Arkadiusz BUKOWIEC, Alexander BARKALOV
UNIWERSYTET ZIELONOGÓRSKI, INSTYTUT INFORMATYKI I ELEKTRONIKI

Realizacja rejestru wyjściowego w układzie cyfrowym automatu z liniowym przekształceniem mikroinstrukcji

Mgr inż. Arkadiusz BUKOWIEC

Ukończył studia inżynierskie (2001) o specjalności inżynieria komputerowa na Politechnice Zielonogórskiej a następnie studia magisterskie (2004) o tej samej specjalności na Uniwersytecie Zielonogórskim. Od roku 2004 pracuje jako asystent oraz jest studentem studiów doktoranckich na Wydziale Elektrotechniki, Informatyki i Telekomunikacji Uniwersytetu Zielonogórskiego. W roku 2006 odbył jeden semestr studiów doktoranckich na Universidade Nova de Lisboa w ramach projektu Sokrates-Erasmus.

e-mail: a.bukowiec@iie.uz.zgora.pl

Prof. dr hab. inż. Alexander A. BARKALOV

W latach 1976-1996 był pracownikiem dydaktycznym w Instytucie Informatyki Narodowej Politechniki Donieckiej. Współpracował aktywno z Instytutem Cybernetyki im. V.M. Glushkova w Kijowie, gdzie uzyskał tytuł doktora habilitowanego ze specjalnością informatyka. W latach 1996-2003 pracował jako profesor w Instytucie Informatyki Narodowej Politechniki Donieckiej. Od 2004 pracuje jako profesor na Wydziale Elektrotechniki, Informatykii Telekomunikacji Uniwersytetu Zielonogórskiego.

e-mail: a.barkalov@iie.uz.zgora.pl

Streszczenie

W artykule została omówiona budowa oraz implementacja w strukturze FPGA rejestru wyjściowego w układzie cyfrowym skończonego automatu stanów z wyjściami typu Mealy'ego przy zastosowaniu liniowego przekształcenia mikroinstrukcji. Przy zastosowaniu liniowego przekształcenia mikroinstrukcji wszystkie mikrooperacje wchodzące w skład jednej mikroinstrukcji generowane są szeregowo. W sytuacji gdy nie zaburzy to działania całego systemu może zostać zastosowany rejestr wyjściowy zbudowany z przerzutników typu D, jednak w sytuacji kiedy wymagane jest aby wszystkie mikrooperacje wchodzące w skład jednej mikroinstrukcji generowane były równolegle niezbędne jest zastosowanie specjalnej organizacji rejestru wyjściowego. Zaproponowany w artykule rejestr zapamiętuje kolejne mikrooperacje wchodzące w skład jednej mikroinstrukcji a po załadowaniu ostatniej mikrooperacji wystawia na wyjściu całą mikroinstrukcję. Taki stan wyjść utrzymywany jest aż do momentu całkowitego zapisania kolejnej mikroinstrukcji, która pojawi się na wyjściu dopiero po jej całkowitym zapisaniu w rejestrze. W celu identyfikacji końca mikroinstrukcji wprowadzony jest dodatkowy sygnał, który ustawiany jest jednocześnie wraz z ostatnią mikrooperacją wchodzącą w skład danej mikroinstrukcji.

Słowa kluczowe: automat stanów, jednostka sterująca, rejestr.

Implementation of output register of digital circuit of FSM with verticalized microinstructions

Abstract

In this paper, the structure and implementation into FPGA device of output register of digital circuit of finite state machine with Mealy outputs and applied verticalization of microinstructions is described. After verticalization of microinstructions all microoperations from this microinstruction are generated serially. If such manipulation do not affect properly working of whole system there can be applied regular output register be means of D type flip-flops. In the case, when there is required parallel execution of all microoperations there is also required applying of special architecture of output register. The proposed architecture of output register is build up two levels of registers. The register (T type) of first level remember serially generated microoperations from one microinstruction. When whole microinstruction is written into this register then it is stored in the register (D type) of second level. Value of the register of second level is not changed until next microinstruction is fully written. The end of microinstruction is indicated by special additional signal y_0. It is generated parallel with last microoperation from particular microinstruction. This signal is used to store whole microinstruction in the register of second level and to reset the register of first level.

