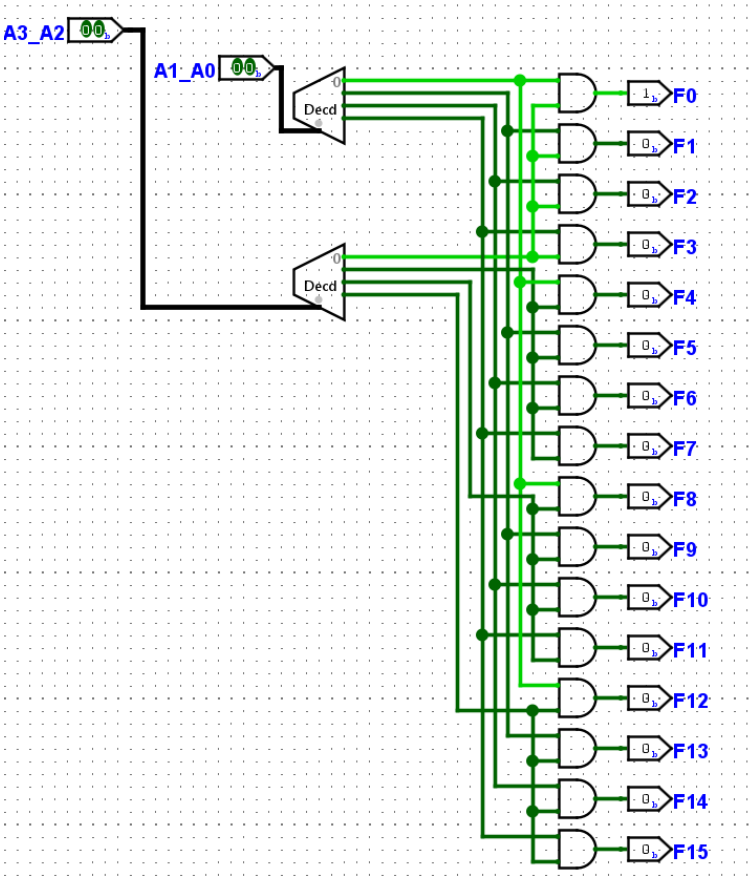
Nama : **Arzaka Raffan Mawardi** **Kode Asdos: RAI**
NPM : **2306152393**
Kelas : **PSD-C**

TUGAS MANDIRI 4

1. Gambarkanlah rancangan (sertakan pula screenshot rangkaian) dari:
   A. 4-to-16 line decoder dengan hanya menggunakan AND gate, NOT gate, dan 2-to-4 decoder (maksimal 4 buah)

