Czujnik przyśpieszenia (1)

Układ odczytu i wyświetlania

Newton, formułując drugą zasadę dynamiki określał zmianę prędkości w czasie „zmianą ruchu" (mutationem motus). Obecnie używamy pojęcia przyśpieszenie i mamy odpowiednie przyrządy do jego pomiaru. Akcelerometry, bo o nich mowa, są elementami stosowanymi w systemach nawigacji lotniczej, ale też w urządzeniach powszechnego użytku, jak telefony komórkowe czy aparaty fotograficzne. W artykule opisano sposób odczytu informacji z akcelerometru o zmierzonym przyśpieszeniu. W pierwszej części przedstawiono realizację interfejsu do komunikacji pomiędzy akcelerometrem Freescale MMA7455L a układem FPGA. W drugiej pokazano jak wygenerować sygnały sterujące dla monitora VGA, aby wyświetlić na nim odczytane wartości przyśpieszenia.

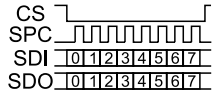
Projekt zrealizowano no płycie testowej HW-SPAR3AN-SK-UNI-G z układem XC3S700AN za pomocą oprogramowania ISE WebPACK oraz ChipScope Pro firmy Xilinx.

Komunikacja z akcelerometrem – interfejs SPI

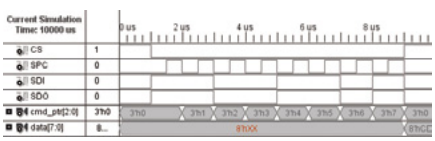
W akcelerometrze wyniki pomiarów są umieszczane w specjalnych rejestrach. Do ich odczytania użyjemy interfejsu SPI, przy czym akcelerometr będzie pracował w trybie *slave*, natomiast układ FPGA w trybie *master*.

W tym interfejsie do komunikacji są używane 4 sygnały SDO: (*Slave Data Output*), SDI (*Slave Data Input*), SCK (*Serial Clock*), tutaj oznaczony jako SPC (*SPI Serial Clock*) oraz CS (*ang. Chip Select*). Przebieg tych sygnałów podczas pojedynczej sesji wymiany danych przedstawiono na **rysunku 1**.

Interfejs SPI jest zbudowany w oparciu o 8-bitowy rejestr przesuwający. Urządzenie *master* przesyła kolejne bity z tego rejestru do urządzenia *slave*. Jednocześnie *master* odczytuje sygnał *SDO* i zapisuje jego kolejne wartości. Opis rejestrów przesuwających w języku Verilog pokazano we fragmencie definicji interfejsu SPI na **listingu 1**.



Rysunek 1. Wysyłanie i odbiór 8 bitów poprzez SPI



Rysunek 2. Symulacja działania modułu SPI

Sterowanie interfejsem odbywa się za pomocą wskaźnika. Określa on, z której pozycji bit rejestru *cmd* zostanie przesłany oraz na której pozycji rejestru *dane_odczytane* należy zapisać wartość odebranego bitu. Co 8 cykli zegarowych dane z rejestru *dane_odczytane* są przepisywane do zewnętrznego rejestru *dane*. Rejestry są taktowane zewnętrznym sygnałem zegarowym, który jest również przesyłany do urządzenia *slave* wówczas, gdy jest aktywna komunikacja. Komunikacja jest uruchamiana, gdy zostanie ustawiony zewnętrzny sygnał *init* (ma poziom wysoki). Sygnał *CS* o poziomie niskim rozpoczyna transfer danych, natomiast przez port *SPC* przesyłany jest sygnał zegarowy do urządzenia *slave*, w takt którego prowadzona jest wymiana danych. Gdy sygnał *init* ma poziom wysoki, to wskaźnik rejestrów zwiększa swoją wartość przy każdej zmianie sygnału zegarowego z 1 na 0. Na **listingu 2** pokazano opis generatora sygnału zegarowego dla modułu *SPI*.

