Projektowanie PLD/FPGA z zestawem Lattice MachXO2 Pico Development Kit (2)

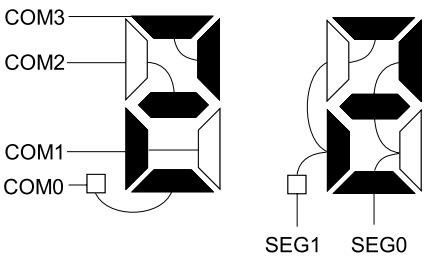
Pod koniec pierwszego kwartału bieżącego roku Lattice wprowadził do seryjnej produkcji nową serię układów CPLD o nazwie MachXO2. Produkowane w procesie 65 nm, bazujące na technologii Flash układy MachXO2, w porównaniu ze swoim poprzednikiem – rodziną MachXO, charakteryzują się 3-krotnym zwiększeniem zasobów logicznych, 10-krotnym zwiększeniem pojemności wbudowanej pamięci RAM, więcej niż 100-krotnym zmniejszeniem poboru mocy (statyczny pobór mocy wynosi zaledwie 19 mikrowatów) oraz 30% spadkiem ceny. Oznacza to, że układy MachXO2, o zasobach logicznych sięgających 7000 tablic LUT, mogą swobodnie konkurować z „mniejszymi" układami FPGA rodzin Actel IGLOO, Altera Cyclone czy Xilinx Spartan. Wraz z nowymi układami Lattice udostępnił również zestaw ewaluacyjny MachXO2 Pico, umożliwiający praktyczne poznanie najważniejszych właściwości układów rodziny MachXO2. Celem niniejszego kursu jest zilustrowanie sposobu projektowania stosunkowo nieskomplikowanych systemów cyfrowych, opartych na nowoczesnych układach programowalnych, z wykorzystaniem zestawu MachXO2 Pico.

Dodatkowe materiały na CD/FTP:
ftp://ep.com.pl, user: 17692, pass: 4yv87ftn
- pierwsza część kursu
- **Plik projektu oraz pełne listingi są dostępne na serwerze FTP**

Kontynuujemy pierwszą część kursu rozpoczętą w listopadowym numerze Elektroniki Praktycznej. W tej części pokażemy projekt sterownika wyświetlacza LCD, opiszemy sposób wykorzystania sprzętowego bloku funkcji wbudowanych EFB – w szczególności kontrolera magistrali I²C.

Sterownik wyświetlacza LCD

Zasadniczym elementem zestawu MachXO2 Pico, służącym do wizualizacji danych, jest 4-cyfrowy, 7-segmentowy wyświetlacz LCD. Wyświetlacz ten nie zawiera specjalizowanego sterownika, tak jak ma to miejsce w przypadku np. popularnych wyświetlaczy alfanumerycznych 2×16 znaków. Aby skorzystać z wyświetlacza należy taki sterownik zaprojektować. Pokażemy teraz jak to zrobić.



Rysunek 17. Połączenia segmentów w wyświetlaczu LCD o współczynniku multipleksu 1/4

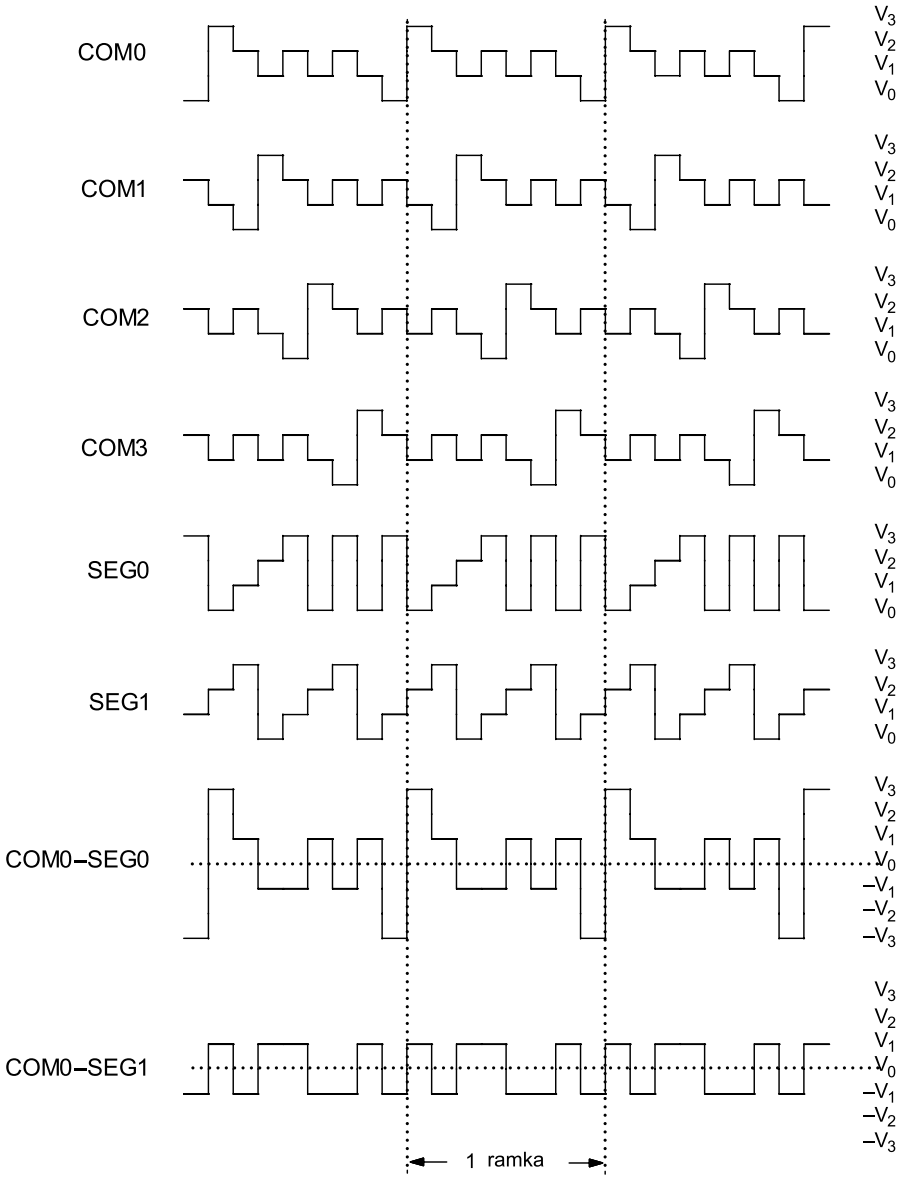
Zasadniczy problem dotyczący sterowania wyświetlaczami LCD polega na tym, że nie mogą być one zasilane napięciem stałym. Doprowadzenie do elektrod wyświetlacza napięcia stałego może spowodować degradację molekuł ciekłego kryształu i w konsekwencji brak odpowiedniej reakcji na pojawiające się pole elektryczne. Dodatkowo, w przeciwieństwie do 7-segmentowych wyświetlaczy LED, segmenty w ramach pojedynczej cyfry wyświetlacza LCD połączone są w matrycę o określonej konfiguracji (**rysunek 17**). Wymusza to multipleksowy sposób sterowania wyświetlaczem, jednak tak aby wartość średnia napięcia pomiędzy elektrodą wspólną (COM) i danym segmentem (SEG) zawsze była równa 0 (brak składowej stałej). Z kolei wartość skuteczna tego napięcia dla załączonego segmentu musi być większa od pewnej wartości progowej, a wartość skuteczna dla segmentu wyłączonego musi być mniejsza od tej wartości. Iloraz tych dwóch wartości skutecznych definiuje tzw. współczynnik dyskryminacji, który określa poziom kontrastu wyświetlacza.