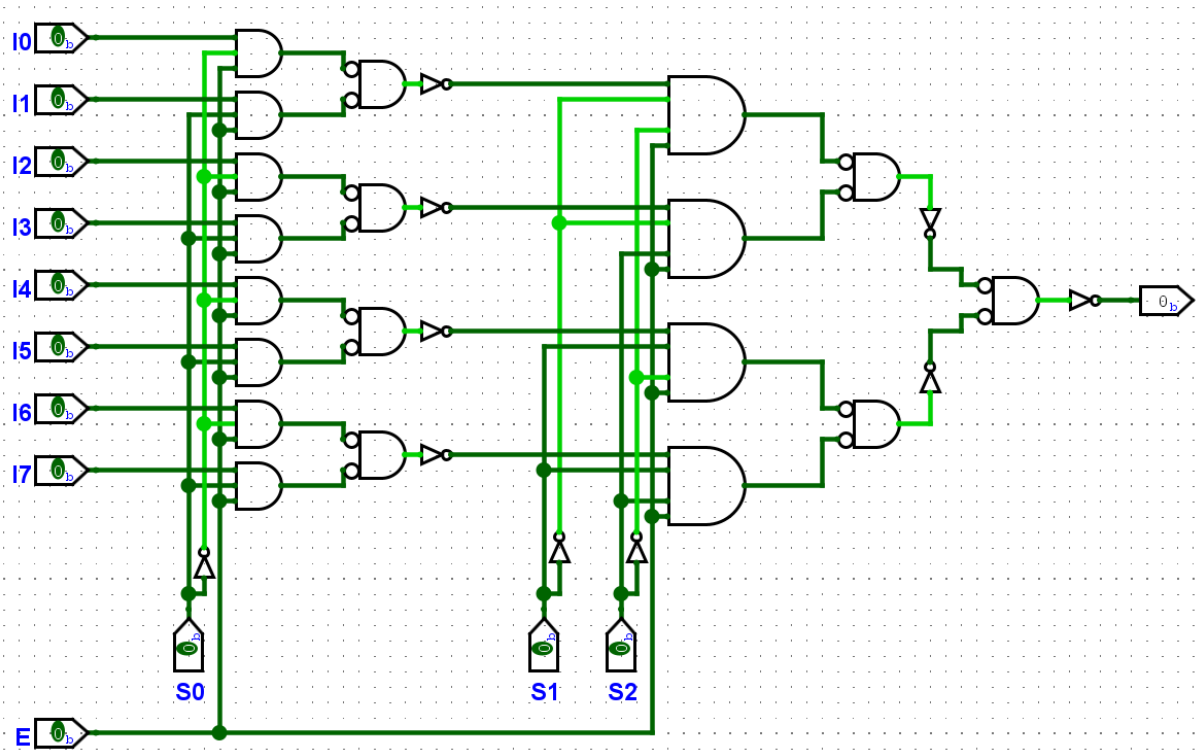


Truth Table

16 of 16 rows shown

A3_A2[1..0]	A1_A0[1..0]	F0	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	F13	F14	F15
0 0	0 0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0 0	0 1	0	1	0	0	0	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	0	0	0	0	0
0 0	1 1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
0 1	0 0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
0 1	0 1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
0 1	1 0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
0 1	1 1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
1 0	0 0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
1 0	0 1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
1 0	1 0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
1 0	1 1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
1 1	0 0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
1 1	0 1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
1 1	1 0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
1 1	1 1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1

B. 8-to-1 multiplexer dengan menggunakan AND gate, NOT gate, serta menggunakan enabler.



2. Buatlah masing-masing rangkaian berikut dengan menggunakan **4-to-1 line multiplexer**, dengan disertai tahapan *specification*, *formulation*, beserta *technology mapping*: (sertakan *screenshot* masing-masing rangkaian)

a. Rangkaian yang mengubah input **4-bit Binary Code** menjadi **4-*bit Gray Code***.

1. Specification
    a. 4-bit Binary Code to 4-bit Gray Code
    b. Transforms Binary code to Gray Code
    c. 4-bit Binary code from 0 through 15: 4-bit patterns from 0000 to 1111, respectively.
    d. 4-bit Gray Code from 0 trough 15: 4-bit patterns from 0000 to 1000, Gray code is an unweighted code
    e. Implementation:
    - Using 4-to-1 Line Multiplexers, OR gates, AND Gates, and NOT Gates
    - Multiple-level circuit

2. Formulations
    - Conversions of any 4-bit code can be formulated by truth table
    - Variables:
        - Binary Code: A, B, C, D
        - Gray Code: W, X, Y, Z

