dr inż. Marian Gilewski
Politechnika Białostocka

ALGORYTM STERUJĄCY SERWOMECHANIZMAMI RAMIENIA ROBOTA AL5A

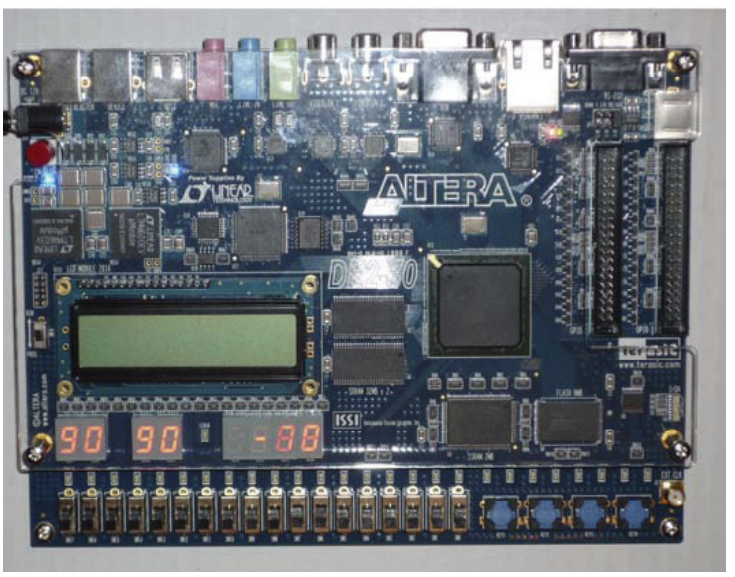
W artykule przedstawiono koncepcję sterowania ramieniem AL5A z zastosowaniem układu FPGA. Serwomechanizmy AL5A posiadają autonomiczne algorytmy zaimplementowane w języku VHDL. Opracowany jednoukładowy system cyfrowy może obsłużyć pracę systemu przez kilka minut.

THE AL5A ROBOT'S CONTROL ALGORITHM

This paper presents an idea of AL5A robot's control algorithm. Servo motors of AL5A have autonomous algorithms. Each algorithm was implemented into VHDL code. The FPGA on-chip control system can maintain AL5A arm for a few minutes.