Konsekwencją wspomnianych wyżej zależności jest to, że przebiegi napięcia na poszczególnych elektrodach wyświetlacza muszą mieć charakter schodkowy – wielopoziomowy. Ilość niezbędnych poziomów napięcia zależna jest od tzw. współczynnika multipleksu (*mux ratio*), który z kolei jest odwrotnością liczby wspólnych elektrod wyświetlacza. Dla przykładu, w przypadku wyświetlacza LCD zastosowanego w zestawie Lattice MachXO2 Pico, o współczynniku multipleksu równym 1/4, niezbędne są trzy poziomy napięcia (nie licząc poziomu odniesienia 0 V), wynoszące odpowiednio 1/3, 2/3 oraz 3/3 napięcia zasilania 3,3 V.

Czas trwania sekwencji przebiegów na elektrodach wyświetlacza, niezbędnej do wyświetlenia kompletnego znaku określany jest jako pojedyncza ramka. Na **rysunku 18** przedstawiono w sposób symboliczny postać pojedynczej ramki dla segmentu załączonego i wyłączonego. Poszczególne cyfry oznaczają poziom napięcia w woltach (dla uproszczenia przyjęto, że napięcie maksymalne to 3 V). Można łatwo wykazać, że dla takiej postaci ramki napięcie pomiędzy dowolną elektrodą wspólną i segmentem, nie zawiera składowej stałej,

COM0	0	3	2	1	2	1	2	1	
COM1	2	1	0	3	2	1	2	1	
COM2	2	1	2	1	0	3	2	1	
COM3	2	1	2	1	2	1	0	3	
SEGx	3	0	3	0	3	0	3	0	wł.
	1	2	1	2	1	2	1	2	wył.

Rysunek 18. Napięcia występujące na elektrodach wyświetlacza dla pojedynczej ramki



Rysunek 19. Przebiegi napięcia na elektrodach wyświetlacza LCD

a współczynnik dyskryminacji jest tutaj równy 1.732 (pierwiastek z trzech).

Na **rysunku 19** przedstawiono przebiegi czasowe na poszczególnych elektrodach pojedynczej cyfry wyświetlacza LCD oraz napięcie występujące pomiędzy wybraną elektrodą wspólną i obydwiema elektrodami segmentowymi – zgodnie z ideą pokazaną na rys. 18. Bardziej szczegółowe informacje dotyczące podstaw działania i sposobów sterowania wyświetlaczy LCD można znaleźć np. w nocie aplikacyjnej Microchip AN658 (*LCD Fundamentals Using PIC16C92X Microcontrollers*).

Listing 2. Opis modułu generatora PWM

```
module LCD_PWM(input CLK, RST,
               input [1:0] VTG,
               output reg PWMV);
   reg [1:0] ST;
   always @(posedge CLK, negedge RST)
   if(~RST) begin ST<=0; PWMV<=0; end
   else
   case (ST)
      0: begin
         if(VTG==2'b00) PWMV<=1'b0;
         else PWMV<=1'b1;
         ST<=1;
         end
      1: begin
         if(VTG==2'b00|VTG==2'b01) PWMV<=1'b0;
         else PWMV<=1'b1;
         ST<=2;
         end
      2: begin
         if(VTG==2'b11) PWMV<=1'b1;
         else PWMV<=1'b0;
         ST<=0;
         end
       default: begin PWMV<=1'b0; ST<=0; end
   endcase
endmodule
```

