Układy FPGA w przykładach, część 8
Projekty przykładowe

Wyświetlacze ciekłokrystaliczne bez sterowników są stosunkowo rzadko stosowane w popularnych aplikacjach, co wynika z – jak sądzimy – z przekonania, że ich obsługa jest trudniejsza niż wyświetlaczy LED. O tym, że nie jest to prawda przekonamy Czytelników w artykule, przy okazji pokazując nieco inny niż dotychczas sposób przygotowywania projektów.

Podstawowa nowość polega na tym, że projekt implementowany w FPGA opiszemy za pomocą schematu, a nie języka VHDL. Jedynie jego niewielki fragment (dekoder BCD–>7 segmentów i preskaler) opiszemy w VHDL, następnie utworzymy ich symbole graficzne i wykorzystamy na schemacie. Pozostałe elementy widoczne na **rys. 1** wzięto ze standardowych bibliotek pakietu WebPack ISE.

Sposób sterowania wyświetlaczy LCD opisaliśmy szczegółowo w EP6/2003 (w artykule „Dekoder–sterownik 7–segmentowego wyświetlacza LCD w VHDL"), wobec czego teraz skupimy się na opisie realizacji projektu: zamierzamy wysterować 3,5–cyfrowy wyświetlacz „multimetrowy" za pomocą układu FPGA.

List. 1. Opis 18–bitowego preskalera służącego do uzyskania przebiegów zegarowych zliczanych przez liczniki (których stany monitorujemy na LCD) oraz sterującego wspólną elektrodę wyświetlacza LCD (*backplane* lub COM)

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

entity preskaler is
    Port ( clk : in  STD_LOGIC;
           q : inout  STD_LOGIC_VECTOR (18 downto 0)
);
end preskaler;

