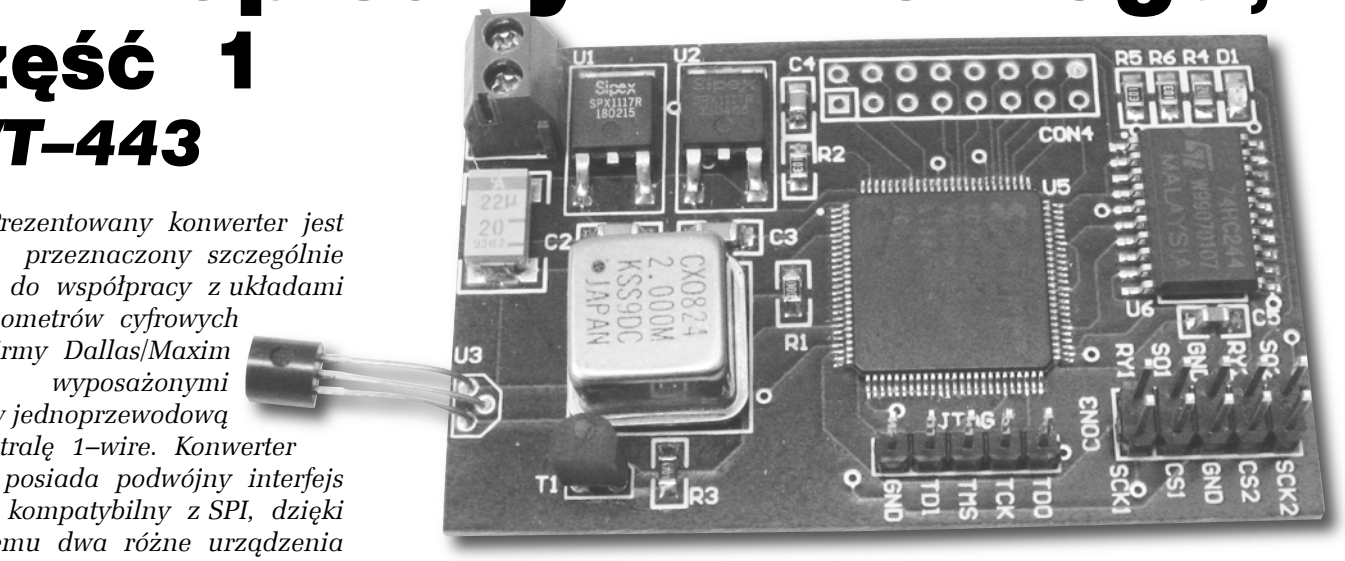
Konwerter 1–Wire –> SPI opisany w Verilogu, część 1

AVT–443

Prezentowany konwerter jest przeznaczony szczególnie do współpracy z układami termometrów cyfrowych firmy Dallas/Maxim wyposażonymi w jednoprzewodową magistralę 1–wire. Konwerter posiada podwójny interfejs kompatybilny z SPI, dzięki czemu dwa różne urządzenia (np. mikrokontrolery) mogą w dowolnym momencie odczytywać wartość temperatury zmierzonej przez termometr w sposób całkowicie od siebie niezależny. Konwerter zrealizowany jest w sposób sprzętowy z wykorzystaniem układów programowalnych i opisany w języku Verilog.

Rekomendacje:

projekt o dużych walorach użytkowych i jeszcze większych edukacyjnych. Przykład niezwykłych możliwości współczesnych układów programowalnych i języków opisu sprzętu (HDL).



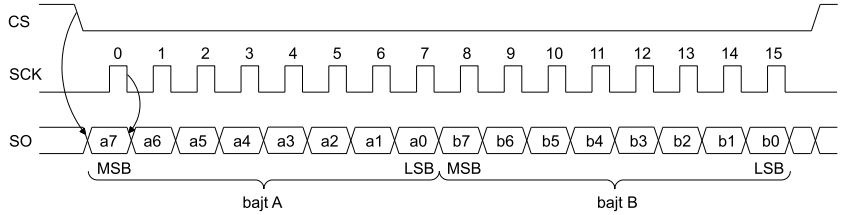
PODSTAWOWE PARAMETRY

- Płytka o wymiarach 61 x 38 mm
- Zasilanie 5...8 V DC
- Jednokierunkowa transmisja danych odczytanych z interfejsu 1-Wire do mikrokontrolera po SPI
- Kontrola CRC poprawności danych odebranych z 1-Wire
- Liczba kanałów 1-Wire: 1
- Liczba kanałów SPI: 1