Interfejs SPI akcelerometru MMA7455L pozwala na zastosowanie zegara o częstotliwości do 8 MHz. Załóżmy, że dysponujemy zegarem o częstotliwości 50 MHz. Aby uzyskać sygnał taktujący dla modułu obsługującego interfejs SPI, zastosujmy licznik, który zlicza impulsy zegara o częstotliwości 50 MHz. Gdy licznik osiągnie odpowiednią zawartość, jest zmieniany poziom logiczny sygnału *tick1Mhz* na przeciwny. Sygnał *tick1Mhz* o częstotliwości 1 MHz jest dołączony do modułu SPI jako taktujący. Prawidłowość działania zaprojektowanego modułu SPI została zweryfikowana za pomocą symulatora. Wynik przeprowadzonej symulacji przedstawiono na **rysunku 2**.

Podczas symulacji było przesyłane 8-bitowe słowo *11001101*. Wyjście modułu *SDI* było dołączone było do wejścia *SDO*, dzięki czemu można było sprawdzić czy moduł prawidłowo odbiera kolejno przychodzące bity oraz czy zapisuje je w odpowiednim rejestrze w chwili, gdy zostanie odebrany ostatni z nich. Początkowo rejestr *data* zawiera wartość nieokreśloną, oznaczoną w systemie szesnastkowym jako *8'hXX*. W chwili zainicjowania komunikacji, sygnał *CS* przyjmuje wartość 0 i zaczyna być nadawany sygnał zegarowy *SPC*. Z każdym kolejnym taktem zegara jest zwiększana zawartość licznika wskaźnika i zmienia się stan wyjścia *SDI* oraz wejścia *SDO*. Gdy zostanie odebrany ostatni bit, komunikacja zostaje przerwana (*CS* na poziomie wysokim), sygnał zegarowy przestaje być przesyłany, a do rejestru *data* jest przepisywana odebrana zawartość rejestru przesuwającego. Na rys. 2 jest szesnastkowa liczba *8'hCD* odpowiadającą wysyłanemu słowu w postaci dwójkowej.

Listing 1. Interfejs SPI

```
assign CS = ~init;                    //Inicjalizacja komunikacji
assign SDI  = cmd[7-cmd_ptr];         //Przypisywanie wyjściu kolejnych stanów
assign dane = (cmd_ptr==3'b000) ? dane_odczytane : dane;
                                      // Gdy licznik wskaźnika przepełniony
                                      //odczytana porcja danych przesyłana dalej
always @ (negedge clk) dane_odczytane[7-cmd_ptr] = SDO;
                                      // wpisanie stanu wejścia do rejestru
always @ (negedge clk)                //Przesuwanie wskaźnika po rejestrze
   if(init)
   cmd_ptr = cmd_ptr + 1;
   else cmd_ptr = 3'b0;
assign SPC = (init) ? clk : 1'b0;  //Przesłanie zegara
```

Listing. 2. Generator sygnału zegarowego

```
always @ (posedge CLK_50MHZ) begin   // Każde pozytywne zbocze zegara
                                     // uruchamia blok
   cnt1MHz <= cnt1MHz + 1;           // Zwiększenie licznika
   if(cnt1MHz==6'd24) begin          // gdy licznik osiągnie żądaną wartość
      tick1MHz <= !tick1MHz;         // zanegowanie sygnału
      cnt1MHz <= 6'b0;               // wyzerowanie licznika
   end
end
```

Automat stanów – generowanie poleceń i odbieranie danych

Jest to moduł układu sekwencyjnego zarządzającego komunikacją pomiędzy akcelerometrem a układem FPGA.

Po dokonaniu pomiarów przyspieszenia przez akcelerometr MMA7455L, ich wyniki są umieszczane w przeznaczonych do ich przechowywania rejestrach, np. *h06*. W rejestrach również są przechowywana dane konfigurujące akcelerometr. Określają one i kontrolują tryb pracy akcelerometru, zakres pomiaru, częstotliwość ich powtarzania, wartości poziomów detekcji oraz wybór rodzaju komunikacji. Dostęp do rejestrów uzyskuje się poprzez przesłanie 6-bitowego adresu. Zawartość rejestrów, w zależności od ich funkcji, możemy modyfikować bądź tylko odczytywać. Komenda, którą należy przesłać do akcelerometru, składa się z 8 bitów: 6 z nich jest wcześniej wspomnianym adresem rejestru, 1 bit informuje czy rejestr ma być odczytany, czy modyfikowany oraz jeden nieistotny bit. Na **rysunku 3** przedstawiono przebieg sygnałów interfejsu SPI przy odczytywaniu (a) oraz zapisywaniu (b) danych do rejestru.