architecture Behavioral of preskaler is
begin
process (clk)
begin
   if clk='1' and clk'event then
      q <= q - 1;
   end if;
end process;
end Behavioral;
```

Realizacja

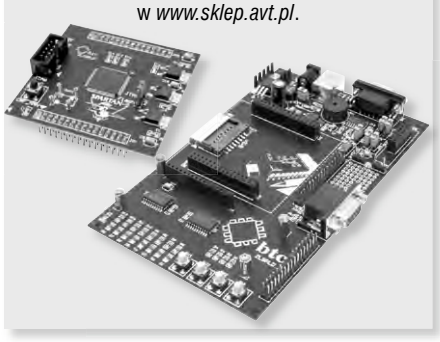
Schemat ilustrujący sposób wykonania projektu pokazano na rys. 1. Do wejść dekoderów sterujących segmentami LCD dołączono trzy liczniki dekadowe, które zliczają impulsy z wyjścia preskalera (którego opis pokazano na **list. 1**).

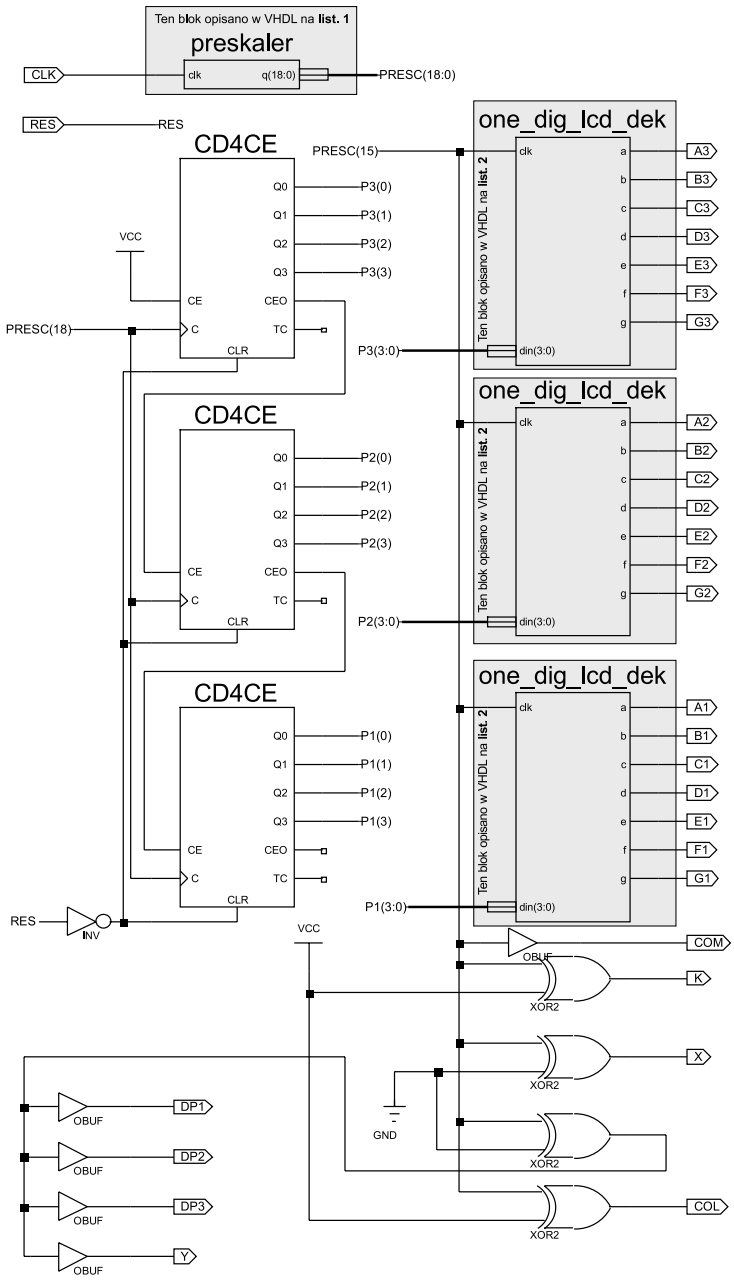
Jak wspomniano, dekoder sterujący także opisano w VHDL (**list. 2**), dzięki czemu oszczędzono sporo czasu, co łatwo ocenić po próbie narysowania (np. z wykorzystaniem podstawowych bramek logicznych) odpowiednika opisu z list. 2.

Ponieważ, zgodnie ze wstępną zapowiedzią, projekt narysujemy w edytorze schematów WebPacka ISE, obydwa fragmenty projektu opisane w VHDL przekonwertujemy do postaci symbolu graficznego i użyjemy ich na planszy schematowej. Robimy to następująco: w oknie zawierającym pliki należące do projektu zaznaczamy plik *one_dig_lcd_dek.vhd* i w oknie procesów (poniżej) wybieramy operację *Create Schematic Symbol* (**rys. 2**). Podobnie postępujemy z plikiem *preskaler.vhd*, w wyniku czego (po przejściu do okna edytora schematów) w bieżącym katalogu roboczym (projektu) są dostępne dwa nowe symbole: *one_dig_lcd_dek* i *preskaler* (rys. 3), które możemy użyć w projekcie w sposób identyczny jak predefiniowane elementy biblioteczne, dostarczane wraz z pakietem WebPack ISE. Na **rys. 4** pokazano wygląd

Kursowa platforma sprzętowa

Wszystkie projekty przedstawione w kursie implementowano i testowano na zestawie składającym się z bazy ZL9PLD i modułu dipPLD ZL10PLD (z układem FPGA z rodziny Spartan 3 – XC3S200). Zestaw oraz programator ISP są dostępne w *www.sklep.avt.pl*.



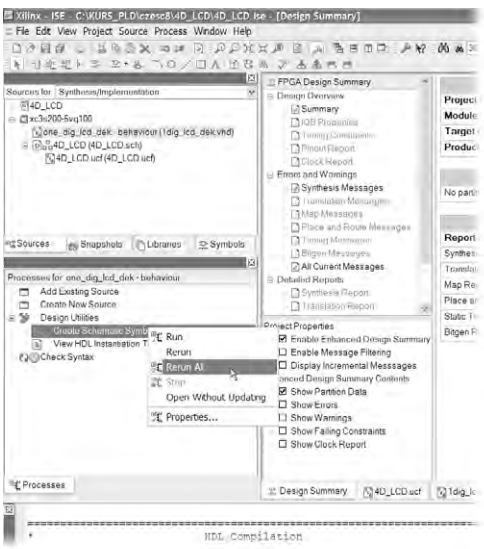


Rys. 1.

Zapraszamy do współpracy!
Wszystkich Czytelników interesujących się układami FPGA zachęcamy do prezentacji na łamach EP własnych rozwiązań. Odpowiemy także na wszelkie pytania związane z kursem, językiem VHDL i układami FPGA.

Plan kursu

1. Wprowadzenie
   - Budowa zestawu uruchomieniowego
   - Programowanie i konfiguracja układu XC3S200
   - Tryby konfiguracji układu XC3S200
   - Zasilanie układu XC3S200
   - Linie I/O w układzie XC3S200
   - JTAG jako uniwersalny interfejs do programowania i konfigurowania
2. Budowa, cechy funkcjonalne i parametry układów FPGA z rodziny Spartan 3
   - CLB
   - IOB
   - Globalne sygnały zegarowe
   - DCM
   - Sprzętowe multiplikatory
   - Pamięć BlockRAM
3. **Projekty przykładowe**
   - Debouncer
   - Klawiatura matrycowa
