Maciej PETKO
AKADEMIA GÓRNICZO-HUTNICZA, WYDZIAŁ INŻYNIERII MECHANICZNEJ I ROBOTYKI, KATEDRA ROBOTYKI I MECHATRONIKI

Sterowanie neuronowe robotem równoległym – projekt i implementacja

Dr inż. Maciej PETKO

Adiunkt w Katedrze Robotyki i Mechatroniki AGH. Jego zainteresowania skupiają się na mechatronice, robotyce, zagadnieniach prototypowania i implementacji algorytmów przetwarzania sygnałów, głównie w sterowaniu i diagnostyce technicznej.

e-mail: petko@agh.edu.pl

Streszczenie

W artykule przedstawiono projekt i implementację sterownika neuronowego równoległego robota o trzech stopniach swobody, przeznaczonego do frezowania. Sterownik jest oparty na neuronowym modelu odwrotnej dynamiki manipulatora uczonego na danych zebranych przy zastosowaniu stabilizującego sterownika wykorzystującego strukturalny model analityczny manipulatora. Po zrealizowaniu wirtualnego i szybkiego prototypowania sterownik został zaimplementowany w układzie FPGA z wprogramowanym mikroprocesorem. Współbieżna implementacja sprzętowo-programowa umożliwiła badanie możliwych realizacji algorytmu.

Neural control of a parallel robot – design and implementation

Abstract

The paper presents design and implementation of neural controller for 3-DOF parallel robot for milling. The controller is based on neural model of the inverse dynamics of the manipulator, trained on data collected with the use of a computed torque stabilizing controller. After successful virtual and fast prototyping, the controller was implemented in a FPGA with a soft-processor. Hardware-Software Co-design allowed for exploration of possible control algorithm realisations.

1. Wstęp

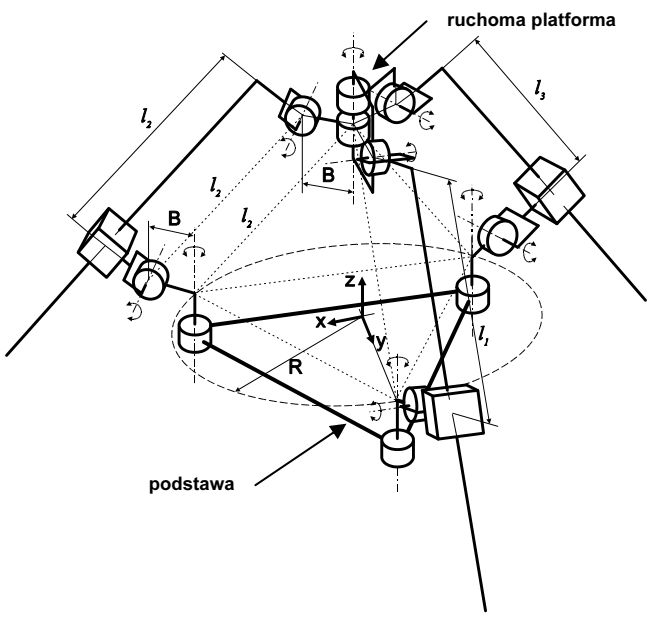
Opracowywanie nowych robotów równoległych jest klasycznym problemem projektowania mechatronicznego [1], przeprowadzanego ze zintegrowanym podejściem interdyscyplinarnym, ponieważ zamknięte łańcuchy kinematyczne komplikują syntezę sterowania i równań kinematyki [2, 3]. Dodatkowo napędy często pełnią również rolę elementów konstrukcji manipulatora. Z powyższych przyczyn, podczas projektowania manipulatorów równoległych, wszystkie aspekty, o różnej naturze fizycznej, razem z zagadnieniami sterowania i jego implementacji, muszą być brane pod uwagę i traktowane równorzędnie od samego początku, poprzez wszystkie fazy procesu projektowania, co jest istotą podejścia mechatronicznego [4, 5].