Keywords: FSM, control unit, register.

1. Wstęp

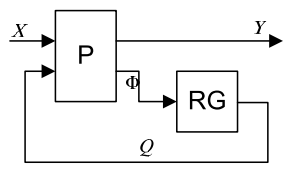
Popularnym sposobem projektowania jednostek sterujących systemami cyfrowymi jak i przemysłowymi jest zastosowanie skończonych automatów stanów z wyjściami typu Mealy'ego [2]. Układy FPGA są bardzo często stosowane do realizacji takiego systemu [9]. Ze względu na specyficzną budowę układów FPGA, podczas implementacji stosowane są różnego rodzaju metody dekompozycji funkcjonalnej [8] lub strukturalnej [1, 4]. W przypadku zastosowania dekompozycji strukturalnej możliwe jest jej połączenie z jednoczesnym liniowym przekształceniem mikroinstrukcji [3, 5, 6]. Zabieg ten powoduje, że mikrooperacje wchodzące w skład jednej mikroinstrukcji generowane są szeregowo. W sytuacji gdy nie zaburzy to działania całego systemu może zostać zastosowany rejestr wyjściowy zbudowany z przerzutników typu D, jednak w sytuacji kiedy wymagane jest aby wszystkie mikrooperacje wchodzące w skład jednej mikroinstrukcji generowane były równolegle niezbędne jest zastosowanie specjalnej organizacji rejestru wyjściowego. Architektura takiego rejestru oraz sposób jego implementacji została zaproponowana w tym artykule.

2. Podstawowe definicje

Definicja: Skończonym automatem stanów z wyjściami typu Mealy'ego nazywamy szóstkę $S = \langle X, Y, A, a_1, \delta, \lambda \rangle$ [2, 8], gdzie:

- X jest skończonym zbiorem zmiennych wejściowych ($X = \{x_1, \ldots, x_L\}$),
- Y jest skończonym zbiorem zmiennych wyjściowych ($Y = \{y_1, \ldots, y_N\}$),
- A jest skończonym zbiorem stanów wewnętrznych automatu ($A = \{a_1, \ldots, a_M\}$),
- a_1 jest początkowym stanem automatu, $a_1 \in A$,
- δ jest funkcją przejść automatu ($A = \delta(A, X)$),
- λ jest funkcją wyjść automatu ($Y = \lambda(A, X)$).

Tak zdefiniowany automat może zostać zrealizowany w strukturach programowalnych z wykorzystaniem jednopoziomowej struktury (rys. 1) [1, 2],



Rys. 1. Schemat blokowy jednopoziomowego układu cyfrowego automatu
Fig. 1. The structural diagram of the single-level circuit of automata

gdzie: skończony zbiór zmiennych $Q = \{q_1, \ldots, q_R\}$ ($R = \lceil \log_2 M \rceil$) reprezentuje kod $K(a_m)$ aktualnego stanu automatu $a_m \in A$; skończony zbiór zmiennych $\Phi = \{D_1, \ldots, D_R\}$ realizuje funkcje wzbudzeń rejestru RG, który stanowi pamięć automatu i jest

zbudowany z R przerzutników typu D. W sytuacji takiej kombinacyjny układ P realizuje funkcje wyjść automatu:

