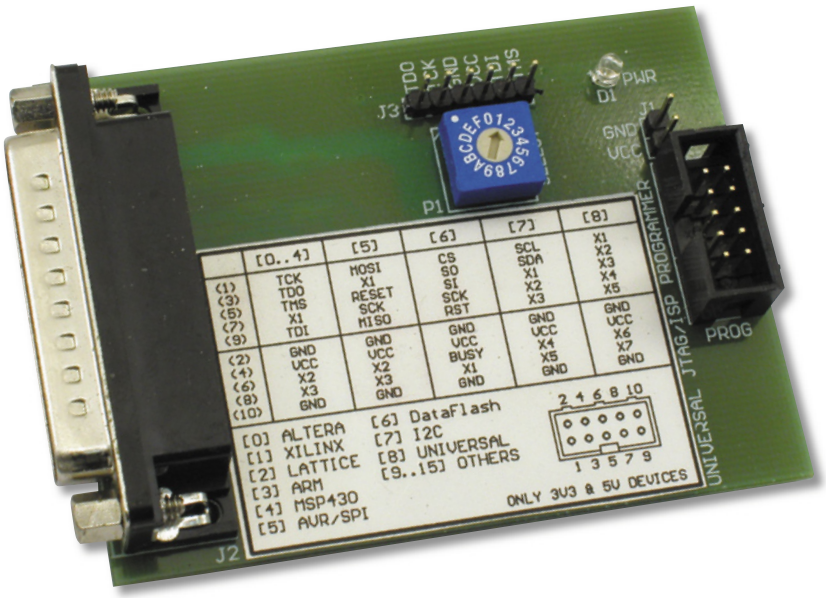
Uniwersalny programator JTAG/ISP



Mimo bogatej oferty na firmowe programatory przeznaczone zarówno dla określonej grupy układów, jak i uniwersalnych, amatorzy nie ustają w projektowaniu własnych konstrukcji. Może to jest tak, jak z posiłkiem, który inaczej smakuje, jak go sami przygotujemy. Własnoręcznie opracowane urządzenie powinno spełnić wszelkie „zachcianki" autora, jeśli tylko możliwe będzie osiągnięcie kompromisu pomiędzy ceną i skomplikowaniem układu, a jego funkcjonalnością.

Rekomendacje:

bardzo przydatne urządzenie dla każdego elektronika zajmującego się programowaniem różnych typów układów, coraz trudniejszym do spełnienia warunkiem jest jednak dostęp do komputera z portem LPT.

AVT-5153

W ofercie AVT:
AVT–5153A – płytka drukowana

PODSTAWOWE PARAMETRY

- Płytka o wymiarach 60x80 mm
- Zasilanie 3...18 VDC (zalecane napięcie 3,3 V)
- Emulacja firmowych interfejsów: Altera ByteBlaster II, Xilinx Parallel Cable III (DLC 5), Lattice ispDownload Cable, ARM Wiggler, TI MSP-FET, Kanda STK 200/300 i innych
- Programowane i konfigurowane układy: CPLD i FPGA: Xilinx, Altera, Lattice oraz mikrokontrolery z rdzeniem ARM, MSP430, AVR, Atmel 8051
- Złącze programujące: IDC10
- Interfejs wyjściowy: JTAG, ISP lub inny
- Wybór interfejsu za pomocą pokrętła
- Podłączany do portu drukarkowego LPT komputera

PROJEKTY POKREWNE wymienione artykuły są w całości dostępne na CD

Tytuł artykułu	Nr EP/EdW	Kit
Programator procesorów 89CX051	EdW 3/2000	AVT-2502
Programator procesorów AVR	EdW 10/2001	AVT-2550
JuPIC – programator mikrokontrolerów PIC współpracujący z programem MPLAB	EP 3/2003	AVT-5100
UnlProg – uniwersalny programator ISP	EP 1/2004	AVT-560
Uniwersalny programator mikrokontrolerów PIC	EP 5-7/2004	AVT-573
Interfejs JTAG do procesorów AVR	EP 6/2004	AVT-581
Miniprogramator AT89Cx051	EP 11/2004	AVT-540
Programator JTAG dla układów MSP430	EP 3/2005	AVT-1409
Flash z ISP – JTAG	EP 3/2006	AVT-921
Programator ISP/ICP dla mikrokontrolerów ST7	EP 7/2006	AVT-937
Programator JTAG dla mikrokontrolerów STR9	EP 9/2006	AVT-947
Uniwersalny adapter dla programatorów AVR-ISP	EP 2/2008	AVT-1462
Programator AVRISP z interfejsem USB (STK500)	EP 7/2007	AVT-988
Adapter dla programatorów AVR ISP	EP 7/2007	AVT-1452
Ulepszony programator STK200	EdW 2/2008	AVT-2855
Programator USB AVR (STK500)	EP 2/2008	AVT-5125