Coraz częściej algorytmy sterowania są implementowane w układach FPGA (ang. Field Programmable Gate Array), ponieważ pozwalają na optymalizację architektury sprzętowej pod kątem prędkości obliczeń [6] i elastyczności w spełnianiu wymagań różnych metod sterowania oraz wymogów funkcjonalnych poszczególnych użytkowników produktu bez potrzeby modyfikacji obwodów drukowanych. Jednakże istniejące podejścia do projektowania układów FPGA nie są wystarczająco efektywne w stosunku do wymagań implementacji złożonych algorytmów sterowania specyfikowanych na poziomie systemowym [7].

2. Konstrukcja manipulatora

Konstrukcja manipulatora wraz ze wstępnym sterownikiem stabilizującym została zaprojektowana zgodnie z procedurą opisaną w [8, 9]. Manipulator ten (rys. 1), o trzech translacyjnych stopniach swobody, przeznaczony jest do pracy jako konstrukcja wsporcza frezarki wysokoobrotowej. Jego strukturę kinematyczną pokazuje rys. 2. l_i oznacza długości napędzanych członów pryzmatycznych, a R promień okręgu opisanego na trójkącie podstawy. Kartezjański układ współrzędnych (x, y, z) jest umieszczony w ten sposób, że oś y jest skierowana w kierunku ramienia nr 1, tak jak pokazuje rys. 2.

Rys. 1. Opracowany manipulator równoległy do frezowania o trzech stopniach swobody



Rys. 2. Struktura kinematyczna manipulatora

Opracowany manipulator posiada konstrukcję będącą kompromisem pomiędzy dwoma dominującymi tendencjami dla kinematyk równoległych: opartych o robota delta i platformę Stewarda-Gougha, i składa się z nieruchomej podstawy o kształcie trójkąta równoramiennego i ruchomej platformy, do której mocowane jest narzędzie. Platforma połączona jest z podstawą za pomocą trzech ramion o strukturze RRPRR (dwa bierne złącza obrotowe złącze pryzmatyczne napędzane liniowymi elektrycznymi silnikami bezpośrednimi, i dwa złącza obrotowe). Platformę stanowi specjalny przegub obrotowy, zapewniający prostopadłość freza do przedmiotu obrabianego, wyposażony w standardowe gniazdo do mocowania elektrowrzeciona. Przestrzeń robocza ma kształt cylindra o średnicy 600 mm i wysokości 300 mm. Maksymalna osiągana prędkość w jej wnętrzu wynosi ok. 2 m/s, a przyspieszenie 2 g, z elektrowrzecionem HFK 95 S 40 firmy IBAG. Osiągana średnia siła frezowania wynosi ponad 100 N, co jest wartością wystarczającą dla szybkoobrotowego frezowania aluminium.

3. Sterowanie

Dominującym źródłem nieliniowości manipulatora jest jego geometria, powodująca duże wahania, w obrębie przestrzeni roboczej, zredukowanych ("widzianych" przez napędy) mas i sił grawitacyjnych. Zaniedbując pozostałe nieliniowości, równania dynamiki manipulatora przyjmują postać:

$$\mathbf{M}(\mathbf{x})\ddot{\mathbf{l}} + \mathbf{G}(\mathbf{x}) = \mathbf{F}\,, \tag{1}$$

gdzie **M** jest macierzą mas zredukowanych, **G** wektorem zredukowanych sił grawitacyjnych (ciężkości), **F** wektorem sił wywieranych przez napędy, $\mathbf{x} = [x, y, z]^{\mathrm{T}}$ jest wektorem współrzędnych narzędzia w układzie kartezjańskim (x, y, z), a $\mathbf{l} = [l_1, l_1, l_1]^{\mathrm{T}}$ wektorem współrzędnych złączowych narzędzia. Dla obliczenia równania (1) konieczna jest znajomość położenia narzędzia we współrzędnych kartezjańskich, które można wyznaczyć z rozwiązania problemu kinematyki prostej. Dla opracowanego manipulatora istnieje analityczna forma tego rozwiązania. Bardziej szczegółowe wyprowadzenie równań kinematyki i dynamiki manipulatora można znaleźć w [8, 9].