Jako pierwszy jest przesyłany bit *R/W* określający rodzaj operacji. Kolejne 6 bitów określa cel operacji, po czym nadawany jest nieistotny bit (oznaczony jako X). Następnie, w przypadku zapisu, przesyłane jest 8 bitów, które zostaną wpisane do rejestru, bądź też akcelerometr zwraca zawartość rejestru, którego odczyt został zażądany.

Akcelerometr MMA7455L ma 31 użytecznych rejestrów, z których ważniejsze to:

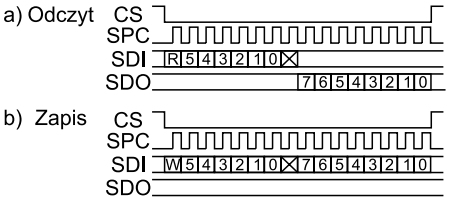
1. Rejestry zawierające wskazania akcelerometru w formacie 8-bitowym o adresach: *h06, h07, h08*, które są tylko do odczytu.
2. Rejestr *h0D* zawierający adres interfejsu I²C, a dokładniej najbardziej znaczący bit *I2CDIS* wyłączający ten interfejs (wyłączamy aby uniknąć błędów komunikacji przy użyciu interfejsu SPI).
3. Rejestr *h16* kontrolujący tryb, w którym działa akcelerometr. Jego zawartość przedstawiono w **tabeli 1**.

Przy komunikacji są istotne cztery bity najmniej znaczące. Pierwsze dwa, oznaczone jako *MODE*, określają tryb w jakim pracuje akcelerometr. Możliwe tryby to:

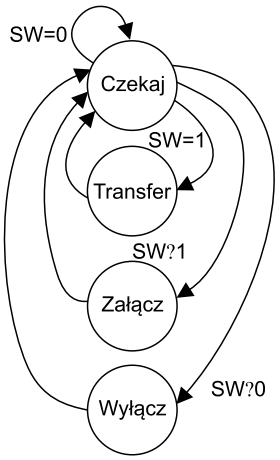
00 – tryb oszczędzania energii,
01 – tryb pomiarów,
10 – tryb detekcji poziomu,
11 – tryb detekcji impulsu.

Bity *GLVL[1]* oraz *GLVL[0]* pozwalają konfigurować zakres pomiarowy będący wielokrotnością g – przyspieszenia ziemskiego:

00 – zakres 8g,
10 – zakres 4g,



Rysunek 3. Wymiana danych



Rysunek 4. Grad automatu obsługującego komunikację

01 – zakres 2g.

Pozostałe bity w tym rejestrze umożliwiają przełączanie pomiędzy trybami SPI, zezwalają na przeprowadzanie testu prawidłowości działania akcelerometru oraz ustawiać wyprowadzanie informacji o odczytach dokonywanych za pomocą przerwań.

Po włączeniu zasilania akcelerometr jest w trybie oszczędzania energii, w którym nie są dokonywane żadne pomiary. Aktywne są tylko interfejsy komunikacyjne I²C i SPI. Aby załączyć akcelerometr, należy zmodyfikować zawartość rejestru *h16*. Najpierw jest przesyłana komenda dotycząca zamiaru modyfikacji rejestru oraz zawierająca adres celu, zakończona bitem bez znaczenia czyli 10101100.

