**Układy cyfrowe i systemy wbudowane**

**SPRAWOZDANIE NR 1**

**PROWADZĄCY:**  
DR INŻ. JACEK MAZURKIEWICZ

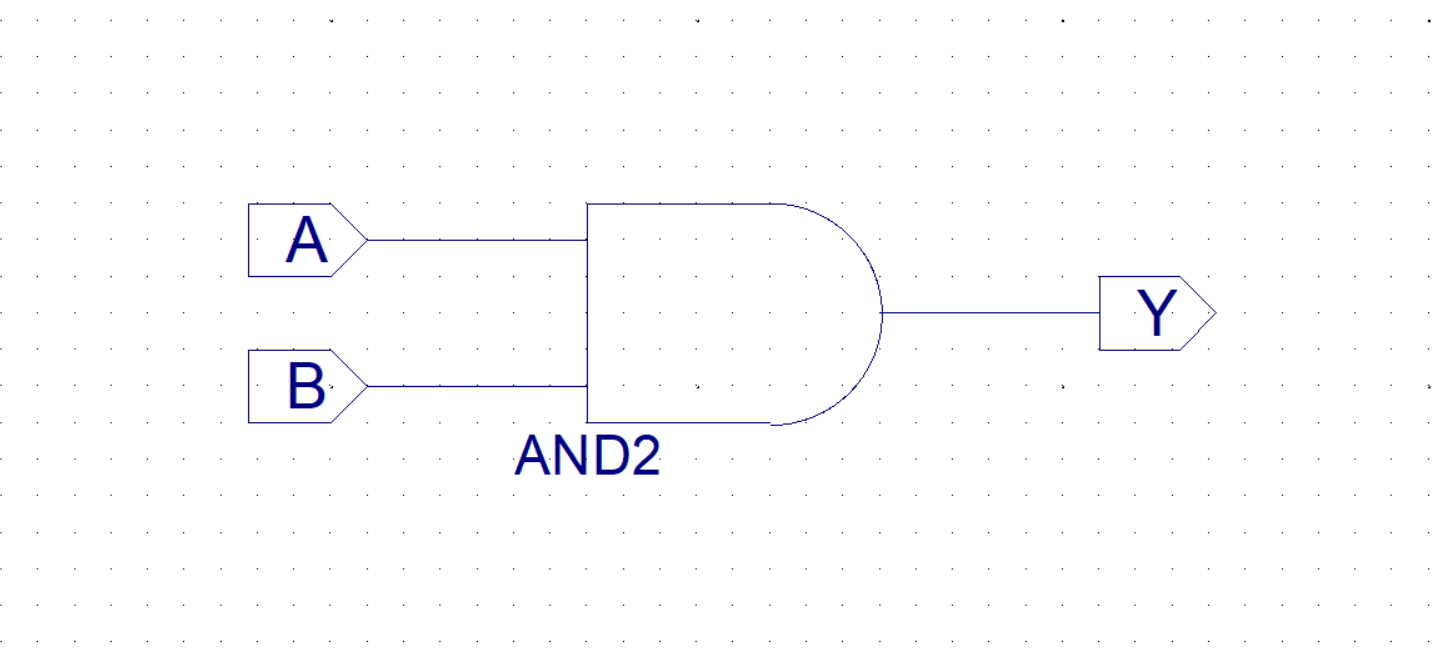
**AUTORZY:**  
MATEUSZ GRUSZKA 249448  
BARTOSZ RUDNIK 248893  
  
**Grupa:** C

**TERMIN ZAJĘĆ:**  
WT 9:15 TP

WROCŁAW, 23.10.2020 r.