   - Obsługa wyświetlacza multipleksowego LED
   - **Obsługa wyświetlacza LCD**
   - Sterownik LCD 2x16 (prosty)
   - Sterownik LCD 2x16 (zaawansowany)
   - Komunikacja via RS232 i USB
   - Sterownik VGA
   - Implementacja mikrokontrolera PicoBlaze



Rys. 2.

symboli graficznych uzyskanych z przygotowanych opisów VHDL.

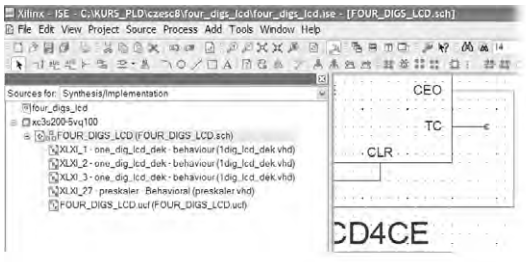
Projekt pokazany na rys. 1 został na potrzeby testów uproszczony: kropki dziesiętne oraz znaki specjalne (oprócz dwukropka) zostały na stałe wygaszone, a cyfra „1" na najstarszej pozycji jest włączona na stałe. Efekt taki uzyskano dzięki bramkom ExOR znajdującym się w dolnej części schematu. Bramki te pracują jako sterowane inwertery, które odwracają fazę sygnału zegarowego (sterującego wspólną elektrodę wyświetlacza) gdy na drugim wejściu bramki jest logiczna „1" – w takim przypadku sterowany segment jest włączony. Wejścia zerujące liczników zastosowanych w projekcie dołączono do mikroprzełącznika S1 na pycie bazowej ZL9PLD, co pozwala w dowolnym momencie wyzerować liczniki uzyskując wynik odczytu „1000", bo – jak wspomniano – najstarsza cyfra jest włączona na stałe. Wątpliwości może budzić sensowność sterowania segmentów wyłączonych na stałe – wynika to z dużej czułości ciekłego kryształu na pole elektryczne, które może powodować niezamierzone

List. 2. Opis VHDL kompletnego dekodera 7–segmentowego przystosowanego do sterowania LCD – na wejście CLK należy podać przebieg prostokątny o częstotliwości 30...300 Hz

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

entity one_dig_lcd_dek is port (
  din: in std_logic_vector(3 downto 0);
  clk: in std_logic;
  a, b, c, d, e, f, g: out std_logic
  );
end one_dig_lcd_dek;

architecture behaviour of one_dig_lcd_dek is
signal segm: std_logic_vector(6 downto 0);
signal segm_o: std_logic_vector(6 downto 0);
signal bp: std_logic;

begin
with din select
--           gfedcba
   segm <= „0111111” when „0000”,   -- 0
           „0000110” when „0001”,   -- 1
           „1011011” when „0010”,   -- 2
           „1001111” when „0011”,   -- 3
           „1100110” when „0100”,   -- 4
           „1101101” when „0101”,   -- 5
           „1111101” when „0110”,   -- 6
           „0000111” when „0111”,   -- 7
           „1111111” when „1000”,   -- 8
           „1101111” when „1001”,   -- 9
           „0000000” when others;   -- wygaszenie

  bp <= clk;

mb_xor:   for I in 0 to 6 generate
           segm_o(i) <= segm(i) xor bp;
         end generate;

  g <= segm_o(6);
  f <= segm_o(5);
  e <= segm_o(4);
  d <= segm_o(3);
  c <= segm_o(2);
  b <= segm_o(1);
  a <= segm_o(0);

end behaviour;
```

