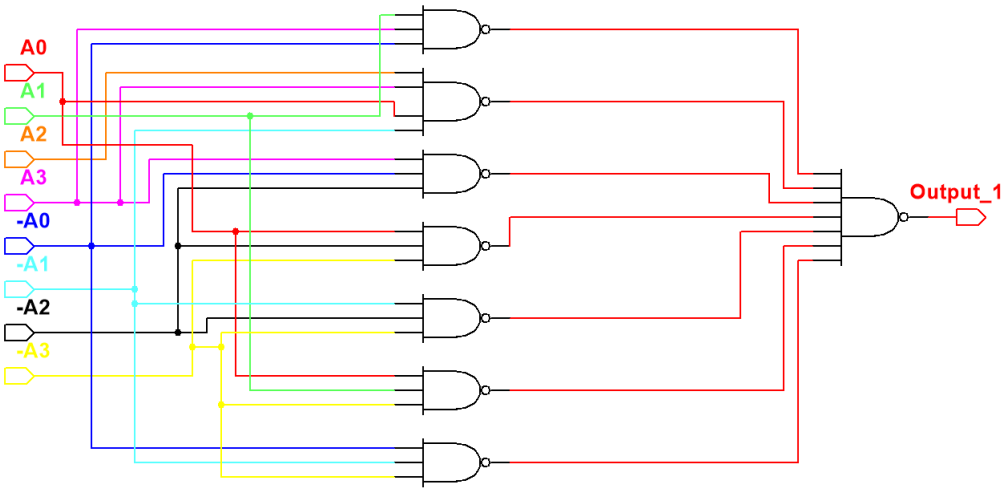


Figure 5: implementacja podobwodu B1

4.3 Implementacja tanskodera bitu B_2

Trzeci podukład wygląda następująco:

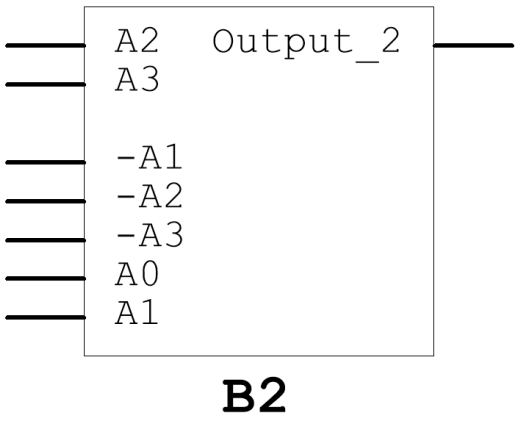


Figure 6: Układ reprezentowany jest jak powyżej; Wejście na siedem bitów : A_0, A_1, A_2, A_3 oraz $\overline{A_1}, \overline{A_2}, \overline{A_3}$; Wyjście wskazujące wartość bitu B_2