Z punktu widzenia zestawu MachXO2 Pico sterowanie wyświetlaczem LCD sprowadza się do wytworzenia przebiegów pokazanych na rys. 19. Zasadniczy problem, który tutaj się pojawia to konieczność wytworzenia napięć pośrednich w stosunku do napięcia zasilania, tak aby uzyskać odpowiedni przebieg schodkowy. Typowe rozwiązanie, stosowane od dawna w przypadku wyświetlaczy LCD polega na zastosowaniu drabinki rezystancyjnej w celu uzyskania napięć pośrednich. Jednak sposób ten charakteryzuje się dość dużymi, niepotrzebnymi stratami mocy, odgrywającymi istotne znaczenie w przypadku aplikacji zasilanych bateryjnie. Innym, dość oczywistym rozwiązaniem, zwłaszcza gdy mamy do dyspozycji układy programowalne, jest zastosowanie generatora PWM. Tym bardziej, że płytka drukowana zestawu MachXO2 Pico zawiera proste filtry dolnoprzepustowe dołączone do końcówek wyświetlacza, konieczne w przypadku zastosowania takiego generatora.

Tabela 2. Opis portów modułu generatora PWM

Moduł: LCD_PWM			
Port	**Kierunek**	**Liczba bitów**	**Opis**
CLK	wejście	1	Sygnał zegarowy o częstotliwości PWM
RST	wejście	1	Sygnał zerujący. Aktywny poziom niski.
VTG	wejście	2	Zakodowana wartość napięcia (dla standardu LVTTL): "00" – 0V, "01" – 1V, "10" – 2V, "11" – 3V.
PWMV	wyjście	1	Przebieg wyjściowy PWM.

Na **listingu 2** przedstawiono kod źródłowy w języku Verilog modułu generatora PWM, w **tabeli 2** zawarto zaś opis portów tego modułu. Generator PWM jest trójstanowym automatem sekwencyjnym, który na podstawie kombinacji sygnałów na 2-bitowym wejściu VTG, generuje periodyczną sekwencję zer i jedynek, pojawiającą się na wyjściu PWMV. Relacja pomiędzy kombinacją stanów na wejściu VTG i sekwencją na wyjściu PWMV jest następująca: "00" -> "000…", "01" -> "100…", "10" -> "110…", "11" -> "111…".

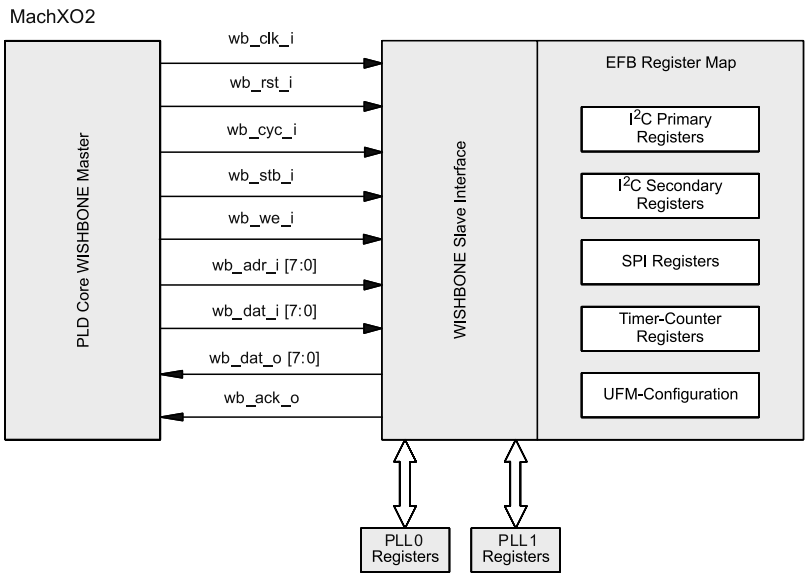
Listing 3 zawiera opis modułu sterownika wyświetlacza LCD. Zasadniczą część kodu tego modułu stanowi opis 8-stanowego automatu sekwencyjnego, generującego zakodowane wartości sygnału odpowiadające napięciu na poszczególnych elektrodach wyświetlacza, według schematu zamieszczonego na rys. 18. Wspomniane wartości sygnału odpowiadające napięciu stanowią następnie informację wejściową dla instancji modułu generatora PWM, sterujących bezpośrednio wyjściami dla elektrod wspólnych wyświetlacza oraz elektrod segmentowych. Sterownik wyświetlacza z list. 3 obsługuje 4-cyfrowy, 7-segmentowy wyświetlacz LCD. Dlatego ma 8-bitowe wejścia DIGIT1, …, DIGIT4, których

poszczególne bity odpowiadają za zapalenie odpowiednich segmentów wyświetlacza dla każdej z czterech cyfr. Moduł sterownika wyświetlacza wymaga również dostarczenia dwóch sygnałów zegarowych: jednego o częstotliwości ramki i drugiego o częstotliwości odpowiedniej dla generatorów PWM.

Tabela 3. Opis portów modułu sterownika wyświetlacza LCD

Moduł: LCD4Digit

Port	Kierunek	Liczba bitów	Opis
LCDFrameCLK	wejście	1	Sygnał zegarowy o częstotliwości ramki
LCDPWMCLK	wejście		Sygnał zegarowy o częstotliwości PWM.
RST	wejście	1	Sygnał zerujący. Aktywny poziom niski.
DIGITj	wejście	7	
LCD_COM0, LCD_COM1, LCD_COM2, LCD_COM3	wyjście	1	Wyjście dla elektrod wspólnych wyświetlacza.
LCD_SEG0_Di, LCD_SEG1_Di,	wyjście	1	Wyjście dla elektrod segmentowych i-tej cyfry 4-cyfrowego wyświetlacza LCD.