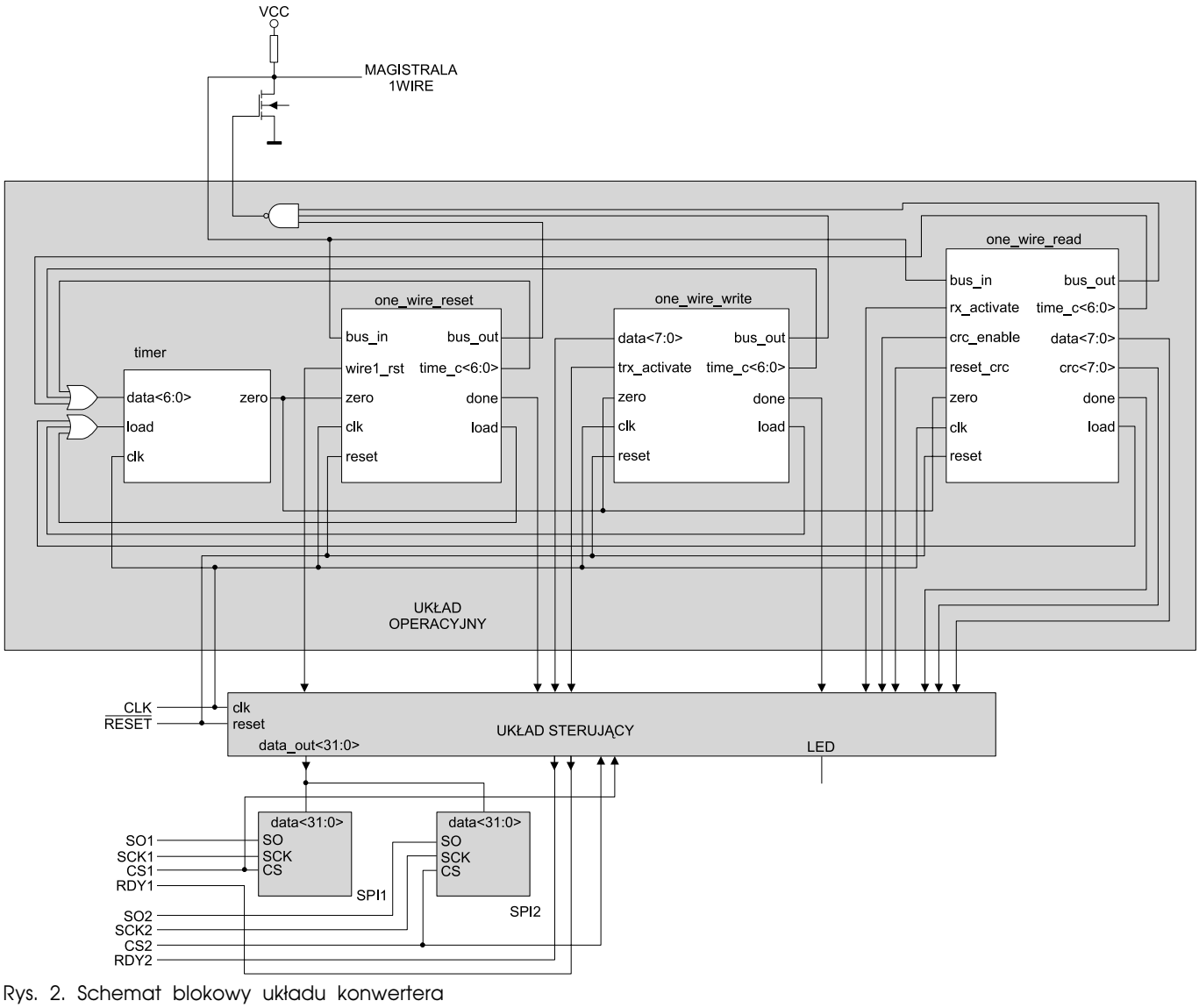
Czasami zachodzi potrzeba niezależnego odczytu temperatury z jednego czujnika przez dwa różne urządzenia. Odczyt temperatury może być dokonywane przez te urządzenia w dowolnej chwili, w sposób niezależny, w tym również w pełni równolegle. W warunkach przemysłowych może to być, na przykład sytuacja, gdy układ sterowania danym procesem (regulator) i układ monitorowania – nadzoru tego procesu wymagają tych samych danych pomiarowych (temperatura) pochodzących z jednego czujnika. Jeżeli stosowanym czujnikiem jest termometr cyfrowy pracujący z magistralą jednoprzewodową, wówczas rozwiązaniem problemu niezależnego odczytu temperatury przez dwa różne układy jest opisany tu konwerter.

