Krzysztof ARNOLD, Sławomir MICHALAK
POLITECHNIKA POZNAŃSKA, WYDZIAŁ ELEKTRONIKI I TELEKOMUNIKACJI,
ul. Polanka 3, 60-965 Poznań

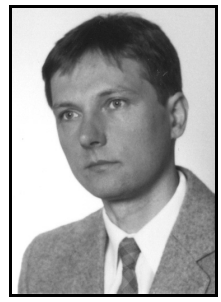
Programowalny kontroler mikroprocesorowych układów transmisji równoległej z interfejsem SPI

Dr inż. Krzysztof ARNOLD

Absolwent Wydziału Elektroniki Politechniki Gdańskiej. Pracuje jako adiunkt w Katedrze Systemów Telekomunikacyjnych i Optoelektroniki na Wydziale Elektroniki i Telekomunikacji Politechniki Poznańskiej. W pracy naukowej zajmuje się problemami pomiarów charakterystyk i parametrów sygnałów stochastycznych, tematyką akwizycji danych w systemach pomiarowych oraz zagadnieniami projektowania, diagnostyki i rozwoju mikroprocesorowych systemów pomiarowych.

e-mail: karnold@et.put.poznan.pl

Dr inż. Sławomir MICHALAK



Pracuje jako adiunkt w Katedrze Systemów Telekomunikacyjnych i Optoelektroniki na Wydziale Elektroniki i Telekomunikacji Politechniki Poznańskiej. W pracy naukowej i dydaktycznej zajmuje się zagadnieniami komputerowego wspomagania projektowania i symulacji układów elektronicznych (ECAD), programowaniem układów mikroprocesorowych i układów programowalnych. Interesuje się tematyką pozyskiwania informacji z inteligentnych czujników pomiarowych.

e-mail: michalak@et.put.poznan.pl

Streszczenie

W pracy wskazano możliwości zwiększenia liczby portów równoległych w systemach mikroprocesorowych i rozszerzenia trybów ich pracy z wykorzystaniem programowalnych układów peryferyjnych. Omówiono istniejące ograniczenia i zaproponowano rozwiązanie problemu multiliniowej komunikacji mikrokontrolerów z otoczeniem przez sterowanie układów PPI z poziomu kontrolera CPLD. Zaprezentowano architekturę kontrolera, komunikującego się z jednostką centralną przez interfejs SPI. Przedstawiono wyniki implementacji kontrolera w strukturze CPLD.

Słowa kluczowe: mikroprocesory, podsystem transmisji równoległej, kontroler magistrali, układy CPLD.

A programmable controller of microprocessor PPI devices with SPI interface

Abstract

In this paper the possibility of increasing parallel inputs and outputs in microprocessor systems with programmable peripheral interface (PPI) is presented. An idea of the PPI subsystem with a central processor unit (CPU), a serial programmed bus/address/interrupt controller and parallel transmission devices is proposed (Fig. 1). The Serial Peripheral Interface (SPI) communication protocol between the CPU and the controller is used for sending instructions and data, where the CPU works as a *master* and the controller as a *slave*. The controller is responsible for address decoding, data transferring and interrupts receiving (Fig. 2). The SPI interface minimizes the necessary I/O ports of CPU, therefore only two additional signals /STR and /INT0 are required. The instruction sequences and the data are composed of two bytes (Fig. 3), the higher one includes codes for creating control signals for the controller and read/write cycles for 82C55A devices (Tab. 1). The block diagram of the PPI subsystem with a CPLD controller and an ATmega 16A microcontroller is shown in Fig. 4. The controller was implemented in the XC9572XL device (Tab.2) and the Behavioral and Post-Fit Simulations were made for functional tests. The Xilinx XC9500XL family is fully 5V (CMOS, TTL) tolerant even though the core power supply is 3.3 volts, so the controller can work in mixed (5V/3.3V/2.5V) systems, with low power supply microprocessors. Use of this one programmable device give us a chance for creating a flexible controller, which can work with any kind of central units supported SPI interface.