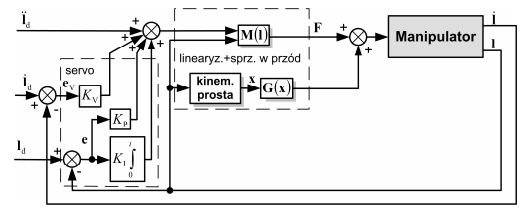
Badania symulacyjne modelu ujawniły znaczące rozbieżności między dokładnym modelem wielomasowym, utworzonym w programie VisualNastran na podstawie rysunków konstrukcyjnych, a modelem analitycznym (1). Można to wyjaśnić zbyt uproszczonym rozkładem masy w tym ostatnim. Dokładny model strukturalny musiały być dużo bardziej skomplikowany z powodu złożonej geometrii ramion i konieczności wzięcia pod uwagę pozostałych nieliniowości, takich jak tarcie, sprężystości, czy nieczułości. Taka rozbudowa uczyniłaby model praktycznie bezużytecznym dla sterowania z powodu zbyt dużych wymagań obliczeniowych dla działania w czasie rzeczywistym. Dlatego do zaprojektowania dokładniejszej wersji sterownika wykorzystany został model typu „czarna skrzynka", opary na sieci neuronowej. Taki model wymaga eksperymentalnej identyfikacji parametrów, a to, z kolei, wymaga zaprojektowania wstępnego sterownika, który zapewniłby stabilność systemu w trakcie przeprowadzania eksperymentu identyfikacyjnego, ponieważ ze względu na silnie nieliniową dynamikę, wynikającą z postaci macierzy **G** i **M**, eksperymenty w otwartej pętli sterowania byłyby niebezpieczne albo wręcz niemożliwe do przeprowadzenia. Taki wstępny sterownik, otrzymany na drodze teoretycznej, powinien zapewnić stabilność systemu i bezpieczne przeprowadzenie eksperymentu.

3.1. Akwizycja danych do zbioru uczącego

Do sterowania położeniem manipulatora **l**, w celu śledzenia zadanej trajektorii $\mathbf{l}_{\mathrm{d}}$, zostało zaproponowane następujące prawo sterowania z linearyzacją w pętli sprzężenia zwrotnego [10]:

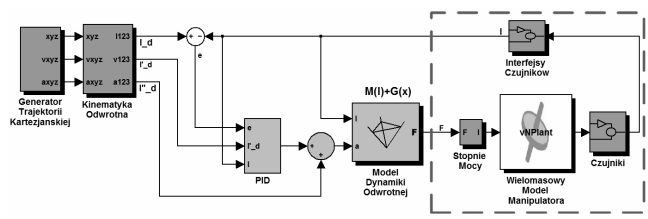
$$\mathbf{F}(t) = \mathbf{M}(\mathbf{l}(t))\Big(\ddot{\mathbf{l}}_{\mathrm{d}}(t) + K_{\mathrm{P}}\mathbf{e}(t) + K_{\mathrm{V}}\mathbf{e}_{\mathrm{V}}(t) + K_{\mathrm{I}}\int_0^t \mathbf{e}(\tau)\,\mathrm{d}\tau\Big) + \mathbf{G}(\mathbf{x})\,, \tag{2}$$

gdzie: $\mathbf{e} = \mathbf{l}_{\mathrm{d}} - \mathbf{l}, \mathbf{e}_{\mathrm{V}} = \dot{\mathbf{l}}_{\mathrm{d}} - \dot{\mathbf{l}}$, a K_{P}, K_{V}, K_{I} są stałymi. Strukturę sterownika pokazuje rys. 3. Część linearyzująca (**M**(**l**) + **G**(**x**)) zawiera wyraz związany z grawitacją, wymagający rozwiązania zagadnienia kinematyki prostej w każdym kroku czasowym.



Rys. 3. Struktura wstępnego sterownika stabilizującego, opartego na analitycznym modelu strukturalnym

W celu nauczenia sieci neuronowej (NN) i oceny jej działania została zebrana pewna ilość danych poprzez wirtualne prototypowanie robota z użyciem sterownika w postaci (2) i bardzo dokładnego modelu wielomasowego manipulatora (rys. 4). Dane składały się z zarejestrowanych wymuszeń (sygnały sterujące – siły) i będących ich rezultatem położeń narzędzia. Wykorzystana trajektoria była trójwymiarową krzywą Lissajoux, ponieważ jest ona zamknięta i gładka, tak we współrzędnych Kartezjańskich, jak i złączowych, ze zmiennymi prędkościami i przyspieszeniami, efektywnie pokrywając znaczną część przestrzeni roboczej. Wszystkie sygnały były próbkowane z częstotliwością 1000 kHz i zarejestrowano 10000 próbek czasowych.



