Adam MILIK
POLITECHNIKA ŚLĄSKA, INSTYTUT ELEKTRONIKI

Dynamicznie rekonfigurowalny sterownik logiczny – łatwo programowalna architektura

Dr inż. Adam MILIK

Ukończył studia na Wydziale Automatyki, Elektroniki i Informatyki Politechniki Śląskiej. Pracę doktorską obronił w 2003 r. Jest adiunktem w Instytucie Elektroniki Politechniki Śląskiej. Jego zainteresowania naukowe to układy logiki programowalnej, sterowniki programowalne, modelowanie i synteza złożonych układów sprzętowo-programowych.

e-mail: adam.milik@polsl.pl

Streszczenie

Artykuł przedstawia architekturę sterownika bitowego implementowanego w strukturze FPGA umożliwiającego wyeliminowanie złożonego procesu implementacji poprzez wykorzystanie odpowiedniej struktury sprzętowej i narzędzi programowania.

Słowa kluczowe: Sterownik Programowalny, Dynamiczna rekonfiguracja, FPGA, synteza logiczna.

Dynamically reconfigurable logic controller - architecture of improved programmability

Abstract

The paper presents an idea of a Programmable Logic Controller for binary control implemented in an FPGA device with use of custom designed architecture and implementation tools. The solution does not require vendor synthesis and implementation tools except for final bitstream generation. It is an extension of the previously proposed architecture (Figs. 1 and 2). The architecture is based on a hardwired set of connections that is formed inside the FPGA device μLC units. The μLC can be programmed by means of LUT table modification. The architecture is mainly limited by the hardwired connection that bases on an invariant set of multiplexed signals delivered to the μLC. A new architecture is proposed, extending programmability of the architecture to programmable connections which are available in FPGAs (Figs. 3 and 4). The μLC architecture has also been modified and exactly fitted into the regular structure of an FPGA (Fig. 5). The new logic resources supplementing architecture modifications of the controller has been defined. They are input (Fig. 6) and output (Fig. 7) cells. The possible computation capabilities of FPGA devices are gathered in Tab. 1. The research task is in progress. A new solution with extended use of programmable connections, better exploitation of logic resources and easiness of logic synthesis and programming is searched for.

Keywords: PLC, FPGA, dynamic reconfiguration, logic synthesis.

1. Wstęp

Sterowniki przemysłowe PLC (ang. Programmable Logic Controller) stosowane są do realizacji algorytmów sterowania od lat siedemdziesiątych ubiegłego wieku. Rozwiązania tego typu, posiadają ugruntowaną pozycję na rynku i są bardzo chętnie wykorzystywane w wielu różnych dziedzinach. Algorytm sterowania w typowym sterowniku realizowany jest przez układ mikroprogramowalny lub mikroprocesorowy. Taki sposób wykonywania algorytmu sterowania, narzuca szeregowy sposób jego realizacji. W przypadku złożonych algorytmów sterowania, czas obiegu głównej pętli programu może być tak długi, że sterownik nie zdąży zareagować na szybkie zmiany sygnałów wejściowych. Rozwiązaniem tego problemu jest wykorzystanie współbieżnej realizacji programu sterowania, która jest możliwa w sterowniku realizującym algorytm sterowania na drodze sprzętowej [1, 2]. Konstrukcja sterownika sprzętowego, oparta jest o układ programowalny z możliwością przeprogramowania (rekonfiguracji układu) – możliwa jest wtedy wymiana programu sterowania w zależności od potrzeb. Najdogodniejszymi do realizacji sterownika sprzętowego są układy FPGA. Układy tego typu charakteryzują się znacznymi zasobami logicznymi, które umożliwiają sprzętową realizację bardzo złożonych algorytmów sterowania. Aby umożliwić wielokrotną zmianę konfiguracji układu, należy zastosować układ FPGA z ulotną pamięcią konfiguracji (RAM). Rozwiązanie takie pozwala na wielokrotną i szybką rekonfigurację układu. Pewną wadą układów tego typu jest opóźnienie uruchomienia po włączeniu napięcia zasilania, związane z pobraniem konfiguracji (zwykle kilkadziesiąt ms) [7]. Układy bez pobrania konfiguracji nie są zdolne do wykonywania funkcji użytkowych.