1. **Cel ćwiczenia**  
    Pierwszym z celów przeprowadzonego ćwiczenia laboratoryjnego było zapoznanie się z zintegrowanym środowiskiem Xilinx ISE służącym do wykonywania wszystkich operacji związanych z opracowaniem projektu układu cyfrowego oraz jego implementacji w układzie CPLD lub FPGA. Drugim z celów ćwiczenia było zaimplementowanie w środowisku Xilinx zadanej funkcji logicznej, układu translatora kodu oraz dowolnej bramki logicznej.   
   W celu zrealizowanie kolejnych układów należało dla poszczególnych zadań stworzyć nowy projekt w środowisku Xilinx ISE. Następnym krokiem było wykonanie schematów poszczególnych układów. Kolejnym krokiem, który należało wykonać dla poszczególnych układów było stworzenie plików VHDL zawierających pobudzenia testowe dla układu. W celu wykonania symulacji działania układu należało wybrać plik VHDL jako obiekt symulacji, a następnie wybrać opcję Simulate Behavioral Model z zakładki Design.
2. **Bramka logiczna**  
   Do zrealizowania wybraliśmy bramkę logiczną AND.

|  |  |  |
| --- | --- | --- |
| **P** | **Q** | **P AND Q** |
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

Tabela 1 Tablica prawdy dla bramki logicznej AND  
  


Rysunek 1 Schemat bramki logicznej AND

**LIBRARY** ieee;

**USE** ieee.std\_logic\_1164.**ALL**;

**USE** ieee.numeric\_std.**ALL**;

**LIBRARY** UNISIM;

**USE** UNISIM.Vcomponents.**ALL**;

**ENTITY** schemat1\_schemat1\_sch\_tb **IS**

**END** schemat1\_schemat1\_sch\_tb;

**ARCHITECTURE** behavioral **OF** schemat1\_schemat1\_sch\_tb **IS**

**COMPONENT** schemat1

**PORT**( Y : **OUT** STD\_LOGIC;

A : **IN** STD\_LOGIC;

B : **IN** STD\_LOGIC);

**END** **COMPONENT**;

**SIGNAL** Y : STD\_LOGIC;

**SIGNAL** A : STD\_LOGIC;

**SIGNAL** B : STD\_LOGIC;

**BEGIN**

UUT: schemat1 **PORT** **MAP**(

Y => Y,

A => A,

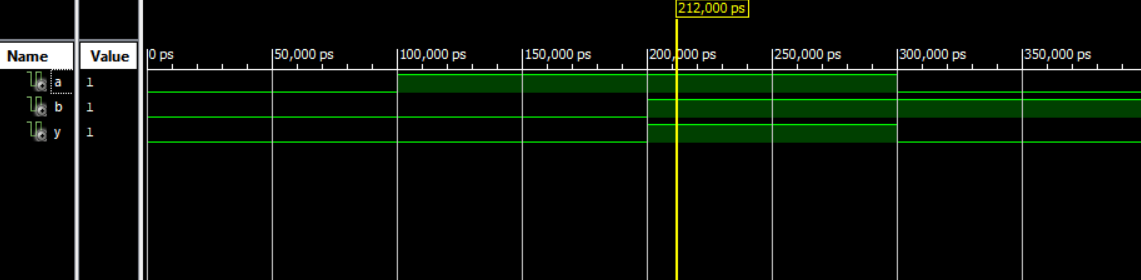
B => B

);

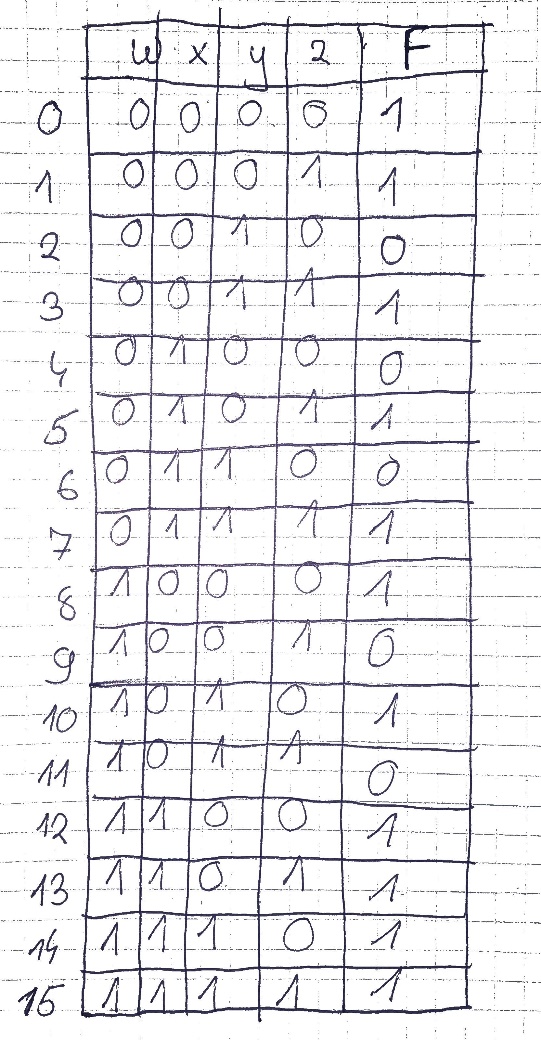
A <= '0', '1' **after** 100 ns, '0' **after** 300 ns;