$$Y = \lambda(Q, X), \tag{1}$$

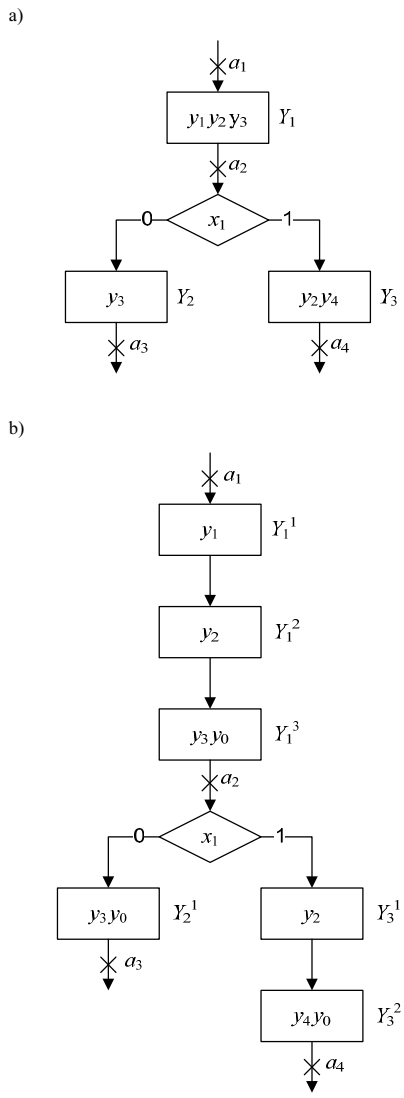
oraz funkcję wzbudzeń przerzutników (przejść automatu):

$$\Phi = \delta(Q, X). \tag{2}$$

Wadą tej struktury jest relatywnie duża liczba funkcji realizowanych przez układ P. Prowadzi to do stosowania dekompozycji [1, 4, 8] układu cyfrowego automatu w celu jego optymalizacji. Jedną z grup metod, stanowią metody oparte o dekompozycję strukturalną i liniowe przekształcenie mikroinstrukcji [3, 5, 6].

3. Liniowe przekształcenie mikroinstrukcji

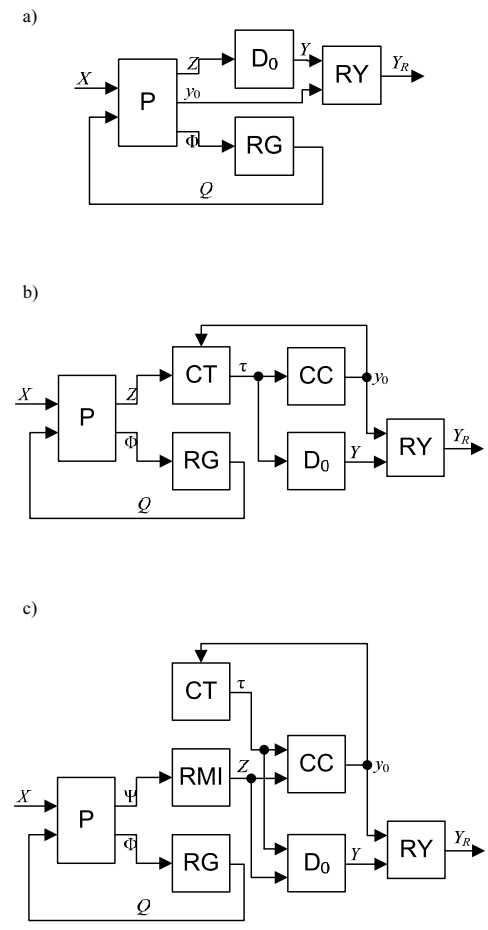
Niech podczas przejścia $\langle a_m, a_s \rangle$ realizowana jest mikroinstrukcja $Y_t \subseteq Y$ ($t = 1,\ldots,T$, gdzie T jest liczbą różnych mikroinstrukcji), która zawiera $N_t = |Y_t|$ mikrooperacji.