Keywords: microprocessors, PPI subsystem, bus controller, CPLD.

1. Wstęp

Przepływ informacji we współczesnych systemach mikroprocesorowych odbywa się z wykorzystaniem standardowej magistrali i modułów peryferyjnych, wspierających poszczególne systemy interfejsu. Zamknięcie magistrali w obrębie mikrokontrolerów i rozwój struktur wbudowanych przesuwają odpowiedzialność za realizację większości zadań systemowych w stronę pojedynczego układu VLSI. Pełna integracja zadań może być jednak dyskusyjna wobec alternatywy efektywnej współpracy kilku jednostek. Dotyczy to w szczególności problemu komunikacji jednostki centralnej systemu mikroprocesorowego z urządzeniami zewnętrznymi, zwłaszcza przy prowadzeniu wielokanałowej transmisji równoległej. Konieczność taka pojawia się podczas obsługi przetworników c/a, wyświetlaczy i klawiatury, a także przyjmowania informacji z przetworników a/c, zespołów kluczy, nastawników kodowych, zadajników stanów logicznych i terminali adresowych.

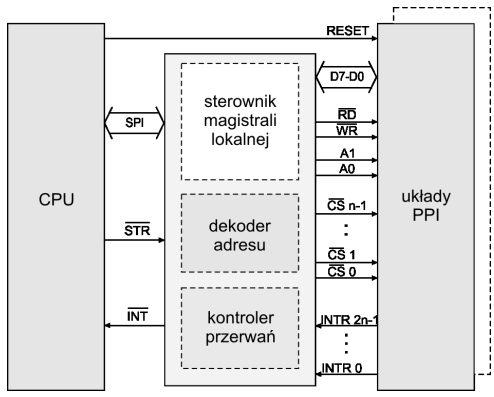
W często wykorzystywanych 8-bitowych mikroprocesorach RISC, takich jak ATmega16A, dostępne są zwykle cztery porty równoległe (32 linie wejścia/wyjścia) [1]. To zbyt mało, aby sprostać wymaganiom komunikacji wielokanałowej. Rozbudowane układy ATmega dysponują znacznie większą liczbą portów, ale i one nie obsługują sprzętowo transmisji równoległej z potwierdzeniem. Mechanizm ten funkcjonuje natomiast w specjalizowanych układach PPI (*Programmable Peripheral Interface*) [2]. Układy PPI zaimplementowano w strukturach PLD [3] i wsparto przez narzędzia symulacyjne [4], ale ich współpraca z mikrokontrolerami wymaga zastosowania dodatkowej logiki sterującej. Dołączenie układów PPI do systemu, z implementacją kontrolerów adresu i przerwań w układzie CPLD, pozwala na zwiększenie liczby portów i zapewnienie obsługi protokołów transmisji z potwierdzeniem przez hardware [5]. Efektywność takiego rozwiązania można znacznie poprawić, integrując również interfejs SPI [6, 7, 8] i sterownik magistrali w układzie programowalnym.

2. Koncepcja organizacji podsystemu PPI

Każdy z układów 82C55A PPI wnosi do systemu trzy 8-bitowe porty PA, PB i PC, zdolne do transferu danych do lub z urządzenia zewnętrznego bez potwierdzenia, albo dwa porty PA i PB, prowadzące transmisję równoległą z potwierdzeniem. Możliwe jest ustawienie różnych kombinacji trybów pracy portów [9]. Magistralowa obsługa układów PPI może jednak wymagać nie tylko zaangażowania zespołu linii procesora, ale i programowego ich sterowania, ponieważ mikrokontrolery najczęściej nie posiadają portów dedykowanych dla komunikacji z zewnętrzną pamięcią i układami wejścia/wyjścia. W tym kontekście zintegrowanie w jednej strukturze programowalnej tak dekodera adresu i kontrolera przerwań [5], jak i sterownika lokalnej magistrali danych podsystemu PPI, jest zdecydowanie korzystnym rozwiązaniem. Powodzenie tej koncepcji zależy jednak od zapewnienia szybkiej transmisji szeregowej między jednostką centralną i programowalnym kontrolerem podsystemu PPI. Warunek ten można spełnić, wykorzystując wbudowany w zasoby mikroprocesorów moduł interfejsu SPI, przy jednoczesnej implementacji również tego modułu w strukturze kontrolera układów transmisji równoległej.