B <= '0', '1' **after** 200 ns, '0' **after** 400 ns;

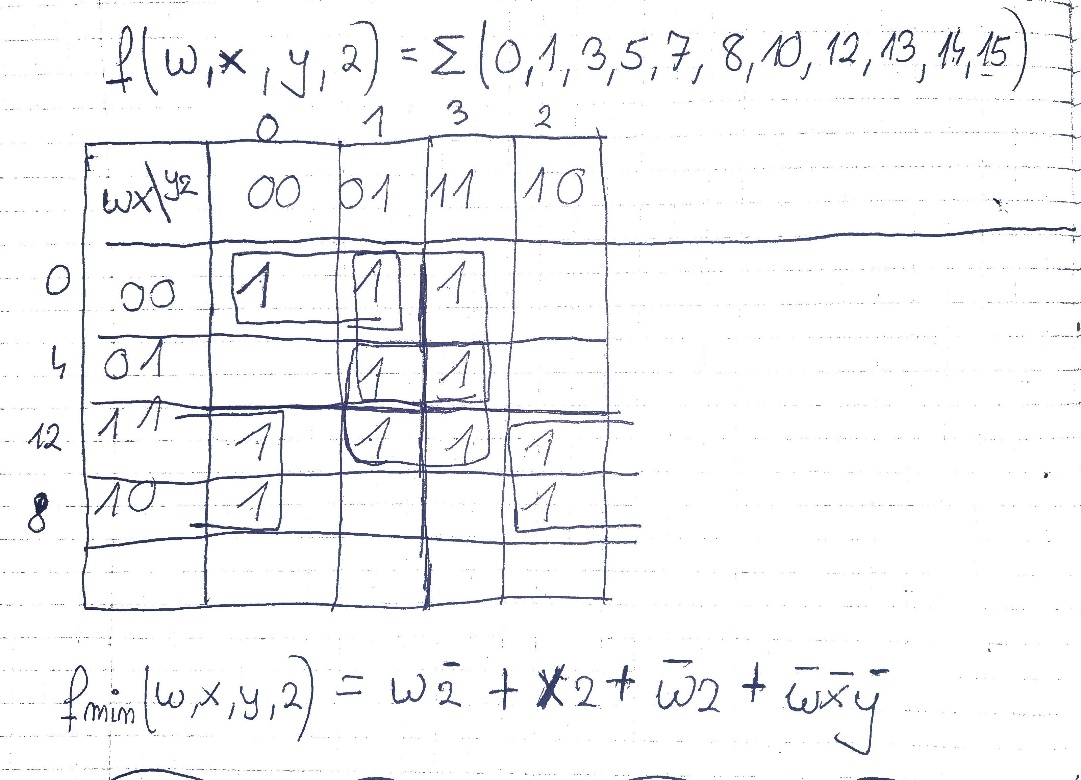
**END**;