Rys. 2. Fragment sieci działań przed (a) i po (b) liniowym przekształceniu mikroinstrukcji
Fig. 2. The subgraph of a flow-chart before (a) and after (b) verticalization of microinstructions

Przekształćmy zatem, każdą taką mikroinstrukcję w sekwencję mikroinstrukcji $\beta_t = \langle Y_t^1, \ldots, Y_t^{N_t} \rangle$, w której każda mikroinstrukcja Y_t^η ($\eta = 1,\ldots,N_t$) zawiera tylko jedną unikalną mikrooperację $y_n = \Pr_\eta Y_t$. Dodatkowo w celu identyfikacji końca sekwencji w każdym ostatnim elemencie $Y_t^{N_k}$ dla każdej sekwencji β_t dodajemy specjalny sygnał y_0. Algorytm liniowego przekształcenia mikroinstrukcji można zastosować zarówno do sieci działań (rys. 2) jak i tablicy przejść-wyjść automatu [7].

Tak przekształcony algorytm sterowania może zostać zaimplementowany w jednej z trzech dwupoziomowych struktur (rys. 3). Pierwsza, ze struktur (rys. 3a)) [3] wymaga zdefiniowania dodatkowych stanów pomiędzy nowo powstałymi mikroinstrukcjami. Układ kombinacyjny P realizuje funkcje wzbudzeń rejestru oraz funkcje kodowania mikroinstrukcji, które następnie są dekodowane w układzie D_0. Druga (rys. 3b)) [5] i trzecia (rys. 3c)) [6] struktura nie wymagają definiowania dodatkowych stanów a kody kolejnych mikrooperacji generowane są przez licznik CT.



Rys. 3. Schematy blokowe dwupoziomowego układu cyfrowego automat
Fig. 3. Structural diagrams of double-level circuits of automata

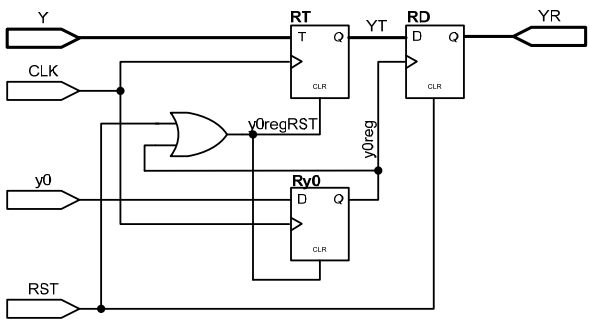
W przypadku drugim, układ kombinacyjny P generuje kod mikrooperacji od którego ma nastąpić zliczanie natomiast w przypadku trzecim generuje kod mikroinstrukcji a kody mikrooperacji zliczane są zawsze od zera. W celu zapewnienia stabilnego działania układu cyfrowego automatu z wyjściami typu Mealy'ego wymagane jest zastosowanie rejestru wyjściowego RY [2, 4, 9]. Zastosowanie liniowego przekształcenia mikroinstrukcji powoduje, że mikrooperacje wchodzące w skład jednej mikroinstrukcji generowane są szeregowo.

W sytuacji gdy nie zaburzy to działania całego systemu może zostać zastosowany normalny rejestr wyjściowy, taki jak dla automatów bez takiego przekształcenia, zbudowany z przerzutników typu D. Jednak w sytuacji kiedy wymagane jest aby wszystkie mikrooperacje wchodzące w skład jednej mikroinstrukcji generowane były równolegle niezbędne jest zastosowanie specjalnej organizacji rejestru wyjściowego.

4. Architektura rejestru wyjściowego

W tym rozdziale zaproponowana zostanie architektura rejestru wyjściowego dla cyfrowego układu automatu implementującego algorytm sterowania z zastosowanie liniowego przekształcenia mikroinstrukcji. Zadaniem takiego rejestru jest szeregowe wczytanie wszystkich mikrooperacji wchodzących w skład jednej mikroinstrukcji i jednoczesne, równoległe, wystawienie ich na wyjściu układu. Stan wyjść nie może natomiast ulec zmianie aż do całkowitego wczytania następnej mikroinstrukcji. Koniec mikroinstrukcji sygnalizowany jest przez dodatkowy sygnał y_0, który generowany jest równolegle z ostatnią mikrooperacją wchodząca w skład danej mikroinstrukcji.