Truth Table

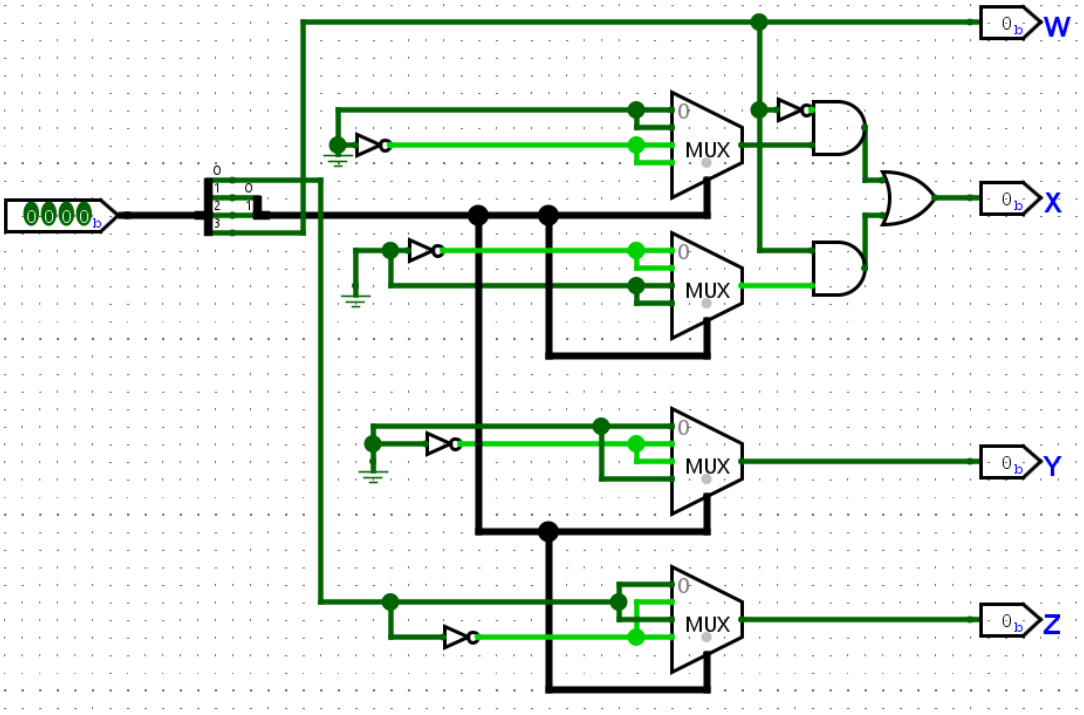
	Input Binary Code				Gray Code			
Index	A	B	C	D	W	X	Y	Z
0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	1
2	0	0	1	0	0	0	1	1
3	0	0	1	1	0	0	1	0
4	0	1	0	0	0	1	1	0
5	0	1	0	1	0	1	1	1
6	0	1	1	0	0	1	0	1
7	0	1	1	1	0	1	0	0
8	1	0	0	0	1	1	0	0
9	1	0	0	1	1	1	0	1
10	1	0	1	0	1	1	1	1
11	1	0	1	1	1	1	1	0
12	1	1	0	0	1	0	1	0
13	1	1	0	1	1	0	1	1
14	1	1	1	0	1	0	0	1
15	1	1	1	1	1	0	0	0

Optimization:

For a more simplified conversion, make a rudimentary function. For this case, D is used to be a rudimentary function for X, Y, and Z

Binary ABCD	Gray Code WXYZ	Rudimentary Function of D for X	Rudimentary Functions of D for Y	Rudimentary Functions of D for Z
0000	0000	F = 0	F = 0	F = D
0001	0001			
0010	0011	F = 0	F = 1	F = D’
0011	0010			
0100	0110	F = 1	F = 1	F = D
0101	0111			
0110	0101	F =1	F = 0	F = D’
0111	0100			
1000	1100	F = 1	F = 0	F = D
1001	1101			
1010	1111	F = 1	F = 1	F = D’
1011	1110			
1100	1010	F = 0	F = 1	F = D
1101	1011			
1110	1001	F = 0	F = 0	F = D’
1111	1000			

3. Technology Mapping



Truth Table

16 of 16 rows shown

a[3..0]	W	X	Y	Z
0 0 0 0	0	0	0	0
0 0 0 1	0	0	0	1
0 0 1 0	0	0	1	1
0 0 1 1	0	0	1	0
0 1 0 0	0	1	1	0
0 1 0 1	0	1	1	1
0 1 1 0	0	1	0	1
0 1 1 1	0	1	0	0
1 0 0 0	1	1	0	0
1 0 0 1	1	1	0	1
1 0 1 0	1	1	1	1
1 0 1 1	1	1	1	0
1 1 0 0	1	0	1	0
1 1 0 1	1	0	1	1
1 1 1 0	1	0	0	1
1 1 1 1	1	0	0	0

Note:

For Y and Z, only one 4-to-1 multiplexers that is needed and necessary because the rudimentary for A = 0 and A = 1 is exactly the same.