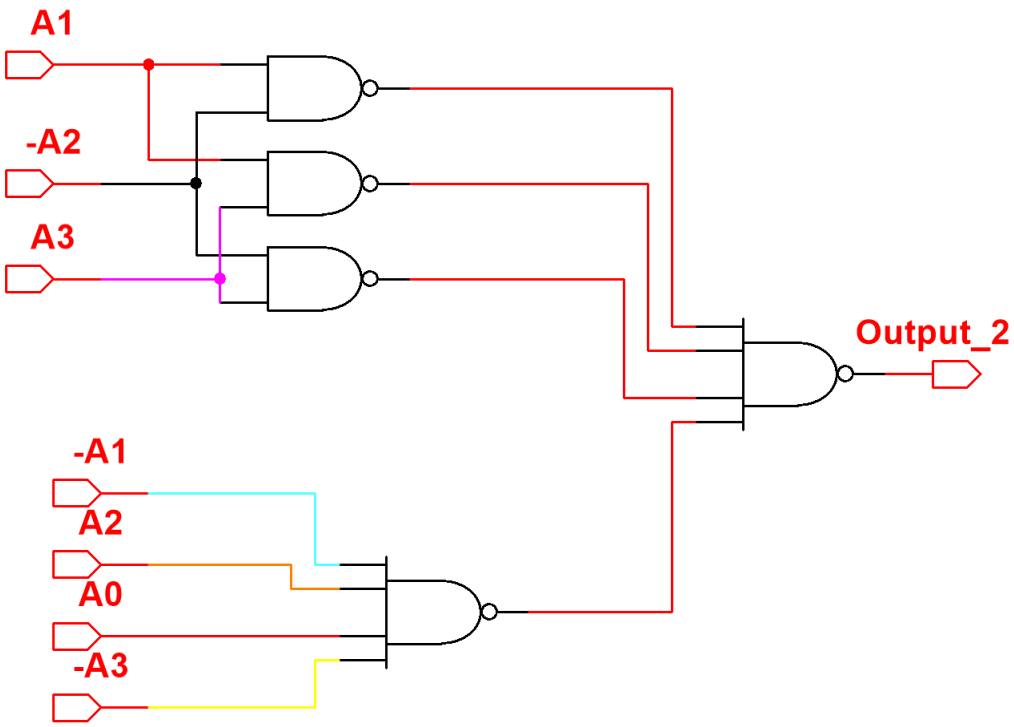


Figure 7: implementacja podobwodu B2

4.4 Implementacja tanskodera bitu B_3

Czwarty podukład wygląda następująco:

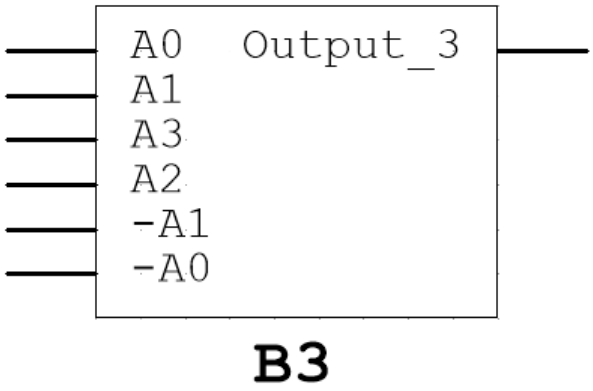


Figure 8: Układ reprezentowany jest jak powyżej; Wejście na cztery bity : A_0, A_1, A_2, A_3 oraz $\overline{A_0}, \overline{A_1}$; Wyjście wskazjące wartość bitu B_3

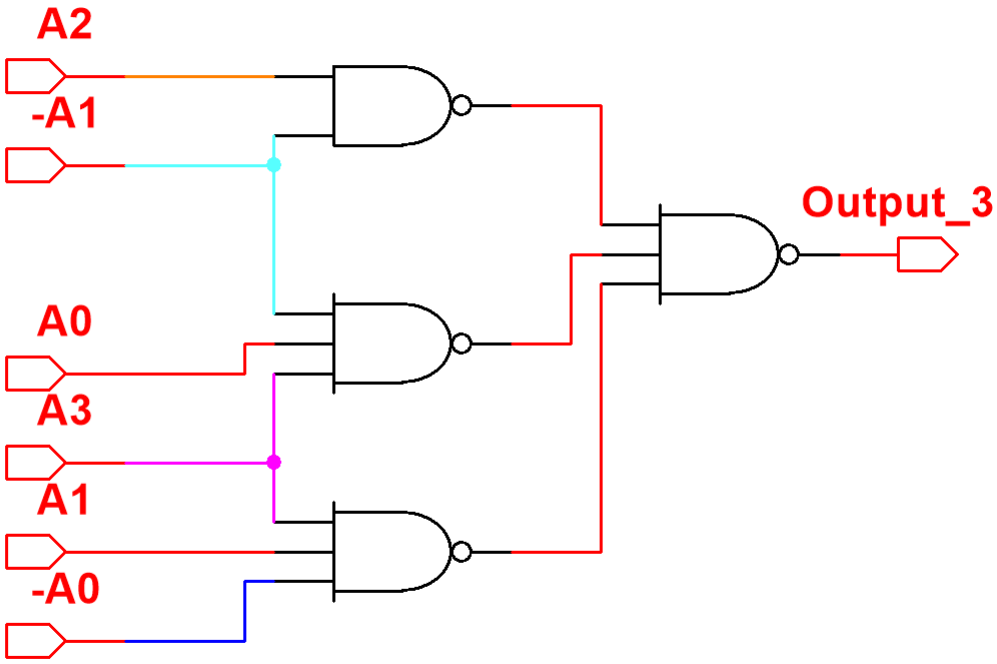


Figure 9: implementacja podobwodu B3

4.5 Implementacja tanskodera bitu B_4

Piąty podukład wygląda następująco:

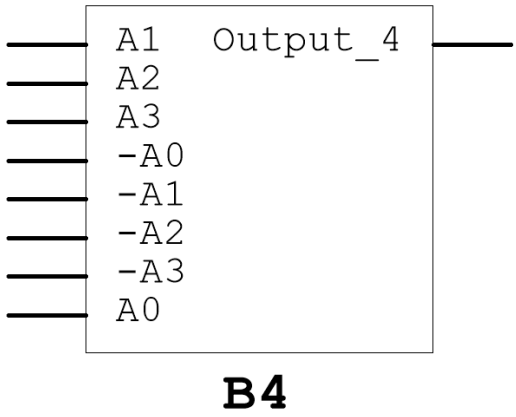


Figure 10: Układ reprezentowany jest jak powyżej; Wejście na osiem bitów : A_i dla i $\in$ {0,1,2,3} oraz ich zaprzeczenia; Wyjście wskazujące wartość bitu B_0

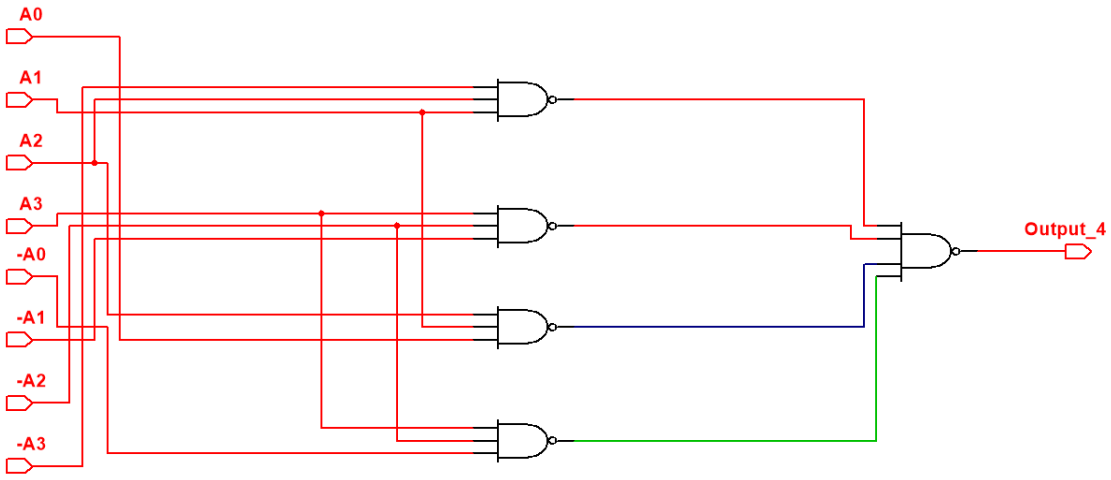


Figure 11: implementacja podobwodu B4

4.6 Implementacja tanskodera bitu B_5

Szósty podukład wygląda następująco:

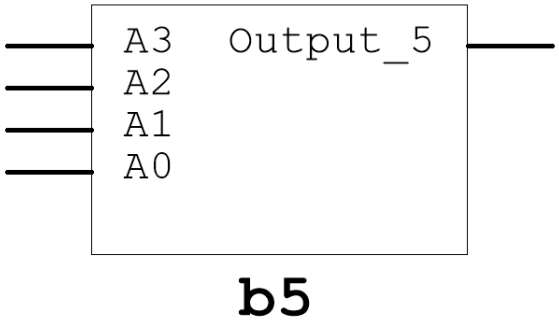


Figure 12: Układ reprezentowany jest jak powyżej; Wejście na cztery bity A_0, A_1, A_2, A_3 ; Wyjście wskazujące wartość bitu B_5