Jednostka centralna systemu komunikuje się wówczas z programowalnym kontrolerem, który nadzoruje układy PPI (rys. 1). Interfejs SPI procesora jest wykorzystywany do przesyłania poleceń dla kontrolera i słów programujących dla układów 82C55A PPI, nadawania i odbioru danych oraz odczytu numerów prze-

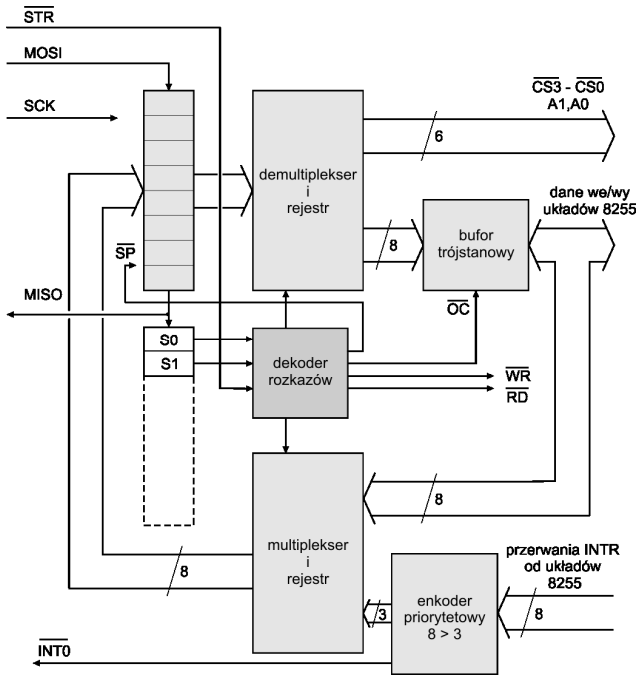
rwań, zgłaszanych przez podsystem. Sygnał /STR synchronizuje funkcjonowanie kontrolera. Kontroler adresuje układy PPI (sygnały /CS) i ich rejestry (linie A1, A0), inicjuje cykle zapisu i odczytu na lokalnej magistrali danych (linie D7-D0, sygnały zapisu /WR i odczytu /RD) oraz przyjmuje przerwania INTR, koduje ich numery i generuje przerwanie zewnętrzne /INT0. Każdy z układów PPI może generować dwa kasowane sprzętowo przerwania INTR (od portów PA i PB). Sygnał RESET wprowadza podsystem w stan początkowy.



Rys. 1. Schemat blokowy podsystemu PPI z implementowanym kontrolerem magistrali, adresu i przerwań
Fig. 1. General block diagram of the PPI subsystem with implemented bus/address/interrupt controller

3. Implementacja sprzętowa kontrolera

Proponowane rozwiązanie kontrolera podsystemu PPI zawiera kilka bloków funkcjonalnych (rys. 2).



Rys. 2. Architektura kontrolera podsystemu PPI
Fig. 2. Architecture of the PPI subsystem controller