But for X, the rudimentary shows that A = 0 isn't the same as A = 1. Therefore, gates such as OR, AND, and NOT gates are needed to make the functions work and suit the truth table.

b. *Rangkaian yang menerima **4-bit Binary Code** kemudian mengubahnya menjadi **3-bit output XYZ**, dengan masing-masing fungsi sebagai berikut:*

- $X(A, B, C, D) = \Sigma\, m(0, 1, 6, 7, 8, 9, 14, 15)$
- $Y(A, B, C, D) = \Sigma\, m(2, 3, 10, 11)$
- $Z(A, B, C, D) = \Sigma\, m(0, 2, 5, 7, 9, 11, 12, 14)$

1. Specification

a. 4-bit Binary code to 3-bit output

b. Transform 4-bit Binary code to 3-bit output as shown before in the functions.

c. 4-bit Binary code from 0 through 15: 4-bit patterns from 0000 to 1111, respectively

d. Implementation:
- Using 4-to-1 Line Multiplexers, OR gates, AND Gates, and NOT Gates
- Multiple-level circuit

2. Formulations
- Conversions of any 4-bit code can be formulated by truth table
- Variables:
  a. Binary Code: A, B, C, D
  b. 3-bit Output: X, Y, Z

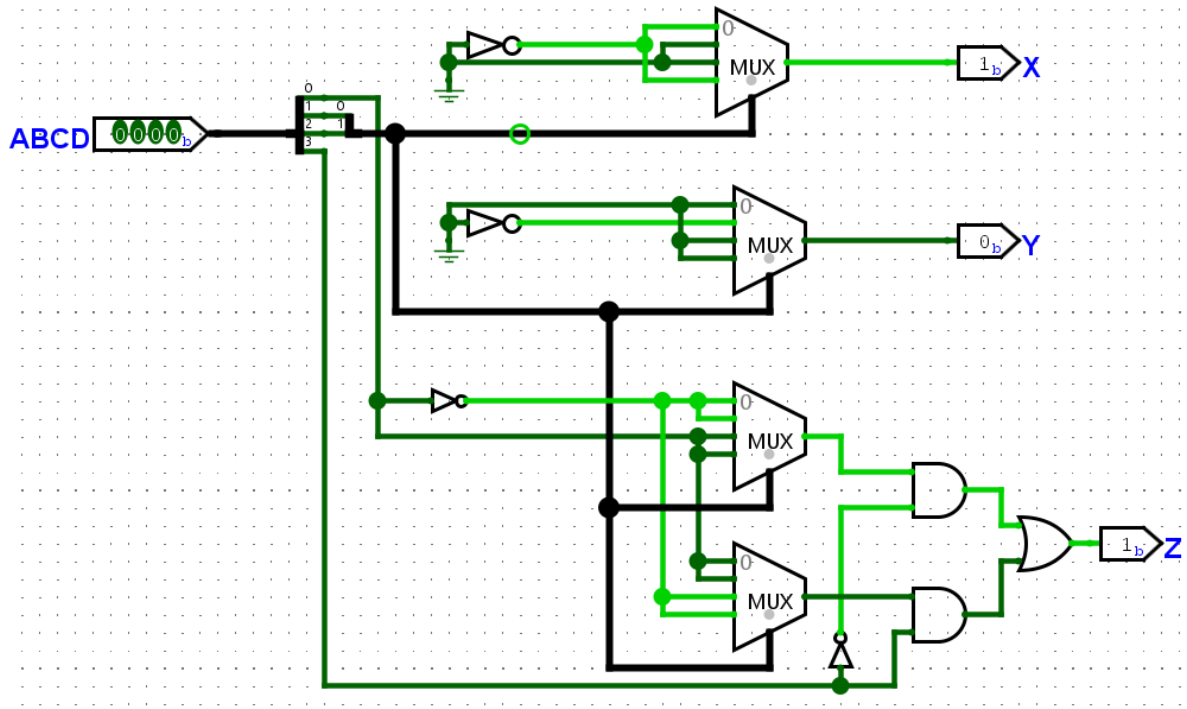
	Input Binary Code				Output		
Index	A	B	C	D	X	Y	Z
0	0	0	0	0	1	0	1
1	0	0	0	1	1	0	0
2	0	0	1	0	0	1	1
3	0	0	1	1	0	1	0
4	0	1	0	0	0	0	0
5	0	1	0	1	0	0	1
6	0	1	1	0	1	0	0
7	0	1	1	1	1	0	1
8	1	0	0	0	1	0	0
9	1	0	0	1	1	0	1
10	1	0	1	0	0	1	0
11	1	0	1	1	0	1	1
12	1	1	0	0	0	0	1
13	1	1	0	1	0	0	0
14	1	1	1	0	1	0	1
15	1	1	1	1	1	0	0