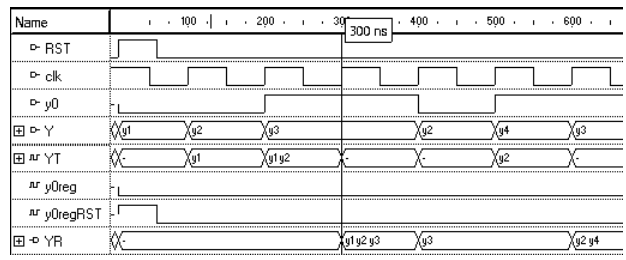
W celu zapewnienia takiego funkcjonowania układu zastosowano architekturę zbudowaną z dwóch *N*-bitowych rejestrów połączonych szeregowo (rys. 4). Pierwszy *N*-bitowy rejestr RT, zbudowany z przerzutników typu T, w kolejnych cyklach zegara zapamiętuje następne mikrooperacje z wczytywanej mikroinstrukcji. Wraz z ostatnią mikrooperacją na wejściu rejestru pojawia się sygnał y_0. Jest on zapisywany w przerzutniku Ry0 (typu D) w celu wygenerowania opóźnienia, które jest niezbędne do zapamiętania ostatniej mikrooperacji. Drugi *N*-bitowy rejestr RD, zbudowany z przerzutników typu D, zatrzaskuje cała mikroinstrukcję.



Rys. 4. Schemat blokowy rejestru wyjściowego
Fig. 4. The structural diagram of the output register

Nie jest on synchronizowany sygnałem zegarowym *CLK* całego układu, a rejestrowanym sygnałem *$y_0$reg*. Opóźnienie jakie generuje przerzutnik Ry0 jest wystarczające aby rejestr RT zapamiętał ostatnią mikrooperację a co za tym idzie rejestr RD zapamiętał cała mikroinstrukcję. Ponieważ w następnym cyklu zegarowym na wejściu układu pojawia się już pierwsza mikrooperacja z kolejnej mikroinstrukcji, wymagane jest asynchroniczne wyzerowania rejestru RT. Do tego celu służy również sygnał *y0reg* (sumowany logicznie z zewnętrznym sygnałem zerującym *RST*). Opóźnienie bramki OR powoduję, że rejestr RT nie zostanie wyzerowany za wcześnie. Ten sam sygnał służy do zerowania przerzutnika Ry0. Powoduje to, że sygnał *y0reg* przyjmuje wartość „1” tylko przez bardzo krótką chwilę (suma czasów propagacji przerzutnika Ry0 i bramki OR) co umożliwia generowanie zbocza narastającego na tym sygnale co cykl sygnału zegarowego *CLK* całego układu. Sytuacja tak jest wymagana w przypadku gdy mikroinstrukcja zawiera tylko jedną mikrooperację.

Tak zaprojektowany rejestr wyjściowy opisany został w języku VHDL i poddany symulacji (rys. 5).

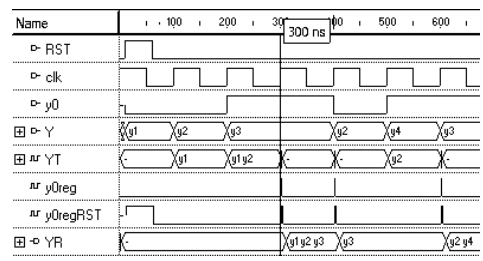


Rys. 5. Symulacja rejestru wyjściowego
Fig. 5. The simulation of the output register

