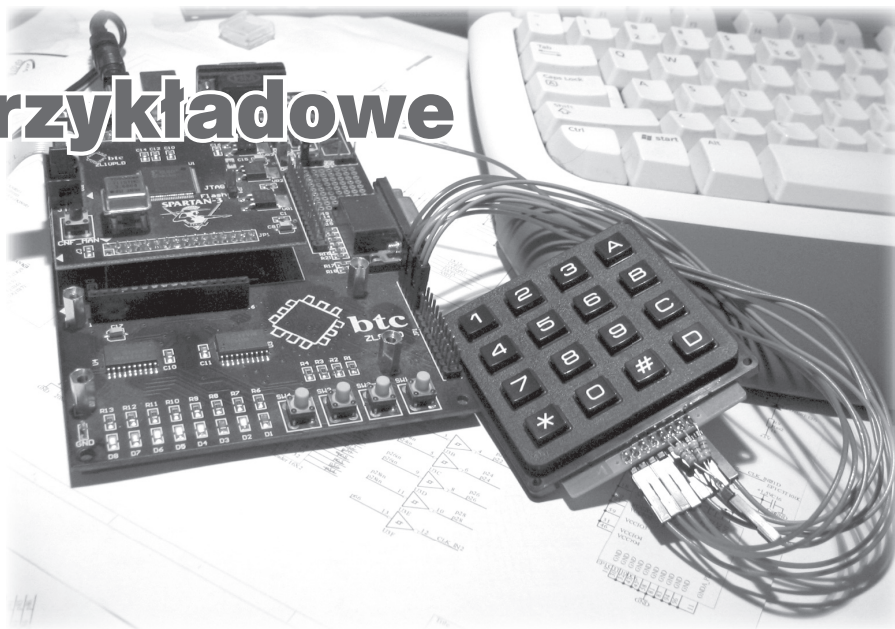


Układy FPGA w przykładach, część 6

Projekty przykładowe

Przedstawiamy kolejny projekt w języku opisu sprzętu VHDL, tym razem jest to moduł obsługujący 16-stykową klawiaturę matrycową o organizacji 4 wiersze x 4 kolumny. Klawiatury tego typu są chętnie stosowane w systemach cyfrowych, ze względu na możliwość odczytania stanu 16 przycisków za pomocą 8 linii FPGA: czterech wejściowych i czterech wyjściowych.

Sposób obsługi pojedynczych przycisków w układach FPGA pokazaliśmy miesiąc temu. W przypadku konieczności zastosowania w projektowanym systemie większej liczby przycisków zazwyczaj stosuje się matrycowe łączenie ich styków. Dzięki temu liczba linii I/O konieczna do odczytania ich stanów jest mniejsza niż niezbędna w przypadku dołączania pojedynczych przycisków bezpośrednio do linii I/O. W prezentowanym projekcie zastosowano 16-przyciskową klawiaturę telefoniczną, której schemat połączeń pokazano na rys. 1. Jak widać, do jej obsługi wystarczy 8 linii.

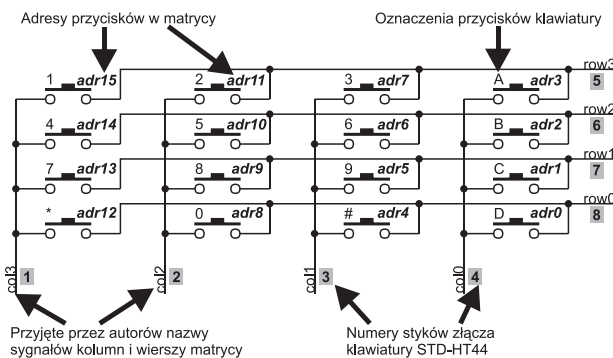


Realizacja

Schemat blokowy proponowanego rozwiązania przedstawiono na rys. 2. Jest to rozwiązanie wręcz podręcznikowe, odpowiadające zarówno klasycznym (dyskretnym) rozwiązaniom sprzętowym jak i im-

plementacjom programowym w mikrokontrolerach.

