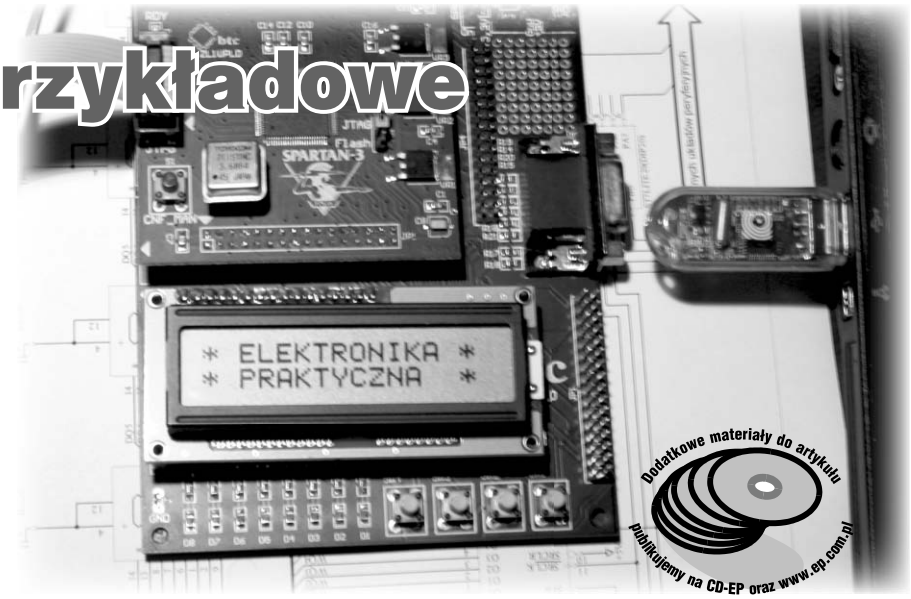
Układy FPGA w przykładach, część 9
Projekty przykładowe



Ze względu na niską cenę i wygodę stosowania, alfanumeryczne wyświetlacze LCD są powszechnie stosowane w panelach operatorskich urządzeń wszelkiej „maści". O ile procedury obsługi sterowników HD44780 (lub podobnych) dla różnych mikrokontrolerów są łatwo dostępne, nieco gorzej wygląda sprawa w przypadku użytkowników FPGA. Nie oznacza to jednak, że musimy rezygnować z możliwości stosowania takich wyświetlaczy w swoich projektach – jedno z najprostszych rozwiązań problemu przedstawiamy w artykule.

Typowe wyświetlacze alfanumeryczne (jak np. WC1602 lub podobne) są wyposażone w sterowniki zgodne z HD44780 i z takim właśnie układem współpracuje opisany w artykule sterownik implementowany w FPGA. Przedstawione rozwiązanie charakteryzuje się prostotą i przejrzystością koncepcyjną, dzięki czemu analiza jego działania nie wymaga zaawansowanej znajomości VHDL, wystarczy podstawowa znajomość sekwencyjnych układów cyfrowych. Maksymalne uproszczenie opisu odbiło się na funkcjonalności projektu, dlatego za miesiąc przedstawimy rozwiązanie sterownika o większej funkcjonalności.