Rys. 4. Schemat Simulinka użyty do zbierania danych do uczenia sieci neuronowej; linią przerywaną obwiedziona jest ta część systemu, która została zamodelowana przez sieć neuronową

3.2. Wybór architektury sieci neuronowej

Z dostępnych danych, po normalizacji, utworzono trzy zbiory: uczący, do przerywania uczenia i testowy. Pierwszy, składający się z 8000 próbek, był używany do obliczania gradientu i modyfikacji wag sieci podczas uczenia. Drugi, składający się z 1000 próbek, służył do wczesnego przerywania uczenia, dla uniknięcia przeuczenia sieci i poprawy możliwości uogólniania sterownika [11]. Pozostała część danych (1000 próbek) nie była używana podczas uczenia, ale do oceny jakości działania sieci i porównywania różnych jej architektur.

Do zaprojektowania sterownika neuronowego została wybrana metoda bezpośredniego modelu odwrotnego [11, 12] ze względu na jej charakterystyczną cechę, polegającą na tym, że sterownik jest otrzymywany bezpośrednio z danych uczących, bez wymogu dysponowania modelem systemu sterowanego. Właściwie, w fazie uczenia zostało zastosowane uczenie wsadowe dla otrzymania neuronowego modelu odwrotnej dynamiki obiektu. Otrzymana sieć była później używana do generowania sygnału sterującego.

W celu uproszczenia fazy implementacji, narzucono na wstępie pewne ograniczenia na rozpatrywane architektury sieci neuronowej. Wybór był dokonywany spomiędzy sieci dwuwarstwowych z neuronami sigmoidalnymi w warstwie ukrytej i liniowymi w warstwie wyjściowej. Zadanie wymagało zastosowania sieci dynamicznej, dlatego przetestowano kilka możliwości: sieci z opóźnionymi wejściami, ze sprzężeniem z opóźnionych wyjść na wejścia, a także inne rodzaje rekurencji. Sygnały wejściowe zostały wybrane na podstawie analizy korelacji. Wszystkie testowane sieci były uczone i symulowane w Matlabie metodą Levenberga-Marquardta z wczesnym przerywaniem procesu uczenia w oparciu o drugi zbiór ciągów par wejście-wyjście pochodzących z symulacji. Ostatecznie zaakceptowany model neuronowy ma strukturę typu NNARX [12], z regresorem φ:

$$\varphi(t) = \left[\mathbf{x}(t-2), \mathbf{x}(t-1), \mathbf{x}(t), \mathbf{F}(t-2), \mathbf{F}(t-1)\right], \tag{3}$$

gdzie t oznacza chwilę czasu, $\mathbf{x}$ wejściem, a $\mathbf{F}$ wyjściem sieci. W sumie sieć posiada 15 wejść, trzy neurony w warstwie ukrytej i trzy w wyjściowej.

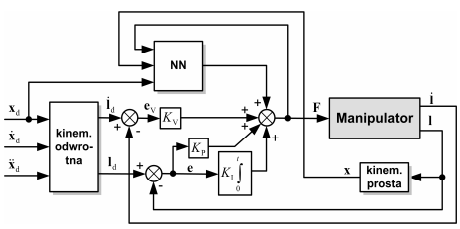
Z wykorzystaniem tego modelu neuronowego zostało zaproponowane następujące prawo sterowania:

$$\begin{aligned}\mathbf{F}(t) = \sum_{j=1}^{3} \mathbf{W}_j \tanh\left(\sum_{l=1}^{15} \mathbf{w}_{jl}\varphi_l + \mathbf{w}_{j,0}\right) + \mathbf{W}_0 + \\ K_\mathrm{P}\mathbf{e}(t) + K_\mathrm{V}\mathbf{e}_\mathrm{V}(t) + K_\mathrm{I}\int_0^t \mathbf{e}(\tau)\ \mathrm{d}\tau\end{aligned}, \tag{4}$$

gdzie $\mathbf{w}$ jest macierzą wag warstwy ukrytej a $\mathbf{W}$ macierzą wag warstwy wyjściowej.