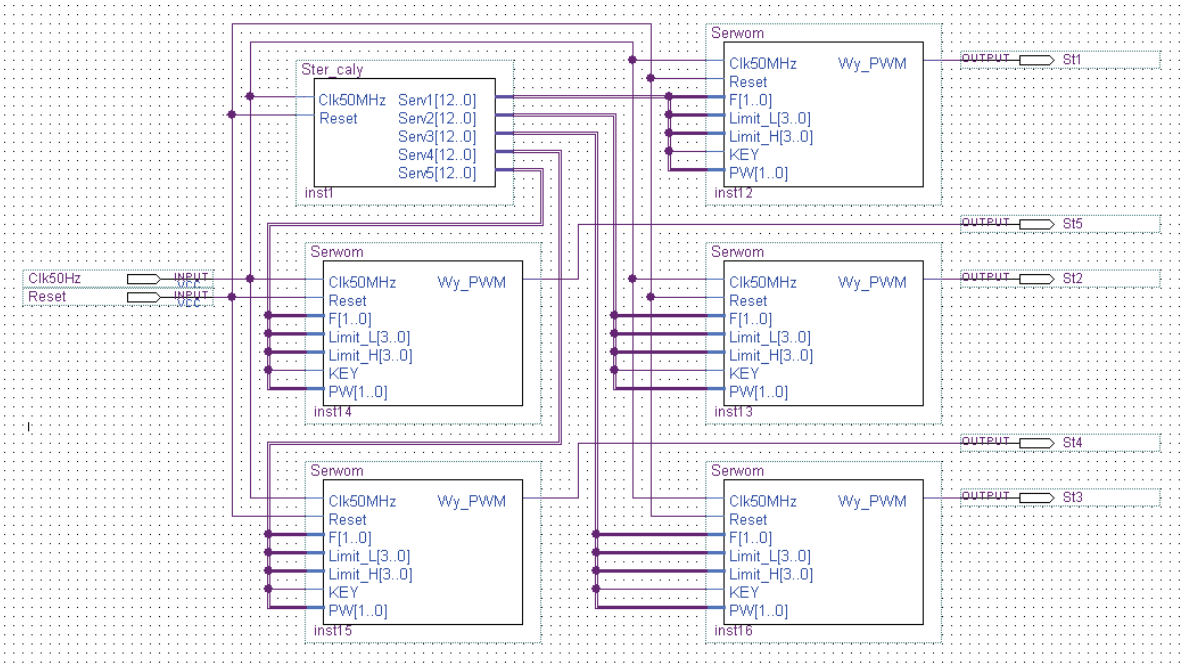
1. WSTĘP

Ramię robota AL5A [1] zawiera pięć niezależnie sterowanych serwomechanizmów. Każdy z serwomechanizmów łączy się z systemem sterującym za pośrednictwem trójprzewodowej magistrali zasilająco-sterującej. Magistrala zawiera przewody doprowadzające: stałe napięcie zasilające, sygnał sterujący oraz wspólny przewód masy. Sygnał sterujący jest cyfrowym przebiegiem prostokątnym w standardzie LVTTL, o częstotliwości 50 Hz i zmiennym współczynniku wypełnienia. Czas trwania impulsu sterującego zawiera się w przedziale od 0,6 ms do 2,4 ms, co zapewnia obrót wału serwomechanizmu o kąt od –90° do +90° w stosunku do położenia neutralnego. Położeniu neutralnemu odpowiada impuls o czasie trwania 1,5 ms. Możliwe są różne implementacje systemu sterującego pracą omawianego ramienia. W publikacji przedstawiono koncepcję z użyciem struktury programowalnej FPGA. Do badań wybrano układ EP2C70F896C6 [2] występujący w module prototypowym DE2-70 [3] firmy Altera (rys. 1).



Rys. 1. Widok modułu DE2-70 użytego do sterowania ramieniem AL5A

Schemat blokowy układu sterującego ramieniem robota, będący jednocześnie zapisem algorytmu sterującego na platformie Quartus II [4], przedstawiono na rys. 2.



Rys. 2. Schemat układu sterującego

Zawiera on nadrzędny moduł sterujący *Ster_caly*, który koordynuje pracę poszczególnych serwomechanizmów. Jego rolą jest ustawienie pozycji początkowych serwomechanizmów a następnie podawanie sygnałów sterujących ich pracą za pośrednictwem 13 bitowych magistral, zawierających sygnały: *F[1..0]*, *Limit_L[3..0]*, *Limit_H[3..0]*, *KEY* i *PW[1..0]*. Do modułu nadrzędnego doprowadzone są dwa sygnały sterujące: sygnał zegarowy o częstoliwości 50 MHz *Clk50MHz* i sygnał zerujący *Reset*. Moduł ten można zrealizować jako pamięć stałą, której komórki przechowują nastawy zmiany położenia wałów serwomechanizmów w kolejnych krokach pracy ramienia. W takim przypadku obsługa modułu nadrzędnego sprowadza się do cyklicznego, z częstotliwością 50 Hz, generowania adresów pamięci.