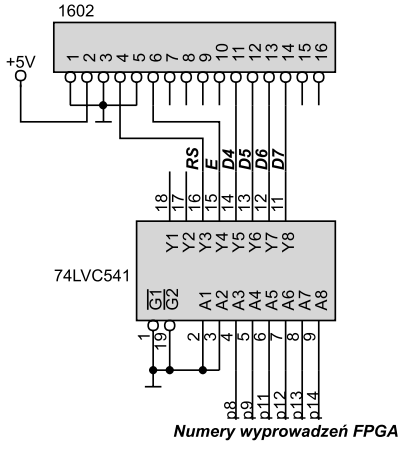
Realizacja

Magistrala danych sterownika HD44780 jest dwukierunkowa, dzięki czemu projektant korzystający z wyświetlacza może (między innymi) sprawdzić gotowość sterownika do przyjęcia kolejnych poleceń (za pomocą sygnału *BusyFlag*, dostępnego na magistrali danych). Ponieważ w większości przypadków wyświetlacze z układami HD44780 są zasilane napięciem 5 V, a linie I/O układów Spartan 3 nie mogą bezpośrednio współpracować z logiką CMOS/TTL5V, konieczne okazało się zastosowanie układu buforującego linie danych D7…D4 oraz sterujące E(NA) i RS (**rys. 1**). Takie rozwiązanie uniemożliwia odczyt przez FPGA magistrali danych sterownika HD44780, co jednak nie uniemożliwia jego obsługi: w praktyce projektanci dość często upraszczają procedury obsługi wyświetlaczy, rezygnując z możliwości monitorowania flagi zajętości, zapewniając takie odstępy czasowe pomiędzy kolejnymi operacjami na magistrali, podczas których sterownik HD44780 na pewno zdąży z wykonaniem poleceń. Zgodnie z danymi katalogowymi tego układu najdłużej trwa wykonanie polecenia *Return Home* (do 1,52 ms), wykonanie pozostałych poleceń trwa nie dłużej niż 37….41 µs.

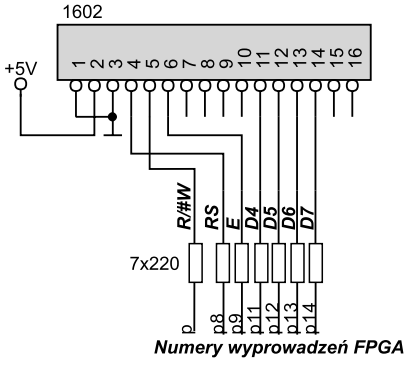
Zapraszamy do współpracy!
Wszystkich Czytelników interesujących się układami FPGA zachęcamy do prezentacji na łamach EP własnych rozwiązań. Odpowiemy także na wszelkie pytania związane z kursem, językiem VHDL i układami FPGA.

Tak więc, dzięki prostemu zabiegowi „czasowemu" udało się uprościć budowę interfejsu obsługującego LCD implementowanego w FPGA, pozwoliło to także na stałe zewrzeć do masy zasilania linię sterującą LCD o nazwie R/#W.

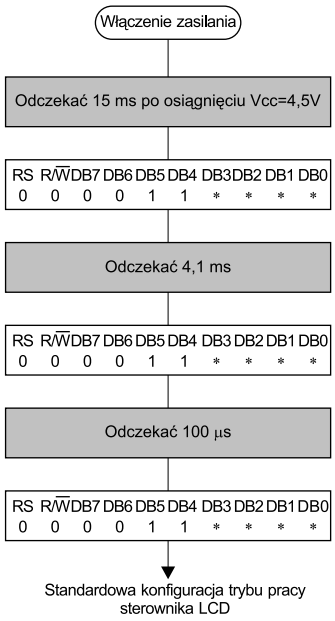
Jeżeli jednak dwukierunkowy dostęp do sterownika HD44780



Rys. 1.



Rys. 2.



Rys. 3.

okaże się niezbędny, można zastosować pasywny układ dopasowujący poziomy napięciowe pomiędzy FPGA i HD44780, jaki pokazano na **rys. 2**.

Sterowniki HD44780 charakteryzują się niezbyt dobrze działającym układem zerującym, który w wielu przypadkach trzeba „wspierać" za pomocą zerowania programowego. Na **rys. 3** pokazano zalecaną przez producenta sekwencję zerującą sterownik, z podanymi minimalnymi czasami trwania każdego etapu. Jak można zauważyć, zakres odcinków czasu jakie trzeba odmierzyć jest bardzo duży: od 37 µs aż do 15 ms. Precyzyjne ich odmierzanie mocno komplikuje budowę interfejsu i – jak pokazały doświadczenia – nie jest niezbędne. Poszliśmy na skróty, przyjmując niską częstotliwość taktowania automatu „zaszytego" w interfejsie,

List. 1. Opis preskalera zastosowanego w projekcie do uzyskania częstotliwości 56 Hz z 3,6864 MHz

```
presc: process (clk)
begin
   if clk='1' and clk'event then
         presk <= presk - 1;
   end if;
end process;

clk_int <= presk(15);
```

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*.

wynosi ona ok. 56 Hz, co daje nam okres taktowania 17,7 ms. Jest to czas wystarczająco długi, żeby zapewnić poprawną pracę sterownika HD44780 – „mieszczą" się w nim wszelkie operacje jakie będziemy wykonywać na wyświetlaczu.