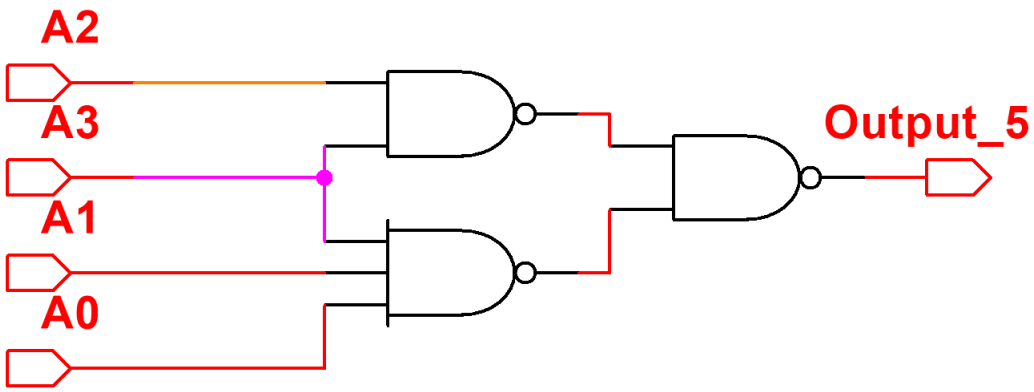


Figure 13: implementacja podobwodu B5

5 Testowanie układu

5.1 Generator słów

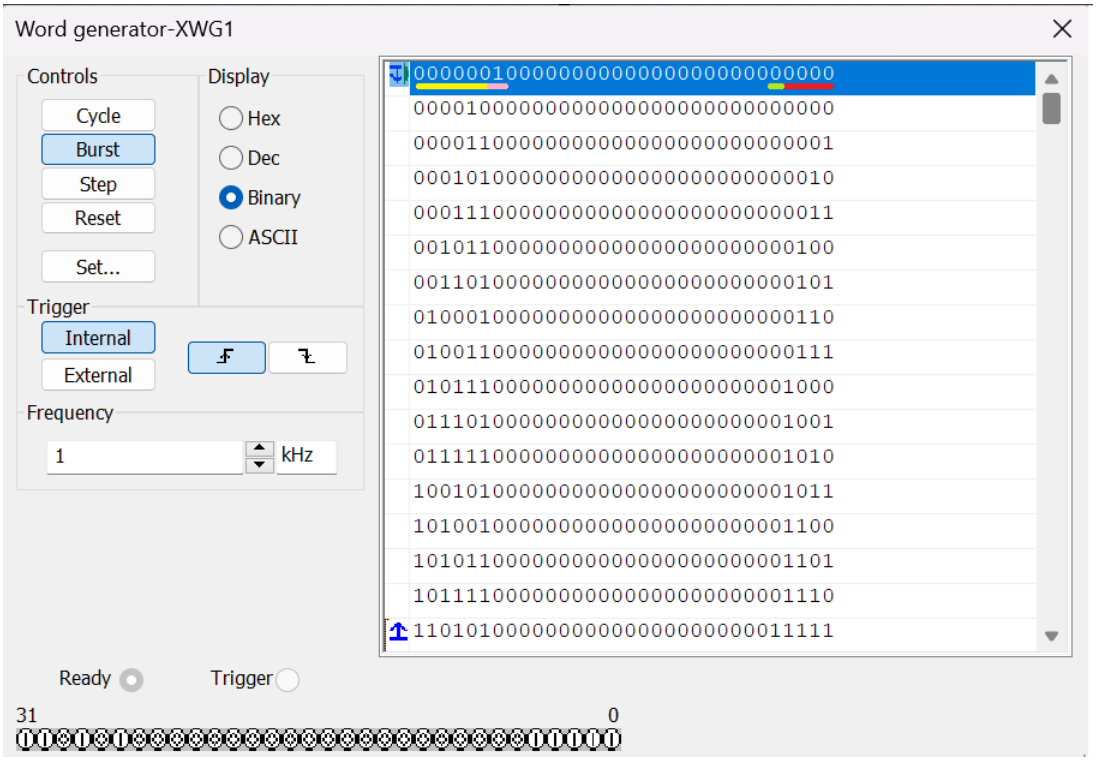


Figure 14: Generator bitów używanych do testowania układu

Generator wysyła sygnały siedemnascie razy - pierwszy wiersz sygnalizuje ropoczęcie testowania, kolejne zawierają między innymi bity wejściowe i oczekiwane bity wyjściowe.