Układ działa następująco: 4-bitowy licznik CNT zlicza impulsy zegarowe podawane na wejście CLK, a jego wyjścia adresują: wyjścia 2-bitowego (4-wyjściowego) dekodera (na rys. 2 DEKODER, linie *kb_cnt3* i *kb_cnt2*) i 4-wejściowego multipleksera (linie *kb_cnt1* i *kb_cnt0*). Na wyjściach dekodera stanem aktywnym jest „0”, a wejścia multipleksera są podciągnięte do plusa zasilania za pomocą zewnętrznych rezystorów (wewnętrzne rezystory *pull-up* w układzie FPGA mają dużą rezystancję co powoduje, że bez zastosowania rezystorów zewnętrznych układ jest podatny na zakłócenia. Logiczne „0” pojawia się kolejno na kolumnach *col0...col3*. W przy-



Rys. 1. Schemat elektryczny połączeń klawiatury STD-HT44 (z oferty www.lcel.com.pl) wykorzystanej w projekcie prezentowanym w artykule

Czytelników zainteresowanych układami FPGA...

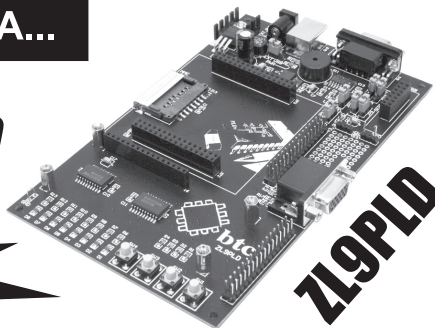
...zapraszamy do naszego sklepu internetowego, w którym oferujemy zestawy wykorzystywane w kursie prezentowanym w EP od numeru 10/2006:

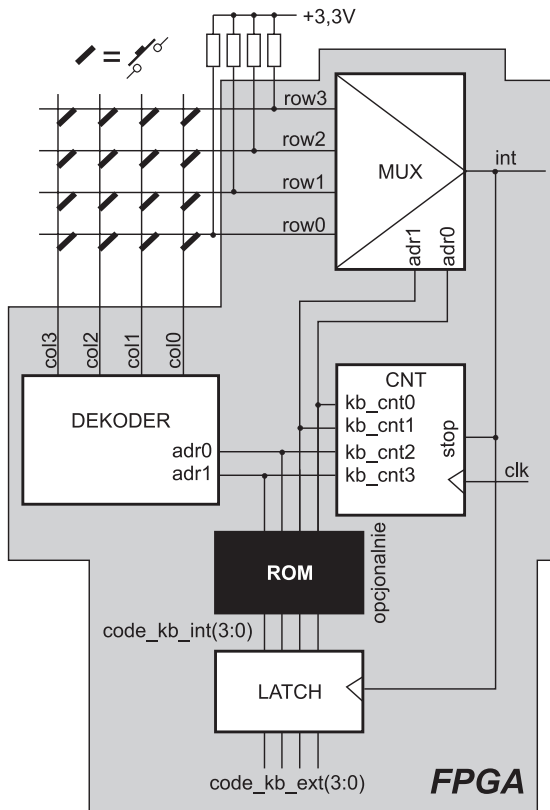
- ZL9PLD - uniwersalna płyta bazowa PLD,
- ZL10PLD - moduł DIP z układem XC3S200, konfiguratory, stabilizator napięcia zasilających, generatorem kwarcowym i przyciskiem rekonfiguracji

www.sklep.avt.pl

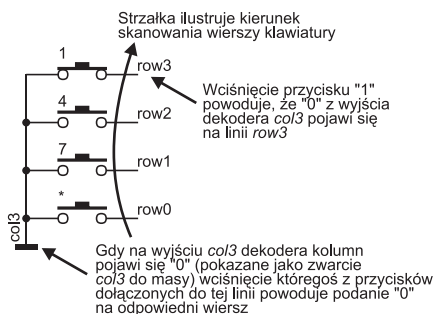
Moduł ZL10PLD wyposażono w układ Spartan 3 (XC3S200) i konfigurator Flash 1 Mb

ZL10PLD





Rys. 2. Schemat blokowy interfejsu klawiatury matrycowej, który zaimplementowano w układzie FPGA (elementy znajdujące się w FPGA znajdują się w szarym obrębie)