Implementacja

Prezentowany projekt zaimplementowano w zestawie ewaluacyjnym składającym się z płyty bazowej ZL9PLD oraz modułu dipPLD ZL10PLD (z zainstalowanym układem XC3S200). W module dipPLD zastosowano generator kwarcowy o częstotliwości przebiegu prostokątnego 3,6864 MHz, który spełnia rolę źródła sygnału taktującego dla projektu implementowanego w FPGA – dzięki preskalerowi (standardowy licznik zliczający w dół opisany na **list. 1**) uzyskujemy z niego przebieg o częstotliwości 56 Hz. Wybór stopnia podziału 16–bitowego preskalera odbywa się w linii *clk_int <= presk(15);*.

Interfejs obsługujący sterownik LCD zaprojektowano jako sekwencyjny automat z kolejno opisanymi stanami. Każdy takt sygnału zegarowego powoduje przejście automatu do kolejnego stanu, a każdemu stanowi przypisano jedną operację. Opis części przerzutnikowej automatu pokazano na **list. 2**, natomiast na **list. 3** pokazano fragment opisu układu decydującego o kolejności wykonywania kroków i operacjach im przypisanych.

Taka realizacja projektu daje możliwość przetestowania różnych sposo-

Kodowanie automatów

Zalety i wady różnych sposobów kodowania automatów są opisane w pomocy programu WebPack ISE.

List. 2. Opis licznika stanów (część „przerzutnikowa" automatu)

```
u1:   process(clk_int)
      begin
        if(clk_int'event and clk_int='1') then
           cs <= ns;
       end if;
      end process;
```

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

List. 3. Opis kombinacyjnej części automatu – tu są podejmowane decyzje o cyklu pracy licznika i operacjom realizowanym w poszczególnych krokach

```
u2:   process(cs)
        begin
             case cs is
             when s1=>
                ns<=s2;
                DC_LCD <= ,0';
                D_LCD  <=  x"30";
             when s2=>
                ns<=s3;
                DC_LCD <= ,0';
                D_LCD  <=  x"30";
             when s3=>
                ns<=s4;
                DC_LCD <= ,0';
                D_LCD  <=  x"30";
             when s4=>
                ns<=s5;
                DC_LCD <= ,0';
                D_LCD  <=  x"38";
.....
```

List. 7. Skrócony opis 4-bitowego sterownika LCD (wyświetla znaki w dwóch liniach wyświetlacza)

```
library IEEE;
use IEEE.std_logic_1164.all;
use IEEE.std_logic_unsigned.all;

entity LCDtest is
port (
       clk:    in std_logic;
--  zegar
       E_LCD:  out std_logic;
--  LCD enable
       DC_LCD: out std_logic ;
     --  data/control
       D_LCD:  out std_logic_vector(7 downto 4)
     );
end LCDtest ;

architecture LCD_architecture of LCDtest  is    --------
---------------------

type stan is (s1,s2,s3,s4,s5,s6,s7,s8,s9,s10,s11,s12,s13
,s14,s15,s16,s17,s18,s19,
s20,s21,s22,s23,s24,s25,s26,s27,s28,s29,s30,s31,s32,s33,
s34,s35,s36,s37,s38,s39,s40,s41,s42,s43,s44,s45,s46,s47,
s48,s49,s50,s51,s52,s53,s54,s55,s56,s57,s58,s59,s60,s61,
s62,s63,s64,s65,s66,s67,s68,s69,s70,s71,s72,s73,s74,s75,
s76,s77,s78);

signal cs: stan;
signal ns: stan;
signal presk: std_logic_vector(15 downto 0);
signal clk_int: std_logic;

begin
presc: process (clk)
begin
   if clk='1' and clk'event then
         presk <= presk - 1;
   end if;
end process;

clk_int <= presk(15);

u1:   process(clk_int)
      begin
        if(clk_int'event and clk_int='1') then
           cs <= ns;
       end if;
      end process;

u2:   process(cs)    stanów
        begin
             case cs is
             when s1=>
                  ns<=s2;
                  DC_LCD <= ,0';
                  D_LCD  <=  x"3";
           when s2=>
                  ns<=s3;
                  DC_LCD <= ,0';
                  D_LCD  <=  x"3";
           when s3=>
                  ns<=s4;
                  DC_LCD <= ,0';
                  D_LCD  <=  x"3";
             when s4=>
                  ns<=s5;
                  DC_LCD <= ,0';
                  D_LCD  <=  x"2";
             when s5=>
                  ns<=s6;
                  DC_LCD <= ,0';
                  D_LCD  <=  x"2";
             when s6=>
                  ns<=s7;
                  DC_LCD <= ,0';
                  D_LCD  <=  x"8";
             when s7=>
                  ns<=s8;
                  DC_LCD <= ,0';
                  D_LCD  <=  x"0";
             when s8=>
                  ns<=s9;
                  DC_LCD <= ,0';
                  D_LCD  <=  x"8";
             when s9=>
                  ns<=s10;
                  DC_LCD <= ,0';
                  D_LCD  <=  x"0";
             when s10=>
                  ns<=s11;
                  DC_LCD <= ,0';
                  D_LCD  <=  x"1";
              when s11=>
                  ns<=s12;
                  DC_LCD <= ,0';
                  D_LCD  <=  x"0";
             when s12=>
                  ns<=s13;
                  DC_LCD <= ,0';
                  D_LCD  <=  x"7";
             when s13=>
                  ns<=s14;
                  DC_LCD <= ,0';
                  D_LCD  <=  x"0";  -- wlacza LCD MS4b
             when s14=>
                  ns<=s15;
                  DC_LCD <= ,0';
                  D_LCD  <=  x"C";  -- wlacza LCD LS4b

             when s15=>
                  ns<=s16;
                  DC_LCD <= ,1';
                  D_LCD  <=  x"2";
             when s16=>
                  ns<=s17;
                  DC_LCD <= ,1';
                  D_LCD  <=  x"A";  -- kod ASCII *

             when s17=>
                  ns<=s18;
                  DC_LCD <= ,1';
                  D_LCD  <=  x"2";
             when s18=>
                  ns<=s19;
                  DC_LCD <= ,1';
                  D_LCD  <=  x"0";  -- kod ASCII SPACJA

             when s19=>
                  ns<=s20;
                  DC_LCD <= ,1';
                  D_LCD  <=  x"4";
             when s20=>
                  ns<=s21;
                  DC_LCD <= ,1';
                  D_LCD  <=  x"5";  -- kod ASCII E
.....
             when s77=>
                  ns<=s78;
                  DC_LCD <= ,0';
                  D_LCD  <=  x"0";  -- ON
             when s78=>
                  ns<=s77;
                  DC_LCD <= ,0';
                  D_LCD  <=  x"3";  -- ON
             when others =>
                  ns<=s1;
                  DC_LCD <= ,0';
                  D_LCD<=  x"0";
             end case;
        end process;

      E_LCD <= clk_int;
end LCD_architecture ;
```

