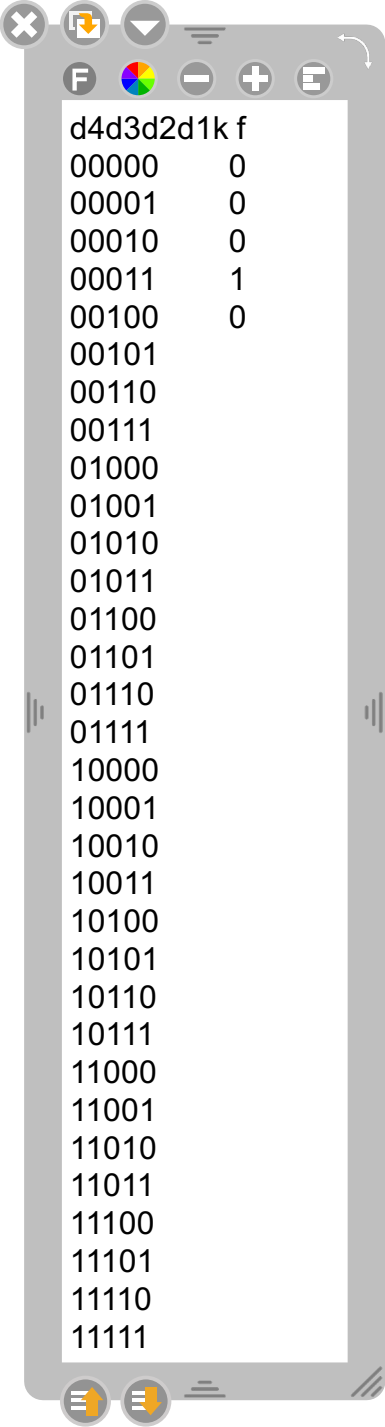


Zaprojektuj układ wysyłający „1” logiczną na wyjście układu kontrolnego jeśli którekolwiek drzwi samochodu są otwarte i kierowca siedzi w środku.

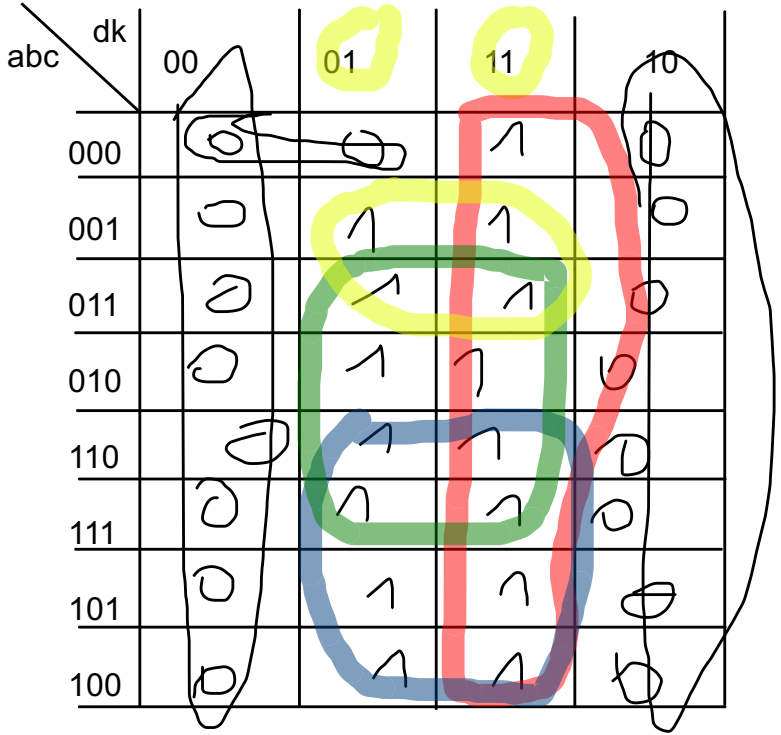


The image shows a screenshot of a logic simulator window. The window has a title bar with standard OS controls (close, maximize, minimize) and a toolbar with icons for file operations (F, color, zoom in, zoom out, print) and a refresh button. The main area displays a truth table for a 5-input logic function. The inputs are labeled d4, d3, d2, d1, and k, and the output is labeled f. The table lists all 32 possible combinations of the inputs, with the output value (0 or 1) for each combination. The output is 1 only for the combination where d4=0, d3=0, d2=0, d1=1, and k=0.