Rys. 3. Ilustracja działania dekodera klawiatury matrycowej przy założeniu, że wciśnięty przycisk o fizycznym adresie „1111” (dwójkowo)

padku zwarcia styków któregoś przycisku na linii odpowiadającego mu wiersza pojawia się „0” – rys. 3. Podczas analizowania pracy układu trzeba pamiętać, że po wybraniu kolumny skanowane są wszystkie wiersze (do ich adresowania wykorzystano dwa młodsze bity licznika CNT). Wykrycie wciśnięcia któregoś z klawiszy powoduje zatrzymanie licznika CNT w stanie, w którym wykryło wciśnięcie – czyli na jego wyjściach znajduje się adres wciśniętego przycisku. Sygnał zatrzymujący licznik (INT) można wykorzystać do zgłoszenia przerwania informującego o konieczności obłożenia klawiatury przez współpracujący np. mikrokontroler, można go także wykorzystać – jak w prezentowanym przykładzie – do wpisania numeru wciśniętego klawisza do rejestru *latch*, na wyjściach którego jest on utrzymywany do kolejnego wciśnięcia.

Jedynym nie zawsze potrzebnym fragmentem projektu jest pamięć ROM (jej opis w VHDL pokazano na list. 1), która spełnia rolę transkodera. Można ją wykorzystać do przypisania dowolnych (niekoniecznie 4-bitowych!) kodów poszczególnym przyciskom klawiatury. Ponieważ w przykładowym projekcie numerom klawiszy przypisano kody odpowiadające ich adresom, obecność tej pamięci nie jest odczuwalna. Jej opisać można zmodyfikować na przykład w sposób pokazany na list. 2 co spowoduje, że wciśnięcie przycisku o fizycznym

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

adresie 0 (mapę adresów pokazano na rys. 1) będzie powodowało wygenerowanie kodu 1111b itd. Na list. 1 i list. 2 pogrubioną czcionką zaznaczono miejsca, w których są określone kody przypisywane poszczególnym przyciskom.

Implementacja

Prezentowany projekt, podobnie do wszystkich pozostałych z naszego

List. 1. Zastosowany w projekcie przykładowym opis VHDL pamięci ROM

```
case kb_cnt is
  when "0000" => code_kb_int <= "0000";
  when "0001" => code_kb_int <= "0001";
  when "0010" => code_kb_int <= "0010";
  when "0011" => code_kb_int <= "0011";
  when "0100" => code_kb_int <= "0100";
  when "0101" => code_kb_int <= "0101";
  when "0110" => code_kb_int <= "0110";
  when "0111" => code_kb_int <= "0111";
  when "1000" => code_kb_int <= "1000";
  when "1001" => code_kb_int <= "1001";
  when "1010" => code_kb_int <= "1010";
  when "1011" => code_kb_int <= "1011";
  when "1100" => code_kb_int <= "1100";
  when "1101" => code_kb_int <= "1101";
  when "1110" => code_kb_int <= "1110";
  when "1111" => code_kb_int <= "1111";
  when others => code_kb_int <= "----";
```

List. 2. Zmodyfikowany opis VHDL pamięci ROM, który można zaimplementować w projekcie

```
case kb_cnt is
  when "0000" => code_kb_int <= "1111";
  when "0001" => code_kb_int <= "1110";
  when "0010" => code_kb_int <= "1101";
  when "0011" => code_kb_int <= "1100";
  when "0100" => code_kb_int <= "1011";
  when "0101" => code_kb_int <= "1010";
  when "0110" => code_kb_int <= "1001";
  when "0111" => code_kb_int <= "1000";
  when "1000" => code_kb_int <= "0111";
  when "1001" => code_kb_int <= "0110";
  when "1010" => code_kb_int <= "0101";
  when "1011" => code_kb_int <= "0100";
  when "1100" => code_kb_int <= "0011";
  when "1101" => code_kb_int <= "0010";
  when "1110" => code_kb_int <= "0001";
  when "1111" => code_kb_int <= "0000";
  when others => code_kb_int <= "----";
```