- Bit startu - siódmy od prawej bit przekazywany do jednego z przerzutników (patrz poniżej), sygnalizuje rozpoczęcie testowania układu;
- Bity wejściowe - cztery pierwsze bity oznaczające kolejne liczby naturalne;
- Oczekiwane bity wyjściowe - ostatnie sześć bitów oznaczające kolejne liczby pierwsze czyli oczekiwane wyniki zwracane przez testowanu układ.
- Bit końcowy - piąty bit od lewej, sygnalizuje koniec testu drugiemu z przerzutników.

5.2 Układ wyświetlający wynik testu

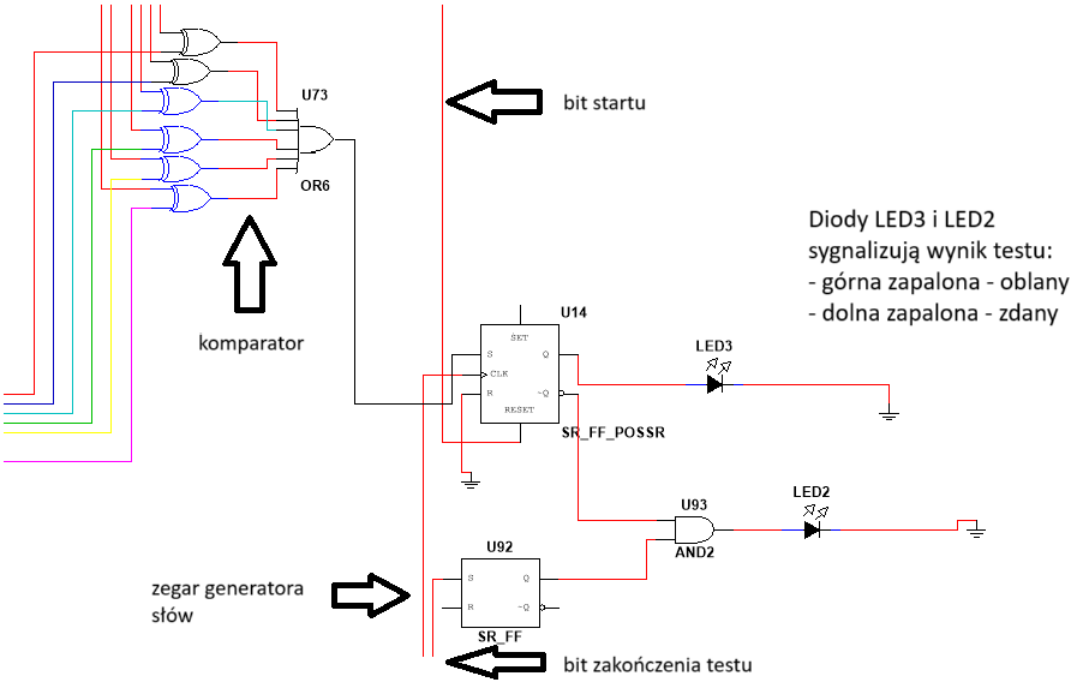


Figure 15: Układ testujący

- **górny przerzutnik** w momencie otrzymania przeczącego sygnału od komparatora zapala LED3; jeśli nie ma błędu przesyła sygnał do bramki AND2
- **dolny przerzutnik** w momencie zakończenia testu wysyła do AND2 odpowiedni sygnał

Komparator porównuje wartości oczekiwane z rzeczywistymi wyjścowych bitów i wysyłą sygnał do górnego przerzutnika. Przed otrzymaniem sygnału od komparatora generator słów wysyła sygnał o kolejnej liczbie. Jeśli przerzutnik otrzyma informację o błednym wyniku testu aktualizuje sygnał wysyłany z Q (i $\overline{Q}$) co zapala górną diodę. W przeciwnym wypadku, po poprawnym przetranskodowaniu liczb dolny przeprzutnik otrzymuje informację o zakończonym tescie i zapala diodę dolną.