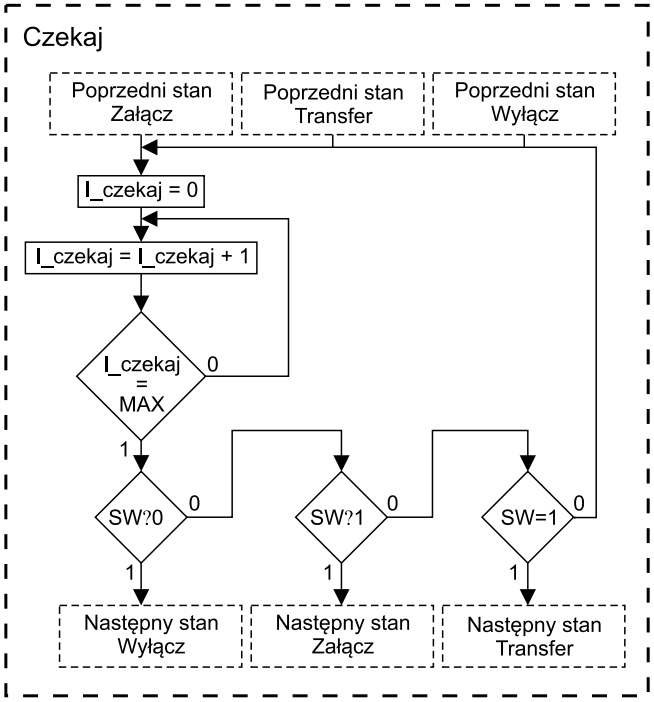
Kolejne 8 bitów jest nową zawartością modyfikowanego rejestru. Akcelerometr zostanie ustawiony w tryb pomiarów (01) w zakresie 2g (01): 00000101.

Następną czynnością jest wyłączenie interfejsu komunikacyjnego I²C. Gdy aktywne są obydwa interfejsy, za pomocą pinu *CS* wybiera się, który z nich jest aktualnie użytkowany. Jednak aby uniknąć błędów komunikacyjnych zaleca się wyłączenie interfejsu I²C podczas użytkowania SPI. Aby tego dokonać należy przesłać do akcelerometru ciąg 16 bitów, składający się z adresu rejestru oraz jego nowej zawartości (11101000 10011101).

Po tej czynności akcelerometr wykonuje pomiary przyśpieszenia w osiach: X, Y oraz Z. Odczyt wyników sprowadza się do przesłaniu żądania odczytania konkretnego rejestru i odebrania jego zawartości.

Aby niezawodnie włączać i wyłączać akcelerometr (przełączanie pomiędzy trybem uśpienia, a pomiarów) zastosujemy układ sekwencyjny (automat stanów, maszyna stanów) obsługujący komunikację między akcelerometrem, a sterownikiem w układzie programowalnym. Do jego zadań należą: włączanie i wyłączanie akcelerometru (przechodzenie z trybu oszczędzania energii do pomiarów i odwrotnie) oraz cykliczny odczyt wskazań. Schemat działania automatu zaprezentowano na **rysunku 4**.

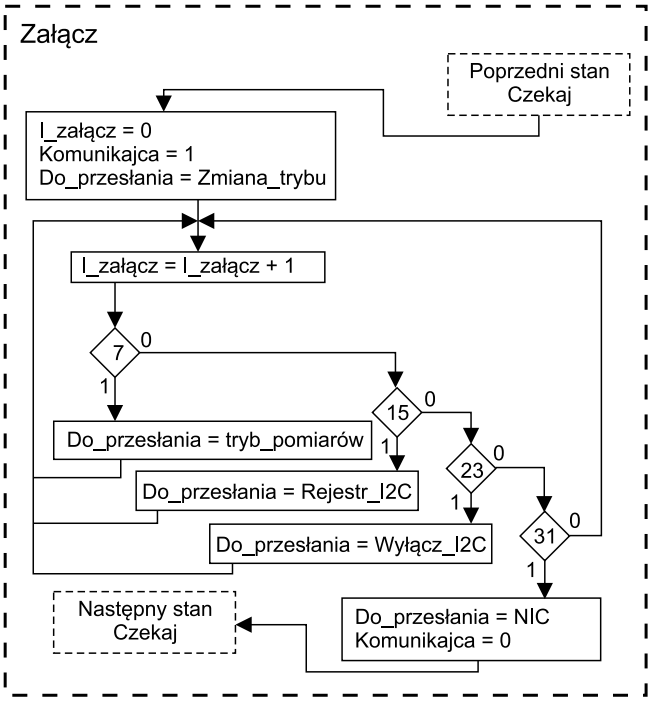
Automat ma 4 stany. Po załączeniu zasilania jest w stanie Czekaj. Gdy przełącznik SW zostanie ustawiony, automat przechodzi w stan Załącz, w którym są przesyłane komendy uruchamiające akcelerometr. Gdy przełącznik SW pozostaje w stanie 1, automat przechodzi pomiędzy stanami Czekaj i Transfer, cyklicznie odczytując wskazania akcelerometru. Gdy przełącznik zostanie wyłączony, automat przejdzie w stan Wyłącz w którym przesłane zostaną komendy przełączające akcelerometr w stan oszczędzania energii. Na-