Zastosowany układ programowalny jest jedynie bazą, która umożliwia realizację programu sterowania. Ze względu na ograniczenia układu FPGA – głównie ograniczone zasoby logiczne – korzystne jest stworzenie szkieletowej symetrycznej architektury sterownika sprzętowego. Architektura szkieletowa jest dostosowana do architektury układu programowalnego oraz klasy realizowanego algorytmu sterowania. Na elementy architektury składają się bloki umożliwiające realizację typowych układów wykorzystywanych do realizacji programu sterowania, a mianowicie: bloki kombinacyjne, czasomierze, liczniki oraz specjalizowane bloki funkcjonalne (np. regulatory). Konkretne elementy architektury szkieletowej, zdefiniowane zostają w momencie zapisania przez użytkownika programu sterowania.

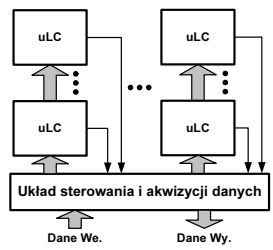
Program sterownia tworzony jest przez użytkownika. Do jego zapisu bardzo wygodnie jest wykorzystać graficzny język diagramów stykowych (LD). Język LD został zdefiniowany formalnie w normie IEC-1131-3 i cieszy się bardzo dużą popularnością wśród projektantów układów automatyki. Stworzony za pomocą języka LD algorytm sterowania, poddany zostaje syntezie logicznej. W wyniku syntezy otrzymujemy wstępny opis układu logicznego realizującego program sterowania [1]. Następnym krokiem implementacji jest umieszczenie otrzymanego opisu logicznego w zdefiniowanej strukturze układu programowalnego.

Implementacja układów logicznych przebiega z wykorzystaniem złożonych programowych narzędzi syntezy logicznej a następnie odwzorowania technologicznego i implementacji układowej. Są to procesy długotrwałe i wymagające wyrafinowanych narzędzi programowych. Czy istnieje możliwość stworzenia własnych narzędzi, pozwalających skutecznie implementować szybkie układy sterowania binarnego?

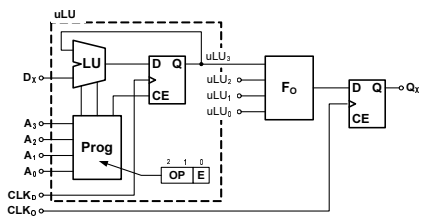
2. Istniejące rozwiązanie

Poszukując rozwiązania eliminującego konieczność prowadzenia procesu implementacji dla zadań sterowania binarnego, zaproponowano strukturę o architekturze symetrycznej złożoną z niewielkich układów logicznych [3]. W rozwiązaniu przyjęto niezmienniczy zestaw połączeń. Realizacja algorytmu przetwarzania oparta jest na równoległo-szeregowym przetwarzaniu informacji z naciskiem na pracę równoległą (rys. 1). Cykl obliczeniowy składa się z 16 kroków, w których jednostki logiczne μLC przetwarzają szeregowo dostarczane informacje, po których następuje cykl przepisania wyników do rejestrów wyjściowych. Bieżący numer cyklu jest dystrybuowany z jednostki sterującej do sterowników poprzez siec połączeń. Opracowana koncepcja opierała się na bloku logicznym, dokładnie wpasowanym w architekturę układu FPGA (rys. 2). W pojedynczym bloku konfigurowalnym (ang. CLB) układów klasy Virtex [7] mieści się dokładnie jedna jednostka logiczna. Składa się on z programowalnego układu logicznego μLU zdolnego do wykonywania 4 predefiniowanych in-

strukcji. Do sterowania układem μLU wykorzystano pamięć PROG złożoną z trzech generatorów tablicowych. Pamięć składa się z 16 komórek opisujących funkcje logiczne realizowane w 16 kolejnych cyklach przetwarzania. Słowo sterujące, dzieli się na dwa fragmenty. Do układu μLU dostarczane są dwa bity (OP) wybierające 1 z 4 możliwych operacji. Bit E kierowany jest do przerzutnika stanu logicznego, umożliwiając wpisanie ostatnio wypracowanego rezultatu. Wynik operacji wyznaczony jest jako dowolna 4 argumentowa funkcja logiczna wyników pochodzących z $\mu LU_{0\text{-}3}$ (blok F_O). Ostatecznie mikrosterownik μLC zajmuje 1.25 bloku CLB układowego. W 5-ciu sąsiadujących w poziomie blokach mieszczą się 4 układy μLC. Przedstawiona struktura, posiada dobre upakowanie i jest zdolna do obliczania wyrażeń logicznych złożonych z maksymalnie 64 zmiennych w 4 blokach równoległych. Programowanie sterownika polega wyłącznie na podmianie zawartości tablicowych generatorów funkcji. Dzięki przedstawionemu podejściu możliwe jest unikniecie procesów syntezy i implementacji.