5.3 Analizator stanu logicznego

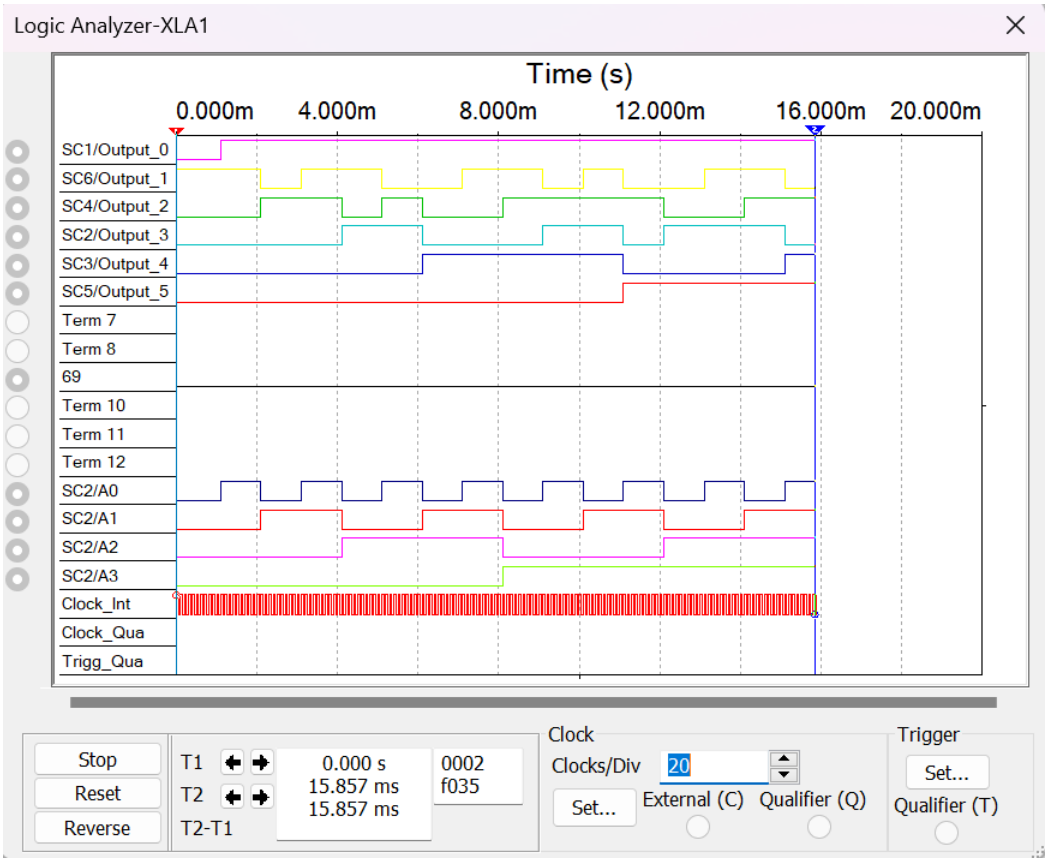


Figure 16: Sześć górnych linii to wartości bitów wyjściowych, środkowa to wartość komparatora, a cztery dolne to wartości bitów wejściowych

Twierdzenie Nyquista

Częstotliwość próbkowania musi być większa niż dwukrotność najwyższej składowej częstotliwości w mierzonym sygnale. U nas częstotliwość sygnału z generatora słów wynosi 1 kHz ,a częstotliwość próbkowania analizatora wynosi 10 kHz.

6 Wnioski i zastosowanie

6.1 Wnioski wyciągnięte podczas implementacji

- W przypadku tegu układu rozsądniejsze jest stosowanie wielu wejściowych bramek NAND - wyprowadzanie do postaci dysjunkcji jest prostsze i łatwiejsze w implementacji;
- Jeśli wejście do urządzenia jest niepodpięte do niczego NIE oznacza to, że nie otrzymuje ono żadnego sygnału - niepodłączone wejście działa jak antena.