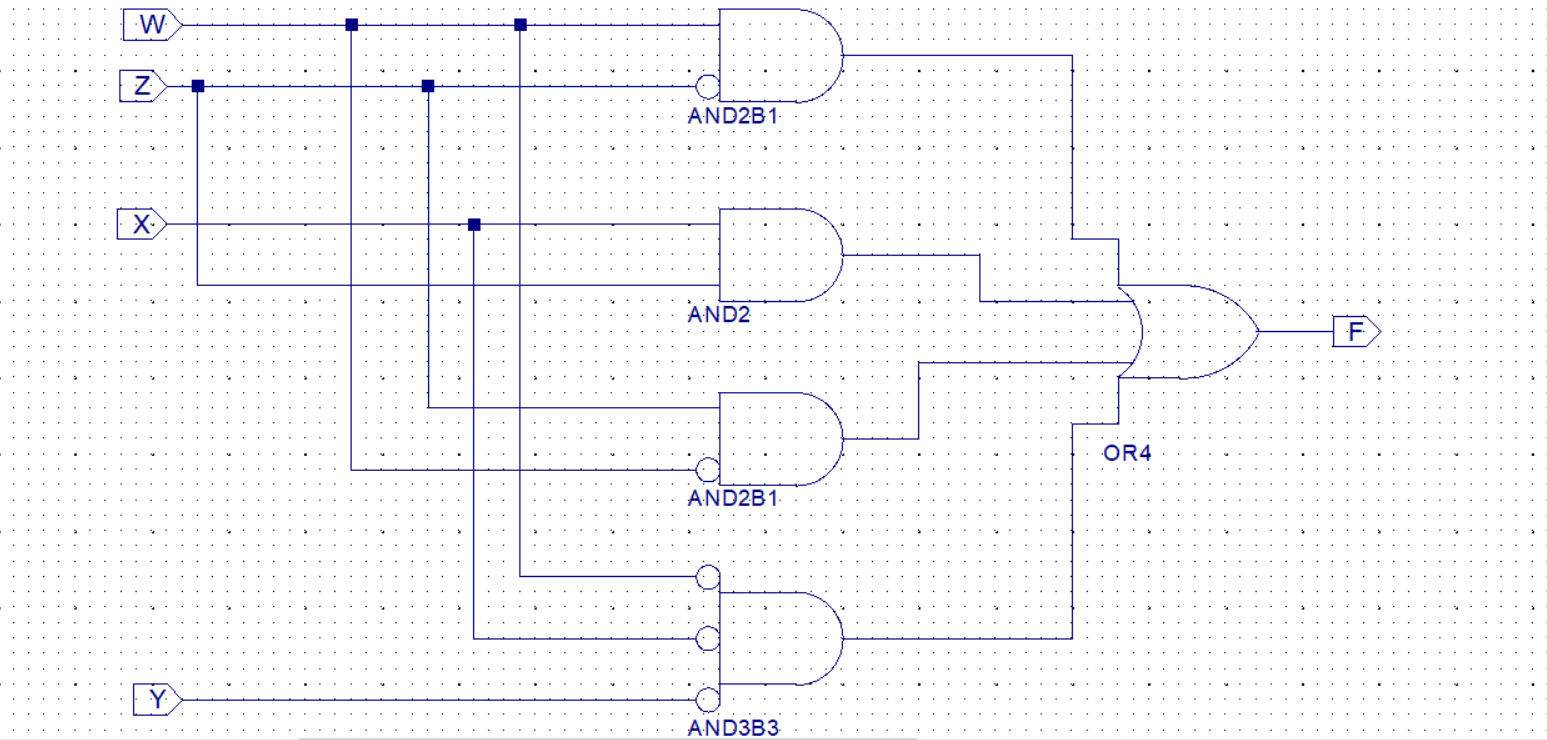


Rysunek 2 Wyniki działania symulacji dla bramki logicznej AND

1. **Implementacja funkcji logicznej**  
   Funkcja logiczna, którą otrzymaliśmy do zaimplementowania miała postać: F(w,x,y,z) = Σ(0, 1, 3, 5, 7, 8, 10, 12, 13, 14, 15)  
   

Rysunek 3 Tabela prawdy dla funkcji logicznej



Rysunek 4 Minimalizacja funkcji za pomocą metody siatki Karnaugh'a  
  
  


Rysunek 5 Schemat funkcji stworzony w programie Xilinx ISE  
  
**LIBRARY** ieee;

**USE** ieee.std\_logic\_1164.**ALL**;

**USE** ieee.numeric\_std.**ALL**;

**LIBRARY** UNISIM;

**USE** UNISIM.Vcomponents.**ALL**;

**ENTITY** fk\_1\_fk\_1\_sch\_tb **IS**

**END** fk\_1\_fk\_1\_sch\_tb;

**ARCHITECTURE** behavioral **OF** fk\_1\_fk\_1\_sch\_tb **IS**

**COMPONENT** fk\_1

**PORT**( F : **OUT** STD\_LOGIC;

W : **IN** STD\_LOGIC;

