mgr inż. Łukasz Bałdyga
dr inż. Marian Gilewski
Politechnika Białostocka

UKŁAD STERUJĄCY ROBOTEM AL5A

W artykule opisano strukturę układu sterującego ramieniem robota AL5A. Układ zawiera obwód zasilający, moduł izolacji galwanicznej oraz moduł wykonawczy z układem FPGA. Matryca FPGA zawiera algorytm sterujący zaimplementowany w języku VHDL. Rozwiązanie charakteryzuje się zwartą konstrukcją oraz stosunkowo małym poborem mocy.

A CONTROL CIRCUITS OF AL5A ROBOT ARM

This paper describes an idea of AL5A robot's control circuit. It includes of power supply circuit, optically coupled isolators and FPGA control chip. A control algorithm was implemented into the FPGA chip using VHDL language. The solution of control circuit has compacted structure and it is characterized by low power consumption.

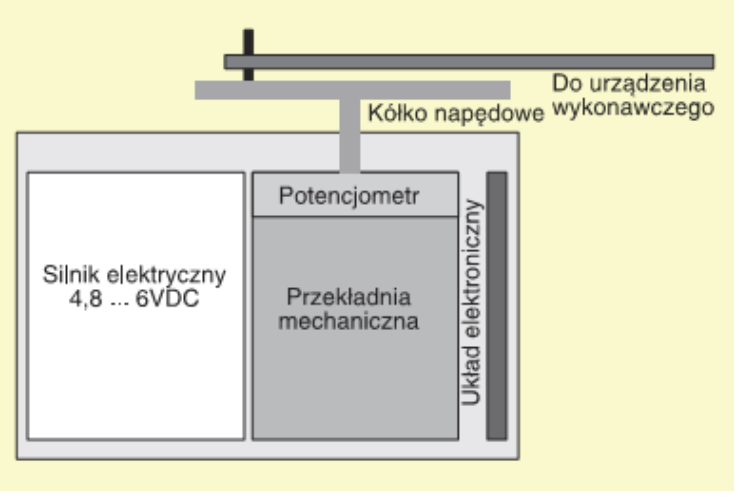
1. WSTĘP

Obiektem badanym w analizowanym przypadku było ramię robota AL5A [1, 2] (rys. 1) sterowane z układu programowalnego FPGA. Ze względu na elastyczność struktury programowalnej zrezygnowano z firmowego rozwiązania sterownika opartego na mikrokontrolerze. Model AL5A zawiera 5 niezależnie kontrolowanych serwomechanizmów.

Rys. 1. Widok ramienia robota AL5A firmy Lynxmotion

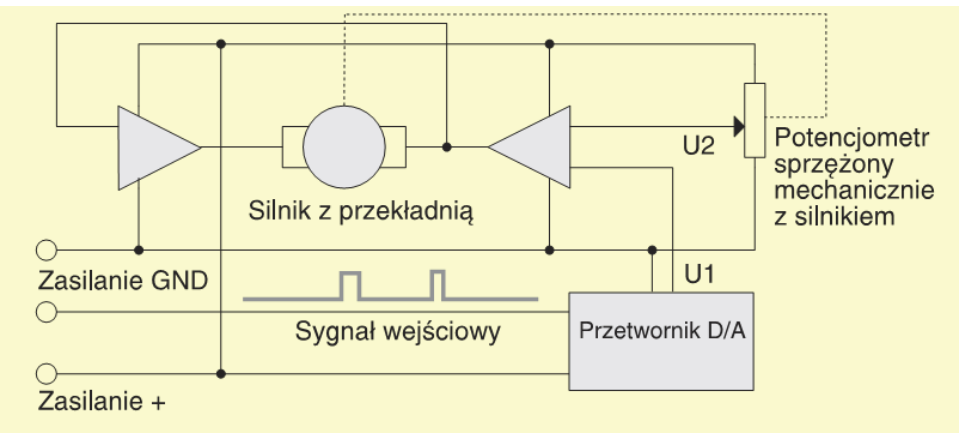
Każdy z nich połączony jest trójprzewodową magistralą zawierającą dwa przewody zasilające oraz linię sygnałową. Konfiguracja wewnętrzna serwomechanizmu zawiera (rys. 2): komutatorowy silnik prądu stałego, przekładnię mechaniczną, potencjometr sprzężony z silnikiem oraz wyspecjalizowany scalony układ sterujący. Użytkownik

ustawiając parametry impulsu w linii sygnałowej wyznacza pozycję, w której wał serwomechanizmu powinien znaleźć się, zaś serwomechanizm próbuje tę pozycję osiągnąć i utrzymać.