Rysunek 20. Magistrala interfejsu WISHBONE łącząca logikę programowalną z blokiem EFB

Tabela 4. Opis sygnałów interfejsu WISHBONE dla urządzenia podrzędnego - modułu EFB

Sygnał	Kierunek	Liczba bitów	Opis
wb_clk_i	wejście	1	Sygnał zegarowy interfejsu WISBONE (aktywne zbocze narastające).
wb_rst_i	wejście	1	Synchroniczny sygnał zerujący (aktywny poziom wysoki). Sygnał przerywa bieżącą komunikację ale nie wpływa na zawartość żadnych rejestrów.
wb_cyc_i	wejście	1	Sygnalizuje obecność obowiązujących danych na magistrali (aktywny poziom wysoki).
wb_stb_i	wejście	1	Sygnał strobujący (aktywny poziom wysoki). Oznacza, że bieżąca transakcja na magistrali kierowana jest do układu podrzędnego (slave). W odpowiedzi na ten sygnał blok EFB aktywuje sygnał potwierdzenia.
wb_we_i	wejście	1	Sygnał sterujący zapisem/odczytem. Poziom niski oznacza operację odczytu danych, a poziom wysoki – zapisu.
wb_adr_i	wejście	8	8-bitowy adres służący do wyboru określonego rejestru modułu EFB.
wb_dat_i	wejście	8	8-bitowe wejście danych zapisywanych do wybranego rejestru modułu EFB.
wb_dat_o	wyjście	8	8-bitwowe wyjście danych odczytanych z wybranego rejestru modułu EFB.
wb_ack_o	wyjście	1	Sygnał potwierdzenia (aktywny poziom wysoki) pochodzący z modułu EFB. Oznacza zaakceptowanie żądania transferu skierowanego do modułu EFB.

Mając zaprojektowany sterownik wyświetlacza LCD możemy teraz go zastosować dla zestawu Lattice MachXO2 Pico. Jako przykład pokażemy aplikację wyświetlającą zawartość starszych 16 bitów, 32-bitowego licznika binarnego, taktowanego częstotliwością generowaną przez wewnętrzny oscylator układu PLD. Zawartość licznika będzie wyświetlana w kodzie heksadecymalnym. Odpowiedni kod źródłowy pokazano na **listingu 4**. Zasadnicza część kodu zamieszczonego na tym listingu, opatrzona odpowiednim komentarzem, zostanie dodatkowo umieszczona w oddzielnym pliku o nazwie *clk_rst_lcd.v*. Plik ten będziemy następnie wykorzystywać przy okazji innych projektów dla zestawu MachXO2 Pico.

Kod zawarty w pliku *clk_rst_lcd.v* odpowiada za wytworzenie głównego przebiegu taktującego o częstotliwości 2.08 MHz uzyskanego przy pomocy instancji wewnętrznego oscylatora (identycznie jak pokazano to na listingu 1, w części I kursu), wytworzenie przebiegów o częstotliwości ramki (około 500 Hz) oraz PWM (około 1.04 MHz) dla wyświetlacza LCD, wygenerowanie impulsu zerującego po włączeniu zasilania lub wciśnięciu przycisku umieszczonego na płytce drukowanej zestawu MachXO2 Pico oraz utworzenie instancji sterownika wyświetlacza LCD z list. 3. W pliku tym zdefiniowana jest też funkcja enkodera dla segmentów wyświetlacza LCD, dzięki której możliwe jest wyświetlanie symboli cyfr 0…9 a także znaków szesnastkowych A…F. Funkcja ta wykorzystywana jest w ostatniej sekcji kodu z list. 4, w przypisaniach ciągłych do zmiennych *lcd1*,…, *lcd4*, odpowiadającym cyfrom od 1 do 4 wyświetlacza LCD. Wywołanie funkcji występuje z parametrem będącym częściową selekcją odpowiednich bitów licznika binarnego *CNTR*.

Interfejs WISHBONE

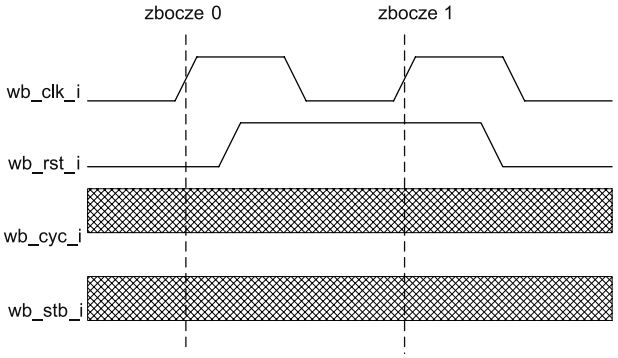
Jako kolejny projekt pokażemy sposób odczytu temperatury z termometru cyfrowego znajdującego się na płycie zestawu MachXO2. Termometr ten (*Burr-Brown* TMP101) wyposażony jest w interfejs I²C. Do odczytu temperatury wykorzystamy zatem blok EFB układu MachXO2 zawierający m. in. kontroler magistrali I²C. Komunikacja bloku EFB z konfigurowalną logiką układu PLD odbywa się za pośrednictwem interfejsu WISHBONE. Stąd najpierw krótko scharakteryzujemy ten interfejs.