Struktura tego sterownika jest pokazana na rys. 5. Sieć neuronowa pracuje jako sprzężenie do przodu, przewidując wymagane wymuszenie, a rolą członu PID jest kompensacja zakłóceń i niedokładności modelu neuronowego. Podczas normalnej pracy wejściami sieci były:

$$\varphi_\mathrm{w}(t) = \left[\mathbf{x}(t-2), \mathbf{x}(t-1), \mathbf{x}_\mathrm{d}(t), \mathbf{F}(t-2), \mathbf{F}(t-1)\right]. \tag{5}$$



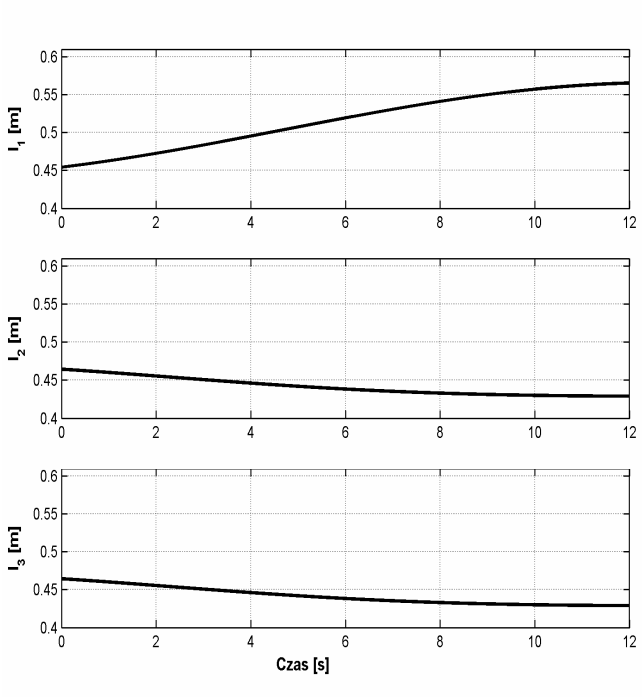
Rys. 5. Struktura sterownika neuronowego

Parametry członu PID zostały dobrane przez wirtualne prototypowanie, z wykorzystaniem struktury systemu podobnej jak ta, z rys. 4, ze sterownikiem wymienionym na neuronowy.

3.3. Szybkie prototypowanie

Działanie algorytmu sterowania zostało sprawdzone z rzeczywistym obiektem w czasie rzeczywistym z zastosowaniem techniki szybkiego prototypowania. W tym podejściu, program realizujący obliczanie prawa sterowania jest generowany automatycznie w Simulinku i wykonywany na specjalizowanym sprzęcie o dużej mocy, zbudowanym wokół mikroprocesora PowerPC. Daje to możliwość szybkiego sprawdzenia algorytmu i dostrojenia jego parametrów, bez wgłębiania się w zagadnienia dotyczące implementacji.

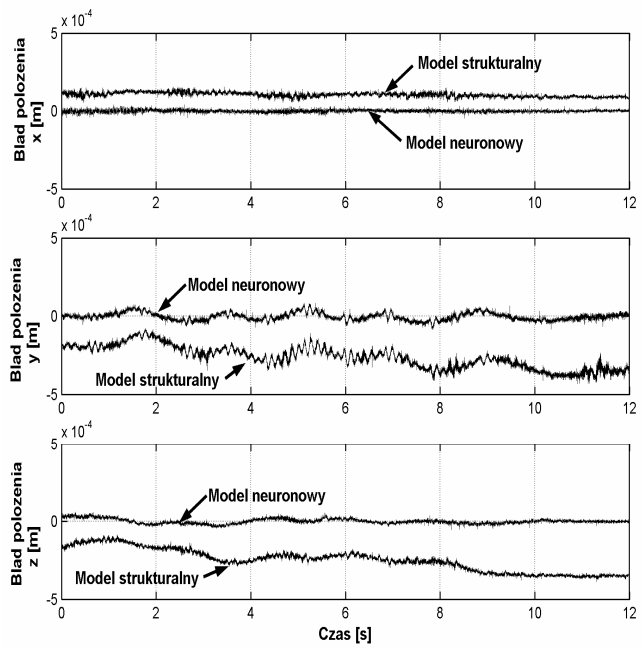
W czasie eksperymentu została użyta pozioma trajektoria prostoliniowa (rys. 6), bez frezowania.