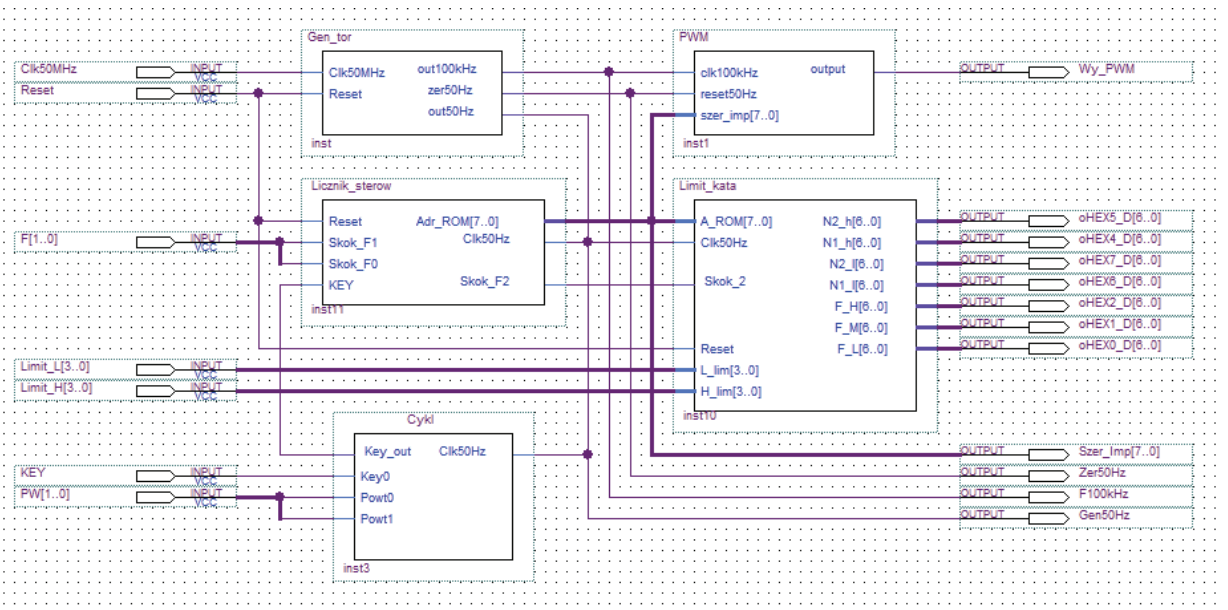
Alternatywą może być implementacja w kodzie VHDL równań opisujących trajektorie zmiany położenia ramienia. Biorąc pod uwagę, iż układ EP2C70F896C6 zawiera około 1 200 000 bitów pamięci wewnętrznej, zastosowanie w omawianej realizacji 65-bitowych słów pozwala na zapamiętanie ponad 18 000 kroków, czyli umożliwia ponad 6-minutową pracę ramienia. Jest to niewątpliwie rozwiązanie i oszacowanie nadmiarowe jeżeli chodzi o wykorzystanie pamięci, gdyż nie wszystkie serwomechanizmy muszą pracować w tym samym czasie oraz zmieniać swoje położenie w maksymalnym zakresie. Wspominany układ FPGA należy do rozwiązań średniej wielkości, największe matryce zawierają pamięci wewnętrzne nawet 10-krotnie większe. Istnieje również możliwość zastosowania dodatkowych pamięci zewnętrznych, wówczas dysponowany czas pracy niepowtarzalnego algorytmu wydłuża się do kilkudziesięciu godzin (pamięci SDRAM o pojemności gigabitów).

W proponowanej aplikacji zastosowano 5 identycznych modułów *Serwom* nadzorujących pracę poszczególnych serwomechanizmów, które sterowane są podobnymi zunifikowanymi sygnałami. Implementacja pojedynczego modułu absorbuje ponad 1000 makrokomórek struktury FPGA, co stanowi około 1 % zasobów logicznych użytego układu.

2. MODUŁ STERUJĄCY POJEDYŃCZYM SERWOMECHANIZMEM

Strukturę modułu obsługującego pojedynczy serwomechanizm przedstawiono na rysunku trzecim. Znaczenie sygnałów zewnętrznych jest następujące:

- dwubitowa magistrala *F[1..0]* umożliwia ustawienie wartości zmiany położenia wału w kolejnym kroku (o 1°, 2°, 5° lub 10°),
- czterobitowe magistrale *Limit_L[3..0]* i *Limit_H[3..0]* określają skrajne lewe i prawe położenie wału względem położenia neutralnego (–90°, –85°, ...; 90°, 85°, ...),
- dwubitowa magistrala *PW[1..0]* wymusza liczbę powtórzeń bieżącego położenia wału (brak powtórzenia, 5 powtórzeń, 10 powtórzeń, dowolna liczba powtórzeń zakończona ręcznym wyzwoleniem),
- *KEY* sygnał ręcznego wyzwalania,
- *Wy_PWM* jest sygnałem wyjściowym, podawanym bezpośrednio na linię sterującą serwomechanizmu,
- pozostałe sygnały (*oHEX0_D[6..0]*...*oHEX7_D[6..0]*, *Szer_Imp[7..0]*, *Zer50Hz*, *Gen50Hz*) pełniły pomocnicze punkty pomiarowe w czasie uruchamiania modułu.