d4	d3	d2	d1	k	f
0	0	0	0	0	0
0	0	0	0	1	0
0	0	0	1	0	0
0	0	0	1	1	1
0	0	1	0	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	0	1	
0	1	0	1	0	
0	1	0	1	1	
0	1	1	0	0	
0	1	1	0	1	
0	1	1	1	0	
0	1	1	1	1	
1	0	0	0	0	
1	0	0	0	1	
1	0	0	1	0	
1	0	0	1	1	
1	0	1	0	0	
1	0	1	0	1	
1	0	1	1	0	
1	0	1	1	1	
1	1	0	0	0	
1	1	0	0	1	
1	1	0	1	0	
1	1	0	1	1	
1	1	1	0	0	
1	1	1	0	1	
1	1	1	1	0	
1	1	1	1	1	

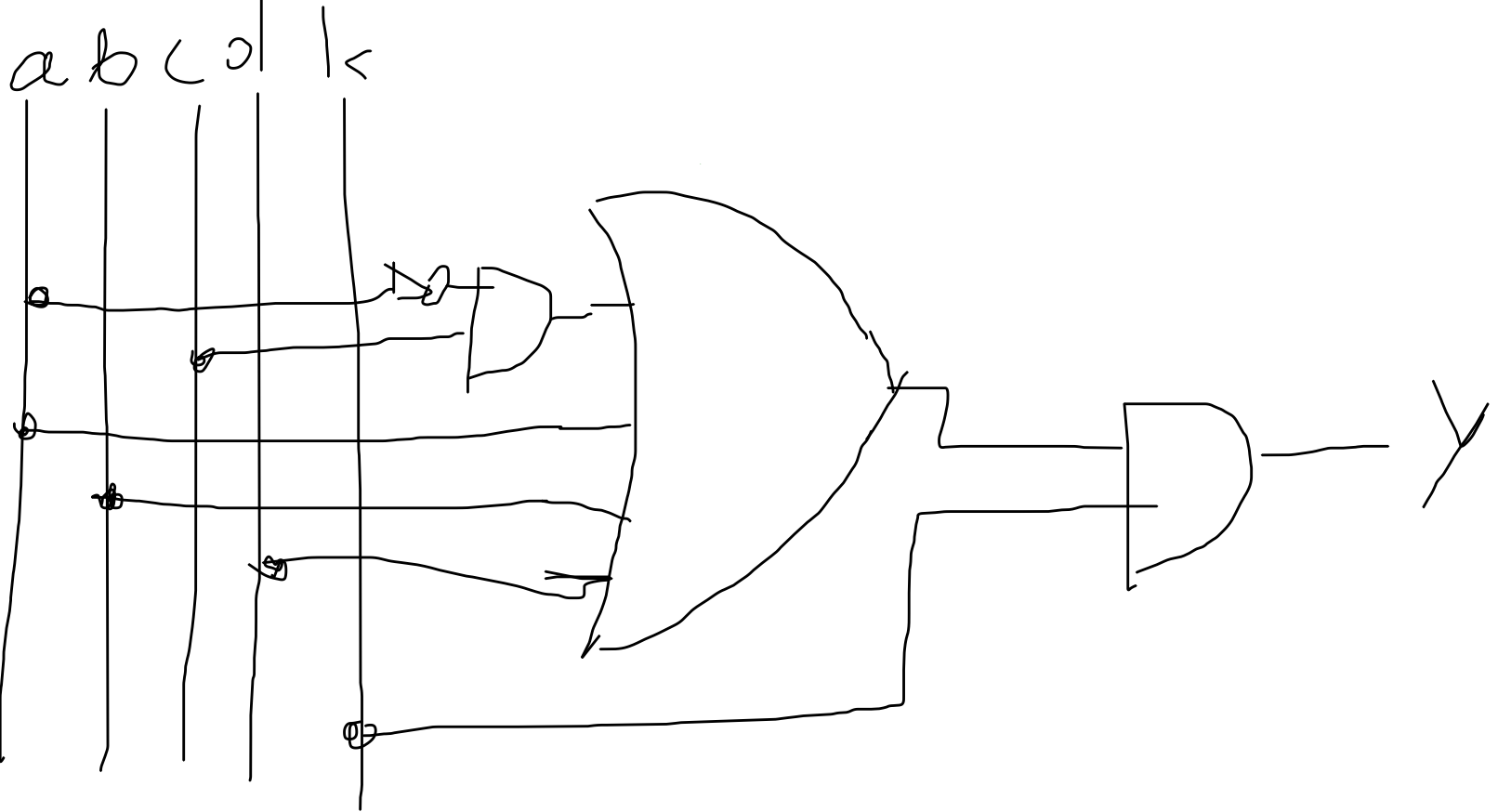
Zaprojektuj układ wysyłający „1” logiczną na wyjście układu kontrolnego jeśli którekolwiek drzwi samochodu są otwarte i kierowca siedzi w środku.