Rys. 1. Ogólna architektura sterownika rekonfigurowanego
Fig. 1. General architecture of the reconfigurable controller



Rys. 2. Architektura bloku uLC
Fig. 2. The uLC cell architecture

W celu zbadania własności architektury w zakresie realizacji funkcji kombinacyjnych, niech będzie dana następująca funkcja:

$$Q = D_0D_4 + D_1D_5 + D_2D_6 + D_3D_7 \tag{1}$$

Do mikrosterownika przekazywane są zmienne w kolejności od D_0 do D_{15} (linia D_x). Wyrażenia (1) nie można przekształcić do postaci szeregowej, w której w kolejności pojawiania się zmiennych będą wykonywane operacje logiczne. Wyrażenie musi zostać rozłożone na cztery wyrażenia cząstkowe, implementowane w czterech blokach μLU przyłączonych do linii z danymi $D_0 - D_{15}$. Każdy ze sterowników μLC ma sztywno zdefiniowane połączenia. Operację należy zrealizować za pomocą 4 jednostek μLC realizujących następujące funkcje:

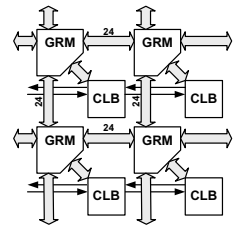
$$\begin{aligned} Q_0 &= D_0D_4 \\ Q_1 &= D_1D_5 \\ Q_2 &= D_2D_6 \\ Q_3 &= D_3D_7 + Q_0 + Q_1 + Q_2 \end{aligned} \tag{2}$$

Prowadzi to do wykorzystania 5 bloków CLB złożonych z 20 generatorów tablicowych LUT w układzie. Przeprowadzone eksperymenty wskazały na znaczne ograniczenie architektury wynikające ze stałego przypisania ciągu zmiennych podawanych do układu przetwarzania, wynikającego ze sztywnej struktury połączeniowej. Uniemożliwia to efektywne wykorzystanie zasobów logicznych mikrosterowników. Możliwość dobierania podzbioru zmiennych przetwarzanych w układach μLU, korzystnie wpłynęła by na efektywność wykorzystania zasobów. Należy postawić pytanie czy można rozszerzyć projektowanie struktury logicznej o możliwość definiowania programowalnych połączeń ?

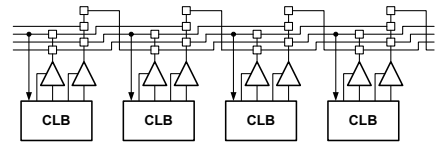
3. Połączenia programowalne FPGA

Obok zasobów logicznych, połączenia programowalne stanowią cechę charakterystyczną układów FPGA. Mając możliwość kształtowania sieci łączącej, można bardziej efektywnie gospodarować programowalnymi zasobami logicznymi, przenosząc część funkcjonalności odpowiadającej za wybór zmiennych do elementów logicznych sterujących siecią połączeń. W ten sposób wykorzystanie zasobów logicznych do przeprowadzania sygnału może zostać znacząco ograniczone. Wykorzystanie programowalnych zasobów łączących wymaga zapoznania się z architekturą układową oraz możliwością ich sterowania. Układy FPGA o architekturze symetrycznej, wykorzystują programowalne matryce połączeń do przenoszenia sygnałów pomiędzy segmentami połączeń biegnącymi ortogonalnie w kanałach poziomych i pionowych. Do badań nad strukturą połączeń wykorzystano układy rodziny Spartan II (o architekturze bazowej opartej o rodzinę Virtex). Informacje o architekturze układu pozyskano na podstawie pliku tekstowego czytelnego dla człowieka, generowanego przez program XDL (ang. Xilinx Design Language) [4, 5, 6]. Warto nadmienić, że w matrycy łączącej GRM przyległej do konfigurowalnego bloku logicznego, można dokonać zestawienia połączeń w oparciu o 1508 programowalnych elementów łączących (na podstawie konfiguracji układu zamieszczonej w pliku XDL). Liczba możliwych do skonfigurowania ścieżek jest znaczna i pozwala na elastyczne prowadzenie połączeń.