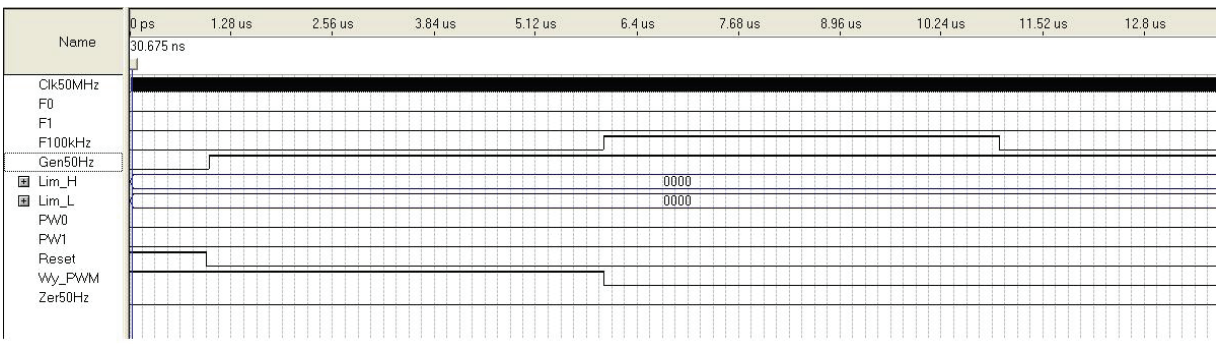
Rys. 3. Schemat modułu sterującego serwomechanizmem AL5A

Generator sygnałów *Gen_tor* na podstawie przebiegu okresowego 50 MHz wytwarza trzy sygnały: *F100kHz* o częstotliwości 100 kHz, *Gen50Hz* o częstotliwości bazowej 50 Hz oraz *Zer50Hz* krótkie impulsy zerujące o częstotliwości 50 Hz. Blok *Cykl* synchronizowany sygnałem *Gen50Hz* na podstawie nastaw *KEY* i *PW[1..0]* wytwarza sygnał *Key_out* inicjujący lub wstrzymujący zmianę położenia wału w następnym kroku. Automat stanów *Licznik_sterow* na podstawie nastawy *F[1..0]* oraz sygnałów *Key_out*, *Skok_2* (z bloku *Limit_kata*) i *Gen50Hz* wytwarza kod położenia wału *Adr_ROM[7..0]* w następnym kroku. Kod ten niesie informację o przyroście kąta oraz kierunku (w prawo lub lewo) położenia. Długość kodu określa minimalną rozdzielczość zmiany położenia w kolejnym kroku; w badanym przypadku zastosowano kod 8-bitowy, co pozwoliło uzyskać 1-stopniową rozdzielczość. *Limit_kata* porównuje wartość kodu położenia z wartościami nastaw *Limit_L[3..0]* i *Limit_H[3..0]*, jeżeli zostało osiągnięte skrajne położenie blok generuje sygnał *Skok_2* zmiany kierunku obrotu wału. Dla potrzeb diagnostycznych podczas badań