abcdk	f
00000	0
00001	0
00010	0
00011	1
00100	0
00101	1
00110	0
00111	1
01000	0
01001	1
01010	0
01011	1
01100	0
01101	1
01110	0
01111	1
10000	0
10001	1
10010	0
10011	1
10100	0
10101	1
10110	0
10111	1
11000	0
11001	1
11010	0
11011	1
11100	0
11101	1
11110	0
11111	1



Handwritten Boolean expression for the function f_A :

$$f_A = d \cdot k + a \cdot k + b \cdot k + \bar{a} \cdot c \cdot k = k(d + a + b + \bar{a} \cdot c)$$



Handwritten Boolean expression for the function f_k :

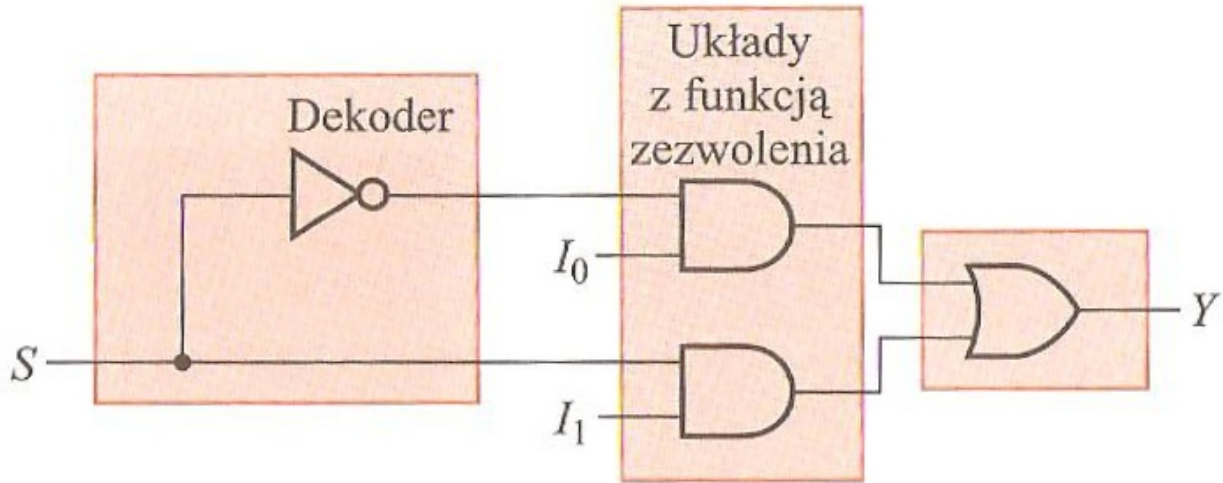
$$f_k = k(a + b + c + d)$$

Multiplekserem nazywamy układ kombinacyjny, który wybiera informację binarną z jednej z pośród wielu linii wejściowych i kieruje ją do jednego wyjścia. Wybór odpowiedniego wejścia jest sterowany zbiorem zmiennych wejściowych zwanych wejściami adresowymi. Zwykle jest 2^n linii wejściowych i n linii adresowych.