Z punktu widzenia sterownika, bliżej zostaną przedstawione połączenia bezpośrednie oraz trójstanowe linie transmisyjne. Połączenia bezpośrednie (rys. 3) to segmenty łączące matryce GRM najbliższych sąsiadów. Składają się one z 24 segmentów. Bardzo korzystną cechą jest możliwość zestawienia połączenia scalającego odpowiednie segmenty. Ułatwia to budowanie magistral dystrybuujących sygnały wejściowe, biegnących pionowo lub poziomo. Duże możliwości łączeniowe stwarzają również trójstanowe poziome linie magistralowe (rys. 4). Każdy z bloków logicznych posiada dwa bufory trójstanowe, pozwalające na wyprowadzenie sygnału na specjalne 4 linie magistralowe biegnące poziomo wzdłuż całej struktury układu. Pozwalają one na bardzo łatwe budowanie konfigurowalnych systemów multipleksowania informacji.



Rys. 3. Zbiór połaczeń bezpośrednich
Fig. 3. Set of direct connections



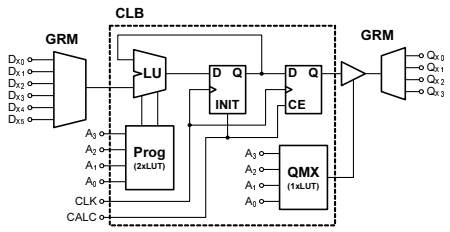
Rys. 4. Magistralowe połączenia trójstanowe
Fig. 4. The tri-state bus lines

4. Zmodyfikowana architektura sterownika

Jak wykazano uprzednio architektura sterownika posiada łatwość programowania, jednakże nie zapewnia efektywnego wykorzystania zasobów logicznych. Przeprojektowanie architektury jednostki µLC powinno zredukować jej rozmiar oraz umożliwić wybór sygnałów wejściowych z możliwie najszerszego zbioru.

Wykorzystując bufory trójstanowe przy ległe do bloku CLB do zbudowania rozproszonego systemu multipleksowania wyników wyjściowych ułatwi dystrybucję sygnału. Powyższe zabiegi poprzez zmniejszenie rozmiaru komórki podstawowej układu, umożliwią pełniejsze wykorzystanie elementów logicznych. Zaproponowane zmiany pozwalają również usunąć scentralizowany układ dystrybucji danych, na rzecz rozproszonego systemu indywidualnie dobieranego dla każdego z bloków funkcjonalnych w oparciu o programowanie matryc GRM. W przedstawionym rozwiązaniu zaproponowano odpowiednio dobrane bloki wejścia i wyjścia, które mogą zostać rozmieszczone wzdłuż lewej i prawej krawędzi układu.

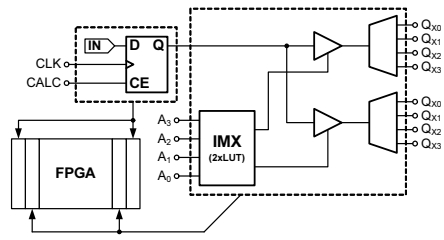
Zmodyfikowany układ µLC został pokazany na rysunku (rys. 5). Architektura układu została starannie wpasowana w pojedynczy blok konfigurowalny układu FPGA. Pozawala to bardo efektywnie spożytkować korzystne cechy symetrycznego układu a w szczególności zasoby połączeniowe. Podobnie jak poprzednio, jednostka logiczna wykonuje dowolnie zdefiniowane 4 instrukcje logiczne. Są one ustalane na drodze programowej w strumieniu konfiguracji. Zrezygnowano ze sterowania sygnałem CE przerzutnika wyniku, co pozwoliło zwolnić jeden z generatorów tablicowych funkcji. Wykorzystano możliwość synchronicznego inicjalizowania przerzutnika do kolejnego cyklu obliczeń poprzez sygnał CALC wprowadzonego na linię inicjalizującą przerzutnik warunku. Na drodze programowej, podczas generacji zbioru konfigurującego wybiera się stan początkowy przerzutnika (0/1). W celu udostępnienia stabilnego stanu logicznego na wyjściu mikrosterownika, nie ulegającego zmianom w czasie procesu obliczeniowego, zastosowano buforowanie poprzez rejestr stanu wyjściowego. Sygnał CALC, którego aktywność oznacza zarówno zakończenie bieżącego cyklu obliczeń i rozpoczęcie kolejnego, przenosi wypracowany wynik do rejestru wyjściowego µLC oraz inicjalizuje przerzutnik wyniku.