List. 3. Opis dekodera klawiatury matrycowej z pamięcią ROM pełniącą rolę transkodera

```

library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;

entity kb_mtx is
port (
    clk      : in  std_logic;
    res      : in  std_logic;
    int      : inout std_logic;
    row      : in  std_logic_vector (3 downto 0);
    col      : out std_logic_vector (3 downto 0);
    code_kb_ext : out std_logic_vector (3 downto 0)
);
end kb_mtx;

architecture keys of kb_mtx is
    signal kb_cnt : std_logic_vector (3 downto 0);
    signal code_kb_int : std_logic_vector (3 downto 0);

begin
    process (clk, int, res)
    begin
        if (res = '0') then kb_cnt <= «0000»;
        elsif (clk'event and clk = '1') then
            if (int = '1') then
                kb_cnt <= kb_cnt + 1;
            end if;
        end if;
    end process;

    col(0) <= '0' when kb_cnt(3 downto 2) = «00» else '1';
    col(1) <= '0' when kb_cnt(3 downto 2) = «01» else '1';
    col(2) <= '0' when kb_cnt(3 downto 2) = «10» else '1';
    col(3) <= '0' when kb_cnt(3 downto 2) = «11» else '1';

    with kb_cnt(1 downto 0) select
    int <= row(3) when «11»,
    row(2) when «10»,
    row(1) when «01»,
    row(0) when others;

    process (kb_cnt)
    begin
        case kb_cnt is
            when «0000» => code_kb_int <= «0000»;
            when «0001» => code_kb_int <= «0001»;
            when «0010» => code_kb_int <= «0010»;
            when «0011» => code_kb_int <= «0011»;
            when «0100» => code_kb_int <= «0100»;
            when «0101» => code_kb_int <= «0101»;
            when «0110» => code_kb_int <= «0110»;
            when «0111» => code_kb_int <= «0111»;
            when «1000» => code_kb_int <= «1000»;
            when «1001» => code_kb_int <= «1001»;
            when «1010» => code_kb_int <= «1010»;
            when «1011» => code_kb_int <= «1011»;
            when «1100» => code_kb_int <= «1100»;
            when «1101» => code_kb_int <= «1101»;
            when «1110» => code_kb_int <= «1110»;
            when «1111» => code_kb_int <= «1111»;
            when others => code_kb_int <= «----»;
        end case;
    end process;

    process (clk, int, code_kb_int, res)
    begin
        if (res = '0') then
            code_kb_ext <= «0000»;
        elsif (clk'event and clk = '1' and int = '0') then
            code_kb_ext <= code_kb_int;
        end if;
    end process;
end;

```

cyklu, uruchomiono i przetestowano na zestawie ZL9PLD i ZL10PLD. Płytki ZL9PLD to uniwersalna baza, natomiast ZL10PLD to moduł DIP z układem XC3S200 z rodziny Spartan 3 firmy Xilinx (nieco więcej pisaliśmy o tych płytkach w pierwszej części cyklu, w EP10/2006).

Sposób podłączenia klawiatury STD-HT44 do płytki bazowej ZL9PLD pokazano na **rys. 4**. Sygnały *col(3:0)* i *row(3:0)* można oczywiście dołączyć do dowolnych innych, dostępnych wyprowadzeń

sobów logicznych układu XC3S200, bowiem wraz z 15-bitowym preskalarem projekt zajął poniżej 1% dostępnych w nim zasobów. Opis w języku VHDL modułu obsługującego klawiaturę w konfiguracji pokazanej na **rys. 2** znajduje się na **list. 3**.

Kompletny projekt, wykorzystany do praktycznego testowania jest nieco bardziej skomplikowany, bowiem zawiera preskaler, który dzieli częstotliwość generatora kwarcowego 3,6864 MHz (taki jest standardowo instalowany na module