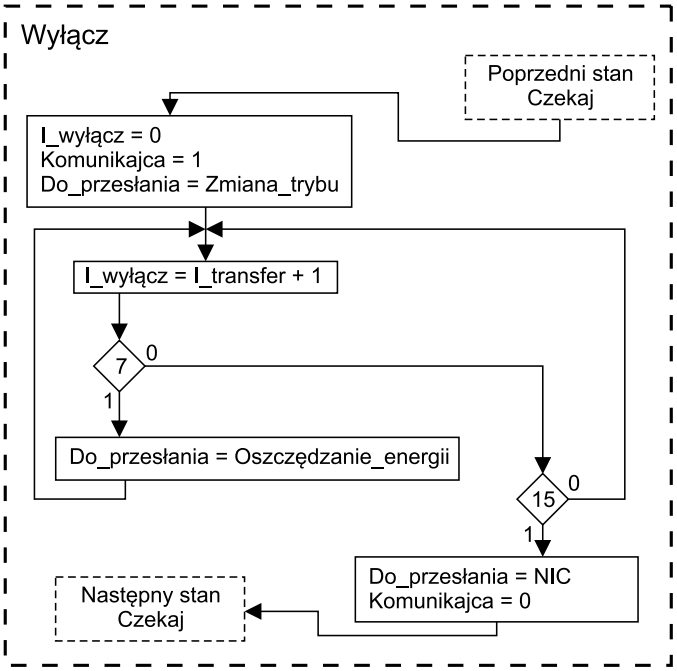
Rysunek 5. Schemat działania w stanie Czekaj

Tabela 1. Zawartość bitów rejestru h16

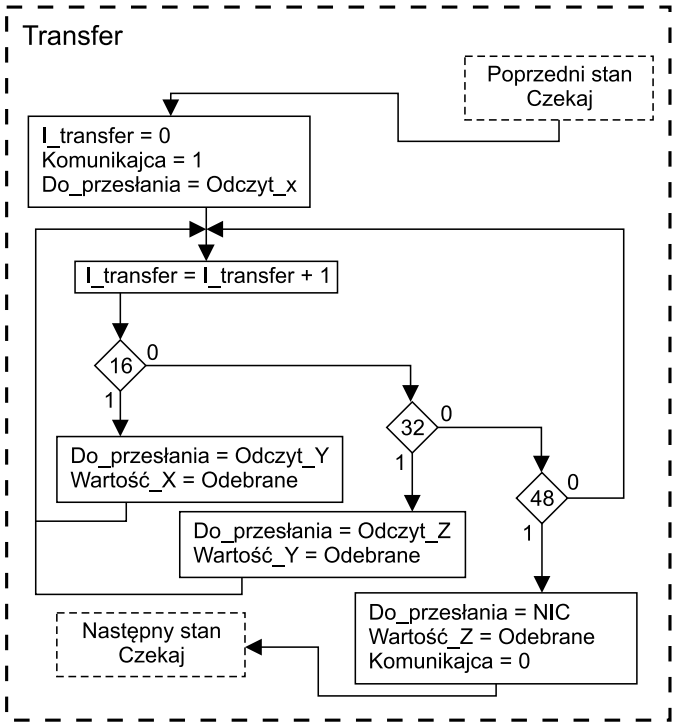
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
-	DRPD	SPI3W	STON	GLVL[1]	GLVL[0]	MODE[1]	MODE[0]