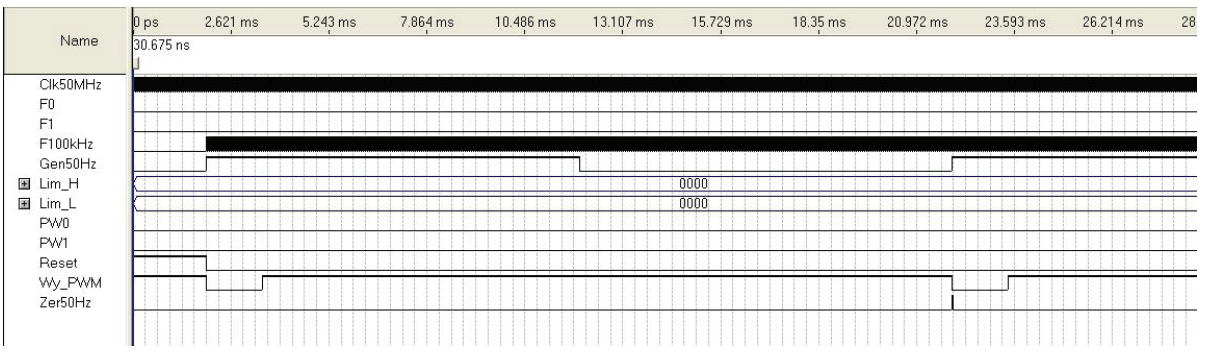
blok wytwarzał dodatkowe sygnały diagnostyczne oraz obsługujące wyświetlacze DE2. *PWM* jest sterowanym przerzutnikiem monostabilnym, który na podstawie kodu położenia *Adr_ROM[7..0]* wytwarza impuls prostokątny o wymaganym czasie trwania z rozdzielczością sygnału *F100kHz* a następnie przechodzi w stan zawieszenia. Ze stanu zawieszenia przerzutnik jest wyprowadzany sygnałem *Zer50Hz* po upływie 20 ms, generując kolejny impuls na narastającym zboczu sygnału *Gen50Hz*.

3. WYBRANE WYNIKI BADAŃ MODUŁU SERWOMECHANIZMU

W tej części pracy przedstawiono wyniki badań symulacyjnych oraz laboratoryjnych w wybranych punktach modułu sterującego serwomechanizmem. Na rys. 4 przedstawiono zachowanie układu po jego wyzerowaniu. Zerowanie układu następuje w czasie wysokiego poziomu na wejściu Reset, w tym stanie serwomechanizm ustawia się w położenie neutralne, tj. 0°. Po zaniku sygnału zerującego pojawia się narastające zbocze sygnału *Gen50Hz* oraz sygnał *F100kHz*, po upływie 0,6 ms rozpoczyna się impuls sterujący serwomechanizmem (poziom niski sygnału *Wy_PWM*). Wartość 0,6 ms jest minimalnym czasem trwania impulsu akceptowanym przez serwomechanizm, odpowiadającym położeniu –90°.