WISHBONE jest standardowym interfejsem, opracowanym przez organizację *OpenCores*, służącym do komunikacji pomiędzy różnymi blokami funkcjonalnymi (wirtualnymi komponentami *IP Cores*, również pochodzącymi od różnych producentów) wewnątrz cyfrowych układów

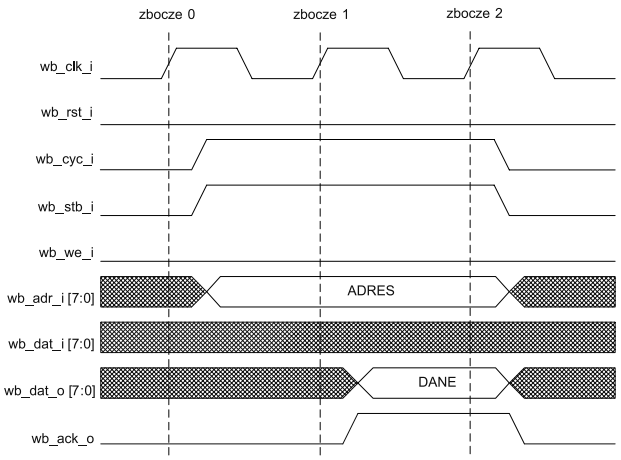
scalonych. Interfejs EFB WISHBONE zastosowany w układach MachXO2 implementuje klasyczną wersję standardu, jednak bez kilku cech, nieistotnych z punktu widzenia użytkownika układu PLD.

Na **rysunku 20** pokazano magistralę WISHBONE łączącą blok EFB z logiką programowalną układu PLD, a w **tabeli 4** zamieszczono opis poszczególnych sygnałów magistrali. Blok EFB dysponuje interfejsem WISHBONE typu *Slave*. Oznacza to, że na użytkowniku układu programowalnego spoczywa konieczność zaimplementowania za pomocą logiki programowalnej części zarządzającej transmisją danych (*Master*). Do tego celu można również wykorzystać „miękki" mikrokontroler LatticeMico8.

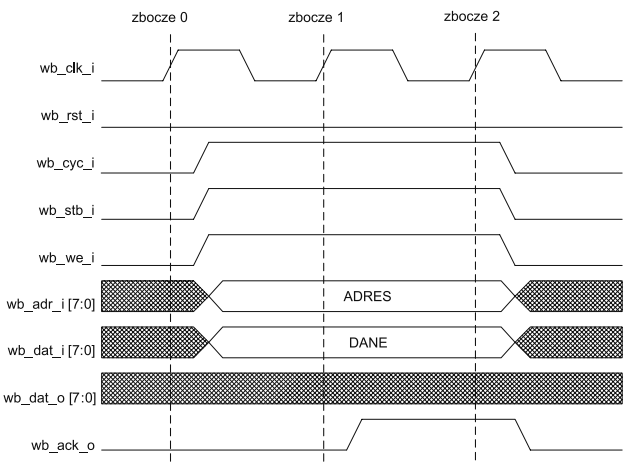
Na **rysunkach 21...23** pokazano przebiegi czasowe podczas wymiany danych za pośrednictwem interfejsu WISHBONE. Pierwszy z wymienionych rysunków przedstawia zerowanie interfejsu. Zerowanie nie ma wpływu na zawartość jakichkolwiek rejestrów, za pomocą których następuje wymiana danych, lecz przerywa jedynie bieżącą transakcję na magistrali. Aktywny poziom na wejściu *clk_rst_i* może trwać dowolnie długi czas.



Rysunek 21. Przebiegi czasowe podczas zerowania interfejsu WISHBONEv



Rysunek 22. Przebiegi czasowe podczas odczytu danych poprzez interfejs WISHBONE



Rysunek 23. Przebiegi czasowe podczas zapisu danych za pośrednictwem interfejsu WISHBONE

Odczyt danych z wybranego rejestru poprzez interfejs WISHBONE polega na podaniu na magistralę *wb_adr_i* adresu tego rejestru i aktywacji sygnałów *wb_cyc_i* oraz *wb_stb_i*. W odpowiedzi układ podrzędny udostępnia na magistrali *wb_dat_o* zawartość oczekiwanego rejestru i jednocześnie aktywuje sygnał *wb_ack_o* potwierdzający ten fakt. Podobnie wygląda procedura zapisu danych. Z tym, że oprócz adresu rejestru do którego dane będą zapisane, należy również na magistralę *wb_dat_i* dostarczyć wartość do zapisu, i wraz z sygnałami *wb_cyc_i* oraz *wb_stb_i* aktywować także sygnał *wb_we_i*. Zapis zostanie potwierdzony sygnałem *wb_ack_o*. Transmisja danych poprzez interfejs WISHBONE odbywa się synchronicznie – stan magistrali sprawdzany jest w czasie narastającego zbocza sygnału taktującego *wb_clk_i*.

Blok sprzętowych funkcji wbudowanych EFB układu MachXO2 zawiera zbiór rejestrów pozwalających, za pośrednictwem interfejsu WISHBONE, na sterowanie działaniem bloku. Każdy z modułów bloku EFB pełniący określoną funkcję, ma dedykowany 8-bitowy rejestr danych oraz rejestr sterujący. Ogólną mapę rejestrów bloku EFB zawiera **tabeli 5**. Bardziej szczegółowa mapa rejestrów dla kontrolera I²C zamieszczona jest w **tabeli 6**.

Tabela 5. Mapa adresów bloku EFB

Adres (Hex)	Funkcja
0x00-0x1F	PLL0
0x20-0x3F	PLL1
0x40-0x49	I²C podstawowy
0x4A-0x53	I²C dodatkowy
0x54-0x5D	SPI
0x5E-0x6F	Licznik/Czasomierz
0x70-0x75	UFM/Konfiguracja
0x76-0x77	Źródło przerwań bloku EFB

Odczyt temperatury z termometru cyfrowego

Płyta zestawu MachXO2 Pico zawiera cyfrowy czujnik temperatury TMP101 (*Burr-Brown*) z interfejsem I²C. Dokładność pomiaru temperatury tego czujnika to ±2 stopnie Celsjusza w przedziale temperatur od –25 do +85 stopni, a jego rozdzielczość wynosi od 9 do 12 bitów, zależnie od ustawień użytkownika. Na płycie zestawu ewaluacyjnego MachXO2 Pico czujnik zamontowany jest pomiędzy układem obsługi interfejsu USB FT2232H a stabilizatorem LDO 3.3 V (NCP1117) i dokonuje pomiaru temperatury powierzchni tej części płyty.