Optimization:

For a more simplified conversion, make a rudimentary function. For this case, D is used to be a rudimentary function for X, Y, and Z

Binary ABCD	Output XYZ	Rudimentary Function of D for X	Rudimentary Functions of D for Y	Rudimentary Functions of D for Z
0000	101	F = 1	F = 0	F = D’
0001	100			
0010	011	F = 0	F = 1	F = D’
0011	010			
0100	000	F = 0	F = 0	F = D
0101	001			
0110	100	F = 1	F = 0	F = D
0111	101			
1000	100	F = 1	F = 0	F = D
1001	101			
1010	010	F = 0	F = 1	F = D
1011	011			
1100	001	F = 0	F = 0	F = D’
1101	000			
1110	101	F = 1	F = 0	F = D’
1111	100			

3. Technology Mapping



Truth Table

16 of 16 rows shown

ABCD[3..0]	X	Y	Z
0 0 0 0	1	0	1
0 0 0 1	1	0	0
0 0 1 0	0	1	1
0 0 1 1	0	1	0
0 1 0 0	0	0	0
0 1 0 1	0	0	1
0 1 1 0	1	0	0
0 1 1 1	1	0	1
1 0 0 0	1	0	0
1 0 0 1	1	0	1
1 0 1 0	0	1	0
1 0 1 1	0	1	1
1 1 0 0	0	0	1
1 1 0 1	0	0	0
1 1 1 0	1	0	1
1 1 1 1	1	0	0

Note:

For X and Y, only one 4-to-1 multiplexers that is needed and necessary because the rudimentary for A = 0 and A = 1 is exactly the same.

But for Z, the rudimentary shows that A = 0 isn't the same as A = 1. Therefore, gates such as OR, AND, and NOT gates are needed to make the functions work and suit the truth table.

3. Lakukan tahapan *formulation*, *optimization*, dan *technology mapping* untuk masing-masing fungsi berikut: (sertakan *screenshot* masing-masing rangkaian)
   a. $F(A, B, C, D) = \Sigma\, m(0, 2, 8, 9, 10, 14)$,
   $d(A, B, C, D) = \Sigma\, m(1, 4, 5, 7, 15)$

Formulation

Truth Table

Index	A	B	C	D	F
0	0	0	0	0	1
1	0	0	0	1	X
2	0	0	1	0	1
3	0	0	1	1	0
4	0	1	0	0	X
5	0	1	0	1	X
6	0	1	1	0	0
7	0	1	1	1	X
8	1	0	0	0	1
9	1	0	0	1	1
10	1	0	1	0	1
11	1	0	1	1	0
12	1	1	0	0	0
13	1	1	0	1	0
14	1	1	1	0	1
15	1	1	1	1	X

Optimization

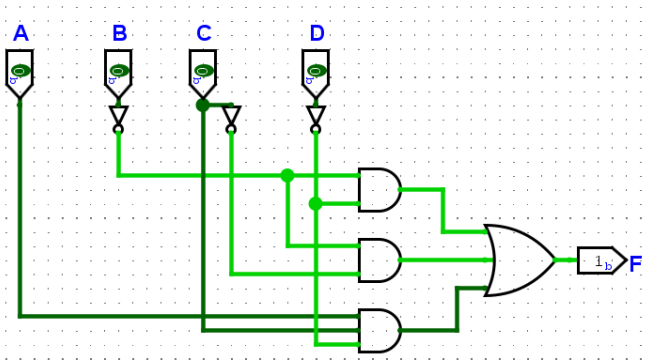
Making the K-Map for optimizing the functions

