ENGINE

Teaching online electronics, microcontrollers and programming in Higher Education

Sprzętowa implementacja algorytmów

7. Enkoder inkrementalny.

Lider projektu: Politechnika Warszawska

Autor: Łukasz Mik

Akademia Nauk Stosowanych w Tarnowie

Erasmus+

GRANT NUMBER: 2020-1-PL01-KA226-HE-095653

This laboratory instruction has been prepared in the context of the ENGINE project. Where other published and unpublished source materials have been used, these have been acknowledged.

Copyright

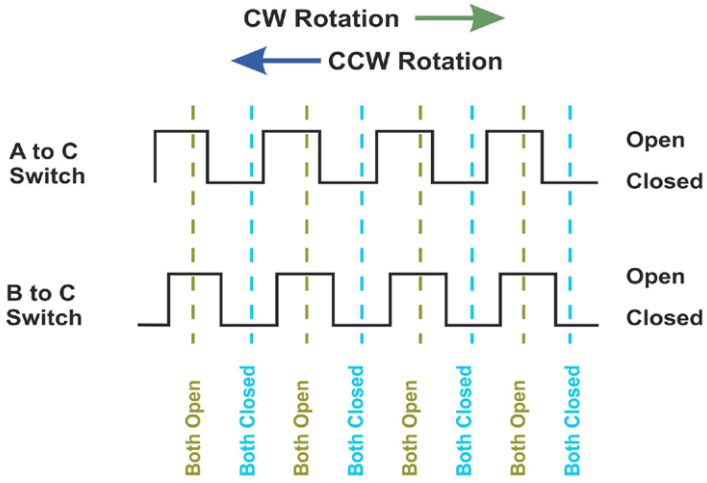
Funding Disclaimer

This project has been funded with support from the European Commission. This report reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

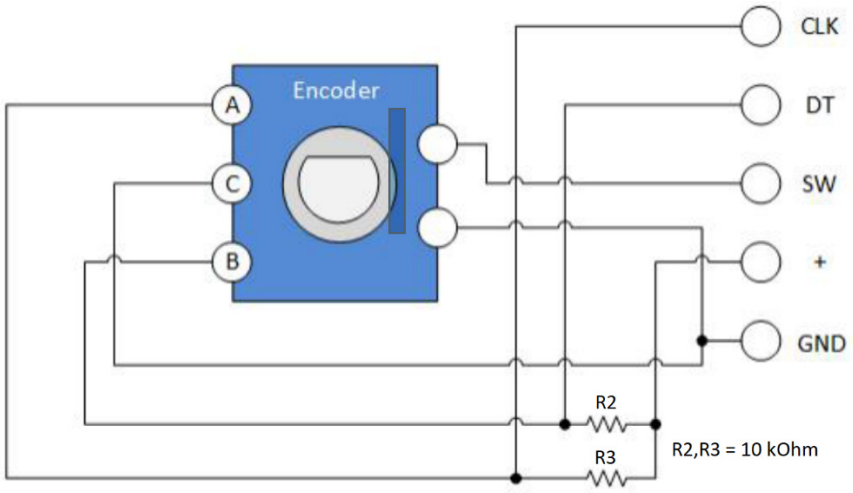
I. Enkoder inkrementalny – zasada działania.

Do realizacji tego ćwiczenia zostanie wykorzystany moduł enkodera obrotowego inkrementalnego firmy Keyes o symbolu KY-040 lub zamiennie HW-040. W tym module zostały już zapewnione rezystory podciągające linie A i B enkodera do zasilania.
Enkoder obrotowy jest elementem, który umożliwia odczyt pozycji obrotu wału (w tym przypadku pokrętła) oraz kierunku jego obrotu. Ma ściśle określoną liczbę pozycji na pełny obrót pokrętła. Sam enkoder posiada 3 wyprowadzenia: A, B i C.