Tablicę prawdy oraz schemat logiczny multipleksera 2 na 1 linię można przedstawić w następujący sposób:

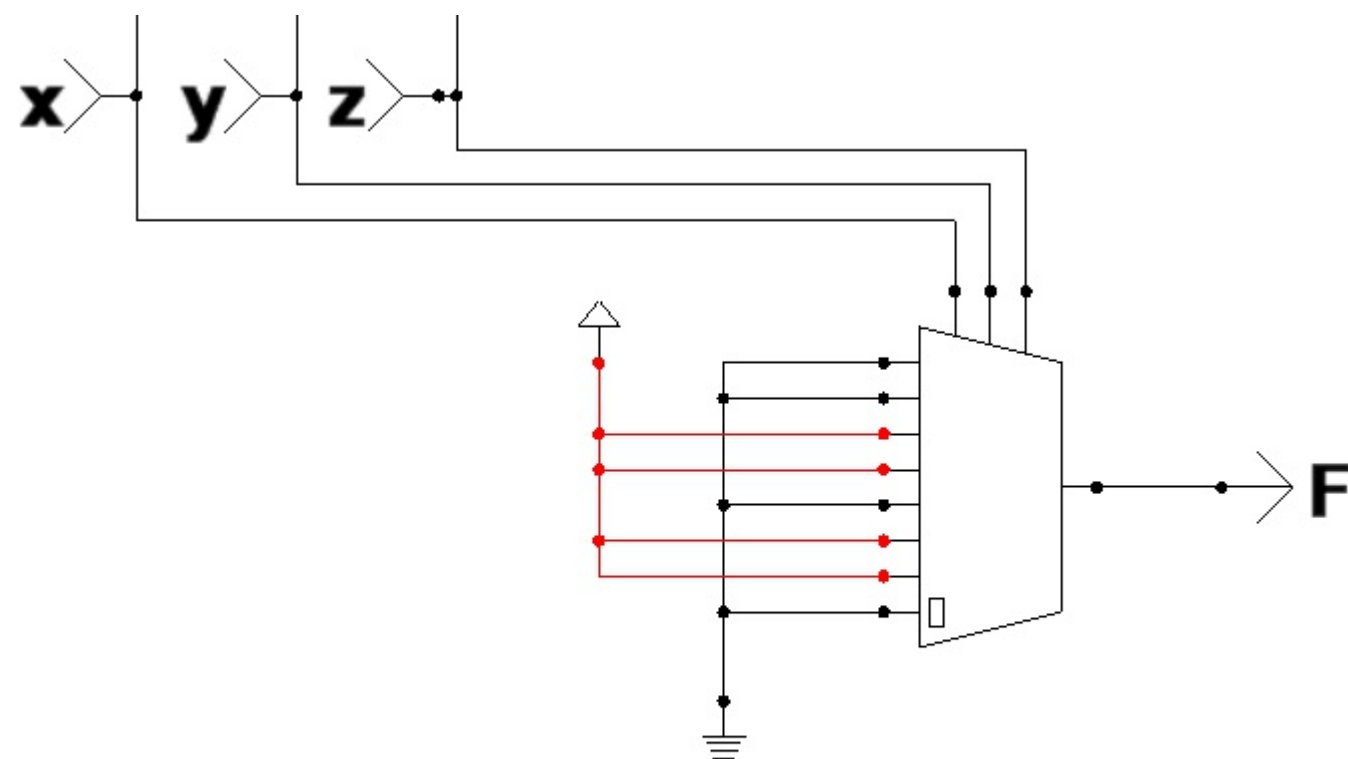
S	I ₀	I ₁	Y
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

$$Y = \bar{S}I_0 + SI_1$$



Jeśli wejście I_i jest równe 1, to minterm m_i jest dołączony do bramki OR, w przeciwnym wypadku minterm m_i zastępowany jest 0. Zastosowanie funkcji wymuszania stałych wartości logicznych na wejściach I_i zapewnia metodę implementacji funkcji boolowskiej n zmiennych na multiplekserze mającym n wejść adresowych i 2^n wejść danych po jednym dla każdego mintermu.