Dostępne obecnie na rynku układy logiki programowalnej (struktury CPLD i matryce FPGA) oraz pamięci nieulotne z interfejsem szeregowym (np. DataFlash), są wyposażone w jeden z wielu specjalizowanych interfejsów ISP (*In–System Programming*), dzięki któremu możliwe jest ich programowanie w systemie. Cecha ta, pozwala na stosowanie mechanizmów konfiguracji lub programowania wbudowanej w strukturę układu scalonego (np. mikrokontrolera) pamięci nieulotnej (typu EEPROM, bądź Flash) już po umieszczeniu go na płytce drukowanej. Co więcej, odbywa się to bez konieczności stosowania drogich i skomplikowanych programatorów zewnętrznych, czy też demontażu układu. Zaletami takiego rozwiązania są: skrócenie czasu uruchomienia pojedynczego układu scalonego (bądź całego systemu cyfrowego – pakietu), a na dalszym etapie ułatwiona staje się również wymiana oprogramowania, skrócenie czasu prototypowania i produkcji urządzenia oraz dostęp użytkownika do programowania zdalnego (np. przez port USB lub Internet) bez konieczności korzystania z autoryzowanego punktu serwisowego.

Typowy interfejs ISP jest szybką, szeregową i dwukierunkową magistralą, po której odbywa się komunikacja z programowanym obiektem – często, jak w przypadku mikrokontrolerów, korzysta się z dostępnych w układzie scalonym zasobów komunikacyjnych (np. portów SPI lub UART). Do zaprogramowania układu za pośrednictwem ISP konieczne jest użycie przystawki programującej, tj. prostego w budowie układu elektronicznego (określanego, nieco na wyrost, programatorem) podłączanego do portu komputera PC (najczęściej LPT), którego zadaniem jest pośredniczenie w wymianie danych z docelowym układem przez odpowiednie do-

pasowanie napięciowe sygnałów. W praktyce jednak stosowanie łącza ISP może utrudniać konstrukcję i prototypowanie urządzeń. Zastosowanie w danym projekcie mikrokontrolera (lub nierzadko kilku różnych), układu programowalnego CPLD oraz struktury FPGA, której plik konfiguracyjny jest przechowywany w niewielkiej pamięci EEPROM, wymaga od twórcy systemu wykorzystywania kilku różnych magistral ISP (najczęściej niekompatybilnych ze sobą). W konsekwencji nie można się obyć bez kilku odmiennych w budowie przystawek (programatorów). Niedogodność ta wynika z faktu, iż każdy z producentów współczesnych układów cyfrowych opracował i wypromował własne mechanizmy programowania i dedykowane im narzędzia. Podobne problemy pojawiają się także podczas realizacji szeregu projektów, w których z reguły stosowane są różnego rodzaju mikroprocesory (od prostych 8–bitowych do złożonych 32–bitowych architektur) oraz logika programowalna – układy te pochodzą często z różnych rodzin, a co za tym idzie, wyposażone są w odmienne łącza ISP. Niedogodności te są często sporą przeszkodą w praktyce amatorskiej i półprofesjonalnej, tym bardziej, że koszt zakupu kilku firmowych programatorów ISP jest dość znaczny.

Powyższe uwarunkowania skłoniły autora do opracowania w miarę uniwersalnego, taniego i łatwego w rozbudowie programatora – przystawki, który posiadałby funkcje dostępne na rynku firmowych interfejsów ISP.

Na marginesie należy wyjaśnić istotną różnicę pomiędzy pojęciami programowania i konfiguracji – wbrew pozorom nie oznaczającymi tego samego. Konfiguracja (w przypadku układów logiki programowalnej) jest terminem opisującym proces inicjalizacji zawartości specjalnej pamięci RAM układu FPGA odpowiedzialnej za przechowywanie tzw. kontekstu. Stąd matryce FPGA bazujące na komórkach RAM są często określane mianem układów rekonfigurowalnych. Programowanie natomiast dotyczy trwałego (nieulotnego) zapisu informacji w układzie scalonym – najczęściej za pomocą ładunku elektrycznego (np. komórek pamięci typu EEPROM, Flash) lub rzadziej z użyciem promienia lasera, bądź napięcia o podwyższonej wartości (np. komórki pamięciowe zawierające miniaturowe bezpieczniki). Dlatego wszystkie układy CPLD i niektóre FPGA (np. Actel czy QuickLogic) wymagają właśnie programowania.

Jako ciekawostkę można dodać, że interfejs JTAG (tj. standard IEEE1149.1) – obecnie powszechnie stosowany jako szyna programująca i konfigurująca układy PLD oraz mikrokontrolery – został stworzony z myślą o testowaniu wzajemnych połączeń pomiędzy układami oraz połączeń pomiędzy układami i elementami stanowiącymi ich otoczenie. Umożliwiał on również testowanie zarówno pojedynczych struktur półprzewodnikowych, jak i złożonych systemów cyfrowych (testowanie metodą tzw. ścieżki krawędziowej). Standard ten był przez wiele lat stosowany wyłącznie do tych celów. Dopiero później wprowadzono jego rozszerzenie o mechanizmy konfiguracji i programowania w systemie, co zaowocowało normą IEEE1532. Szerzej na ten temat można przeczytać w [2, 3].