Rys. 2. Poglądowy widok układu serwomechanizmu

Zakres zmian obrotu wału badanych serwomechanizmów zawiera się w przedziale od 0° do 180°. Sygnał linii sterującej jest okresowym, prostokątnym przebiegiem sterującym o okresie 20 ms i zmiennym współczynniku wypełnienia. Wartość czasu trwania impulsu z przedziału od 0,6 ms do 2,4 ms jednoznacznie określa położenie wału serwomechanizmu. Wewnętrzny układ sterujący serwomechanizmu (rys. 3) stara się minimalizować uchyb błędu przemieszczenia na podstawie wejściowego sygnału sterującego oraz sygnału zwrotnego z potencjometru sprzężonego z wałem.

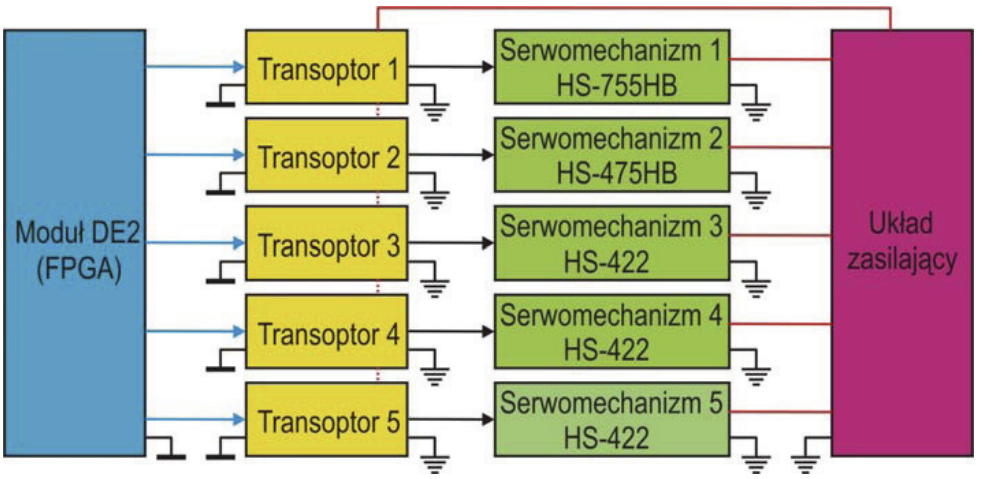


Rys. 3. Schemat blokowy serwomechanizmu

Model ramienia robota zawiera pięć niezależnie zasilanych i sterowanych serwomechanizmów. Stąd naturalnym rozwiązaniem centralnego układu sterującego jest zastosowanie pracujących współbieżnie, niezależnych algorytmów dla poszczególnych serwomechanizmów. Implementacja takiego rozwiązania wydaje się naturalna w strukturze FPGA. Synteza elastycznego algorytmu w strukturze jednoprocesorowej jest zbyt skomplikowana, z tego powodu zrezygnowano z firmowej aplikacji układu sterującego opartej na mikrokontrolerze ATMEGA168-20PU.

2. SCHEMAT FUNKCJONALNY UKŁADU STERUJĄCEGO

Elektroniczny układ sterujący pracą poszczególnych serwomechanizmów (rys. 4) zawiera: moduł prototypowy DE2 [3] z układem FPGA, moduł izolacji galwanicznej złożony z pięciu transoptorów oraz układ zasilający. W pracy wykorzystano moduł uruchomieniowy [4] z układem FPGA EP2C70F896C6N, którego zasoby są wystarczające do implemetacji algorytmów sterujących. Układ FPGA dostarcza sygnały sterujące poszczególne serwomechanizmy poprzez złącze GPIO_1 modułu DE2, który dodatkowo zawiera min. źródło sygnału synchronizującego o częstotliwości 50 MHz oraz elementy zasilające strukturę. Sygnał sterujący z wyjścia DE2 zasila obwód wejściowy transoptora CNY-17 izolujący galwanicznie FPGA z wybranym serwomechanizmem.