	C’		C		
A’	1 (0)	X (1)	0 (3)	1 (2)	B’
	X (4)	X (5)	X (7)	0 (6)	B
A	0 (12)	0 (13)	X (15)	1 (14)	
	1 (8)	1 (9)	0 (11)	1 (10)	B’
	D’	D		D’	

Optimized functions will be:

F = B’D’ + B’C’ + ACD’

Technology Mapping



Truth Table

A	B	C	D	F
0	0	0	0	1
0	0	0	1	1
0	0	1	0	1
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	1
1	0	0	1	1
1	0	1	0	1
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	1
1	1	1	1	0

$$F = \overline{B}\cdot\overline{D} + \overline{B}\cdot\overline{C} + A\cdot\overline{C}\cdot\overline{D}$$

b. $F(A, B, C, D) = \Pi M(3, 4, 5, 7, 8, 10, 11)$
$d(A, B, C, D) = \Sigma m(1, 2, 6, 9, 15)$

Formulation

Truth Table

Index	A	B	C	D	F
0	0	0	0	0	1
1	0	0	0	1	X
2	0	0	1	0	X
3	0	0	1	1	0
4	0	1	0	0	0
5	0	1	0	1	0
6	0	1	1	0	X
7	0	1	1	1	0
8	1	0	0	0	0
9	1	0	0	1	X
10	1	0	1	0	0
11	1	0	1	1	0
12	1	1	0	0	1
13	1	1	0	1	1
14	1	1	1	0	1
15	1	1	1	1	X

Optimization

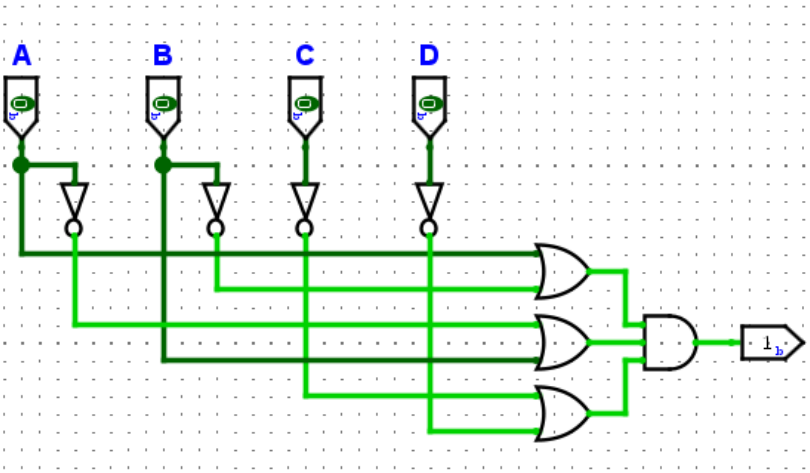
Making the K-Map for optimizing the functions

	C		C'		
A	1 (0)	X (1)	0 (3)	X (2)	B
	0 (4)	0 (5)	0 (7)	X (6)	B'
A'	1 (12)	1 (13)	X (15)	1 (14)	
	0 (8)	X (9)	0 (11)	0 (10)	B
	D	D'		D	

Optimized functions will be:

F = (A+B').(A'+B).(C'+D')

Technology Mapping



Truth Table

A	B	C	D	F
0	0	0	0	1
0	0	0	1	1
0	0	1	0	1
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	0	1	0
1	0	1	0	0
1	0	1	1	0
1	1	0	0	1
1	1	0	1	1
1	1	1	0	1
1	1	1	1	0

$$F = (A+\overline{B})\cdot(\overline{A}+B)\cdot(\overline{C}+\overline{D})$$

Link to author's Logisim file: [Link to Logisim (Google drive)]

😉 PSD IS FUN 😉