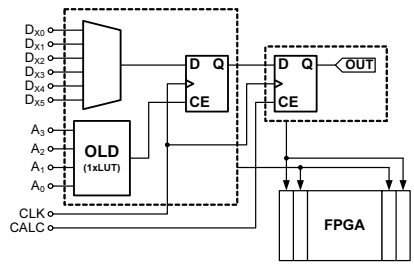
Rys. 5. Zmodyfikowany układ µLC
Fig. 5. Modified µLC structure

Kolejną modyfikacją jest sposób wyboru danych wejściowych. Mimo iż w układzie są dostępne 24 linie połączeń bezpośrednich (rys. 3), pojedyncze wejście generatora tablicowego LUT może zostać dołączone do 6 linii sygnałowych. Przy pracy multipleksowanej jednostki, podejście takie pozwala wprowadzić do pojedynczej kolumny mikrosterowników aż do 96 sygnałów zorganizowanych w 6 grup po 16 sygnałów. Sygnały te pobierane są z poziomych linii trójstanowych wprowadzonych do wybranych wertykalnych linii danych. Wyniki obliczeń wyprowadzane są na linie magistralowe poprzez bufory trójstanowe a następnie dystrybuowane są dalej do wybranych linii wertykalnych. Moment wyprowadzenia danych, kontrolowany jest przez generator tablicowy QMX. W wybranych momentach cyklu obliczeniowego, dana wyjściowa może zostać wyprowadzona. Struktura układowa pozwala na wybranie 4 linii magistralowych.

Obok modyfikacji układów mikrosterownikowych µLC wprowadzono programowalne komórki sygnałów wejściowych i wyjściowych. Zostały one dostosowane do zaproponowanej architektury, zapewniając spójność funkcjonalną układu. Układy wejściowe i wyjściowe są ulokowane w skrajnej prawej lub lewej kolumnie bloków wejścia-wyjścia (IOB) oraz w przylegającej kolumnie bloków logicznych. W każdym wierszu lewej lub prawej krawędzi układu, w fazie projektowej zostały zaplanowane 4 komórki IOB. Ostatecznie zostały zaimplementowane 3 komórki przy czym ograniczona liczba wyprowadzeń w obudowie spowodowała, że dołączone do faktycznych wyprowadzeń są conajwyżej 2 komórki. Na tej podstawie, architekturę dostosowano do 2 komórek pracujących jako wejścia i wyjścia w każdym wierszu układu. W układach wejściowych zrealizowano możliwość multipleksowanego wprowadzania sygnałów na wertykalne linie transmisyjne. Do dyspozycji układu są 4 generatory tablicowe oraz 4 bufory trójstanowe (2–CLB + 2–IOB) dla dwóch sygnałów wejściowych. Wykorzystując dostępne zasoby, każde z wejść może zostać wprowadzone na dowolną z linii wertykalnych



Rys. 6. Programowalna komórka wejściowa
Fig. 6. Programmable input cell



Rys. 7. Programowalna komórka wyjściowa
Fig. 7. Programmable output cell

Model programistyczny zaproponowanego sterownika, zapewnia znaczną elastyczność w sposobie prowadzenia połączeń jak i programowania funkcji. Proces kompilacji i odwzorowania programu sterowania dokonuje minimalizacji i dekompozycji funkcji. Zminimalizowane funkcje pozwalają na wyznaczenie zmiennych, które są wykorzystywane. Następnie określona zostaje kolejność multipleksowania zmiennych.

W tabeli (tab. 1) zestawiono własności obliczeniowe układów sterowania, zbudowanych w oparciu o układy rodziny Spartan II. Wskaźnik liczby wejść-wyjść jest wskaźnikiem potencjalnych możliwości implementacyjnych struktury. Ze względu na ograniczoną liczbę wyprowadzeń w obudowie układu w pewnych wypadkach liczba ta może być nieco mniejsza.

Tab. 1. Zasoby logiczne wykorzystywane przez układ i wybrane elementy
Tab. 1. Logic resources allocated by circuit and selected components

Typ układu	Matryca [wier., kol.]	Liczba wejśc wyjść [4w]	Liczba układów µLC [w(k-2)]
XC2S50	24, 16	96	336
XC2S100	30, 20	120	540
XC2S150	36, 24	144	792
XC2S200	42, 28	168	1092

5. Podsumowanie i dalsze badania

Przedstawiona architektura znacząco poprawia własności sterownika bitowego umożliwiając wyeliminowanie procesów implementacyjnych. Dodatkowo, atutem jest dynamiczna podmiana algorytmu sterowania poprzez częściową rekonfigurację. Planuje się dalszy rozwój architektury poprzez ograniczenie lub wyeliminowanie multipleksowania danych oraz rozszerzenie modyfikacji połączeń w obrębie struktury układowej.