Rysunek 6. Schemat czynnościowy w stanie Załącz



Rysunek 7. Schemat czynnościowy w stanie *Wyłącz*



Rysunek 8. Schemat czynnościowy w stanie *Transfer*

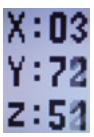
stępnie automat pozostanie w stanie Czekaj aż do zmiany stany przełączenia SW.

W stanie *Czekaj*, którego schemat pokazano na **rysunku 5**, jest realizowany odstęp czasu między wysyłaniem kolejnych komend do akcelerometru, gdyż potrzebuje on czasu, aby przejść z trybu oszczędzania energii do trybu pomiarów. Ponadto, pomiary są wykonywane z określoną częstotliwością, dlatego próby zbyt częstego odczytu rejestrów, gdy akcelerometr umieszcza w nich wyniki pomiarów, będą powodować błędy. Gdy rejestry zawierające ostatnio dokonane pomiary zostaną odczytane, ich zawartość jest zerowana, a więc próba zbyt wczesnego odczytu skutkuje błędnym wynikiem. Stan *Czekaj* jest początkowym po załączeniu zasilania. W tym stanie jest inkrementowany jest licznik *l_czekaj* i gdy osiągnie wartość maksymalną, jest sprawdzany stan przełącznika *SW*. W zależności od jego stanu czy zmiany, następuje odpowiednia akcja. Jeżeli *SW* pozostaje wyłączony, to jest zerowany licznik *l_czekaj* i odliczanie rozpoczyna się od początku.

Gdy przełącznik *SW* zostanie załączony, automat przejdzie do stanu *Załącz*. W tym stanie są przesyłane komendy załączające akcelerometr i ustawiające go w żądanym trybie. Dla uniknięcia błędów komunikacyjnych jest wyłączany interfejs I²C. Po przejściu do tego stanu sygnał *Komunikacja* jest ustawiany, co powoduje rozpoczęcie przesyłania sygnału zegarowego do akcelerometru. Jednocześnie w rejestrze przesuwającym modułu SPI umieszczana jest pierwsza komenda modyfikacji rejestru kontrolującego tryb działania akcelerometru. Wyzerowany licznik *l_załacz* jest inkrementowany. Gdy licznik osiągnie wartość 7, co będzie oznaczać, że licząc od 0 zliczył już 8 taktów zegara, w rejestrze przesuwającym jest umieszczone słowo konfigurujące akcelerometr. Jest on przełączany do trybu pomiarów o zakresie 2g. Następnie wysyłane są komendy wyłączające interfejs I²C, aby po zliczeniu 32 taktów zakończyć połączenie i przejść do stanu *Czekaj*.

W stanie *Wyłącz* czynności są analogiczne jak w stanie *Załącz*. Różnią się tyme, że do rejestru kontrolującego tryb działania jest wysyłany ciąg bitów powodujący przejście w tryb oszczędzania energii.

Na **rysunku 8** przedstawiono schemat działań, które są wykonywane w stanie *Transfer*. Do akcelerometru są wysyłane komendy powodujące odczyt zawartości rejestrów zawierających wskazania akcelerometru. Podobnie jak w stanach *Załącz* oraz *Wyłącz* najpierw następuje zerowanie licznika oraz inicjalizacja komunikacji. Pierwsza jest przesyłana komenda odczytu rejestru zawierającego wartość przyśpieszenia w osi X. Ta komenda, umieszczona w rejestrze przesuwającym, gdy licznik *l_transfer* był wyzerowany spowoduje, że odpowiedź akcelerometru będzie dostępna w chwili, gdy licznik będzie miał wartość 16. Jednocześnie, podczas przepisywania zawartości rejestru przesuwającego *Odebrane* do rejestru *Wartość_X*, rozpoczynane jest wysyłanie komendy odczytu rejestru z wynikiem dla osi Y.