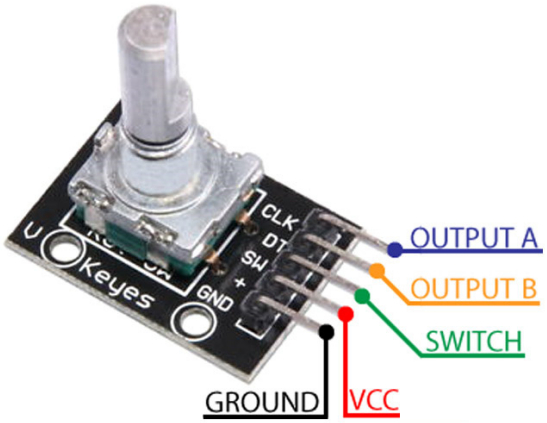
Na pinach A i B są generowane impulsy podczas obrotu pokrętła. Te impulsy są przesunięte względem siebie w fazie co umożliwia określenie kierunku obrotu. Pin C to wspólna masa dla obu sygnałów A i B, które generowane są na zasadzie przełączników.



Podczas obrotu pokrętła zgodnie z ruchem wskazówek zegara powoduje, że sygnał na wyjściu przełącznika A (piny A i C) pojawia się wcześniej niż sygnał na wyjściu przełącznika B (piny B i C). Szybkość obrotu wpływa też na częstotliwość pojawiania się impulsów. Schemat modułu enkodera został przedstawiony na poniższym rysunku.



Moduł został tak zaprojektowany, aby w chwili zwarcia styków A lub B na ich wyjściach pojawiło się napięcie 0V, a przypadku rozwarcia – napięcie zasilania dzięki rezystorom podciągającym o rezystancji 10 kOhm. Opis wyprowadzeń na module wygląda tak:

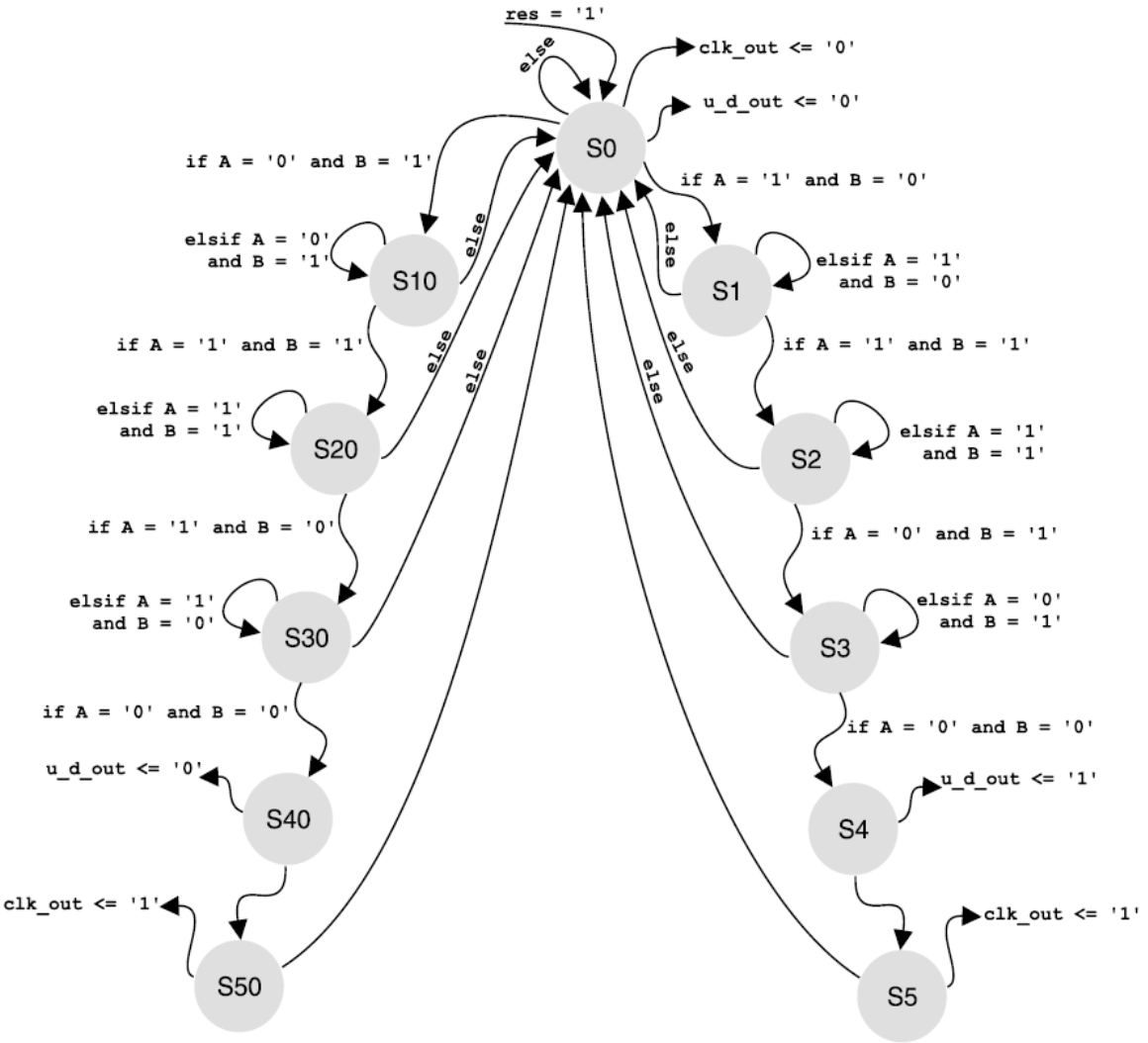


W celu dostosowania aplikacji modułu enkodera do układu FPGA należy podłączyć napięcie zasilania do pinów + i GND. Piny CLK (A) i DT (B) należy podłączyć do wybranych pinów układu FPGA, dostępnych na złączach typu goldpin. Pomijamy pin SW, który jest wyjściem przełącznika, zwieranego w chwili naciśnięcia pokrętła.

Wziąwszy pod uwagę nieuniknione zakłócenia impulsowe powstające podczas zwierania i rozwierania styków, taka para sygnałów nie nadaje się do wykorzystania bezpośrednio w innych układach cyfrowych. Do likwidacji drgań styków w układach FPGA potrzebne są układy odmierzające czas. Najprostszym sposobem realizacji układu odmierzającego czas w układzie FPGA jest zbudowanie automatu stanów, taktowanego zewnętrznym sygnałem zegarowym. Należy też ustalić taki cykl pracy tego automatu, który zminimalizuje wpływ drgań styków na jakość dekodowania.

II. Implementacja dekodera sygnałów dla enkodera inkrementalnego.

We współczesnych językach HDL, także w VHDL, opisanie pracy automatu jest możliwe na wiele sposobów. Aby zapewnić przejrzystość opisu działania, automatu został on przedstawiony w postaci grafu, którego poszczególne stany są jawnie predefiniowane za pomocą deklaracji **constant Sx**, gdzie **x** identyfikuje stan. Jakkolwiek jawne przypisywanie wartości poszczególnym stanom wymaga nieco pracy, to łatwiejsze jest sprawdzenie poprawności opisu logicznego (weryfikacja opisu), ponieważ na podstawie wartości wektora **state** łatwo określić, w którym stanie (w jakim miejscu grafu) znajduje się w danej chwili automat.



Ze względu na zasadę działania, projektowany automat wyposażono w wejście zerowania (asynchroniczne) **res** – każdorazowo po włączeniu zasilania należy podać na nie krótki impuls o poziomie wysokim. Sprzętowe zerowanie wymusza przejście automatu do

stanu S0, który jest stanem początkowym - automat pozostaje w nim do czasu zmiany poziomu na jednym z wejść: A lub B. W zależności od tego, na którym z nich poziom wysoki wystąpi jako pierwszy (co określa kierunek obracania osi enkodera) automat przechodzi do stanu S1 (kierunek zliczania w górę) lub S10 (kierunek zliczania w dół). Począwszy od tych stanów rozpoczyna się kontrola drgań styków (stany inne od oczekiwanych powodują przejście automatu do stanu początkowego S0), a także generację sygnału **enkoder_rol** (stany S4 lub S40) i **enkoder_clk** (stany S5 lub S50).