Z : **IN** STD\_LOGIC;

X : **IN** STD\_LOGIC;

Y : **IN** STD\_LOGIC);

**END** **COMPONENT**;

**SIGNAL** F : STD\_LOGIC;

**SIGNAL** W : STD\_LOGIC;

**SIGNAL** Z : STD\_LOGIC;

**SIGNAL** X : STD\_LOGIC;

**SIGNAL** Y : STD\_LOGIC;

**BEGIN**

UUT: fk\_1 **PORT** **MAP**(

F => F,

W => W,

Z => Z,

X => X,

Y => Y

);

W <= '0','1' **after** 400 ns;

X <= '0', '1' **after** 200 ns, '0' **after** 400 ns, '1' **after** 600 ns; Y <= '0', '1' **after** 100 ns, '0' **after** 200 ns, '1' **after** 300 ns, '0' **after** 400 ns, '1' **after** 500 ns, '0' **after** 600 ns, '1' **after** 700 ns;

Z <= '0', '1' **after** 50 ns, '0' **after** 100 ns,'1' **after** 150 ns,'0' **after** 200 ns,'1' **after** 250 ns,'0' **after** 300 ns,'1' **after** 350 ns,'0' **after** 400 ns,'1' **after** 450 ns,'0' **after** 500 ns,'1' **after** 550 ns,'0' **after** 600 ns,'1' **after** 650 ns,'0' **after** 700 ns,'1' **after** 750 ns;  
**END**;