W celu dokonania odczytu temperatury z czujnika wykorzystamy oczywiście kontroler I²C będący na wyposażeniu bloku EFB układu MachXO2. Ogólny sposób obsługi tego kontrolera za pośrednictwem rejestrów interfejsu WISHBONE pokazany jest na **rysunku 24**. W pierwszym kroku należy w rejestrze TXDR (WISHBONE I2C_1_TXDR lub I2C_2_TXDR), przechowującym dane do zapisu, umieścić adres urządzenia I²C slave, jednocześnie zerując najmłodszy bit rejestru TXDR (zapis na magistrali I²C). Zainicjowanie transmisji na magistrali I²C następuje po przesłaniu interfejsem WISHBONE polecenia o kodzie 90h (wygenerowanie sekwencji START na magistrali I²C). Następnie należy zaczekać na zakończenie realizacji bieżącego polecenia sprawdzając stan bitu TRRDY w słowie statusu kontrolera I²C. W tym celu za pomocą interfejsu WISHBONE należy zażądać odczytu rejestru I2C_1_SR (I2C_2_SR) i sprawdzić czy bit 2 (o wadze 4) jest zapalony. Jeżeli tak, to na magistralę I²C można przesłać teraz dane, wpisując do rejestru WISHBONE TXDR 8-bitową wartość transmitowanej danej i przesyłając polecenie (zawartość rejestru WISHBONE CMDR) o kodzie 10h. Następnie znowu należy zaczekać na ustawienie bitu TRRDY w słowie statusu kontrolera I²C. Jeżeli teraz chcemy odczytać dane

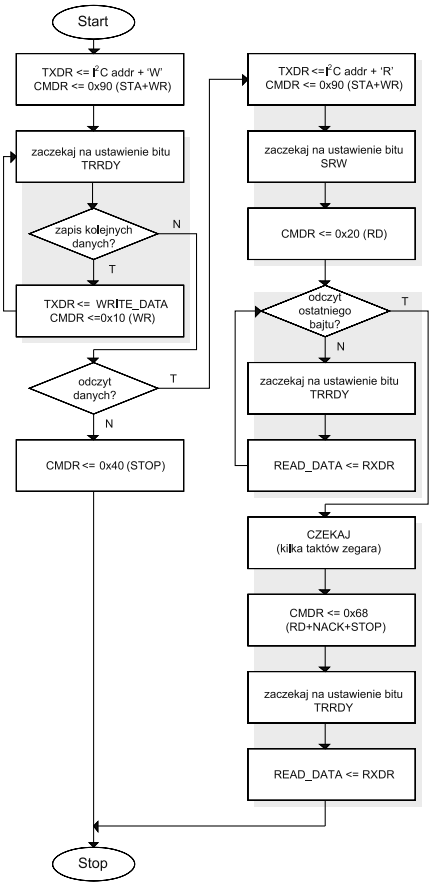
Tabela 6. Rejestry kontrolera I²C

kontroler I²C podstawowy		kontroler I²C dodatkowy		Funkcja	Dostęp
Nazwa	Adres	Nazwa	Adres		
I2C_1_CR	0x40	I2C_2_CR	0x4A	Sterowanie	Odczyt/Zapis
I2C_1_CMDR	0x41	I2C_2_CMDR	0x4B	Rejestr polecenia	Odczyt/Zapis
I2C_1_BR0	0x42	I2C_2_BR0	0x4C	Preskaler	Odczyt/Zapis
I2C_1_BR1	0x43	I2C_2_BR1	0x4D	Preskaler	Odczyt/Zapis
I2C_1_TXDR	0x44	I2C_2_TXDR	0x4E	Dane wysyłane	Zapis
I2C_1_SR	0x45	I2C_2_SR	0x4F	Status	Odczyt
I2C_1_GCDR	0x46	I2C_2_GCDR	0x50	Wywołanie ogólne (General Call)	Odczyt
I2C_1_RXDR	0x47	I2C_2_RXDR	0x51	Dane odebrane	Odczyt
I2C_1_IRQ	0x48	I2C_2_IRQ	0x52	Żądanie przerwania	Odczyt/Zapis
I2C_1_IRQEN	0x49	I2C_2_IRQEN	0x53	Zezwolenie na przerwanie	Odczyt/Zapis