List. 4. Kodowanie automatu *one–hot*

```
Optimizing FSM <cs> on signal <cs[1:42]> with one-hot encoding.
-----------------------------------------------------
 State | Encoding
-----------------------------------------------------
 s1    | 000000000000000000000000000000000000000001
 s2    | 000000000000000000000000000000000000000010
 s3    | 000000000000000000000000000000000000000100
 s4    | 000000000000000000000000000000000000001000
 s5    | 000000000000000000000000000000000000010000
 s6    | 000000000000000000000000000000000000100000
 ......................
 s40   | 001000000000000000000000000000000000000000
 s41   | 010000000000000000000000000000000000000000
 s42   | 100000000000000000000000000000000000000000
```

bów kodowania automatów i oceny wpływu kodowania na parametry projektu. W języku VHDL można ręcznie przypisać kombinacje bitów licznika automatu do konkretnych stanów (za pomocą dyrektywy *attribute enum_encoding of state_type: type is „0001 0010 0101 1000 1010”;*, oczywiście liczba podanych kombinacji, długość każdej z nich i kolejność przypisania zależy od projektanta i wymogów projektu), ale w naszym przypadku nie jest to konieczne. Na **list. 4** pokazano wynik automatycznego kodowania automatu *one–hot* (z jednym stanem aktywnym, czego zakodowanie wymaga 42 przerzutników), na **list. 5** pokazano wynik kodowania *compact* (do zakodowania wystarczy sześć przerzutników), a na **list. 6** kodowanie *Gray–encoding* (kolejne stany różnią się tylko jednym bitem). Wybór sposobu kodowania jest możliwy poprzez kliknięcie prawym przyciskiem w gałąź *Synthesize>Propertries* (w oknie procesów – **rys. 4**). W wyświetlonym oknie trzeba wybrać kategorię *HDL Options* i w niej *FSM Encoding Algorithm* (**rys. 5**).