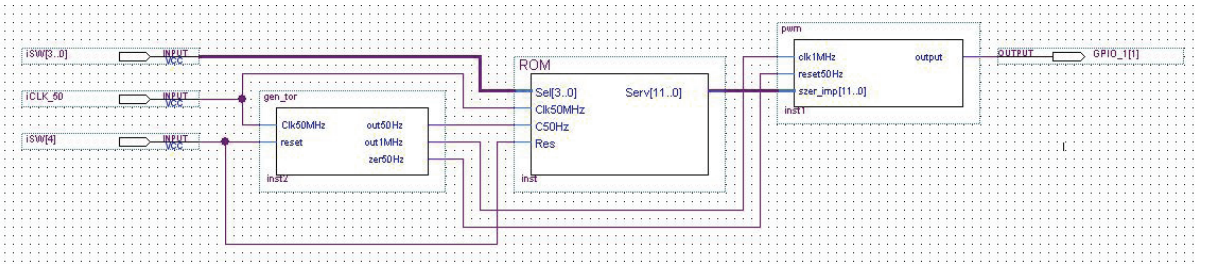


Rys. 4. Schemat układu sterującego ramienia AL5A

Układ zasilający wytwarza napięcia stałe zasilające poszczególne serwomechanizmy oraz obwody wyjściowe transoptorów. W celu uniknięcia sprzężeń poprzez obwód zasilania każdy z serwomechanizmów ma oddzielne stabilizowane źródło zasilania o napięciu 6 V i wydajności prądowej 1 A. W zależności od wartości obciążenia serwomechanizmu oraz amplitudy zmiany kąta obrotu wału natężenie prądu zasilającego może zmieniać się w zakresie od kilkudziesięciu mA do 1 A. Istotna jest zatem mała wartość rezystancji wyjściowej oraz duża pojemność wyjściowa układu zasilającego w celu zapewnienia wymaganej dynamiki zmian prądu zasilającego.

Sygnały sterujące poszczególne serwomechanizmy w strukturze FPGA można wytwarzać implementując niezależne algorytmy lub korzystać ze wspólnej tablicy (pamięci), której elementy zawierają wartości nastaw położeń wałów w kolejnych krokach położenia chwytaka. Autonomiczne algorytmy efektywniej wykorzystują zasoby struktury programowalnej, eliminując nadmiarowość informacji przechowywanej w układzie tablicowym. Układ tablicowy jest prostszy w implementacji, gdyż przechowuje jedynie wartości współczynników wyznaczonych wcześniej w sposób analityczny. Niezależnie od zastosowanej metody uzyskiwania nastaw położeń serwomechanizmów, końcowymi elementami układu sterującego pozostają bloki wytwarzające sygnał prostokątny 50 Hz o wymaganej wartości współczynnika wypełnienia. Strukturę przykładowego bloku przedstawiono na rys. 5; jego projekt przygotowano z użyciem pakietu projektowego Quartus II Web Edition. Zawiera on trzy elementy funkcjonalne: generator sygnałów sterujących (*gen_tor*), pamięć współczynników (*ROM*) oraz generator impulsów wyjściowych (*pwm*). Każdy z elementów zdefiniowany został z użyciem bibliotek

źródłowych pakietu projektowego lub kodów źródłowych napisanych w języku VHDL. Generator sygnałów sterujących, na podstawie sygnału odniesienia 50 MHz z portu *iCLK_50*, wytwarza stabilny sygnał 50 Hz (wyjście *out50Hz*) oraz sygnał 1 MHz (wyjście *out1MHz*) i krótki impuls zerujący 50 Hz (wyjście *zer50Hz*). Do wejścia układu doprowadzony jest również sygnał zerujący z portu *iSW[4]*. Pamięć współczynników przechowuje 12-bitowe dane określające położenia wału serwomechanizmu w kolejnych krokach. Wejściowa magistrala *iSW[3..0]* wybiera wartość skoku położenia wału – amplitudę zmiany kąta położenia. W badanym przypadku zakodowano cykliczną zmianę położenia wału od 0° do 180° i z powrotem z możliwością zadawania skoku: 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1, 2, 5, 10, 30, 60 i 90 stopni. Generator impulsów wyjściowych wytwarza okresowy sygnał prostokątny o współczynniku wypełnienia zależnym od wartości 12-bitowych danych z wyjścia pamięci. Sygnał z generatora wyprowadzany jest przez port wyjściowy *GPIO_1[1]*. Przykładowy kod opisujący funkcjonowanie generatora impulsów przedstawiono w następnym rozdziale.