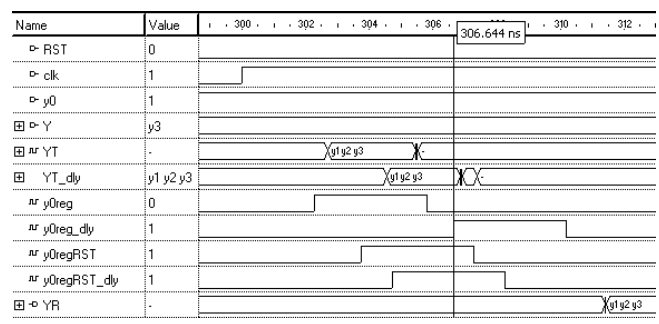
Ponieważ opis behawioralny w języku VHDL jest opisem idealnym (nie zawiera opóźnień czasowych) w symulacji nie widać krótkotrwałych wartości „1” na sygnałach *y0reg* i *y0regRST*, jednak można zauważyć, że układ działa poprawnie. Np. w 300 ns symulacji następuje jednoczesne zapamiętanie zmiennej y_3 w rejestrze RT, zapamiętanie całej mikroinstrukcji $Y_1 = \{y_1, y_2, y_3\}$ w rejestrze RD i wyzerowanie rejestrów RT i Ry0 – przez co nie widać na magistrali *YT* momentu zapisania sygnału y_3 oraz chwilowych wartości „1” na sygnałach *y0reg* i *y0regRST*.

Sytuację tą mogło by zilustrować dodanie opóźnień czasowych w kodzie opisującym układ rejestru, jednak nie zwiększyło by to wiarygodności symulacji. Dlatego układ poddano syntezie i implementacji do układów FPGA oraz następnie symulacji czasowej (rys. 6). W celu zwiększenia wiarygodności tego kroku wybrano dwa układy różnych producentów: *Xilinx Virtex v50* i *Altera Stratix ep1s10*.

a)



b)



Rys. 6. Symulacja czasowa rejestru wyjściowego (a) i zbliżenie synchronizacji sygnałem y_0 (b)
Fig. 6. The timing simulation of the output register (a) and zoom into synchronization of the signal y_0 (b)

Na przedstawionej symulacji (rys. 6a)) można zauważyć, że pojawiają się chwilowe wartości „1” na sygnałach *y0reg* i *y0regRST* oraz krótkotrwałe zapamiętanie ostatniej mikrooperacje (rys. 6b)) przez rejestr RT. Ponadto tak zaprojektowany rejestr zaimplementowano (wraz z przykładowym automatem) do trzech układów FPGA *Xilinx Spartan 2E xc2s200e*, *Xilinx Spartan 3E xc3s500e* i *Altera Stratix II ep2s60* (wchodzących w skład płytek prototypowych *Digilab 2E*, *Spartan-3E Starter Kit* i *DSP Development Kit – Stratix II*

edition). We wszystkich przypadkach mikroinstrukcje generowane były poprawnie.

5. Podsumowanie

W artykule zaproponowano architekturę rejestru wyjściowego w układzie cyfrowym automatu z wyjściami typu Mealy'ego z zastosowanym liniowym przekształceniem mikroinstrukcji. Zastosowanie zaproponowanego rejestru umożliwia równoległe generowania wszystkich mikrooperacji wchodzących w skład jednej mikroinstrukcji, tak jak w układach w których nie zastosowano liniowego przekształcenia mikroinstrukcji. Umożliwia to więc zastosowanie automatu z liniowym przekształceniem mikroinstrukcji w każdej sytuacji, bez obaw, że szeregowe generowanie mikrooperacji wpłynie niekorzystnie na sterowany obiekt.

W zaproponowany rejestrze zastosowano niekonwencjonalną metodę asynchronicznego sterowania przerzutnikami wchodzącymi w jego skład. Metoda ta wykorzystuje czasy propagacji poszczególnych elementów logicznych wchodzących w skład rejestru. Weryfikacja układu z wykorzystaniem analizy czasowej i prototypu wykazała, że układ taki działa poprawnie.