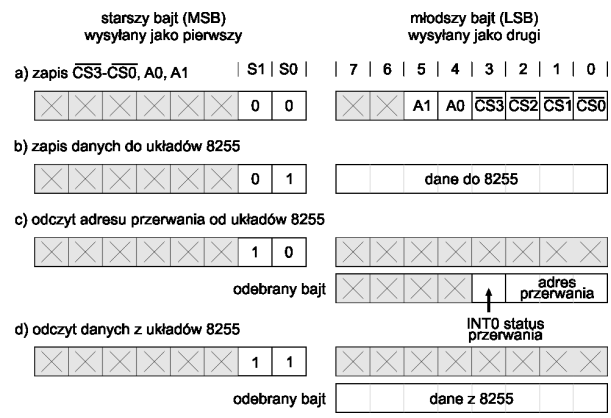
Komunikację z jednostką centralną zapewnia rejestr szeregowo-równoległy, który pełni funkcję nadajnika i odbiornika interfejsu SPI. W trybie transmisji SPI rejestr pracuje szeregowo jako *slave*, a układem *master* jest nadrzędny mikrokontroler. Dane wprowadzane są przez wejście MOSI, od strony najmniej znaczącego bitu, a wyprowadzane przez wyjście MISO. Przesuw danych jest taktowany sygnałem SCK. Do rejestru wprowadzane są dwa bajty. Pierwszy z nich (starszy) zawiera kod rozkazu, a drugi (młodszy) dane. Ponieważ rozkaz kodowany jest na dwóch najmłodszych bitach S1 i S0, długość rejestru wynosi 10 bitów (8 bitów danych i 2 bity kodu rozkazu).

Podczas wpisu równoległego 8-bitowe dane są ładowane do młodszej części rejestru, z której wyprowadzono wyjście MISO. Pozwala to na szeregowy odczyt danych przez przesunięcie jednego, a nie dwóch bajtów w pętli SPI. Wyjście MISO może także służyć do testowania wprowadzanych rozkazów i danych.

Tab. 1. Sygnały sterujące dekodera rozkazów
Tab. 1. Control signals of the instruction decoder

STR	S1	S0	WR	RD	SP	OC
0	0	0	1	1	1	1
0	0	1	0	1	1	0
0	1	0	1	1	0	1
0	1	1	1	0	0	1
1	0	0	1	1	1	1
1	0	1	1	1	1	1
1	1	0	1	1	1	1
1	1	1	1	1	1	1

Dekoder rozkazów jest strobowanym układem kombinacyjnym, generującym sygnały sterujące pracą pozostałych bloków (tab. 1). Bity rozkazowe S1 i S0, zawarte w starszej części rejestru szeregowo-równoległego, determinują działanie dekodera. Stan tych bitów decyduje o drodze przepływu danych w blokach multipleksera i demultipleksera. Sygnał strobujący /STR aktywizuje wyjścia /SP, /WR, /RD oraz /OC. Sygnał /SP określa tryb pracy rejestru szeregowo-równoległego. Wysoki stan logiczny zezwala na przesuw szeregowy rejestru, a stan niski umożliwia wpis równoległy danych. Sygnały /WR i /RD sterują odpowiednio zapisem i odczytem rejestrów w układach 82C55A. Sygnał /OC otwiera bufory trójstanowe, umożliwiając stworzenie lokalnej dwukierunkowej magistrali danych.



Rys. 3. Sekwencje poleceń dla kontrolera podsystemu PPI
Fig. 3. Instruction sequences for the PPI subsystem controller

W pracy kontrolera podsystemu PPI można wyróżnić cztery tryby, określone przez stan bitów S1, S0. W trybie adresowania układu 82C55A (rys. 3a) w młodszym przesłanym bajcie zawarte są bity /CS3-0 (wybór układu) oraz A1, A0 (wybór rejestru). Demultiplekser kieruje je na 6-bitową magistralę adresową.

W trybie zapisu danych (rys. 3b) młodszy bajt, zawierający dane, jest wystawiany przez bufory trójstanowe na dwukierunkową magistralę danych. Sygnał /OC z dekodera rozkazów otwiera bufory, generowany jest też sygnał /WR do układów 82C55A.