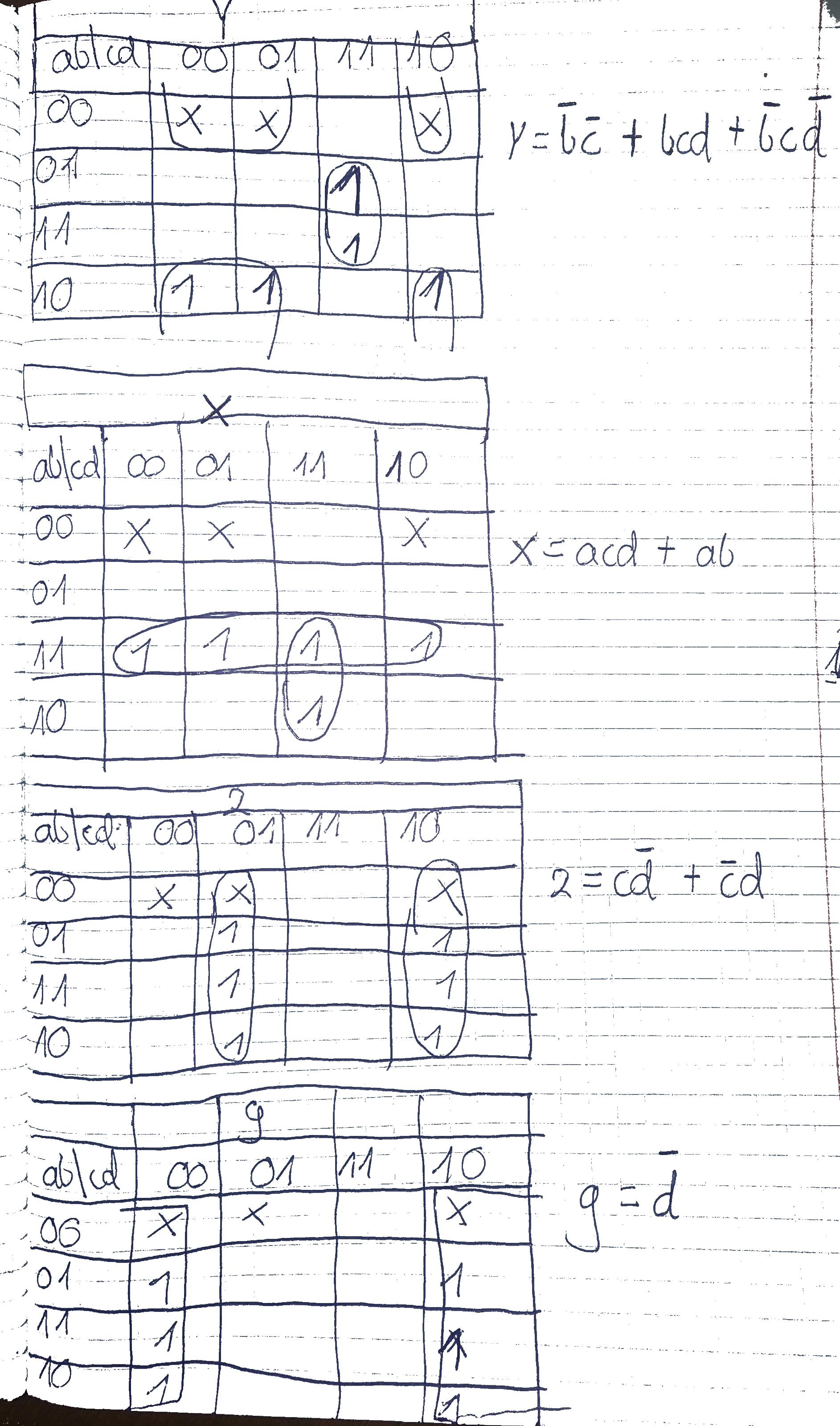


Rysunek 6 Wyniki działania symulacji dla funkcji logicznej

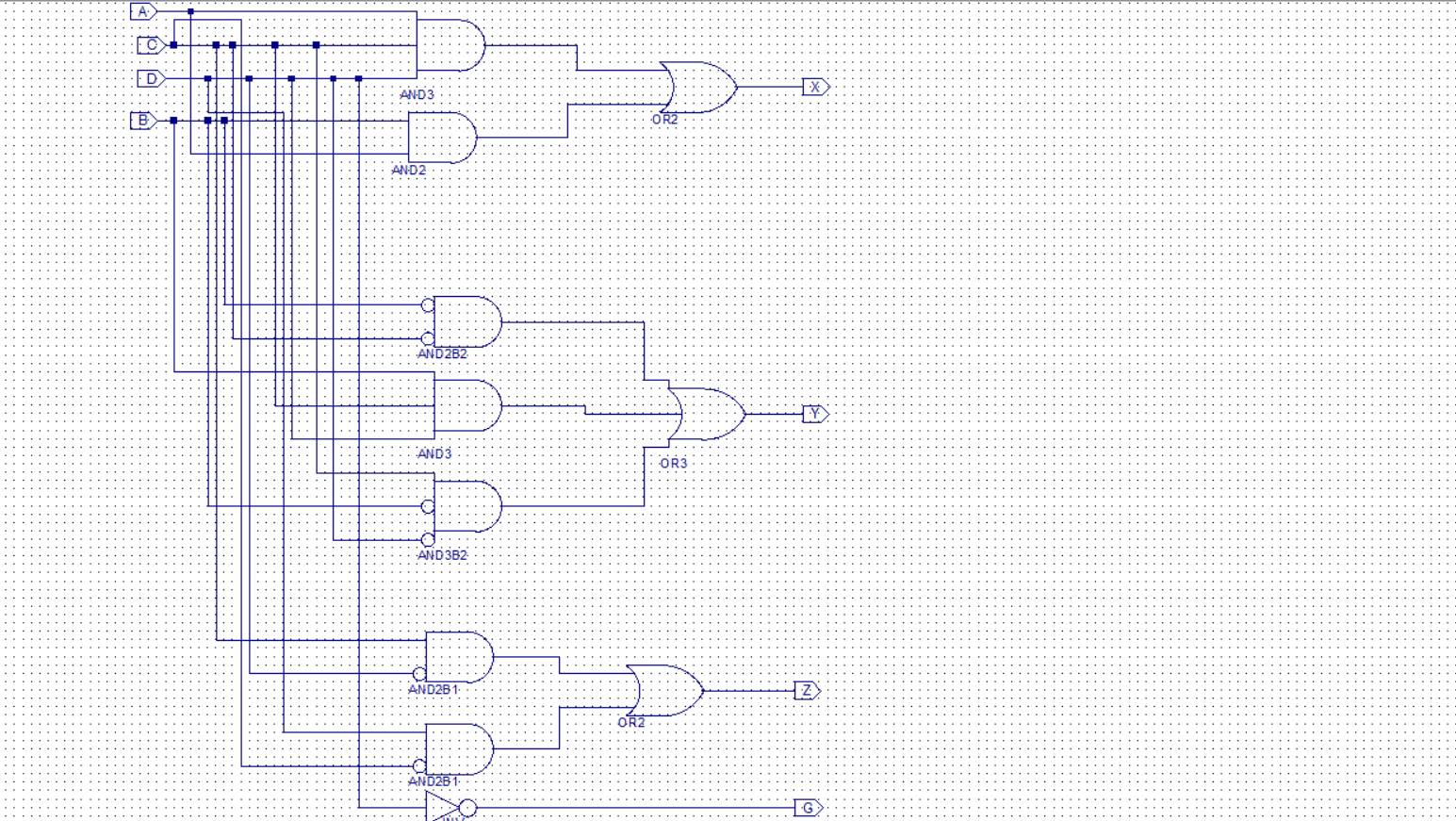
1. **Implementacja układu translatora kodu**  
     