włączanie segmentów, które nie są sterowane.

Pokazany sposób opisania projektu nie jest jedynym możliwym, można oczywiście całość (modułowo lub w pojedynczym pliku) opisać w VHDL, jak robiliśmy to w poprzednich miesiącach.



Rys. 5.

Implementacja

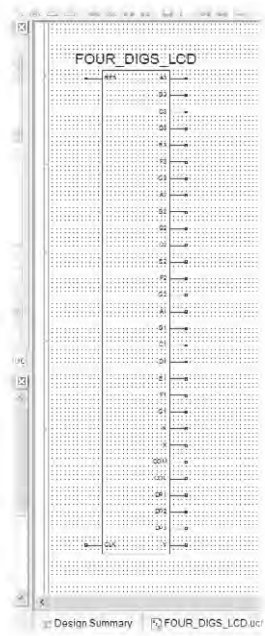
Projekt dekodera ma strukturę hierarchiczną (**rys. 5**), najwyżej w hierarchii jest ulokowany schemat identyczny z pokazanym na rys. 1. Elementami podrzędnymi są: preskaler i dekoder (jak widać na rys. 5, zastosowany na schemacie trzykrotnie), z opisów których wykonaliśmy schematowe elementy biblioteczne.

Interesująco wyglądają wyniki syntezy logicznej wykonanej przez pakiet Web PackISE, syntezer „zauważył” bowiem, że w projekcie zastosowaliśmy 3 pamięci ROM (transkodery BCD–>kod 7–segmentowy), 19–bitowy preskaler, 3 liczniki 4–bitowe i 21 sterowanych inwerterów (bramek ExOR) – po jednym dla każdego sterowanego segmentu.

Dokładny opis korzystania z bezpłatnego pakietu WebPack ISE przedstawiono w książce „Układy programowalne – pierwsze kroki”, dostępnej w sklepie AVT (nr katalogowy KS–220604). Spis treści tej książki jest dostępny pod adresem: *http://www.sklep.avt.pl/photo/_pdf/KS–220604.pdf*.

Podsumowanie

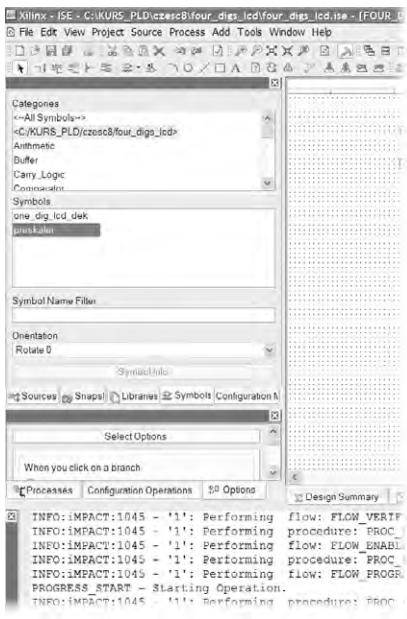
Projekt przedstawiony w artykule można (także po wykonaniu ewentualnych modyfikacji) wykorzystać jako uniwersalny „IP core” w dowolnych innych projektach, na przykład poprzez wykonanie symbolu schematowego (jak na **rys. 6**).



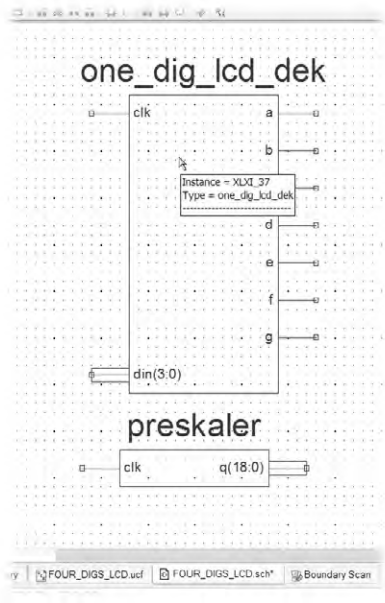
Rys. 6.

Jak się z pewnością przekonają Czytelnicy zamierzający prowadzić własne próby z modyfikowaniem prezentowanego projektu, VHDL zapewnia dużą łatwość wprowadzania zmian. Modyfikowanie schematów jest nieco trudniejsze, czego jedną z przyczyn są drobne narowy edytora schematów, z drugiej – konieczność myślenia o projekcie na poziomie bramek co powoduje, że sporą część nie zawsze łatwej w wykonaniu syntezy logicznej Czytelnik musi przeprowadzić samodzielnie. Warto więc pokonać pierwsze trudności w nauce VHDL–a, żeby w kolejnych projektach było wygodnie.

Jacek Majewski
jacek.majewski@pwr.wroc.pl
Piotr Zbysiński, EP
piotr.zbysinski@ep.com.pl



Rys. 3.



Rys. 4.