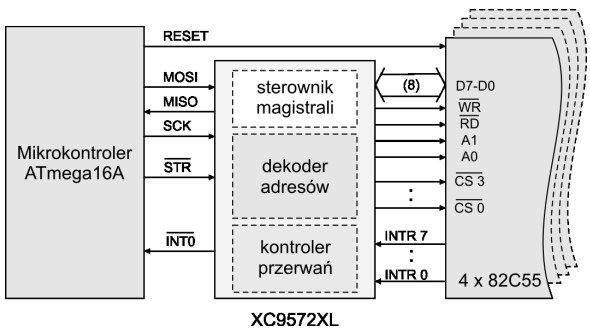
W trybie odczytu adresu przerwania (rys. 3c) przekazywany jest numer portu zgłaszającego przerwanie. Prezentowany kontroler zawiera enkoder priorytetowy, który obsługuje 8 przerwań INTR7-INTR0, zgłaszanych przez układy 82C55A. Blok enkodera sygnalizuje wystąpienie przerwania (aktywny niski poziom wyj-

ścia /INT0) oraz koduje jego numer na 3 bitach, wskazując aktualnie najważniejsze przerwanie (najwyższy priorytet przyjęto dla INTR7, najniższy dla INTR0). Podczas obsługi zgłoszonego przerwania procesor programuje najpierw tryb odczytu adresu przerwania. Zakodowany adres z enkodera kierowany jest przez multiplekser do młodszej części rejestru szeregowo-równoległego. Wygenerowany sygnał /SP, aktywny w stanie niskim, wpisuje dane równolegle do rejestru. Kolejnym krokiem jest szeregowe odczytanie zawartości rejestru przez SPI.

W trybie odczytu danych (rys. 3d) dekoder rozkazów wystawia sygnał /RD, multiplekser kieruje dane do rejestru szeregowo-równoległego, a ujemny impuls /SP wymusza wpis równoległy.

Strukturę kontrolera podsystemu SPI zaimplementowano w układzie reprogramowalnym XC9572XL [10]. Do opisu działania układu i programowania wykorzystano język Verilog oraz środowisko Xilinx ISE 11.

Zaimplementowana wersja kontrolera pozwala na pełną obsługę czterech dołączonych do magistrali lokalnej układów 82C55A, z przerwaniami od wszystkich portów (rys. 4).



Rys. 4. Schemat blokowy podsystemu PPI z kontrolerem zaimplementowanym w układzie XC9572XL
Fig. 4. Block diagram of the PPI subsystem with controller implementation in XC9572XL circuit

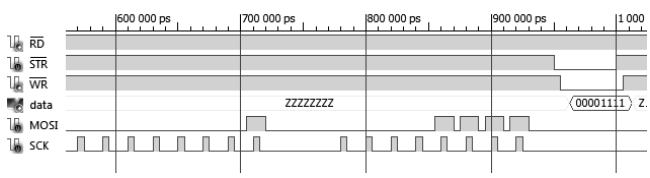
4. Wyniki badań

Przedstawione rozwiązanie zajmuje mniej niż 50% zasobów wewnętrznych, ale angażuje prawie wszystkie wyprowadzenia układu programowalnego XC9572XL (tab. 2).

Tab. 2. Wykorzystanie zasobów układu XC9572XL
Tab. 2. Use of resources of XC9572 device

Macrocells Used	Pterms Used	Registers Used	Pins Used	Function Block Inputs Used
XC9572XL				
35/72 (49%)	115/360 (32%)	32/72 (45%)	29/34 (86%)	61/216 (29%)

Funkcjonowanie kontrolera podsystemu PPI symulowano w środowisku Xilinx ISim 11 dla układu XC9572-5-VQ44 (*Behavioral Simulation* i *Post-Fit Simulation*), a jednostki testowe napisano w języku Verilog. Na rys. 5 przedstawiono przykładowe wyniki symulacji cyklu zapisu danych.