6. Literatura

[1] A. Milik: High Level Synthesis – Reconfigurable Hardware Implementation of Programmable Logic Controller, PDeS 2006 Programable Devices and Embedded Systems, Brno 14-16 February 2006.
[2] M. Chmiel, E. Hrynkiewicz, A. Milik: Remarks on Improving of Operation Speed of The PLCs, 16th IFAC World Congress, Prague, 4-8 July 2005.
[3] E. Hrynkiewicz, A. Milik, J. Mocha: Dynamicznie rekonfigurowana współbieżna realizacja sterowania binarnego, VII Krajowa Konf. Elektroniki, Darłowko Wsch. 2-4 Czerw. 2008.
[4] C. Claus, B. Hang, M. Hubner, C. Schmutzler, J. Becker, W. Stechele: An XDL-based busmacro generator for customizable communications interfaces for dynamically and partially reconfigurable systems, RC Education, Porto Allegre, Brasil, May 2007.
[5] A. Ehliar, D. Liu: Thinking Outside the Flow: Creating Customized Backend Tools for Xilinx Based Designs, FPGA World 2007, Stockholm, Sweden, September 2007.
[6] N. Sedcole PhD Dissertation: Reconfigurable Platform-Based Design in FPGAs for Video Image Processing, University of London, January 2006.
[7] Xilinx: XAPP 151: Virtex Series Configuration Architecture User Guide, v.1.5 09.2000.

Artykuł recenzowany

INFORMACJE

Informacje dla Autorów

Redakcja przyjmuje do publikacji tylko prace oryginalne, nie publikowane wcześniej w innych czasopismach. Redakcja nie zwraca materiałów nie zamówionych oraz zastrzega sobie prawo redagowania i skracania tekstów oraz streszczeń.

Artykuły naukowe publikowane w czasopiśmie PAK są formatowane jednolicie zgodnie z ustaloną formatką zamieszczoną na stronie redakcyjnej www.pak.info.pl. Dlatego artykuły przekazywane redakcji należy przygotowywać w edytorze Microsoft Word 2003 (w formacie DOC) z zachowaniem:

- wielkości czcionek,
- odstępów między wierszami tekstu,
- odstępów przed i po rysunkach, wzorach i tabelach,
- oznaczeń we wzorach, tabelach i na rysunkach zgodnych z oznaczeniami w tekście,
- układu poszczególnych elementów na stronie.

Osobno należy przygotować w pliku w formacie DOC notki biograficzne autorów o objętości nie przekraczającej 450 znaków, zawierające podstawowe dane charakteryzujące działalność naukową, tytuły naukowe i zawodowe, miejsce pracy i zajmowane stanowiska, informacje o uprawianej dziedzinie, adres e-mail oraz aktualne zdjęcie autora o rozmiarze 3,8 x 2,7 cm zapisane w skali odcieni szarości lub dołączone w osobnym pliku (w formacie TIF).

Wszystkie materiały:

- artykuł (w formacie DOC),
- notki biograficzne autorów (w formacie DOC),
- zdjęcia i rysunki (w formacie TIF lub CDR),

prosimy przesyłać w formie plików oraz dodatkowo jako wydruki na białym papierze (lub w formacie PDF) na adres e-mail: wydawnictwo@pak.info.pl lub pocztą zwykłą, na adres:

Redakcja Czasopisma
Pomiary Automatyka Kontrola
Asystent Redaktora Naczelnego
Agnieszka Skórkowska
ul. Akademicka 10, p.21A
44-100 Gliwice

Wszystkie artykuły naukowe są dopuszczane do publikacji w czasopiśmie PAK po otrzymaniu pozytywnej recenzji. Autorzy materiałów nadesłanych do publikacji są odpowiedzialni za przestrzeganie prawa autorskiego. Zarówno treść pracy, jak i wykorzystane w niej ilustracje oraz tabele powinny stanowić dorobek własny Autora lub muszą być opisane zgodnie z zasadami cytowania, z powołaniem się na źródło cytatu.

Przedrukowywanie materiałów lub ich fragmentów wymaga pisemnej zgody redakcji. Redakcja ma prawo do korzystania z utworu, rozporządzania nim i udostępniania dowolną techniką, w tym też elektroniczną oraz ma prawo do rozpowszechniania go dowolnymi kanałami dystrybucyjnymi.