X	Y	Z	F
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	0



minterm - pełny iloczyn
logiczny wyrażenia

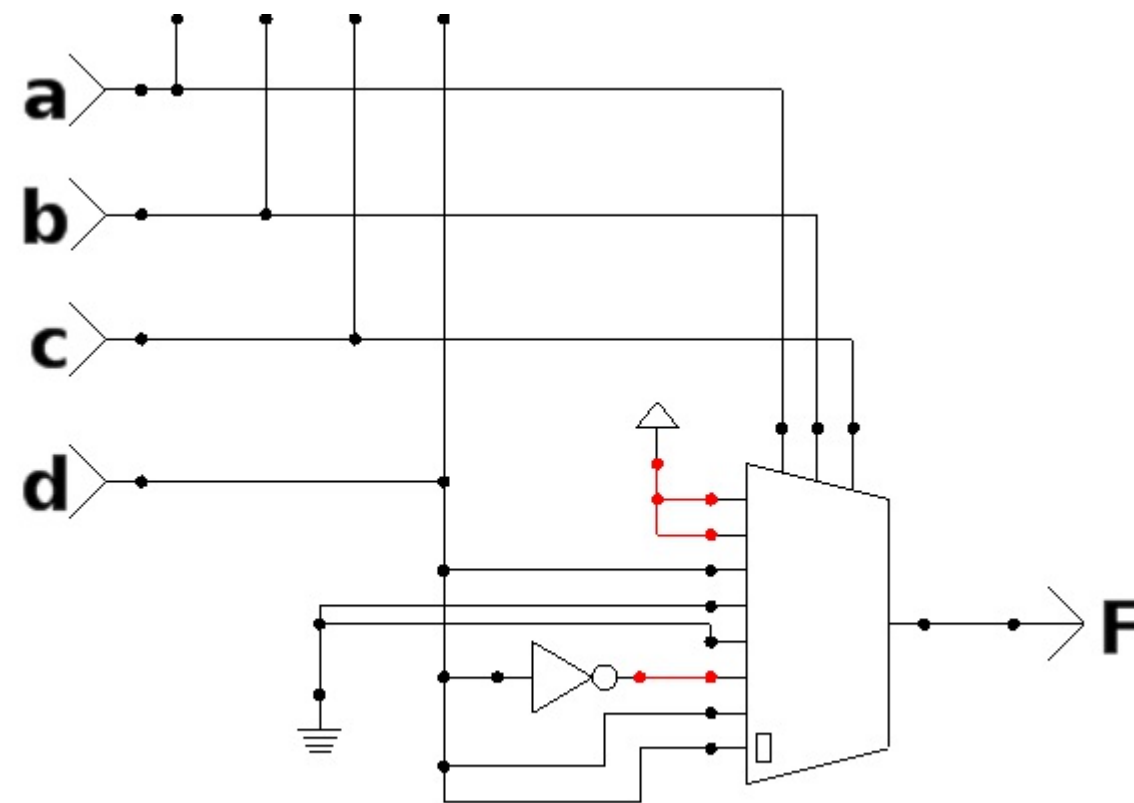
Przy pomocy multipleksera o n wejściach adresowych oraz 2^n wejściach danych można zrealizować funkcję boolowską o $n+1$ zmiennych.

Wykonać to można w następujący sposób: funkcję boolowską wpisujemy do tablicy prawdy, pierwsze n zmiennych z tablicy jest podanych na wejście adresowe multipleksera, dla każdej kombinacji zmiennych adresowych obliczamy wyjście jako funkcję ostatniej zmiennej.

Funkcja ta może być 0, 1, ta zmienna lub negacja tej zmiennej.

$$F(A, B, C, D) = \sum m(1, 3, 4, 11, 12, 13, 14, 15)$$

A	B	C	D	F	
0	0	0	0	0	F = D
0	0	0	1	1	
0	0	1	0	0	F = D
0	0	1	1	1	
0	1	0	0	1	F = ~D
0	1	0	1	0	
0	1	1	0	0	F = 0
0	1	1	1	0	
1	0	0	0	0	F = 0
1	0	0	1	0	
1	0	1	0	0	F = D
1	0	1	1	1	
1	1	0	0	1	F = 1
1	1	0	1	1	
1	1	1	0	1	F = 1
1	1	1	1	1	