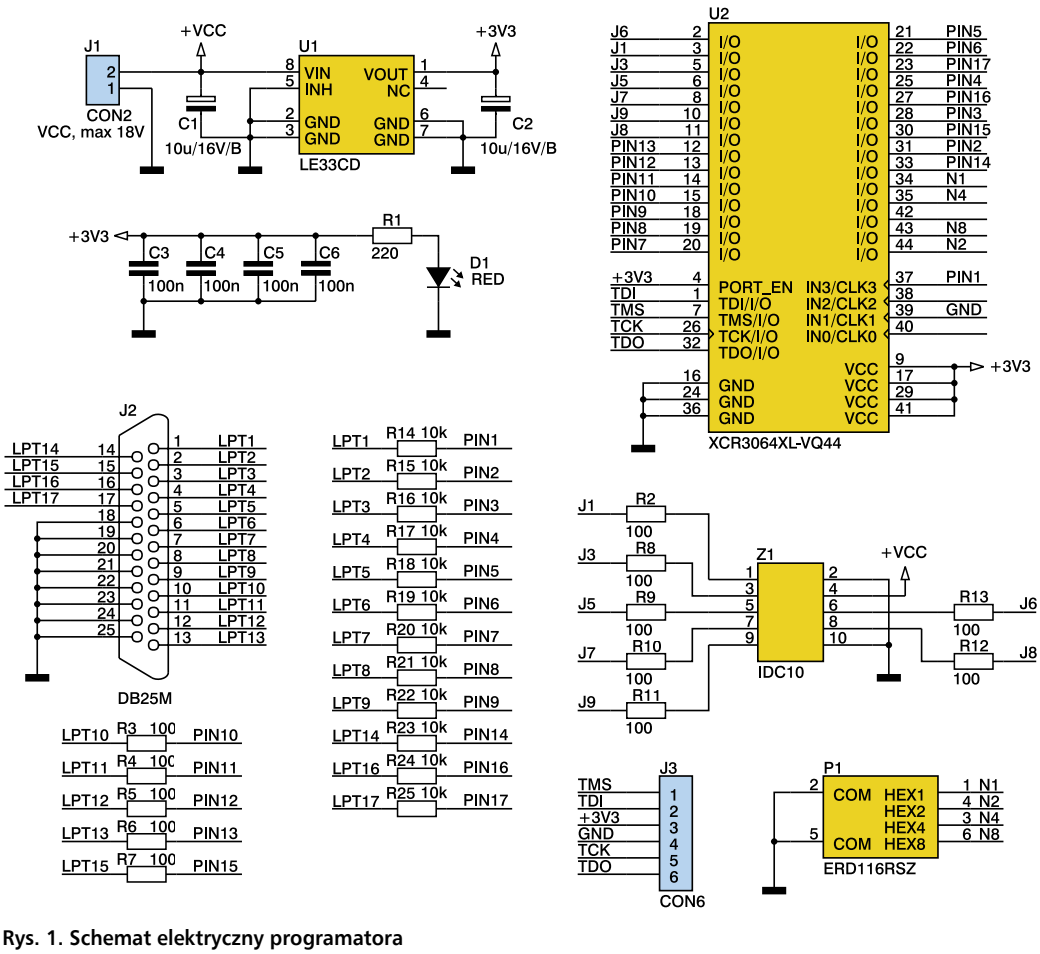
Opis układu

Moduł służy do programowania w systemie układów cyfrowych (takich jak mikrokontrolery, struktury programowalne PLD, pamięci Flash), które są wyposażone w typowy szeregowy interfejs programujący (np. JTAG, SPI lub I²C). Programator ten spełnia rolę emulatora firmowej przystawki (konwertera połączeń elektrycznych) pomiędzy portem równoległym LPT komputera PC, na którym jest uruchomione środowisko projektowe z narzędziami wspierającymi programowanie układu w systemie (np. WebPack Xilinx dla matryc programowalnych PLD), oraz docelowym układem (np. procesorem) wlutowanym na płycie PCB w uruchamianym urządzeniu.

Schemat elektryczny programatora przedstawiono na **rys. 1**. Dzięki zastosowaniu logiki programowalnej CPLD firmy Xilinx z mikromocowej rodziny *CoolRunner* (układ U2) uproszczona została jego budowa wewnętrzna – wykorzystuje się tylko niewielką liczbą elementów do jego konstrukcji. Dodatkowo pobór prądu urządzenia jest bardzo niski.