Implementacja dekoder sygnałów enkodera w języku VHDL

KROK 1: W programie ISE Design Suite 14.7 (Webpack) tworzymy nowy projekt o nazwie encoder, ustawiamy odpowiednie parametry docelowego układu FPGA.

KROK 2: W projekcie tworzymy nowy plik typu *VHDL Module* o nazwie **encoder.** Po jego utworzeniu otworzy się okno edycyjne z zawartością pliku **encoder.vhd**. W pierwszej kolejności uzupełniamy część nagłówkową z bibliotekami.

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

KROK 3: Następnie deklarujemy nazwę jednostki jako encoder projektowej wraz z jej portami wejścia/wyjścia.

```
entity encoder is
port ( A : in std_logic;          -- input A
       B : in std_logic;          -- input B
       res: in std_logic;         -- reset input
       clk: in std_logic;         -- clock input
       EN : out std_logic_vector(2 downto 0); -- enable for 7-segment display
       SSEG: out std_logic_vector(7 downto 0) -- 7-segment display
);
end encoder;
```

KROK 4: Po deklaracji jednostki przechodzimy do opisu architektury. W części deklaracyjnej architektury definiujemy stałe od S0 do S5 oraz od S10 do S50 reprezentujące poszczególne stanu z przedstawionego wcześniej diagramu automatu stanów. Dodatkowo deklarujemy sygnał **enkoder_rol** (*enkoder right or left*), który wskazuje kierunek obrotu pokrętła enkodera oraz **enkoder_clk**, na którego wyjściu są generowane impulsy prostokątne w stanach S5 lub S50. Dodatkowo deklarujemy sygnał **bcd_counter**, odpowiedzialny za zliczanie impulsów z enkodera. Sygnał ten będzie bezpośrednio sterował wejściem dekodera kodu BCD na kod wyświetlacza siedmiosegmentowego.

```
architecture arch of encoder is
  signal state: std_logic_vector(3 downto 0);
  constant S0 : std_logic_vector(3 downto 0):= "0000";
  constant S1 : std_logic_vector(3 downto 0):= "0001";
  constant S2 : std_logic_vector(3 downto 0):= "0010";
  constant S3 : std_logic_vector(3 downto 0):= "0011";
  constant S4 : std_logic_vector(3 downto 0):= "0100";
  constant S5 : std_logic_vector(3 downto 0):= "0101";
  constant S10 : std_logic_vector(3 downto 0):= "1001";
  constant S20 : std_logic_vector(3 downto 0):= "1010";
  constant S30 : std_logic_vector(3 downto 0):= "1011";
  constant S40 : std_logic_vector(3 downto 0):= "1100";
  constant S50 : std_logic_vector(3 downto 0):= "1101";
  signal encoder_clk : std_logic;
  signal encoder_rol : std_logic;
  signal clk_1kHz : std_logic:='0';
  signal bcd_counter : std_logic_vector(3 downto 0):="0000";
```

KROK 5: W opisie architektury pierwszy proces jest odpowiedzialny za obsługę sygnałów z enkodera. Rozpoczyna się on od obsługi sygnału zerującego – powrót do stanu zerowego, jak również obsługi przejścia ze stanu S0 do S1 (obrót enkodera o 1 takt w prawo) lub ze stanu S0 do S10 (obrót enkodera o 1 takt w lewo).

```
BEGIN
process (clk_1kHz, res)
begin
  if res = '0' then
       state <= "0000";
  elsif clk_1kHz'event and clk_1kHz = '1' then
    case state is
      when S0 =>
        encoder_clk <= '0';
        encoder_rol <= '0';
        if A = '1' and B = '0' then
          state <= S1;
        elsif A = '0' and B = '1' then
          state <= S10;
        else state <= S0;
        end if;
```