   Translator kodu, który otrzymaliśmy do zrealizowania to: 4-bit kod +3 na 4-bit kod NKB  
     
   Obraz zawierający tekst, klawiatura

   Opis wygenerowany automatycznie

Rysunek 7 Tablica prawdy dla układu translatora kodu



Rysunek 8 Minimalizacja metodą siatek Karnaugh'a



Rysunek 9 Schemat układu translatora kodu

**LIBRARY** ieee;

**USE** ieee.std\_logic\_1164.**ALL**;

**USE** ieee.numeric\_std.**ALL**;

**LIBRARY** UNISIM;

**USE** UNISIM.Vcomponents.**ALL**;

**ENTITY** tk\_tk\_sch\_tb **IS**

**END** tk\_tk\_sch\_tb;

**ARCHITECTURE** behavioral **OF** tk\_tk\_sch\_tb **IS**

**COMPONENT** tk

**PORT**( X : **OUT** STD\_LOGIC;

Y : **OUT** STD\_LOGIC;

Z : **OUT** STD\_LOGIC;

G : **OUT** STD\_LOGIC;

A : **IN** STD\_LOGIC;

C : **IN** STD\_LOGIC;

D : **IN** STD\_LOGIC;

B : **IN** STD\_LOGIC);

**END** **COMPONENT**;

**SIGNAL** X : STD\_LOGIC;

**SIGNAL** Y : STD\_LOGIC;

**SIGNAL** Z : STD\_LOGIC;

**SIGNAL** G : STD\_LOGIC;

**SIGNAL** A : STD\_LOGIC;

**SIGNAL** C : STD\_LOGIC;

**SIGNAL** D : STD\_LOGIC;

**SIGNAL** B : STD\_LOGIC;

**BEGIN**

UUT: tk **PORT** **MAP**(

X => X,

Y => Y,

Z => Z,

G => G,

A => A,

C => C,

D => D,

B => B

);

A <= '0','1' **after** 400 ns;

B <= '0', '1' **after** 200 ns, '0' **after** 400 ns, '1' **after** 600 ns;

C <= '0', '1' **after** 100 ns, '0' **after** 200 ns, '1' **after** 300 ns, '0' **after** 400 ns, '1' **after** 500 ns, '0' **after** 600 ns, '1' **after** 700 ns;

D <= '0', '1' **after** 50 ns, '0' **after** 100 ns,'1' **after** 150 ns,'0' **after** 200 ns,'1' **after** 250 ns,'0' **after** 300 ns,'1' **after** 350 ns,'0' **after** 400 ns,'1' **after** 450 ns,'0' **after** 500 ns,'1' **after** 550 ns,'0' **after** 600 ns,'1' **after** 650 ns,'0' **after** 700 ns,'1' **after** 750 ns;

**END**;  
  
Obraz zawierający zielony, monitor, ekran, granie

Opis wygenerowany automatycznie

Rysunek 10 Wyniki działania symulacji dla układu translatora kodu

1. **Wnioski**