z magistrali I²C wówczas należy ponownie zainicjować transmisję na magistrali poprzez wpisanie do rejestru TXDR adresu urządzenia podrzędnego z ustawionym najmłodszym bitem i przesłanie polecenia o kodzie 90h. Ustawienie najmłodszego bitu rejestru TXDR przy jednoczesnej wartości rejestru CMDR wynoszącej 90h oznacza przestawienie kontrolera I²C w stan odbioru. Potwierdzenie tego faktu sygnalizowane jest ustawieniem bitu SRW (bit 4, waga 16) w rejestrze statusu. Przed przesłaniem kolejnych poleceń interfejsem WISHBONE należy w tym momencie zaczekać na ustawienie tego bitu. Zainicjowanie odczytu pojedynczego bajtu z magistrali I²C inicjowane jest poleceniem o kodzie 20h. Jeżeli zamierzamy odczytać więcej bajtów wówczas, po przesłaniu polecenia o kodzie 20h, należy odczekać na ustawienie bitu TRRDY i wówczas rejestr RXDR interfejsu WISHBONE zawierał będzie odczytaną daną. Jeżeli zamierzamy odczytać ostatni bajt z magistrali I²C wówczas, po wysłaniu polecenia odczytu (20h), należy odczekać określoną liczbę taktów zegara, następnie przesłać polecenie o kodzie 68h (odczyt bez bitu potwierdzenia ACK, z jednoczesną sekwencją STOP na magistrali I²C), zaczekać na ustawnie bitu TRRDY i dopiero wówczas rejestr RXDR będzie zawierał ostatni odczytany bajt. Liczba taktów zegara określająca czas oczekiwania po przesłaniu polecania 20h, podczas odczytu ostatniego bajtu, zależy od liczby odczytanych bajtów. Jeżeli odczytywaliśmy tylko jeden bajt, wówczas wspomniana liczba taktów zegara *wb_clk_i* powinna być większa od 2 i mniejsza od 7. W przypadku odczytu dwóch bajtów liczba ta powinna być większa od 0 i mniejsza od 7. W pozostałych przypadkach liczba taktów oczekiwania powinna być większa od 0 (jeden takt lub więcej).

Na listingu 5 przedstawiono kod modułu głównego projektu realizującego odczyt temperatury z czujnika I²C i prezentację tej wartości na ekranie wyświetlacza LCD. Wykorzystano tutaj dodatkowy kontroler I²C, którego instancję utworzono przy pomocy aplikacji IPexpress. Zasadniczą część kodu stanowią opisy dwóch automatów sekwencyjnych: jednego odpowiedzialnego za komunikację z interfejsem WISHBONE, oraz drugiego realizującego ciąg czynności podobnych do przedstawionych za pomocą algorytmu z rys. 24.

Pierwszy automat sekwencyjny generuje przebiegi czasowe zapisu i odczytu na magistrali WISHBONE, tak jak zilustrowano to na rys. 22 i rys. 23. Jest on uruchaminy z drugiego automatu sekwencyjnego poprzez aktywację odpowiednich sygnałów sterujących. Jeżeli drugi automat sekwencyjny ustawi w stan wysoki (przez minimum jeden takt zegara) sygnał *send_cmd1*, wówczas zawartość zmiennej *CMDR* zostanie przesłana do rejestru WISHBONE I2C_2_CMDR (por. tab. 6) lub rejestru o adresie zawartym w zmiennej *REGA* – zależnie od stanu sygnału *reg_sw*. Ustawienie w stan wysoki zmiennej *send_cmd2* spowoduje zapisanie w rejestrze WISHBONE I2C_2_TXDR wartości przechowywanej w zmiennej *TXDR* a następ-



Rysunek 24. Algorytm opisujący realizację zapisu/odczytu na magistrali I²C poprzez interfejs WISHBONE

nie wykonanie takiej samej czynności jak przy aktywacji sygnału *send_cmd1*. Z kolei ustawienie w stan wysoki zmiennej *wait_trdy* powoduje cykliczne odczytywanie rejestru statusu WISHBONE I2C_2_SR i sprawdzanie stanu bitu TRRDY. Jeżeli bit ten zostanie ustawiony wówczas automat sekwencyjny przez jeden takt zegara ustawi w stan wysoki zmienną *trdy_ok*. Analogiczna funkcja realizowana jest po uaktywnieniu zmiennej *wait_srw*, z tym, że wówczas sprawdzany jest bit SRW w rejestrze statusu i gdy jest on zapalony, przez jeden takt zegara ustawiany jest bit *srw_ok*. Aktywacja zmiennej *read_byte* powoduje odczytanie zawartości rejestru WISHBONE I2C_2_RXDR i przepisanie jej do zmiennej *wb_dat_rd*.

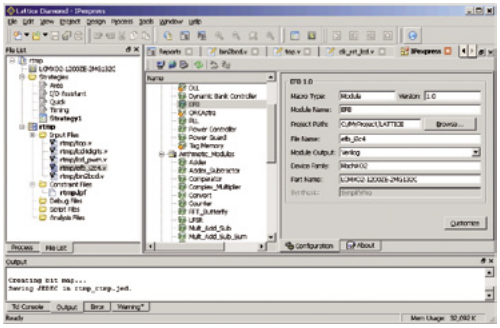
Drugi automat sekwencyjny realizuje sekwencję czynności podobną do tej opisanej algorytmem z rys. 24, dostosowaną do obsługi termometru TMP101. Na początku wykonywany jest zapis do rejestru konfiguracji czujnika temperatury TMP101, określający 12-bitową rozdzielczość pomiaru. Wymaga to najpierw nadania odpowiedniej wartości rejestrowi wskaźnikowemu termometru, tak aby wskazywał on na rejestr konfiguracji, następnie przesłania wartości rejestru konfiguracji. Po wznowieniu transmisji na magistrali I²C znowu należy zmienić wartość rejestru wskaźnikowego, tak aby pokazywał on teraz na rejestr zawierający wartość zmierzonej temperatury. Po ponownym wznowieniu transmisji na magistrali I²C można już odczytać dwa bajty zmierzonej temperatury. Wspomniane dwa bajty zawierają zarówno część całkowitą odczytanej temperatury, jak i część ułamkową. Część całkowita zapisana jest w naturalnym kodzie dwójkowym, stąd aby można ją było wyświetlić na wyświetlaczu LCD konieczne jest jej przetworzenie do postaci dwóch (lub trzech) cyfr w kodzie BCD. Operację tę przeprowadza dodatkowy moduł konwertera, którego kod pokazano na **listingu 6**. Uruchomienie procesu konwersji kodu realizuje drugi automat sekwencyjny po odczytaniu dwóch bajtów z czujnika I²C zawierających wartość zmierzonej temperatury. Część ułamkowa jest bezpośrednio kodowana za pomocą prostej funkcji kombinacyjnej. Po wykonaniu konwersji, automat sekwencyjny powraca do stanu w którym następuje ponowny odczyt temperatury z czujnika I²C.