Konwerter opisany w artykule umożliwia odczyt temperatury za pomocą synchronicznego szeregowego interfejsu SPI. Zaletą tego interfejsu jest jego niezwykle prosta obsługa programowa (za pomocą mikrokontrolerów), nie wymagająca precyzyjnego odmierzania czasu. Dlatego też konwerter może być również stosowany w sytuacji, gdy programowa obsługa magistrali 1–wire termometru jest ze względów czasowych (np. bardzo intensywnego zaangażowania mikroprocesora w inne zadania) utrudniona lub wręcz niemożliwa.

Prezentowany tutaj projekt układu konwertera posiada też pewien aspekt dydaktyczny. Pokazuje sposób współczesnego konstruowania sprzętu cyfrowego oparty na wykorzystaniu języków opisu sprzętu (Verilog), komputerowych narzędzi syntezy i układów programowalnych. O ile, dla sprawnego programisty, napisanie odpowiednich procedur obsługi magistrali jednoprzewodowej dla wybranego typu mikrokontrolera, nawet w języku maszynowym, nie powinno zająć więcej czasu niż 0,5...1 godziny (nie wspominając już o możliwości wykorzystania gotowych bibliotek), o tyle zaprojektowanie i przetestowanie analogicznie działającego sprzętu (wirtualnych komponentów) jest już procesem znacznie bardziej skomplikowanym



Rys. 1. Jednokierunkowa transmisja danych za pomocą interfejsu SPI



Rys. 2. Schemat blokowy układu konwertera

i zajmuje nieporównywalnie więcej czasu. Opisywana tu realizacja konwertera ilustruje również pewien problem związany ze żmudną konstrukcją wielostanowych automatów sterujących, zasygnalizowany przy okazji opisu „miękkiej" realizacji mikrokontrolera PicoBlaze (Elektronika Praktyczna 5 i 6/2005).

Chociaż została tu zaproponowana implementacja konwertera w układach programowalnych firmy Xilinx z rodziny CoolRunner–II, to wirtualny komponent opisujący konwerter (kod w języku Verilog) można – bez żadnych zmian – wykorzystać do implementacji w dowolnych innych układach PLD. Możliwe jest również wykorzystanie, opisanych dalej, poszczególnych wirtualnych komponentów odpowiedzialnych za obsługę magistrali jednoprzewodowej, czy też interfejsu SPI jako części składowej pewnego większego projektu.

Magistrala 1–Wire

Na początku przypomnimy podstawowe zagadnienia związane z magistralą jednoprzewodową. Magistrala ta, z definicji, wykorzystuje tylko pojedynczą linię danych. Każde urządzenie (nadrzędne lub podrzędne) dołączane jest do magistrali poprzez port trójstanowy lub typu otwarty dren. Umożliwia to danemu urządzeniu zwolnienie linii danych magistrali, podczas, gdy nie jest ono zaangażowane w proces transmisji danych, pozwalając tym samym na użycie magistrali przez inne układy. W czasie, gdy magistralą nie są transmitowane żadne dane, linia danych magistrali znajduje się w stanie wysokim wymuszonym przez zewnętrzny rezystor podciągający (zazwyczaj o wartości około 5 kΩ).

Każda transakcja (proces wymiany danych) na magistrali inicjowana jest przez urządzenie nadrzędne (*master*) poprzez wysłanie do urządzeń podrzędnych tzw. impulsu zerującego. Układ nadrzędny wymusza poziom niski na magistrali przez czas, co najmniej 480 µs, po czym zwalnia magistralę. Jeżeli do magistrali dołączone są jakieś urządzenia to powinny one, średnio po czasie 15...60 µs od momentu zwolnienia magistrali przez układ nadrzędny, ustawić magistralę w stan niski przez czas 60...240 µs. Jest to tzw. impuls obecności układu (układów) podrzędnego.

Transmisja poszczególnych bitów danych na magistrali 1–Wire odbywa się podczas trwania szczelin czasowych zapisu i odczytu. Czas trwania każdej szczeliny musi wynosić minimum 60 µs, a czas przedzielający dwie sąsiednie szczeliny nie może być krótszy niż 1 µs. Istnieją dwa typy szczelin czasowych zapisu: szczelina zapisu 1 (transmitowany przez układ nadrzędny bit ma wartość logiczną 1) oraz szczelina zapisu

0 (transmitowany bit ma wartość 0). Aby wygenerować szczelinę czasową zapisu 1 układ nadrzędny wymusza poziom niski na magistrali przez czas 1...15 µs, po czym zwalnia magistralę. W przypadku szczeliny czasowej zapisu 1 układ nadrzędny wymusza poziom niski na magistrali przez czas co najmniej 60 µs (jednak nie dłużej niż 120 µs). Układ podrzędny próbkuje stan magistrali w czasie 15...60 µs od momentu zainicjowania transmisji szczeliny czasowej przez układ nadrzędny (ustawienia magistrali w stan niski). Jeżeli podczas próbkowania magistrala znajduje się cały czas w stanie wysokim, oznacza to, że odczytany bit ma wartość 1, w przeciwnym przypadku 0.

