Dekoder wyświetlacza 7-segmentowego (with ... select).

1. Kod źródłowy (zał. dekoder.vhd).

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity dekoder is
    Port ( adres : in  STD_LOGIC_VECTOR (3 downto 0);
           kod : out  STD_LOGIC_VECTOR (6 downto 0));
end dekoder;

architecture Behavioral of dekoder is

begin

        with adres select

                kod <=   "1110111" when "0000",
                                        "0010010" when "0001",
                                        "1011101" when "0010",
                                        "1011011" when "0011",
                                        "0111010" when "0100",
                                        "1101011" when "0101",
                                        "1101111" when "0110",
                                        "1010010" when "0111",
                                        "1111111" when "1000",
                                        "1111011" when "1001",
                                        "1101101" when "1010"|"1011"|"1100"|
                                        "1101"|"1110"|"1111",
                                        "0000000" when others;

end Behavioral;
```

2. Test bench (zał. test1_tb.vhd).

```
LIBRARY ieee;
USE ieee.std_logic_1164.ALL;
use IEEE.STD_LOGIC_unsigned.ALL;

ENTITY test1 IS
END test1;

ARCHITECTURE behavior OF test1 IS

    -- Component Declaration for the Unit Under Test (UUT)

    COMPONENT dekoder
```

```
   PORT(
        adres : IN  std_logic_vector(3 downto 0);
        kod : OUT  std_logic_vector(6 downto 0)
       );
   END COMPONENT;


  --Inputs
  signal adres : std_logic_vector(3 downto 0) := (others => '0');

          --Outputs
  signal kod : std_logic_vector(6 downto 0);
  -- No clocks detected in port list. Replace <clock> below with
  -- appropriate port name


BEGIN

          -- Instantiate the Unit Under Test (UUT)
  uut: dekoder PORT MAP (
         adres => adres,
         kod => kod
        );


  -- Stimulus process
  stim_proc: process
  begin
     -- hold reset state for 100 ns.
adres<= (others =>'0');
            wait for 10 ns;
for i in 0 to 15 loop
adres<= adres+'1';
wait for 1 ns;
end loop;


     wait;
  end process;