Wspomniany moduł konwertera kodów z binarnego na BCD, pokazany na list. 6 (w **tabeli 7** zawarto opis parametrów i portów tego modułu) oparty jest na realizacji idei zawartej w nocie aplikacyjnej Xilinx XAPP029 (*Serial Code Conversion between BCD and Binary*). Zasadniczo ten sam moduł bardziej szczegółowo przedstawiany był w EP 1/2008 przy okazji prezentacji projektu uniwersalnego miernika częstotliwości, czasu i okresu na FPGA, stąd nie będziemy go tutaj omawiać.

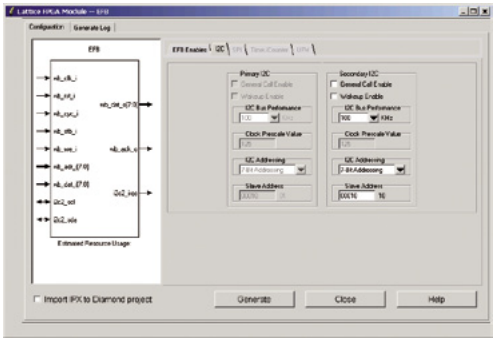
Tabela 7. Opis parametrów i portów modułu z listingu 6

Moduł: bin2bcd			
Parametr	**Opis**		
NO_BITS_IN	Liczba bitów słowa danych wejściowych (domyślnie 8)		
NO_BCD_DIGITS	Liczba cyfr BCD (domyślnie 3)		
BIT_CNT_WIDTH	Minimalna liczba bitów za pomocą której da się zapisać wartość będącą liczbą bitów słowa wejściowego (domyślnie 4)		
Port	**Kierunek**	**Liczba bitów**	**Opis**
clk	wejście	1	Sygnał taktujący
start	wejście	1	Sygnał inicjujący konwersję. Powinien być aktywny (w stanie wysokim) przez co najmniej jeden takt zegara jednak nie dłużej niż wynosi czas konwersji
done	wyjście	1	Stan wysoki na tym wyjściu oznacza zakończenie konwersji liczby i utrzymuje się aż do ponownego zainicjowania konwersji
data_in	wejście	NO_BITS_IN	Wejście danych konwertowanej liczby binarnej
data_bcd	wyjście	4*NO_BCD_DIGITS	Wyjście danych BCD

Drobnego wyjaśnienia wymaga jeszcze sposób użycia narzędzia generatora wirtualnych komponentów IPexpress, za pomocą którego utworzona została instancja kontrolera I²C bloku EFB. Aby skorzystać z tego narzędzia należy z menu *Tools* środowiska *Lattice Diamond* wybrać opcję *IPexpress* lub kliknąć odpowiedni przycisk na pasku narzędzi. W głównej części okna aplikacji *Lattice Diamond* zostanie utworzona nowa zakładka *IPexpress*, pokazana na **rysunku 25**. Teraz z rozwijanego drzewa modułów w oknie *Name* zakładki *IPexpress* należy wybrać blok *EFB*. W prawej części okna zakładki *IPexpress* wyświetlone zostaną ogólne parametry wybranego komponentu, z których użytkownik może dokonywać zmian w polach ścieżki projektu (*Project Path*), nazwy pliku (*File Name*) generowanego wirtualnego komponentu oraz języka HDL w którym generowany będzie wirtualny komponent (*Module Output*) – rys. 25. Po modyfikacji i uzupełnieniu wspomnianych pól należy kliknąć przycisk *Customize* (dostosuj). Wówczas otwarte zostanie kolejne okno (**rysunek 26**), w którym będzie można dokonać wyboru z jakich funkcji bloku EFB chcemy skorzystać (zakładka *EFB enable*) oraz podać odpowiednie parametry dotyczące wybranej funkcji (pozostałe zakładki). Zaznaczenie opcji *Import IPX to Diamond project*, znajdującej się w lewym dolnym rogu okna z rys. 25, spowoduje, że w folderze *Input Files* zakładki *File List* lewego okna aplikacji *Diamond*, pojawi się plik o nazwie takiej jak została podana w polu *File Name* w zakładce *IPexpress* (rys. 25) i rozszerzeniu IPX. Dwukrotne kliknięcie tego pliku spowoduje otwarcie okna modyfikacji funkcji i parametrów wybranego wirtualnego komponentu, w naszym przypadku bloku EFB – rys. 26. Wygenerowanie odpowiednich plików wirtualnego komponentu nastąpi po kliknięciu przycisku *Generate*. Wówczas w folderze, którego lokalizację wybraliśmy uzupełniając pole *Project Path*, utworzony zostanie plik z kodem wirtualnego komponentu, który należy dołączyć do bieżącego projektu, a także plik (z przyrostkiem *_tmpl*), zawierający szablon utworzenia instancji wirtualnego komponentu. Z zawartości tego ostatniego pliku skorzystaliśmy w przypadku kodu pokazanego na listingu 5.



Rysunek 25. Generator wirtualnych komponentów IPexpress aplikacji *Lattice Diamond*



Rysunek 26. Okno konfiguracji parametrów bloku EFB

Zbigniew Hajduk
hajduk@kia.prz.edu.pl