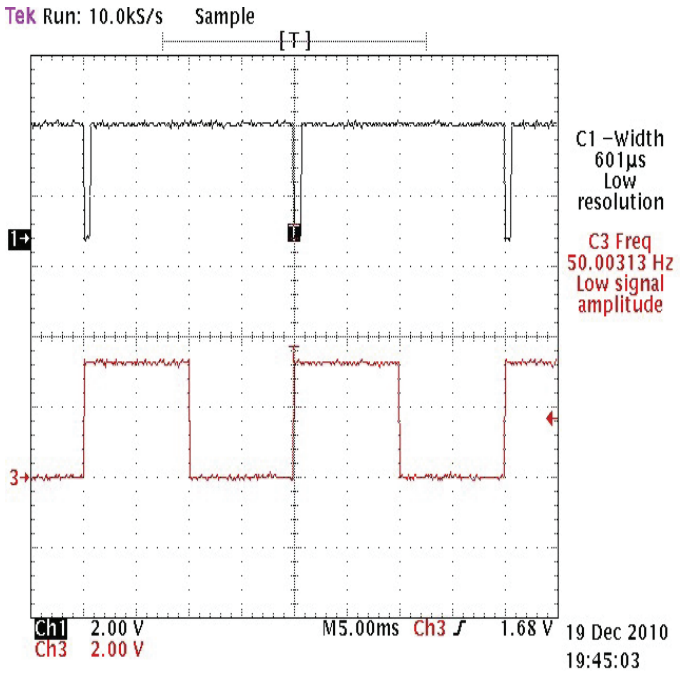
Rys. 4. Zerowanie modułu



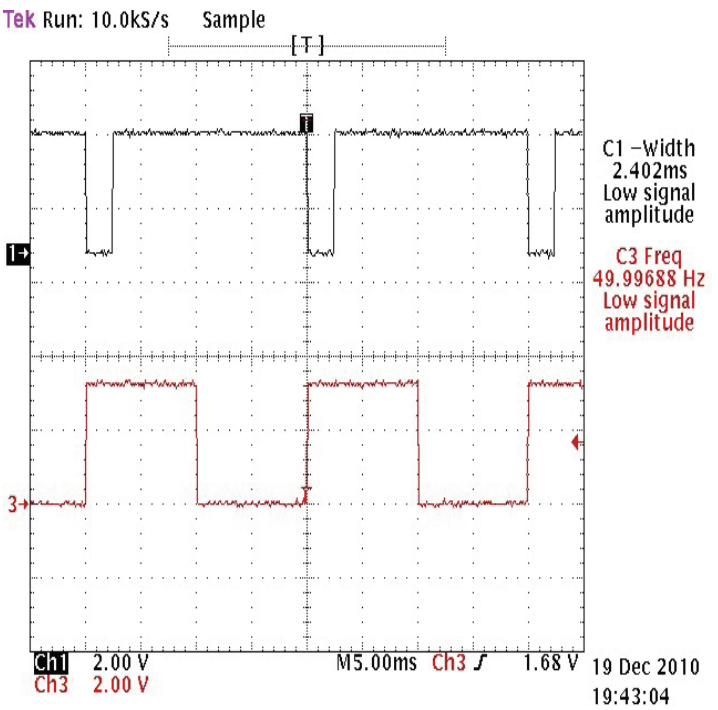
Rys. 5. Inne sygnały sterujące

Symulacja w dłuższym przedziale czasowym (rys. 5) pokazuje synchronizację wytwarzania impulsów sterujących *Wy_PWM* narastającymi zboczami przebiegu *Gen50Hz*. Impulsy zerujące *Zer50Hz* wyprowadzają przerzutnik sterowany *PWM* ze stanu zawieszenia tuż przed pojawieniem się kolejnego narastającego zbocza *Gen50Hz*. Przedstawione wyniki symulacji dotyczyły sytuacji zatrzymania serwomechanizmu w tym samym położeniu, czas trwania impulsów *Wy_PWM* nie ulegał zmianie. Lewe i prawe skrajne położenia były ustawione w wartości maksymalne, tzn. –90° i +90° (*Lim_L* = "0000", *Lim_H* = "0000"). Kolejne dwie charakterystyki (rys. 6 i rys. 7) przedstawiają wyniki badań laboratoryjnych, odpowiadające położeniu wału w pozycji –90° oraz +90°. Ujemna notacja sygnałów

Wy_PWM (aktywny sygnał niski) wynika z faktu iż w układzie elektronicznym elementy izolacji galwanicznej odwracają fazę sygnału.



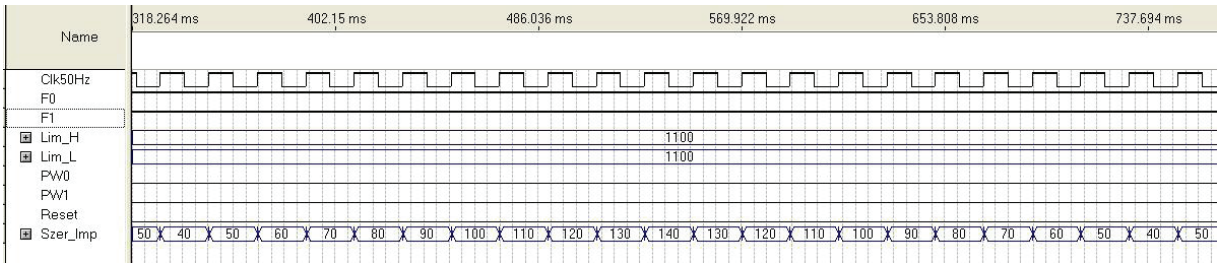
Rys. 6. Oscylogramy sygnału *Wy_PWM* (przebieg 1) oraz *Gen50Hz* (przebieg 3) w pozycji wału –90°



Rys. 7. Oscylogramy sygnału *Wy_PWM* (przebieg 1) oraz *Gen50Hz* (przebieg 3) w pozycji wału +90°