6. Literatura

[1] Adamski M., Barkalov A.: Architectural and Sequential Synthesis of Digital Devices, University of Zielona Góra Press, Zielona Góra, 2006.
[2] Baranov S.: Logic Synthesis for Control Automata, Kluwer, 1994
[3] Barkalov A., Bukowiec A.: Synteza automatów skończonych z wyjściami typu Mealy'ego z zastosowaniem liniowego przekształcenia sieci działań, Materiały VIII konferencji RUC'05, Szczecin, 2005, s. 9-16
[4] Barkalov A., Węgrzyn M.: Design of Control Units with Programmable Logic, University of Zielona Góra Press, Zielona Góra, 2006.
[5] Bukowiec A.: Synteza skończonych automatów Mealy'ego z liniowym przekształceniem sieci działań i adresowaniem mikrooperacji, Pomiary Automatyka Kontrola, 2007, nr 5, s. 27-29.
[6] Bukowiec A., Barkalov A.: Synteza automatów stanów typu Mealy-'ego z liniowym przekształceniem sieci działań i adresowaniem mikroinstrukcji, Pomiary Automatyka Kontrola, 2007, nr 7, s. 115-117.
[7] Bukowiec A., Barkalov A.: Verticalization of Direct Structural Table in Synthesis of Mealy FSMs for FPGAs, Proceedings of the 13th Conference MIXDES'05, Gdynia, 2006, s. 407-411
[8] Łuba T. (ed.): Synteza układów cyfrowych, WKŁ, Warszawa 2003.
[9] Salcic Z.: VHDL and FPLDs in Digital Systems Design, Prototyping and Customization, Kluwer, 1998.

Artykuł recenzowany

INFORMACJE

Studia Podyplomowe

Wydział Elektryczny Politechniki Śląskiej w Gliwicach, Instytut Metrologii, Elektroniki i Automatyki
ogłasza nabór na Dwusemestralne Zaoczne Studia Podyplomowe

Systemy Pomiarowe i Sterowniki Programowalne (SPSP)

Cel Studiów

Celem studiów jest przekazanie wiedzy teoretycznej i umiejętności praktycznych w zakresie: projektowania, wdrażania i utrzymania ruchu systemów automatyki, programowania sterowników PLC oraz systemów nadrzędnych (SCADA), projektowania, programowania i eksploatacji automatycznych systemów pomiarowych w laboratoriach badawczych i przemysło-wych, metod opracowania danych w systemach zapewnienia jakości procesów przemysłowych.

Profil uczestnika studiów

Studia przeznaczone są dla pracowników o różnych specjalnościach, z wyższym wykształceniem o kierunku elektrycznym, elektronicznym, informatycznym lub pokrewnym, zajmujących się organizacją pomiarów w laboratoriach badawczych i przemysłowych lub eksploatacją oraz modernizacją systemów starowania. Ich ukończenie pozwoli uczestnikom na podwyższenie kwalifikacji niezbędnych do sprawnego opracowywania i wdrażania nowoczesnych systemów. Absolwent Studiów otrzymuje Świadectwo Ukończenia Studiów Podyplomowych w zakresie objętym nazwą studiów.

Studia prowadzone są na Wydziale Elektrycznym Politechniki Śląskiej w Gliwicach, w systemie zaocznym w każdą sobotę lub w co drugi weekend (do wyboru) przez dwa semestry. Zajęcia prowadzone są przez nauczycieli akademickich ze stopniem co najmniej doktora oraz przez zaproszonych Gości o uznanym dorobku i autorytecie. Studia obejmują 200 godzin dydaktycznych. Rozpoczęcie Studiów nastąpi po skompletowaniu odpowiedniej liczby kandydatów na dany rodzaj studiów.

Organizator studiów:

Instytut Metrologii, Elektroniki i Automatyki Politechniki Śląskiej, 44-100 Gliwice, ul. Akademicka 10, tel. 032 237 12 41, fax: 032 237 20 34, e-mail: re2@polsl.pl lub agnieszka.skorkowska@polsl.pl, http://imeia.elektr.polsl.pl

Kierownik studiów:

Prof. dr hab. inż. Tadeusz SKUBIS