List. 4. Opis w języku VHDL zawierający opis połączeń pomiędzy preskalarem i modułem obsługi klawiatury matrycowej

```

library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_arith.all;
use ieee.std_logic_unsigned.all;

entity kb_top is port (
    clk      : in  std_logic;
    res      : in  std_logic;
    int      : inout std_logic;
    q_p      : out std_logic;
    row      : in  std_logic_vector (3 downto 0);
    col      : out std_logic_vector (3 downto 0);
    code_kb_ext : out std_logic_vector (3 downto 0)
);
end kb_top;

architecture behavioral of kb_top is
    signal q_presc : std_logic_vector (15 downto 0);

    component kb_mtx port (
        clk      : in  std_logic;
        res      : in  std_logic;
        int      : out std_logic;
        row      : in  std_logic_vector (3 downto 0);
        col      : out std_logic_vector (3 downto 0);
        code_kb_ext : out std_logic_vector (3 downto 0)
    );
    end component kb_mtx;

    component preskaler port (
        clk : in  std_logic;
        q : inout std_logic_vector (15 downto 0)
    );
    end component preskaler;

begin
    presc: preskaler port map (
        clk => clk,
        q => q_presc
    );

    kb_d: kb_mtx port map (
        clk => q_presc(14),
        res => res,
        int => int,
        row => row,
        col => col,
        code_kb_ext => code_kb_ext
    );

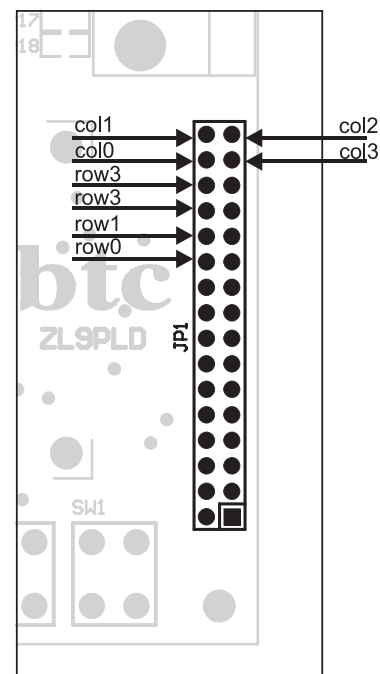
    q_p <= q_presc(14);
end behavioral;

```

układu FPGA, ale w projekcie (będzie dostępny na CD-EP4/2007B) zastosowano takie właśnie przypisanie sygnałów.

Implementacja prezentowanego projektu nie nadwyrężyła zasobów logicznych układu XC3S200, bowiem wraz z 15-bitowym preskalarem projekt zajął poniżej 1% dostępnych w nim zasobów. Opis w języku VHDL modułu obsługującego klawiaturę w konfiguracji pokazanej na **rys. 2** znajduje się na **list. 3**.

Kompletny projekt, wykorzystany do praktycznego testowania jest nieco bardziej skomplikowany, bowiem zawiera preskaler, który dzieli częstotliwość generatora kwarcowego 3,6864 MHz (taki jest standardowo instalowany na module



Rys. 4. Sposób dołączenia klawiatury do płytki bazowej ZL9PLD

ZL10PLD) do wartości ok. 112 Hz ($3686400 : 2^{15}$). Preskaler także opisano w VHDL, a obydwie moduły połączono w całość za pomocą opisu pokazanego na list. 4. W celach diagnostycznych, poza niezbędnymi sygnałami wynikającymi ze schematu pokazanego na rys. 1, w projekcie dodano wejście asynchronicznego zerowania *res* oraz wyjście sygnału zegarowego (wyjście preskaler), oznaczone jako *q_p*. Wyjścia rejestru *latch*, w którym jest zatrzaśkiwany numer (z wyjścia pamięci ROM) wciśniętego klawisza, dołączono do diod LED (rys. 5). Diody LED monitorują także sygnał taktujący moduł obsługi klawiatury oraz sygnał zgłoszenia przerwania *INT*.