KROK 6: Jeśli nastąpiła zmiana z S0 na S1 to przechodzimy do ścieżki automatu stanu, odpowiedzialnej za detekcję ruchu obrotowego w prawo („zwiększ”). Ta ścieżka automatu jest uruchamiana również przy każdym takcie (kombinacji) sygnałów A i B dla kierunku obrotu w prawo (stany od S1 do S5).

```
-- path of the state machine for motion detection in the "increase" direction
      when S1 =>
        if A = '1' and B = '1' then
          state <= S2;
        elsif A = '1' and B = '0' then
          state <= S1;
        else state <= S0;
        end if;
      when S2 =>
```

```
        if A = '0' and B = '1' then
          state <= S3;
        elsif A = '1' and B = '1' then
          state <= S2;
        else state <= S0;
        end if;
      when S3 =>
        if A = '0' and B = '0' then
          state <= S4;
        elsif A = '0' and B = '1' then
          state <= S3;
        else state <= S0;
        end if;
      when S4 =>
        encoder_rol <= '1';
        state <= S5;
      when S5 =>
        encoder_clk <= '1';
        state <= S0;
```

KROK 7: Jeśli nastąpiła zmiana z S0 na S10 to przechodzimy do ścieżki automatu stanu, odpowiedzialnej za detekcję ruchu obrotowego w lewo („zmniejsz”). Ta ścieżka automatu jest uruchamiana również przy każdym takcie (kombinacji) sygnałów A i B dla kierunku obrotu w lewo (stany od S10 do S50).

```
-- path of the state machine for motion detection in the "decrease" direction
      when S10 =>
        if A = '1' and B = '1' then
          state <= S20;
        elsif A = '0' and B = '1' then
          state <= S10;
        else state <= S0;
        end if;
      when S20 =>
        if A = '1' and B = '0' then
          state <= S30;
        elsif A = '1' and B = '1' then
          state <= S20;
        else state <= S0;
        end if;
      when S30 =>
        if A = '0' and B = '0' then
          state <= S40;
        elsif A = '1' and B = '0' then
          state <= S30;
        else state <= S0;
        end if;
      when S40 =>
        encoder_rol <= '0';
        state <= S50;
      when S50 =>
        encoder_clk <= '1';
        state <= S0;
      when others =>
        state <= S0;
      end case;
  end if;
end process;
```

W obu ścieżkach automatu stanów, jeśli wystąpi inna kombinacja sygnałów A i B niż zakładana (przy zmianie kierunku obrotu) następuje powrót do stanu S0.

KROK 8: Po procesie obsługującym enkoder dodajemy proces odpowiedzialny za generowanie sygnału taktującego o nazwie **clk_1kHz** i częstotliwości 1kHz. Taka częstotliwość (okres równy 10 ms) jest odpowiednia do wyeliminowania wpływu drgań styków enkodera.

```
--clock 1kHz
process(clk)
  variable clock_cnt : integer:=0;
begin
  if rising_edge(clk) then
    if clock_cnt < 6000 then
         clock_cnt := clock_cnt+1;
       else
         clock_cnt := 0;
         clk_1kHz <= not(clk_1kHz);
       end if;
  end if;
end process;
```

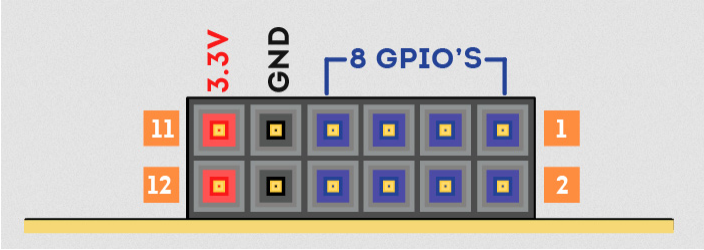
KROK 9: Kolejny proces w architekturze będzie odpowiedzialny za obsługę licznika mod 10, taktowanego za pomocą sygnału **encoder_clk**. Kierunek zliczania licznika będzie sterowany sygnałem **encoder_rol**. Jeśli oś enkodera będzie obracać się w lewo to licznik będzie zwiększany o 1 z każdym taktem sygnału **encoder_clk**, w przeciwnym wypadku zmniejszany o 1.

```
process(encoder_clk)
begin
  if rising_edge(encoder_clk) then
       if bcd_counter < 10 then
         if(encoder_rol = '1') then
              bcd_counter <= bcd_counter + 1;  --encoder knob rotates right
         else
              bcd_counter <= bcd_counter - 1;  --encoder knob rotates left
         end if;
       else
         bcd_counter <= "0000";
       end if;
  end if;
end process;
```