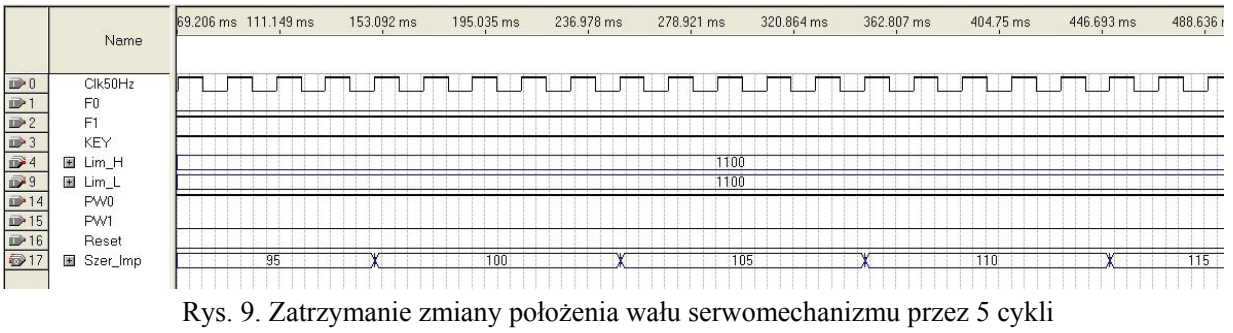
Dalsze wyniki badań symulacyjnych dotyczą odcinków czasu rzędu setek sekund. Z tego względu nie będzie prezentowany sygnał wyjściowy *Wy_PWM* lecz jego odpowiedniki w postaci dziesiętnej kodów położenia *Szer_Imp[7..0]*. Takie podejście nie wymaga symulacji całego układu w tak długim czasie. Na rys. 8 przedstawiono wynik symulacji

realizacji algorytmu cyklicznej zmiany położenia serwomechanizmu ze skokiem 10° między skrajnym lewym położeniem określonym kodem *Lim_L* = "1100" (–50°, *Szer_Imp* = 40) a skrajnym prawym położeniem określonym kodem *Lim_H* = "1100" (+50°, Szer_Imp = 140).

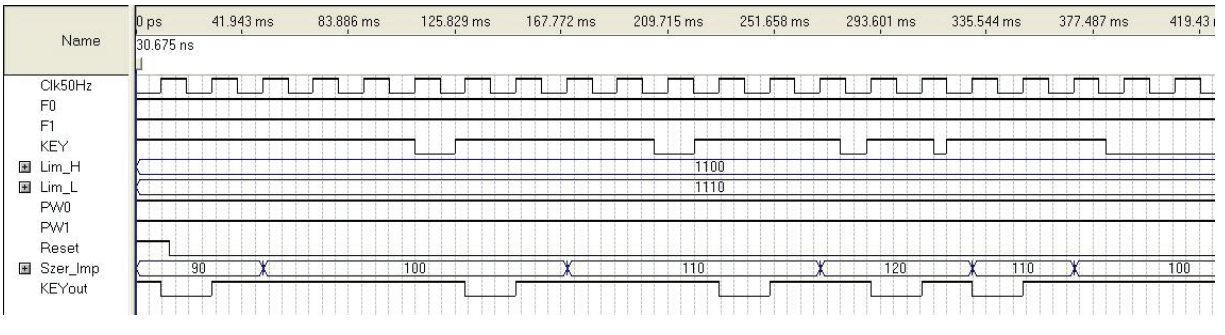


Rys. 8. Kolejne zmiany położenia wału ze skokiem 10°

Zmiana wartości magistrali *F[1..0]* pozwala wielokrotnie powtórzyć to samo położenie wału w kolejnych krokach (rys. 9). Ustawienie *F[1..0] = "11"* powoduje przejście w tryb ręcznego wyzwalania. W tym trybie układ po wygenerowaniu impulsu powiela go w kolejnych cyklach (rys. 10) aż do pojawienia się niskiego poziomu sygnału na wejściu *KEY*.