Rys. 5. Struktura bloku sterującego serwomechanizmem

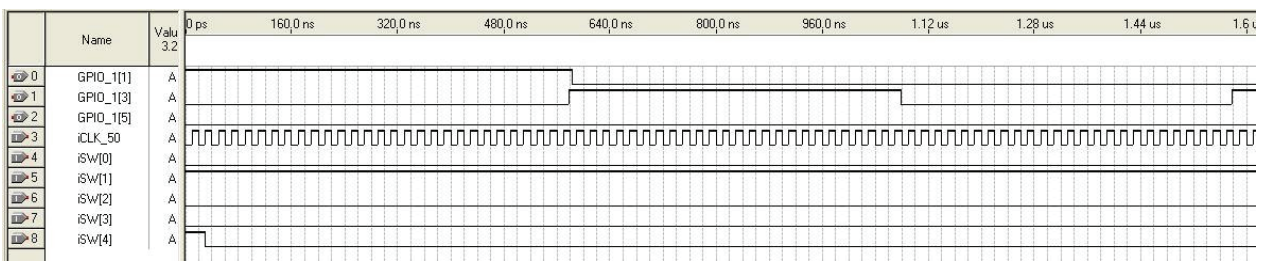
3. OPROGRAMOWANIE WYBRANEGO MODUŁU

Algorytm pracy generatora impulsów wyjściowych zakodowano w języku VHDL stosując technikę maszyny stanów. Sygnałami wejściowymi są: przebieg okresowy 1 MHz (port *clk1MHz*), 20 ns sygnał zerujący o częstotliwości 50 Hz (port *reset50Hz*) oraz 12-bitowa magistrala wartości czasu trwania impulsu wyjściowego (port *szer_imp*).

W stanie *s0* odmierzana jest minimalna wartość czasu trwania impulsu wyjściowego 0,6 ms, która odpowiada skrajnemu lewemu położeniu wału serwomechanizmu. Po upływie wymaganego czasu układ przechodzi do stanu *s_low* uzupełniającego czas trwania impulsu o wartość odczytaną z pamięci *ROM*. Po wygenerowaniu impulsu o żądanym czasie trwania moduł przechodzi w stan zawieszenia *s_high*. Po upływie okresu 20 ms układ wyzerowany impulsem *reset50Hz* przechodzi ponownie do stanu *s0*. Poniżej zawarto kompletny kod definiujący pracę modułu. Sygnał wyjściowy *output* zapisano w notacji odwrotnej (czasowi trwania impulsu przypisano stan niski) gdyż transoptor odwraca fazę.

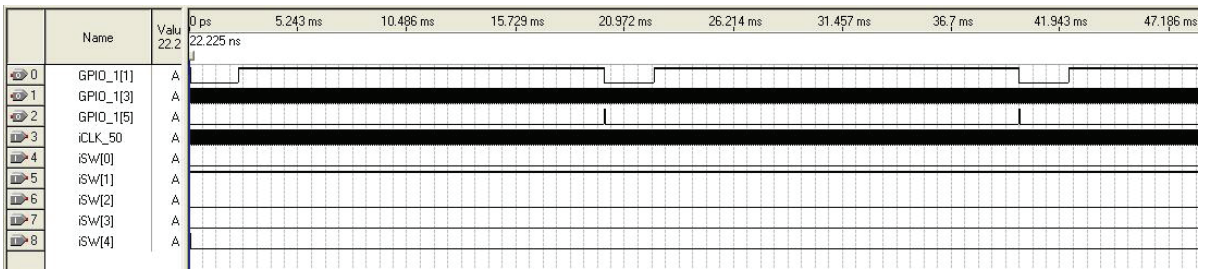
```
library ieee; use ieee.std_logic_1164.all;  use ieee.numeric_std.all;
entity pwm is
 port(clk1MHz, reset50Hz     : in    std_logic;
       szer_imp              : in    unsigned(11 downto 0);  -- nastawa szerokości impulsu
       output                : out   std_logic);
end entity;
architecture rtl of pwm is
 type state_type is (s0, s_low, s_high);
 signal state   : state_type;
  begin
   process (clk1MHz, reset50Hz)
```

```
  variable tmp : unsigned(11 downto 0) := "000000000000";
  begin
    if reset50Hz = '1' then
      output <= '1'; tmp := "000000000000"; state <= s0;
    elsif (rising_edge(clk1MHz)) then
      case state is
        when s0 =>                                  -- pomiar 0,6 ms = 600 okresów
          tmp := tmp + 1; output <= '0';
          if tmp = "001001011000" then    tmp := "000000000000"; state <= s_low;
          else   state <= s0;
          end if;
        when s_low =>
          output <= '0'; tmp := tmp + 1;
          if tmp = "011100001000" then          -- 1,8 ms - przekroczenie nastaw
            state <= s_high;                    -- 1,8 + 0,6 = 2,4 ms
          elsif tmp = szer_imp then state <= s_high;
          else  state <= s_low;
          end if;
        when s_high => output <= '1'; state <= s_high;
        when others => state <= s0;
      end case;
    end if;
  end process;
end rtl;
```