Ogólnie rzecz biorąc, układ U2 odpowiada za sprzętową emulację firmowych interfejsów programatorów. Projekt zaszytego w nim układu logicznego został opracowany w języku opisu sprzętu VHDL, dzięki czemu możliwa jest łatwa jego modyfikacja do własnych wymagań.

Do poprawnej pracy układu U2 wymagane jest napięcie zasilania o wartości 3,3 V. Jest ono uzyskiwane za pomocą popularnego stabilizatora LDO (U1). Napięcie zasilające programator (o wartości z zakresu 3...18 V) może być podane w dwojaki sposób: zewnętrznie – poprzez złącze J1 lub doprowadzone wprost z obiektu programowanego, tj. przez linię nr 4 złącza Z1 (zalecane rozwiązanie). Ponieważ układ U1 jest regulatorem napięcia typu LDO, napięcie zasilające programator może posiadać obniżoną wartość z zakresu 3...4 V. Poniżej wartości punktu stabilizacji 3,3 V układ ten przenosi napięcie z niewielkim spadkiem (rzędu miliwoltów). Cecha ta pozwala na zasilanie programatora ty-



Rys. 1. Schemat elektryczny programatora

powym napięciem zasilania układów cyfrowych o wartości 3,3 V.

Bardzo istotną i pożyteczną właściwością zastosowanego układu PLD (U2) jest przystosowanie jego linii I/O do współpracy ze standardami napięciowymi 3,3 V i 5 V (tj. LVTTL i TTL) – układ ten posiada tolerancję napięcia 5 V (*5V Tollerant*).

Wybór typu emulowanego interfejsu programatora odbywa się za pomocą nastawnika binarnego (P1), który może wskazywać jedną z szesnastu pozycji (tym samym można wybrać jeden z szesnastu różnych programatorów).

Programator jest podłączany do komputera PC przez złącze drukarkowe (LPT), które jest najpopularniejszym sposobem komunikacji pomiędzy środowiskiem projektowym (aplikacją) i obiektem docelowym, programowanym przez złącze ISP.

Wszystkie linie I/O złącza LPT są doprowadzone do układu PLD, co pozwala na odwzorowanie struktury logicznej dowolnego programatora firmowego. Na potrzeby interfejsu ISP przewidziano siedem dwukierunkowych linii, dostępnych na złączu Z1 – linie te odpowiadają za sterowanie programowanym układem. Wszystkie wyprowadzenia I/O układu U2, pełniące funkcję sygnałów emulowanych interfejsów, zostały zabezpieczone przed ich przypadkowym zwarciem lub podaniem na nie zbyt wysokiego napięcia. Osiągnięto to przez włączenie w ich obwód szeregowych rezystorów (R2...R25). Rezystancja tych elementów została tak dobrana, aby ograniczyć również negatywny wpływ długości zewnętrznych połączeń na jakość parametrów sygnału cyfrowego przy zapewnieniu odpowiednio dużej szybkości narastania napięcia w liniach ISP programatora.

Dioda D1 sygnalizuje obecność napięcia zasilania 3,3 V oraz poprawną pracę programatora. Na górnej stronie płytki PCB programatora umieszczono tabelkę opisową, zawierającą skrócony opis pełnionych funkcji przez wyprowadzenia złącza Z1 w zależności od aktualnego położenia nastawnika P1. Płytka drukowana została zaprojektowana z jednostronną warstwą ścieżek i obustronnym obrysem elementów, natomiast do budowy układu elektrycznego wykorzystano ogólnie dostępne elementy SMD (z wyjątkiem złącz), dzięki czemu uzyskano znaczną miniaturyzację urządzenia, uproszczono jego montaż i obniżono całkowity koszt wykonania.

Montaż i uruchomienie

Rozmieszczenie elementów po obu stronach płytki drukowanej przedstawiono na **rys. 2**. Montaż elementów należy rozpocząć od przylutowania elementów biernych. Następnie należy zamontować złącza oraz dwa mostki (zworki) na dolnej stronie elementów, oznaczone odcinkami i zakończone obustronnie punktami lutowniczymi. W ostatniej kolejności montowane powinny być układy scalone. W przypadku montażu układu U2 należy zachować szczególną ostrożność, ponieważ posiada on niewielki raster wyprowadzeń.