6.2 Zastosowanie

Generator liczb pierwszych można wykorzystać do bezpiecznego szyfrowania wiadomości podczas przesyłania danych. Sposób na jaki wpadł nasz zespół działa następująco. Najpierw wymyśliliśmy klucz umożliwiający zakodowanie podstawowych liter alfabetu łacińskiego. Do uzyskania tego wykorzystujemy trzy ostatnie bity

jednej cyfry pierwszej i trzy pierwsze bity następnej liczby pierwszej w sposób ukazany w poniższej tabeli. W ten sposób uzyskujemy 35 możliwych kombinacji kodujących unikalne symbole.

Żółta oś	Bit B_5	Bit B_4	Bit B_3	Bit B_2	Bit B_1	Bit B_0	Czerwona oś
1	0	0	0	0	1	0	1
2	0	0	1	0	1	1	2
3	0	1	0	1	0	1	3
4	0	1	1	1	1	1	4
5	1	0	0	0	0	1	5
6	1	0	1	Nie dotyczy	Nie dotyczy	Nie dotyczy	Nie dotyczy
7	1	1	0	Nie dotyczy	Nie dotyczy	Nie dotyczy	Nie dotyczy

Table 1: Tabela z bitami, które generują cały język.

bity czerwone / bity żółte	1	2	3	4	5	6	7
1	a	b	c	d	e	z	!
2	f	g	h	i	j	" "	?
3	k	l	m	n	o	"endl"	@
4	p	q	r	s	t	+	*
5	u	v	w	x	y	-	/

Table 2: Rozszyfrowanie co oznaczają poszczególne konfiguracje bitów, z tabeli powyżej.

Dodatkowo, aby móc w dowolny sposób korzystać z podanych kombinacji tylko co druga szóstka bitów koduje nam litery, a reszta bitów ma na celu spowodować, że jedną literę będziemy w stanie kodować na więcej niż jeden sposób, co spowoduje, że najprostsze metody łamania szyfrów polegające na porównywaniu występowania danych liter będą nieskuteczne. Na przykład literę a można zakodować na 3 sposoby.

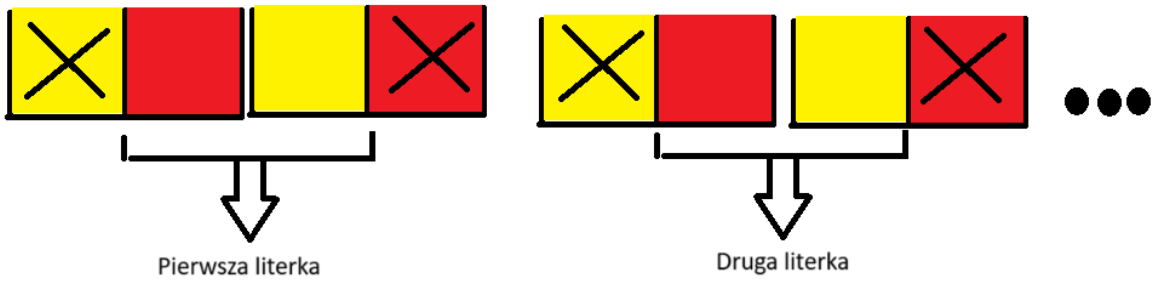


Figure 17: Sposób odszyfrowania ciągu cyfr

Przykładowe użycie maszyny szyfrującej do zakodowania słowa maciek na dwa różne sposoby. Wartości wejścia napisane będą w systemie dziesiętnym dla skrócenia zapisu.

9,6,0,0,5,3,10,5,12,0,0,7 => 100101010001000010000010001101000111011111001101101001000010000010010011 => maciek

2,8,0,1,15,2,3,4,6,1,0,8 => 000101010111000010000011110101000101000111001011010001000110000100101111 => maciek

Podsumowując największe zalety takiej metody szyfrowania to fakt, że do złamania go potrzeba aż trzech elementów. Zarówno wiadomości wejścia, naszego przyrządu do transforamcji cyfr oraz klucza, co powoduje, że jak na swoją prostotę działania jest on bardzo bezpieczny. Dodatkowo dzięki wprowadzeniu tego, że wiele różnych inputów jest w stanie być przerabianych na ten sam output zabezpieczamy się przed najprostszymi metodami łamania szyfrów, na które nie byłoby odporne na przykład przypisanie konretnym wartościom wejścia kolejnych liter.