Rys. 6. Przebiegi czasowe generatora po wyzerowaniu

Na rys. 6 przedstawiono wyniki symulacji zachowania układu tuż po jego wyzerowaniu (*iSW[4]* = ‘1’). Widoczny jest sygnał synchronizujący *Icek_50* o częstotliwości 50 MHz oraz wytworzony na jego podstawie sygnał *GPIO_1[3]* o częstotliwości 1 MHz. Narastające zbocze sygnału *GPIO_1[3]* rozpoczyna fazę *s0* trwającą 0,6 ms. Z kolei na poniższym rysunku przedstawiono wyniki symulacji w dłuższej skali czasowej. Widoczne są na nim okresowe impulsy wyjściowe generatora (*GPIO_1[1]*) o czasie trwania poziomu niskiego proporcjonalnym do wartości słowa wyjściowego *ROM* i skoku zadanego nastawami *iSW[0]* – *iSW[3]*. Sygnał *GPIO_1[5]* odzwierciedla okresowe impulsy zerujące 50 Hz.



Rys. 7. Przebiegi czasowe generatora w dłuższej skali czasowej

4. PODSUMOWANIE

W pracy wykazano, iż można zaimplementować algorytmy sterujące pracą pięciu serwomechanizmów ramienia robota w pojedyńczym układzie FPGA średniej mocy obliczeniowej. W analizowanym przypadku, jak wynika z zestawienia przedstawionego na rys. 8, do ponad 1,5 minutowej pracy serwomechanizmu ze zmianą kąta obrotu wału, co 0,1° wykorzystano poniżej 1 % dostępnych makrokomórek (591 total logic elements) oraz poniżej 15 % pamięci wbudowanej (172416 total memory bits). Zaletą powyższego rozwiązania jest zwarta konstrukcja umożliwiająca syntezę jednoukładowego cyfrowego systemu sterującego typu on-chip. Dłuższe cykle pracy ramienia robota wymagałyby zastosowania większych struktur programowalnych FPGA. Można w tym celu zastosować układy rodziny Stratix (Altera) lub Virtex (Xilinx), mające matryce logiczne i wbudowane pamięci ponad 10-krotnie większe niż zastosowana rodzina Cyclone II. Rozwiązaniem problemu byłoby również zastosowanie pamięci zewnętrznych, których pojemność jest elementem decydującym o czasie pracy algorytmu, rozdzielczości zmiany położenia wału serwomechanizmu oraz jej dynamice. Istotny jest też rodzaj algorytmu sterującego, jest to jednak odrębne zagadnienie wykraczające poza zakres niniejszej publikacji.

```
+-------------------------------------------------------------------------------------+
; Flow Summary                                                                       ;
+-----------------------------------+-------------------------------------------------+
; Flow Status                       ; Successful - Sat Dec 04 20:58:32 2010           ;
; Quartus II Version                ; 9.1 Build 304 01/25/2010 SP 1 SJ Web Edition    ;
; Revision Name                     ; I_tor                                           ;
; Top-level Entity Name             ; I_tor                                           ;
; Family                            ; Cyclone II                                      ;
; Device                            ; EP2C70F896C6                                    ;
; Timing Models                     ; Final                                           ;
; Met timing requirements           ; No                                              ;
; Total logic elements              ; 591 / 68,416 ( < 1 % )                          ;
;     Total combinational functions ; 451 / 68,416 ( < 1 % )                          ;
;     Dedicated logic registers     ; 344 / 68,416 ( < 1 % )                          ;
; Total registers                   ; 344                                             ;
; Total pins                        ; 9 / 622 ( 1 % )                                 ;
; Total virtual pins                ; 0                                               ;
; Total memory bits                 ; 172,416 / 1,152,000 ( 15 % )                    ;
; Embedded Multiplier 9-bit elements ; 0 / 300 ( 0 % )                                ;
; Total PLLs                        ; 0 / 4 ( 0 % )                                   ;
+-----------------------------------+-------------------------------------------------+
```

Rys. 8. Raport wykorzystania zasobów struktury EP2C70

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. Bałdyga Ł.: *Implementacja algorytmu sterującego ramieniem robota w układzie FPGA - praca magisterska,* Politechnika Białostocka, Wydział Elektryczny, 2010.
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: *Cyclone II Device Family Data Sheet*, 101 Innovation Drive, San Jose, California, 95134 USA, 2004.