Rys. 5. Przykład symulacji cyklu zapisu danych dla implementacji kontrolera w strukturze XC9572XL
Fig. 5. An example of data write simulation for controller implementation using XC9572XL device

Testy funkcjonalne układów logicznych, zaimplementowanych w strukturze XC9572XL, przeprowadzono z wykorzystaniem nadrzędnego komputera, wyposażonego w interfejs SPI, oraz modułu zadajnika stanów logicznych, który dołączono do wyjść układu XC9572XL od strony podsystemu PPI. Trójstanowy zadajnik emulował funkcjonowanie bufora wyjściowego magistrali danych i sterownika przerwań w układach PPI. Programowa komparacja danych zwracanych linią MISO do komputera z danymi uprzednio wysyłanymi dowiodła poprawności działania rejestru szeregowo-równoległego kontrolera i pozwoliła na śledzenie protokołów transmisji, podejmowanej przez komputer na liniach interfejsu SPI. Pozostałe bloki kontrolera podsystemu PPI testowano dla wszystkich trybów pracy, generując z poziomu komputera polecenia i weryfikując przebieg cykli sterujących podsystem PPI oraz kodowanie wektora przerwań w trybie pracy krokowej. Sprawdzono kolejność obsługi i zagnieżdżanie zgłaszanych przerwań. Przeprowadzone testy potwierdziły prawidłowość implementacji kontrolera.

5. Podsumowanie

Implementacja dekodera adresu, enkodera priorytetowego, sterownika 8-bitowej dwukierunkowej magistrali danych, uproszczonego interfejsu SPI oraz układów logiki sterującej w obrębie jednego układu programowalnego znacznie zmniejsza stopień komplikacji systemów z mikrokontrolerami i układami PPI. Jednostka centralna komunikuje się z kontrolerem podsystemu PPI przez interfejs SPI i odpowiada za obsługę tylko jednej linii przerwania zewnętrznego. Pozwala to na efektywne wykorzystanie podsystemów z układami PPI, przez zwiększenie liczby kanałów transmisji równoległej i jednoczesne odciążenie jednostki centralnej. Opracowana wersja kontrolera może obsługiwać 8 portów prowadzących transmisję równoległą z potwierdzeniem lub aż 12 portów w przypadku transmisji bez potwierdzenia. W trybie transmisji z potwierdzeniem każdy z kanałów dostosowuje się indywidualnie do szybkości urządzenia zewnętrznego.

Warto zauważyć, że koncepcja kontrolera CPLD podsystemu PPI może być przydatna dla współpracy innych układów wejścia/wyjścia z większością standardowych mikrokontrolerów, wyposażonych w interfejs SPI.

6. Literatura

[1] ATmega16A. 8-bit AVR Microcontroller with 16K Bytes In-System Programmable Flash. Atmel Corporation 2009.
[2] A.K. Ray, K.M. Bhurchandi: Advanced Microprocessors and Peripherals. Tata McGraw-Hill, New Delhi 2007.
[3] Microperipheral MegaCore Function. Data Book, Altera 1997.
[4] Pradeep Kumar Jaisal, Anant G. Kulkarni, Srikant B. Burje: Design and Simulation of 8255 Programmable Peripheral Interface Adapter using VHDL. International Journal of Computer Science and Technology, IJCST Vol.2, Issue 1, March 2011.
[5] Arnold K., Michalak S.: Implementacja kontrolera mikroprocesorowych układów transmisji równoległej w strukturach CPLD. Pomiary Automatyka Kontrola, vol.58, nr 7/2012, s.635-637.
[6] Oudjida A. K., Berrandjia M. L., Tiar R., Liacha A., Tahraoui K.: FPGA Implementation of I2C and SPI protocols: A comparative study. Electronics, Circuits and Systems, ICECS 2009, 16th IEEE International Conference, pp.507-510, 2009.
[7] Jamro E., Wielgosz M., Cioch W., Bieniasz S.: Efektywna komunikacja ARM-FPGA z użyciem interfejsu SPI. Pomiary Automatyka Kontrola, vol. 57, nr 8/2011, s.874-876.
[8] Maśnicki R., Hallmann D.: Akwizycja danych z ADC z wykorzystaniem FPGA. Pomiary Automatyka Kontrola, vol. 58, nr 11/2012, s. 912-915.
[9] 82C55A Data Sheet FN2969.10. Intersil 2006.
[10] XC9572XL High Performance CPLD. Product Specification. Xilinx 2007.

otrzymano / received: 21.05.2013
przyjęto do druku / accepted: 03.07.2013 *artykuł recenzowany / revised paper*