KROK 10: Na końcu opisu architektury dodajemy proces odpowiedzialny za dekodowanie kodu BCD na kod wyświetlacza siedmiosegmentowego. Niezbędne jest też ustawienie bitów wektora EN, odpowiedzialnego za wybór aktywnego wyświetlacza.

```
process(bcd_counter)
begin
  case bcd_counter is
       --------------------abcdefgp
       when "0000"=>SSEG<="00000011";
       when "0001"=>SSEG<="10011111";
       when "0010"=>SSEG<="00100101";
       when "0011"=>SSEG<="00001101";
       when "0100"=>SSEG<="10011001";
       when "0101"=>SSEG<="01001001";
       when "0110"=>SSEG<="01000001";
       when "0111"=>SSEG<="00011111";
       when "1000"=>SSEG<="00000001";
       when "1001"=>SSEG<="00001001";
       when others=>SSEG<="11111111";
  end case;
end process;
EN <= "110";
END arch;
```

Po wykonaniu wszystkich powyższych kroków pozostało jeszcze podłączenie enkodera do pinów układu FPGA. Enkoder łączymy z pinami gniazda P1.



Schemat połączenia został opisany w poniższej tabeli.

Encoder pin	GPIO HEADER „P1”
+	**3.3V**
GND	**GND**
CLK	**1**
DT	**2**

Do prawidłowego działania projektu brakuje jeszcze pliku z przypisaniem pinów układu FPGA do portów we/wy jednostki projektowej. Tworzymy zatem plik o nazwie **encoder.ucf**, (podczas tworzenia nowego pliku wybieramy opcję *Implementation Constraints File*) i wklejamy poniższą zawartość.

```
NET "A"  LOC = P31   | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 12;
NET "B"  LOC = P32   | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 12;
NET "clk"  LOC = P129  | IOSTANDARD = LVCMOS33 | PERIOD = 12MHz;
NET "res" LOC = P80  | PULLUP  | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 12;

NET "SSEG[7]"  LOC = P117  | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 12;
NET "SSEG[6]"  LOC = P116  | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 12;
NET "SSEG[5]"  LOC = P115  | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 12;
NET "SSEG[4]"  LOC = P113  | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 12;
NET "SSEG[3]"  LOC = P112  | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 12;
NET "SSEG[2]"  LOC = P111  | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 12;
NET "SSEG[1]"  LOC = P110  | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 12;
NET "SSEG[0]"  LOC = P114  | IOSTANDARD = LVCMOS33 | SLEW = SLOW | DRIVE = 12;

NET "EN[2]"  LOC = P124 | IOSTANDARD = LVCMOS33 | DRIVE = 8 | SLEW = FAST ;
NET "EN[1]"  LOC = P121 | IOSTANDARD = LVCMOS33 | DRIVE = 8 | SLEW = FAST ;
NET "EN[0]"  LOC = P120  | IOSTANDARD = LVCMOS33 | DRIVE = 8 | SLEW = FAST ;
```

Po ponownym zapisaniu i zsyntezowaniu projektu generujemy plik konfiguracyjny dla układu FPGA.

UWAGA: Zaobserwuj ile taktów enkodera odpowiada pojedynczej zmianie stanu licznika wyświetlanego na wyświetlaczu? Czy potrafisz wyjaśnić co się dzieje?

ZADANIE: Zmodyfikować projekt tak, aby obrót enkodera zwiększał lub zmniejszał stan licznika w zakresie od 0 do 99 (należy zastosować dwa wyświetlacze siedmiosegmentowe).

References

- J. Majewski, P. Zbysiński – *Układy FPGA w przykładach.* Wydawnictwo BTC, Legionowo 2007.

- Components 101 – *KY-040 - Rotary Encoder Module,* https://components101.com/modules/KY-04-rotary-encoder-pinout-features-datasheet-working-application-alternative