WYKAZ ELEMENTÓW

programator

Rezystory
R1: 220 Ω, SMD 0805
R2...R13: 100 Ω, SMD 0805
R14...R25: 10 kΩ, SMD 0805
Kondensatory
C1, C2: 10 µF/16V, SMD, tantalowy, typ B
C3...C6: 100 nF, SMD 0805
Półprzewodniki
U1: LE33CD, SMD SO8
U2: XCR3064XL–10VQ44C
D1: dioda LED, czerwona, 3 mm
Inne
J1: listwa kołkowa gold–pin, 2x1, raster 2,54 mm
J2: złącze DB25M: DHP8–25M, męskie, kątowe, do druku
J3: listwa kołkowa gold–pin, 6x1, raster 2,54 mm
P1: zadajnik kodu ERD116RSZ (lub z gałką ERD216RSZ), 16 położeń
Z1: gniazdo IDC10: proste ZL231–10PG (lub kątowe ZL231–10GK), do druku

moduł XJTAG

Rezystory
R1..R7: 220 Ω, SMD 0805
R8...R14: 10 kΩ, SMD 0805
Kondensatory
C1: 10 µF/16 V, SMD, tantalowy, typ B
C2: 100 nF, SMD 0805
Półprzewodniki
D1: BAT43, SMD
D2: dioda LED, czerwona, 3 mm
U1: 74HC125, SMD
Inne
J1: złącze DB25M: DHP8–25M, męskie, kątowe, do druku
J2: listwa kołkowa gold–pin, 6x1, raster 2,54 mm
wiązka przewodów połączeniowych CAB_A(B)

Na CD karty katalogowe i noty aplikacyjne elementów oznaczonych kolorem czerwonym

Uruchomienie programatora polega na zaprogramowaniu układu U2 plikiem w formacie JEDEC (*.jed), zawierającym obraz docelowej konfiguracji. W tym celu należy podłączyć przez łącze J3 (wyprowadzenia magistrali JTAG układu U2) firmowy programator Xilinx (np. *Parallel Cable III*) lub jego funkcjonalny odpowiednik – zakupiony, albo wykonany samodzielnie (w Internecie dostępnych jest wiele amatorskich konstrukcji, firma Xilinx również udostępnia schemat swojej oryginalnej konstrukcji). Autorskie opracowanie takiej przystawki umieszczono też w tym artykule jako moduł uruchomieniowy XJTAG (schemat ideowy oraz montażowy zostały przedstawione na **rys. 3** i **4**). Jest on bardzo prosty w budowie i nie wymaga szerszego omówienia.

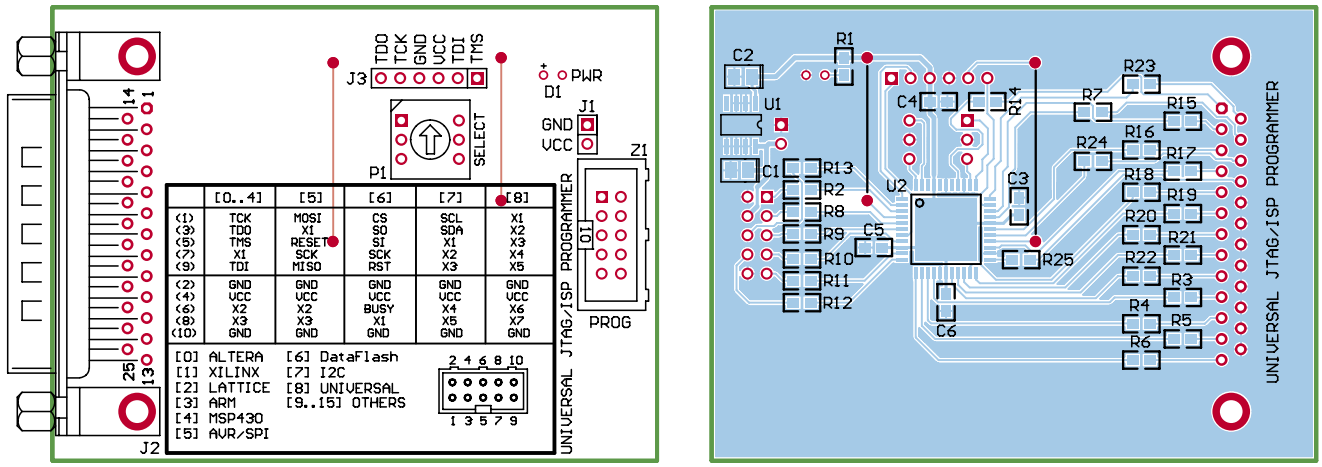
Moduł XJTAG należy podłączyć poprzez złącze LPT do komputera PC, na którym zostało zainstalowane darmowe środowisko projektowe *Xilinx ISE WebPack*. Program ten umożliwia kompilację i implementację projektów dla układów programowalnych CPLD i FPGA firmy Xilinx – jest on niezbędny w przypadku, gdy zaistnieje potrzeba modyfikacji opisu sprzętu (pliku VHDL) programatora oraz jego kompilacji. Darmową wersję pakietu instalacyjnego *ISE WebPack* można pobrać ze strony Xilinx.com po wcześniejszej rejestracji. Autor projektu zaleca pobranie jednej ze starszych wersji środowiska (dostępnych w dziale *ISE Classics* – kategoria *Design Tools*), np. v7.1i z dodatkowym plikiem poprawek *ServicePack 4* (objętość plików instalacyjnych kolejnych wersji znacznie przekracza 0,5 GB – w wersji v9.1i jest to już 1,4 GB!). W przypadku gdy nie zamierzamy ingerować w strukturę logiki programowalnej, a jedynie chcemy zaprogramować układ U2, możemy zainstalować specjalną, zubożoną i darmową wersję środowiska *ISE WebPack*, zawierającą niezbędne komponenty do programowania i konfigurowania układów PLD Xilinx w postaci aplikacji *iMPACT*. Taka dystrybucja oprogramowania zajmuje w formie instalacyjnej zaledwie około 40 MB dla wersji v6.3i (niestety, nie są one obecnie dostępne na stronie WWW producenta, autor artykułu dysponuje wersją v6.3i).