END;
```

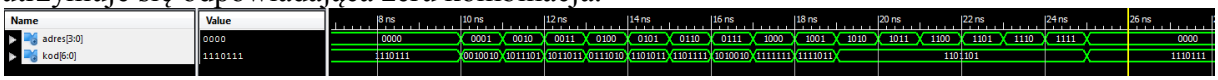
3. Opis projektu.

Projekt posiada:

- 4-bitowe wejście adres, które określa w kodzie binarnym liczbę do wyświetlenia.
- 7-bitowe wyjście kod, które przyjmuje w zależności od wartości wejścia adres odpowiednie kombinacje mają powodować podanie pożądanych stanów na segmenty wyświetlacza.

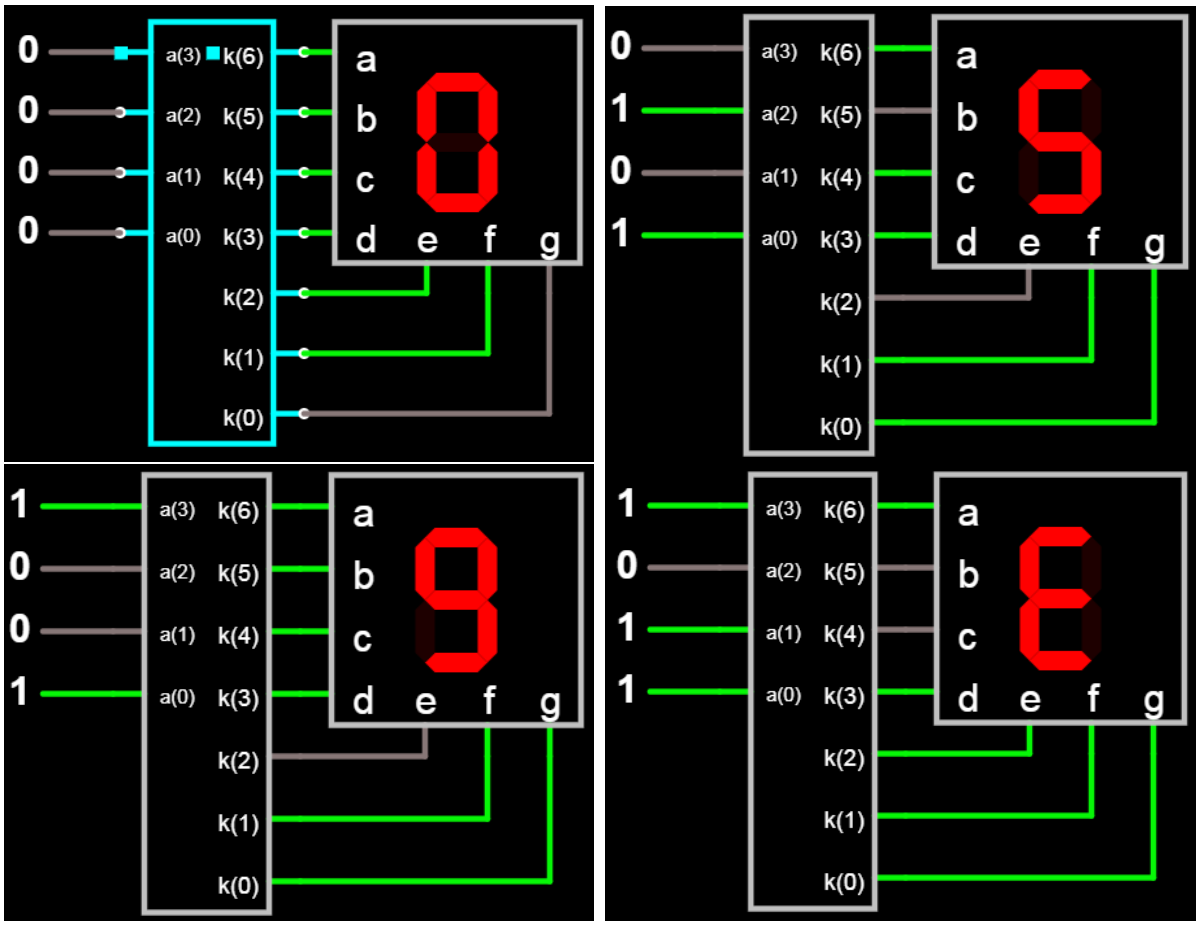
4. Symulacja.

Na początku, przez 10 ns wartość parametru adres ustalona jest na 0, więc wartość wyjścia kod przyjmuje wartość 1110111. Następnie co 1ns wartość adresu rośnie o 1 i kod przyjmuje odpowiednie kombinacje, a więc dla wartości adresu od 10 do 15 nie zmienia się (na wyświetlaczu E). Potem adres przyjmuje wartość 0 do końca symulacji, przez co na wyjściu utrzymuje się odpowiadająca zeru kombinacja.



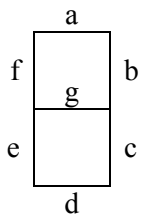
5. Wizualizacja.

Z powodu braku dostępu do odpowiedniego układu FPGA z wyświetlaczem, na którym moglibyśmy zobaczyć działanie układu, postanowiliśmy zwizualizować go przy pomocy witryny https://www.falstad.com/circuit/circuitjs.html (plik do otworzenia na stronie: zał. falstad). Składa się z układu programowalnego i wyświetlacza 7-segmentowego. Na wejścia a podajemy wektor adres za pomocą wejść cyfrowych , a na wyjściach k otrzymujemy odpowiednie kombinacje (inne niż w kodzie VHDL z powodu innej kolejności segmentów w wyświetlaczach, ale dające taki sam skutek). Kilka przykładowych kombinacji:



Oznaczenia segmentów wyświetlacza (a-najbardziej znaczący bit):

- użytego na falstad.com



- obsługiwanego przez projekt w VHDL