Rysunek 9. Wskazania wyświetlane na ekranie monitora

Po odczytaniu wszystkich bitów przyśpieszenia w osi Y, wysyłana jest komenda odczytu wartości osi Z, a po ich odczytaniu komunikacja jest przerywana i automat przechodzi w stan *Czekaj*.

Listing 3. Fragment opisu automatu stanu w Verilogu

```
     . . .
always @ (posedge CLK_50MHZ)
   state_reg  <=  state_next;

always @ (negedge tick1MHz) begin
   state_next <= state_reg;
   case(state_reg)
     załącz: begin
       case(cmd_num)
       8'h00: begin
          komunikacja    <= 1'b1;
          Do_przesłania  <= {1'b1,6'h16,1'b0};
       end
       8'h07: Do_przesłania <= {4'b0000,2'b10,2'b01};

       8'h0f: Do_przesłania <= {1'b1,6'h0d,1'b0};

       8'h17: Do_przesłania <= 8'b10011101;

       8'h1f: begin
          Komunikacja <= 1'b0;
          state_next <= Czekaj;
          Do_przesłania <= 8'b0;
          now_on <= 1'b1;
          now_off <= 1'b0;
          cmd_num = 8'b0;
       end
       endcase
       cmd_num = (komunikacja) ? cmd_num + 1 :  8'b0 ;
     end
     . . .
     transfer: begin
       case(cmd_num)
       8'h00: begin
          Komunikacja         <= 1'b1;
          Do_przesłania       <= {1'b0,6'h06,1'b0};
          end
       8'h07: Do_przesłania  <= 8'h00;
       8'h10: begin
          Wartość_X<= Odebrane;

          Do_przesłania  <= {1'b0,6'h07,1'b0};

       end
       8'h17: Do_przesłania  <= 8'h00;
       8'h20: begin
          Wartość_Y <= Odebrane;

          Do_przesłania  <= {1'b0,6'h08,1'b0};
       end
       8'h27: Do_przesłania  <= 8'h00;
       8'h30: Wartość_Z <= Odebrane;

       8'h37: Do_przesłania  <= 8'h00;
       8'h3f: begin
          cmd_num = 8'h00;
          Komunikacja <= 1'b0;
          state_next <= Czekaj;
       end
       endcase
       cmd_num = (Komunikacja) ? cmd_num + 1 : 8'b0 ;
     end
     . . .

SPI spi1(
     .clk (tick1MHz),
     .init (Komunikacja),
     .cmd (Do_przesłania),
     .SDO (SDO),
     .CS (CS),
     .SDI (SDI),
     .SPC (SPC),
     .data (Odebrane)
    );
```

Fragment opisu implementacji automatu stanu zarządzającego komunikacją zamieszczono na **listingu 3**. Jest on zgodny ze schematami zaprezentowanymi wcześniej. Różnicą jest tylko to, że w każdym ze stanów jest używany ten sam licznik sterujący umieszczaniem w rejestrze *Do_przesłania* odpowiedniej komendy. Zastosowanie jednego rejestru o nazwie *cmd_num* pozwoliło na skrócenie długości kodu i jego ujednolicenie. Nie przeszkadza to, gdyż przy przejściach z jednego stanu do drugiego licznikjest zerowany.

Po zaimplementowaniu w FPGA modułu obsługi połączenia z akcelerometrem, który umożliwia wysyłanie poleceń i odbieranie wyników pomiaru mamy zapisywane w rejestrach układu programowalnego: *Wartość_X*, *Wartość_Y*, *Wartość_Z*. Aby zobaczyć te wyniki, wyświetlmy je na ekranie monitora. Zrzut ekranu wyświetlającego wyniki pomiaru przedstawiono na **rysunku 9**, a realizację modułu obsługi wyświetlania na ekranie monitora VGA zaprezentujemy w następnym artykule.

Chrystian Ruminowicz
Piotr Pietrzyk