Po zaprogramowaniu układu U2 należy sprawdzić całkowity pobór prądu, który powinien zawierać się w zakresie 5...10 mA. Elementy użyte do budowy są ogólnodostępne. Oferowane są przez większość witryn internetowych zajmujących się dystrybucją elementów elektronicznych. Nastawnik kodowy P1 występuje w kilku wersjach – z pokrętłem lub z nacięciem na śrubokręt, a liczba jego pozycji wynosi 10 lub 16 – wyboru właściwego modelu należy dokonać po ewentualnej modyfikacji projektu zawartego w układzie PLD programatora.

Eksploatacja

W logice programowalnej programatora zaszyto kilka najbardziej popularnych obecnie firmowych interfejsów wspomagających uruchamianie układów z wbudowanym łączem ISP, takich jak struktury FPGA czy też mikrokontrolery. W **tab. 1** przedstawiono zależność położenia przełącznika P1 od aktualnie wybranego interfejsu. Dla ułatwienia pracy z urządzeniem, na górnej stronie płytki drukowanej programatora umieszczono tabelkę opisową, w której znajduje się krótka charakterystyka wyprowadzeń złącza Z1 oraz nastawnika kodowego P1. Połączenia pomiędzy obiektem docelowym a programatorem można wykonać za pomocą wiązki przewodów lub przy użyciu wykonanej we własnym zakresie 10–żyłowej tasiemki, zakończonej obustronnie złączami zaciskanymi typu IDC.

Przed przystąpieniem do pracy należy ustawić przełącznikiem P1 właściwy tryb, podłączyć gniazdo J2 (DB25) do portu równoległego LPT komputera PC przy pomocy przedłużacza (zalecane jest, aby długość kabla nie przekraczała 2 m) oraz doprowadzić tasiemką wieloprzewodową (o długości nie większej niż 50 cm) odpowiednie sygnały sterujące złącza IDC10 do programowanego elementu. Działanie pro-



Rys. 2. Schemat montażowy programatora (strona górna i dolna)

gramatora jest sygnalizowane czerwoną diodą LED (D1).

Programator współpracuje poprawnie z układami, których interfejs ISP jest zgodny ze standardem napięciowym LVTTL (3,3 V) lub TTL (5 V), a także jest przystosowany do komunikacji z portem równoległym LPT, operującym na logice 5 V. Dopuszcza się również możliwość programowania i konfigurowania układów z magistralą ISP o poziomach logicznych zgodnych z LVTTL (2,5 V), np. matryc FPGA firmy Xilinx z interfejsem JTAG (układy serii Spartan 3). W tym przypadku należy na liniach interfejsu komunikacyjnego dobrać wartość rezystancji szeregowych, ograniczających płynący przez nie prąd (zaleca się wartości z zakresu 200...300 Ω). Problem ten został szerzej opisany na łamach EP w [1].

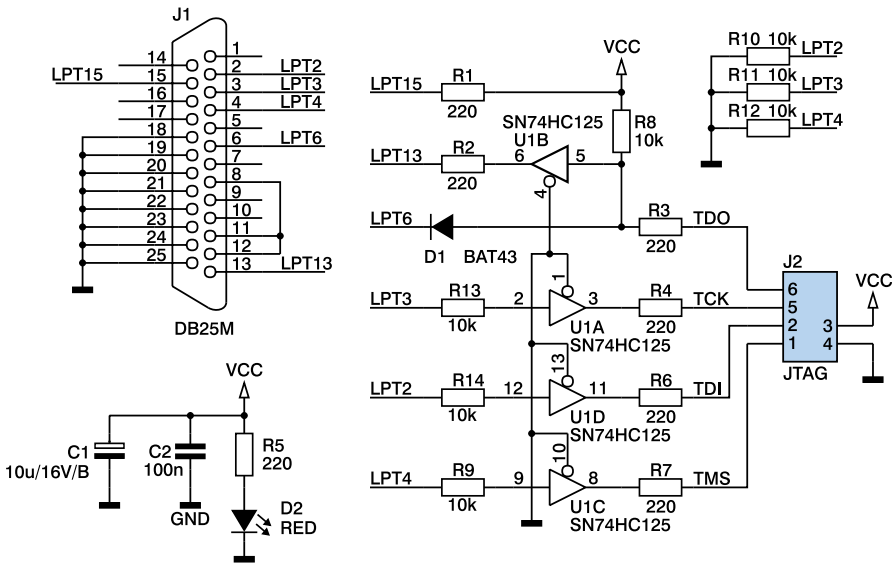
Schematy logiczne wszystkich emulowanych przez programator interfejsów zostały zawarte w materiałach dodatkowych, dołączonych na płycie CDEP10/2008B. Schematy elektryczne interfejsów firmowych można znaleźć również w Internecie. Jedynie w przypadku dwóch magistral ISP umożliwiających programowanie układów (tj. DataFlash i I²C) opracowano własny schemat logiczny (**rys. 5**).