List. 5. Kodowanie automatu *compact*

```
Optimizing FSM <cs> on signal
<cs[1:6]> with compact encoding.
-------------------
 State | Encoding
-------------------
 s1    | 001010
 s2    | 000110
 s3    | 100010
 s4    | 010010
 s5    | 010000
 s6    | 100000
 ..............
 s40   | 111110
 s41   | 000010
 s42   | 000000
```

List. 6. Kodowanie automatu *Gray–encoding*

```
Optimizing FSM <cs> on signal
<cs[1:6]> with gray encoding.
-------------------
 State | Encoding
-------------------
 s1    | 000000
 s2    | 000001
 s3    | 000011
 s4    | 000010
 s5    | 000110
 s6    | 000111
..............
 s40   | 110100
 s41   | 111100
 s42   | 111101
```

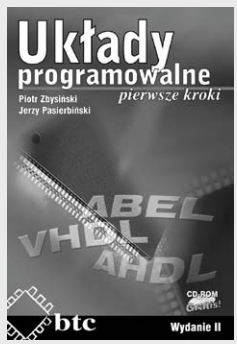
Uważni Czytelnicy dostrzegli z pewnością pewną niezgodność pomiędzy opisem pokazanym na list. 3 i schematami ilustrującymi sposób dołączenia wyświetlacza do FPGA: wyświetlacz jest bowiem sterowany w trybie 4–bitowym, a opis interfejsu operuje na bajtach! Ostateczną postać projektu interfejsu, pracującego w trybie 4–bitowym (narzuca nam to budowa płyty bazowej zestawu testowego ZL9PLD) pokazano na **list. 7**. Najpoważniejszą różnicą w stosunku do wcześniej opisanego, jest transfer bajtów w dwóch częściach (kolejno MS i LS) oraz inna procedura inicjalizacyjna, podczas której wybierany jest interfejs 4–bitowy.

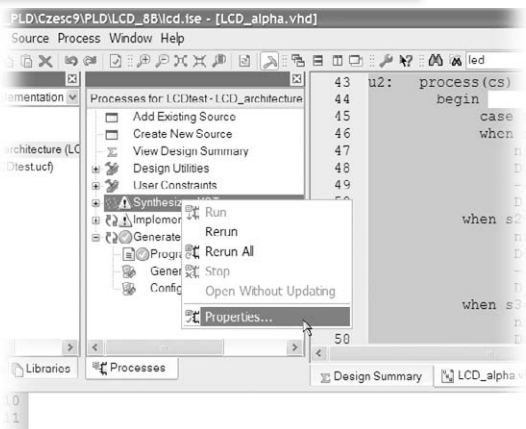
Podsumowanie

Skutkiem działania projektu opisanego w artykule jest wyświetlony w dwóch liniach tekst „* ELEKTRONIKA * * PRAKTYCZNA *”. Modyfikacja tego tekstu wymaga zmiany kodów ASCII wysyłanych do wyświetlacza w kolejnych stanach pracy automatu, co jest zajęciem dość żmudnym. Dlatego za miesiąc przedstawimy bardziej rozbudowany sterownik o większych możliwościach praktycznych.

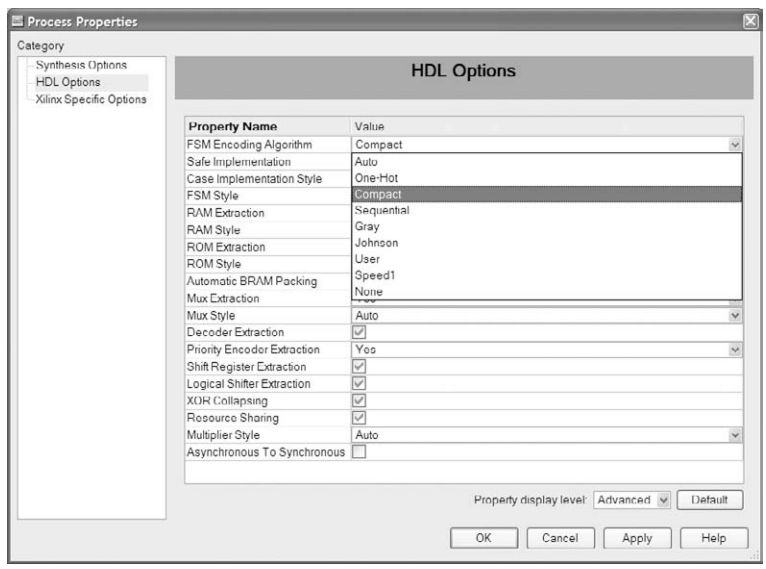
Jacek Majewski
jacek.majewski@pwr.wroc.pl
Piotr Zbysiński, EP
piotr.zbysinski@ep.com.pl

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*.





Rys. 4.



Rys. 5.