Rys. 6. Trajektoria we współrzędnych złączowych użyta w trakcie szybkiego prototypowania

Całe prawo sterowania było obliczane w każdym okresie próbkowania z częstotliwością 1 kHz. Okazało się, że sterownik pracował najlepiej z tymi samymi parametrami, które zostały dobrane w trakcie wirtualnego prototypowania, co wskazuje na wysoką dokładność wielomasowego modelu manipulatora.

Wyniki, pokazane na (rys. 7), pokazują satysfakcjonującą dokładność śledzenia trajektorii, o wiele lepszą niż dla wstępnego sterownika stabilizującego.

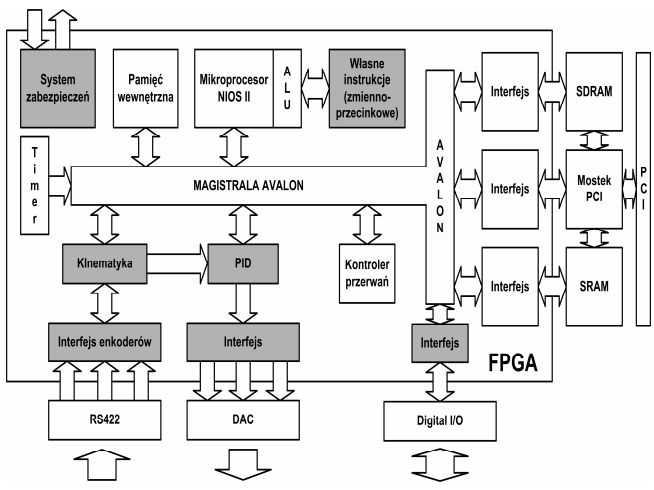


Rys. 7. Porównanie działania wstępnego sterownika stabilizującego, opartego na modelu strukturalnym (2) i sterownika neuronowego (4) w postaci błędów śledzenia trajektorii we współrzędnych Kartezjańskich

4. Implementacja sterowania

Algorytm sterowania został zaimplementowany na platformie sprzętowej zbudowanej w standardzie PC104+. Składa się ona z modułu komputera typu PC, wykorzystywanego do programowania układu FPGA, gromadzenia i transmisji danych, oraz głównego modułu z układem FPGA Stratix EP1S30 firmy Altera z peryferiami.

W pierwszej kolejności algorytm został przeanalizowany w celu zidentyfikowania fragmentów, które mogą być obliczone z zastosowaniem operacji stałoprzecinkowych. Następnie te fragmenty (równania kinematyki i algorytm PID) zostały zaimplementowane sprzętowo, z wykorzystaniem procedury opisanej w [13] tak, że wykonują się w pojedynczym cyklu zegara. Obliczenia pozostałej części prawa sterowania wykonywane są programowo na mikroprocesorze Nios II [14] wprogramowanym w układzie FPGA. W celu przyspieszenia obliczeń jednostka arytmetyczno-logiczna (ALU) Niosa II została sprzętowo rozbudowana o instrukcje użytkownika, aby szybko wykonywać najczęściej występujące w algorytmie operacje zmiennoprzecinkowe: mnożenie, dodawanie, odwrotność i pierwiastek kwadratowy, stosując techniki opisane w [15]. Architektura głównego modułu sterownika pokazana jest na rys. 8. Niemal cała elektronika cyfrowa o stosunkowo złożonej strukturze zawarta jest w pojedynczym układzie FPGA i dlatego może on być uważany za „system w układzie" (ang. System-on-Chip – SoC).