Kod źródłowy

Projekt logiki programowalnej urządzenia został opracowany w języku VHDL. Do modyfikacji kodu źródłowego (poniżej umieszczono jego fragment) jest wymagana jego podstawowa znajomość (choć nie jest to konieczne – na podstawie krótkiej analizy kodu można wprowadzać w nim dowolne zmiany bez potrzeby wgłębiania się w szczegóły języka VHDL). W części deklaracyjnej ciała architektury zdefiniowano stałe wektory bitowe, odwzorowujące nastawy zadajnika kodowego P1 oraz sygnał PROGRAMMER, który niesie informację o trybie pracy układu.

```
[...]
architecture RTL of UNIPROG is
signal PROGRAMMER:  std_logic_vector(3 downto
0);
constant ALTERA:  std_logic_vector(3 downto 0)
:= „0000”;
constant XILINX:  std_logic_vector(3 downto 0)
:= „0001”;
constant LATTICE:  std_logic_vector(3 downto
0) := „0010”;
constant ARM:  std_logic_vector(3 downto 0) :=
„0011”;
constant MSP430:  std_logic_vector(3 downto 0)
:= „0100”;
constant AVR:  std_logic_vector(3 downto 0) :=
„0101”;
constant DATAFLASH:  std_logic_vector(3 downto
0) := „0110”;
constant I2C:  std_logic_vector(3 downto 0) :=
„0111”;
constant UNIVERSAL:  std_logic_vector(3 downto
0) := „1000”;

[...]

PROGRAMMER <= not (N8 & N4 & N2 & N1);

[...]

--sterowanie wyjściem LPT13
process(PROGRAMMER, J3, J6, J7, LPT8)
begin
 case PROGRAMMER is
  when ALTERA =>
   LPT13 <= J7;
  when XILINX =>
   LPT13 <= J3;
  when LATTICE =>
   LPT13 <= LPT8;
  when ARM =>
   LPT13 <= '1';
  when DATAFLASH =>
   LPT13 <= J6;
  when I2C =>
   LPT13 <= J3;
  when others =>
   LPT13 <= '-';
 end case;
end process;
[...]
```

W ciele architektury opisano procesy kombinacyjne za pomocą konstrukcji `CASE-WHEN`,



Rys. 3. Schemat elektryczny modułu XJTAG

W ciele architektury opisano procesy kombinacyjne za pomocą konstrukcji `CASE-WHEN`,

Tab. 1. Nastawy przełącznika P1 i odpowiadający mu wybór firmowych programatorów ISP oraz układów

Ustawienie przełącznika P1	Interfejs firmowy programatora	Wsparcie programowania układów
0	Altera ByteBlaster II	Altera FPGA & CPLD
1	Xilinx Parallel Cable III	Xilinx FPGA & CPLD
2	Lattice ispDownload Cable	Lattice FPGA & CPLD
3	ARM Wiggler Interface	Mikrokontrolery z rdzeniem ARM
4	Texas Instruments MSP–FET	Mikrokontrolery MSP430
5	Kanda STK 200/300	Mikrokontrolery ATMEL AVR
6	własny	ATMEL DataFlash
7	własny	I²C Bus
8...15	brak	dla dalszej rozbudowy

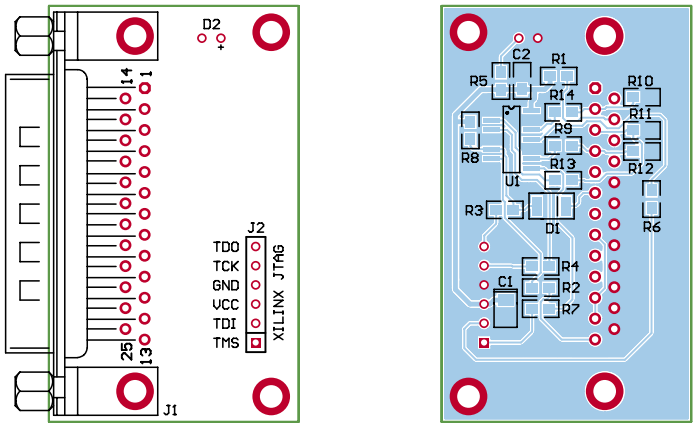
realizujące właściwy układ logiczny. Taka budowa kodu źródłowego pozwala w przejrzysty i czytelny sposób opisać strukturę projektu oraz umożliwia jego adaptację do potrzeb użytkownika. Podczas modyfikacji projektu należy pamiętać o stosowaniu stanów nieokreślonych ('–'), co wydatnie wpłynie na stopień minimalizacji funkcji logicznych niezbędnych do realizacji struktury logicznej i przyczyni się do pomyślnego wpasowania projektu w docelowy układ CPLD.