Uważni Czytelnicy, zwróć uwagę na niezbyt czytelny – na pierwszy rzut oka – sposób taktowania rejestru wyjściowego *latch*. Ponieważ rejestr jest – co jest najbardziej logiczne – taktowany sygnałem *INT*, najprostszym sposobem opisanego źródła sygnału taktującego byłoby:

```
...elsif (int'event and int = '0')
then...
```

natomiast na list. 3 widać:

```
...elsif (clk'event and clk = '1' and
int = '0') then...
```

Zastosowano bowiem synchronizację sygnałem zegarowym *clk*, który zapobiega asynchronizacji sygnału *INT*, wytwarzanego w układzie kombinacyjnym, co wynika z opisu:

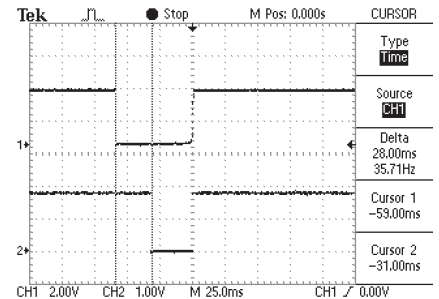
```
with kb_cnt(1 downto 0) select
int <= row(3) when „11”,
row(2) when „10”,
row(1) when „01”,
row(0) when others;
```

Taki sposób synchronizacji ma szczególny sens w przypadkach, kiedy układ FPGA pracuje na skraju swoich możliwości czasowych (czyli z całą pewnością nie w tym projekcie). Ta drobna komplikacja ma na celu pokazanie Czytelnikom poprawnego stylu opisu, przy czym obydwie przedstawione wersje będą działały, przy tak niskiej częstotliwości taktowania, prawidłowo.

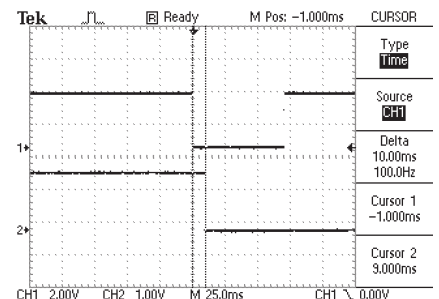
Na rys. 6 i rys. 7 pokazano przykładowe przebiegi na wyjściach układu FPGA po zaimplementowaniu w nim projektu.

Podsumowanie

Przedstawiony w artykule projekt – jak większość przygotowanych w językach HDL – jest podatny na różne modyfikacje. Jak wspomniano wcześniej, można zmodyfikować zawartość transkodującej pamięci ROM, sposób adresowania styków klawiatury czy przypisanie sygnałów do wyprowadzeń układu FPGA, można także dodać sygnalizację akustyczną wciśnięcia przycisku (brzęczyk pie-



Rys. 6. Sygnał *INT* (na dole) pojawia się z opóźnieniem, zależnym od chwili naciśnięcia przycisku (na górze linia *col3*)



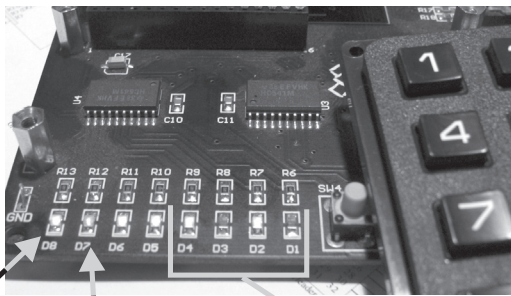
Rys. 7. Przykładowe zależności czasowe pomiędzy sygnałem *INT* (na górze) i odpowiedzią na wyjściu rejestru *latch*

zoelektryczny znajduje się na płycie ZL9PLD).

Wszystkich Czytelników, którzy podejmą samodzielne próby modyfikacji udostępnionego projektu lub przygotują własne – zapraszamy do podzielenia się z nami swoimi opracowaniami.

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

Kompletny projekt dla WebPacka 8.2i wraz z plikami źródłowymi opublikujemy na CD-EP4/2007B.



D8 – dioda monitorująca stan wyjścia *INT* D7 – dioda monitorująca wyjście preskaler D4-D0 – diody monitorujące stany wyjść rejestru *latch*

Rys. 5. Funkcje LED na płycie ZL9PLD po zaimplementowaniu projektu opisanego w artykule

Serdecznie zapraszamy naszych Czytelników do odwiedzenia naszego stoiska E6 na targach Automaticon 2007 w warszawskim Centrum EXPO XXI znajdującym się przy ul. Prądzyńskiego 12/ 14 (Wola, wjazd ul. Bema od ul. Kasprzaka) w dniach 13...16 marca w godz. 9.00 – 17.00, piątek 9.00 – 15.00.

Wstęp wolny