Zaprojektuj konwerter kodu zamieniający kod 2 z 10 na naturalny kod binarny

y9 y8 y7 y6 y5 y4 y3 y2 y1 y0	a b c d
0 0 0 0 0 0 0 0 1 1	0 0 0 0
0 0 0 0 0 0 0 1 1 0	0 0 0 1
0 0 0 0 0 0 1 1 0 0	0 0 1 0
0 0 0 0 0 1 1 0 0 0	0 0 1 1
0 0 0 0 1 1 0 0 0 0	0 1 0 0
0 0 0 1 1 0 0 0 0 0	0 1 0 1
0 0 1 1 0 0 0 0 0 0	0 1 1 0
0 1 1 0 0 0 0 0 0 0	0 1 1 1
1 1 0 0 0 0 0 0 0 0	1 0 0 0

$$d = (y_0 \oplus y_1) + (y_2 \oplus y_3) + (y_4 \oplus y_5) + (y_6 \oplus y_7)$$

$$c = y_3 + y_7$$

$$b = y_5 + y_7$$

$$a = y_9$$

Zaprojektuj dekodер zamieniający informację z kodu naturalnego binarnego na kod 1 z 10

a b c d	y9 y8 y7 y6 y5 y4 y3 y2 y1 y0
0 0 0 0	0 0 0 0 0 0 0 0 0 1
0 0 0 1	0 0 0 0 0 0 0 0 1 0
0 0 1 0	0 0 0 0 0 0 0 1 0 0
0 0 1 1	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 0 1	0 0 0 0 1 0 0 0 0 0
0 1 1 0	0 0 0 1 0 0 0 0 0 0
0 1 1 1	0 0 1 0 0 0 0 0 0 0
1 0 0 0	0 1 0 0 0 0 0 0 0 0
1 0 0 1	1 0 0 0 0 0 0 0 0 0

DEKODER:

1. Odrzucający fałszywe kombinacje wejściowe Równania wprost z tabelki zerojedynkowej.

2. Nieodrzucający fałszywych kombinacji wejściowych Tabela Karnaugh, prostsze równania.

$y_0 = \overline{a} \overline{b} \overline{c} \overline{d}$

$y_1 = \overline{a} \overline{b} \overline{c} d$

$y_2 = \overline{a} \overline{b} c \overline{d}$

$y_3 = \overline{a} \overline{b} c d$

odrzucający

$y_0 = \overline{a} \overline{b} \overline{c} \overline{d} \quad y_1 = \overline{a} \overline{b} \overline{c} d$

$y_3 = \overline{a} \overline{b} c d$

$y_2 = \overline{a} \overline{b} c \overline{d}$

$y_4 = \overline{a} b \overline{c} \overline{d}$

$y_5 = \overline{a} b \overline{c} d$

$y_7 = b c d$

$y_6 = b c \overline{d}$

$y_8 = a d$

$y_9 = a \overline{d}$

Handwritten Karnaugh map for variables a, b, c, d. The map is a 4x4 grid with rows labeled ab (00, 01, 11, 10) and columns labeled cd (00, 01, 11, 10). The cells contain the output values y0 through y9, with some cells marked as '-' for unused combinations.

00	y0	y1	y2	y3
01	y4	y5	y6	y7
11	-	-	-	-
10	y8	y9	-	-

Nieodrzucający

Zaprojektuj układ konwertujący kod naturalny binarny NKB na kod Aikena

NKB (8421) $b_3 b_2 b_1 b_0$	Aikena (2421) $a_3 a_2 a_1 a_0$
0000	0000
0001	0001
0010	0010
0011	0011
0100	0100
0101	1011
0110	1100
0111	1101
1000	1110
1001	1111

a_0

$b_3 b_2 \backslash b_1 b_0$	00	01	11	10
00	0	1	1	0
01	0	1	1	0
11	-	-	-	-
10	0	1	-	-

a_1

$b_3 b_2 \backslash b_1 b_0$	00	01	11	10
00	0	0	0	0
01	1	0	1	1
11	-	-	-	-
10	1	1	-	-

a_2

$b_3 b_2 \backslash b_1 b_0$	00	01	11	10
00	0	0	1	1
01	0	1	0	0
11	-	-	-	-
10	1	1	-	-

a_3

$b_3 b_2 \backslash b_1 b_0$	00	01	11	10
00	0	0	0	0
01	0	1	1	1
11	-	-	-	-
10	1	1	-	-

$a_0 = b_0$

$a_1 = \overline{b_2} b_1 + b_3 \overline{b_1} + b_2 \overline{b_1} b_0$

$a_2 = b_2 \overline{b_0} + b_2 b_1 + b_3 \overline{b_1}$

$a_3 = b_3 \overline{b_1} + b_2 b_0 + b_2 b_1$