Szerzej temat programowania i projektowania układów cyfrowych z użyciem układów PLD oraz języka VHDL został opisany na łamach Elektroniki Praktycznej oraz w literaturze [4, 5, 6, 7].

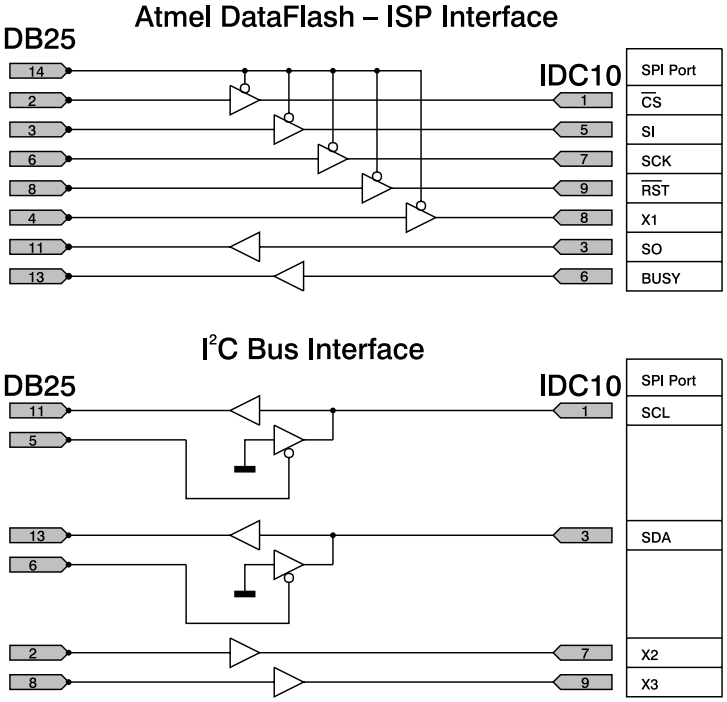
Podsumowanie

Autor korzysta z urządzenia od ponad dwóch lat – spełnia ono swoją funkcję bez zastrzeżeń. Użytkownik może w dowolny sposób rozbudowywać programator (i korzystać z niego niekoniecznie w celu programowania), np. zaadaptować interfejs kart pamięci SD, MMC, SmartCard lub SIM, podłączyć sieć rozproszoną czujników (np. termometrów), sterować wyświetlaczami graficznymi LCD. Możliwości funkcjonalne urządzenia są spore. Pełna lista układów, które można zaprogramować za pomocą niniejszego programatora, jest dość długa i obejmuje z pewnością wiele tysięcy pozycji. Jedynym znaczącym mankamentem korzystania z programatora jest konieczność posiadania komputera wyposażonego w port LPT, co obecnie niestety jest coraz rzadsze z powodu zdominowania rynku PC przez magistralę USB.

Mirosław Folejewski



Rys. 4. Schemat montażowy modułu XJTAG (strona górna i dolna)



Rys. 5. Schemat logiczny zaimplementowanych w programatorze interfejsów pamięci szeregowych DataFlash oraz szyny I²C

Literatura:

[1] Zbysiński P.: PicoBlaze: sposoby konfiguracji układów FPGA z rodziny Spartan–3, Elektronika Praktyczna, 10–11/2005
[2] Zbysiński P, Pasierbiński J.: Nowoczesne metody programowania i konfigurowania układów PLD, Elektronizacja, 7–8/2001
[3] Zbysiński P.: JTAG – światowy standard testowania i programowania układów cyfrowych, Elektronika Praktyczna, 1–2/1998
[4] Kalisz J.: Język VHDL w praktyce, WKŁ, Warszawa 2002
[5] Skahill K.: Język VHDL. Projektowanie programowalnych układów logicznych, WNT, Warszawa 2001
[6] Zbysiński P, Pasierbiński J.: Układy programowalne – pierwsze kroki, Wydawnictwo BTC, Warszawa 2002
[7] Pasierbiński J., Zbysiński P.: Układy programowalne w praktyce, WKŁ, Warszawa 2001
[8] A. Dybkowski: ISP Programmer, http://www.amwaw.edu.pl/~Adybkows/elka/ispprog.zip