Rys. 9. Zatrzymanie zmiany położenia wału serwomechanizmu przez 5 cykli



Rys. 10. Ręczne wyzwalanie zmiany położenia wału

Sygnały aktywujące, o różnych czasach trwania, mogą pojawiać się z różnymi interwałami czasowymi co jest cechą ręcznego wyzwalania. W publikacji nie przedstawiono kodów źródłowych opisujących w VHDL funkcjonowanie poszczególnych bloków ze względu na ich obszerność. Każdy z elementów struktury hierarchicznej (rys. 3) został zdefiniowany w wersji VHDL'93 [5] i uruchomiony z zastosowaniem platformy projektowej Quartus II Web Edition dedykowanej do syntezy systemów w układach PLD firmy Altera.

4. PODSUMOWANIE

Celem pracy było sprawdzenie możliwości implementacji algorytmu sterującego ramieniem robota AL5A w strukturze programowalnej FPGA. Jako docelowy układ wybrano matrycę rodziny Cyclone II typu EP2C70F896C6 firmy Altera. Był to średniej wielkości układ programowalny. Algorytm kontrolujący pracę jednego serwomechanizmu zawierał kod obejmujący około 2500 wierszy w języku VHDL. W wyniku jego implementacji wykorzystano tylko 1 % dostępnych zasobów matrycy logicznej FPGA. Oznacza to, iż w badanej strukturze można zaimplementować zarówno 5 algorytmów kontrolujących pracę wszystkich serwomechanizmów jak też moduł nadrzędny koordynujący pracę całego układu. Badania przeprowadzono w stosunku do rozwiązania "oszczędnego" funkcjonalnie, tzn. zastosowano rozdzielczość zmiany położenia wału na poziomie 1°, możliwość zmiany przyrostu kąta zmiany położenia wału w zakresie 4 wartości oraz realizację nastaw skrajnych położeń z 5-stopniowym krokiem. Badania laboratoryjne wykazały, iż w celu uzyskania większej płynności ruchu serwomechanizmu należałoby zastosować większą rozdzielczość, np. rzędu 0,1 stopnia. Do algorytmu sterującego nie wprowadzono mechanizmów linearyzacji charakterystyki odpowiedzi układu na szerokość impulsów wyzwalających. Przybliżone pomiary wykazały zasadność zastosowania linearyzacji układu, jednak ze względu na brak odpowiedniej jakości aparatury pomiarowej w tym przypadku nie uwzględniono tego aspektu.

Odrębnym zagadnieniem pozostaje możliwy do osiągnięcia czas pracy algorytmu. Zastosowany układ FPGA pozwalał na kilkuminutową pracę niepowtarzalnej trajektorii ruchu. Rozwiązaniem problemu mogłoby być zastosowanie większego układu lub zewnętrznych pamięci, co nie stanowiłoby ograniczeń przy tak małych prędkościach zmiany położenia. Język VHDL dzięki rozbudowanym bibliotekom umożliwia syntezę zaawansowanych funkcji sterujących. Ze względu na większą przewidywalność położenia ramienia wybrana została wersja oparta na pamięciach i automatach stanu, gdyż nie wszystkie konstrukcje opisu behawioralnego w tym języku są w pełni syntezowalne.

Publikację przygotowano w ramach realizacji pracy statutowej S/WE/1/2006 finansowanej przez Ministerstwo Nauki i Szkolnictwa Wyższego.

5. BIBLIOGRAFIA

1. Hitec Robotics, http://www.lunxmotion.com/c-124-al5a.aspx [opis robota, dostęp online: 1 października 2010].
2. Altera Corporation: *Cyclone II Device Family Data Sheet*, 101 Innovation Drive, San Jose, California, 95134 USA, 2004.
3. Terasic Technologies Inc.: *DE2-70 Development and Education Board - User Manual*, ver. 1.08, huBei City, HsinChu County, Taiwan 302, http://www.terasic.com, 2009.
4. Altera Corporation: *Introduction to the Quartus II Software,* 101 Innovation Drive, San Jose, California, 95134 USA, www.altera.com, 2010.
5. IEEE Computer Society: *IEEE Standard VHDL Language Reference Manual*, New York, NY 100016-5997, USA, 2002.