Rys. 8. Architektura sterownika; niestandardowe bloki użytkownika oznaczone są szarym tłem

Cała aplikacja zajmuje ok. 46% elementów logicznych, 100% bloków DSP i ok. 33% pamięci w układzie FPGA EP1S30, pozostawiając wystarczającą ilość zasobów na ewentualne przyszłe modyfikacje algorytmu sterowania. Pracując z zegarem 55 MHz system wykonuje obliczenia prawa sterowania w 25 μs, co pozwala na maksymalną częstotliwość próbkowania równą 40 kHz. Przy obecnej częstotliwości próbkowania wynoszącej 1 kHz, zapas czasu jest wystarczający do realizacji znacznie bardziej wymagających obliczeniowo metod sterowania.

5. Podsumowanie

W artykule przedstawiono konstrukcję robota równoległego do frezowania o trzech stopniach swobody wraz ze sterownikiem neuronowym wykorzystującym metodę bezpośredniego modelu odwrotnego. W celu zebrania danych do zbioru uczącego zastosowano wstępny sterownik stabilizujący oparty na analitycznym modelu strukturalnym manipulatora. Opisane zostało wirtualne prototypowanie robota, szybkie prototypowanie sterowników z porównaniem efektów ich działania oraz implementacja sterownika neuronowego.

Platforma sprzętowa sterownika oparta jest na nowoczesnym układzie FPGA. Algorytm sterowania został zaimplementowany w sposób sprzętowo-programowy, tzn. jego część wykonywana jest sprzętowo, a część programowo, na mikroprocesorze Nios II wprogramowanym w tym samym układzie FPGA. Opracowany sterownik jest przykładem systemu w pojedynczym układzie scalonym, co pozwala na wysoki poziom integracji i jednocześnie elastyczności. Zastosowana metodologia obniżyła koszt i czas trwania projektu.

Praca naukowa finansowana ze środków Komitetu Badań Naukowych jako projekt badawczy 4 T07B 077 26.

6. Literatura

[1] Petko M., Wybrane zagadnienia projektowania mechatronicznego, Rozprawy Monografie, nr 153, Wyd. AGH, Kra-ków 2005.

[2] J.-P. Merlet, Parallel Robots. Kluwer Academic Publishers, 2000.

[3] Tzafestas, S.G., Schmidt G., (red.) Progress in system and robot analysis and control design. Springer, 1999.

[4] R. Iserman Mechatronic Systems: Fundamentals, Springer, 2003.

[5] Bishop, R.H. (ed.), The Mechatronics Handbook. CRC Press, 2002.

[6] M. Wegrzyn, M. A. Adamski and J. L. Monteiro, The application of reconfigurable logic to controller design, Control Engineering Practice, Vol. 6, 1998, pp. 879-887

[7] T. Kambe, A. Yamada and M. Yamaguchi, Trend of system level design and approach to C-based design, Microelectronics Journal, Vol. 33, 2002, pp. 875-880

[8] M. Petko and G. Karpiel, Mechatronic design of a parallel manipulator for milling, in Proceedings of the 2005 IEEE/ASME International Conference on Advanced Intelligent Mechatronics, pp. 759-764

[9] Petko M., Manipulator równoległy do frezowania o trzech stopniach swobody, PAK nr 5/2005, s. 24-27

[10] Craig, J.J., Wprowadzenie do robotyki: Mechanika i sterowanie. WNT, 1995.

[11] Jang, J.-S. R., C.-T. Sun, E. Mizutani, Neuro-Fuzzy and Soft Computing: A Computational Approach to Learning and Machine Intelligence. Upper Saddle River: Prentice Hall, 1997

[12] Nørgaard, M., Ravn, O., Poulsen, N. K., Hansen, L. K. Neural Networks for Modelling and Control of Dynamic Systems. Springer-Verlag, London, 2000.

[13] M. Petko, and G. Karpiel, Semi-Automatic Implementation of Control Algorithms in ASIC/FPGA, in 2003 IEEE Conf. on Emerging Technologies and Factory Automation: Pro-ceedings, Lisbon, 2003, vol. 1, pp. 427-433

[14] http://www.altera.com

[15] A. R. Omondi, Computer Arithmetic Systems. Prentice Hall, 1994.

Artykuł recenzowany