Szczelina czasowa odczytu inicjowana jest przez układ nadrzędny poprzez wymuszenie poziomu niskiego na magistrali przez czas minimum 1 µs (jednak nie dłużej niż 15 µs) i zwolnienie magistrali. Po zainicjowaniu szczeliny czasowej odczytu, układ podrzędny rozpoczyna transmisję poszczególnych bitów. Bit o wartości 1 transmitowany jest poprzez pozostawienie magistrali w stanie wysokim, a bit o wartości 0 – poprzez wymuszenie na magistrali stanu niskiego. Podczas przesyłania bitu o wartości 0 układ podrzędny zwalnia magistralę na końcu czasu trwania szczeliny czasowej odczytu. Dane z układu podrzędnego mogą być odczytywane przez układ nadrzędny po czasie minimum 15 µs od wystąpienia opadającego zbocza na magistrali wyznaczającego początek szczeliny czasowej odczytu. Magistralą 1–wire bit najmniej znaczący (LSB) transmitowany jest jako pierwszy.

Interfejs SPI

Drugi z wykorzystywanych w projekcie interfejsów to szeregowy interfejs SPI (*Serial Peripheral Interface*). System wykorzystujący interfejs SPI składa się z urządzenia nadrzędnego (*master*), zarządzającego transmisją danych i jednego lub więcej urządzeń podrzędnych (*slave*). SPI jest czteroprzewodowym szeregowym interfejsem synchronicznym umożliwiającym w pełni dwukierunkową transmisję danych pomiędzy urządzeniem nadrzędnym i urządzeniami podrzędnymi. Transmisja danych przebiega z udziałem trzech linii interfejsu: szeregowego wejścia danych (MOSI), szeregowego wyjścia danych (MISO) oraz synchronizującego sygnału taktującego (SCK), generowanego przez układ nadrzędny. Czwarta linia interfejsu to linia CS wyboru danego układu podrzędnego.

Każda transmisja danych rozpoczyna się wraz z opadającym zboczem sygnału CS i kończy się wraz ze zboczem narastającym. Bit najbardziej znaczący (MSB) transmitowany jest jako pierwszy. Kolejne transmitowane bity danych przesuwane są na linię wyjściową (MISO) układu podrzędnego podczas opadającego zbocza sygnału SCK. Układ nadrzędny próbkuje stan tej linii podczas narastającego zbocza sygnału SCK.

List. 1. Opis behawioralny modułu układu czasowego

```
module timer(data,load,zero,clk);
//deklaracje portów we/wy
input [6:0] data;
input load,clk;
output zero;
reg zero,mload,z1;
reg[6:0] cntr;
reg [2:0] div;
wire cntr0;

assign cntr0=|cntr;
//cntr0=0, gdy wszystkie bity cntr wyzerowane

always @(posedge clk)
begin
 mload<=load;
 //zapamiętanie stanu wejścia wpisu load
 if(load&&~mload) //gdy narastające zbocze na load
 begin
    cntr=data; //wpis równoległy
    zero<=1'b0; div=1'b0; z1<=1'b0;
 end else
 begin
   div=div+1;
   if(div==3'd5) //dzielnik przez 5
   begin
    div=0;
    if(cntr0) cntr=cntr-1; //licznik liczy wstecz
    else
     if(!z1) begin zero<=1'b1; z1<=1'b1; end
     //ustawienie wyjścia zero, gdy licznik wyzerowany
   end
 end
   if(z1) zero<=1'b0;
   //wyzerowanie wyjścia zero (w następnym takcie)
end

endmodule
```

W prezentowanym projekcie wykorzystywana będzie jedynie uproszczona wersja interfejsu umożliwiająca jednokierunkową transmisję danych od układu podrzędnego (opisywanego tu konwertera) do układu nadrzędnego (np. mikrokontrolera). Przebiegi czasowe ilustrujące taką transmisję pokazano na **rys. 1**. Linie standardowo oznaczaną jako MISO (*Master Input Serial Output*) nazwano na rysunku w uproszczeniu SO. Oznaczenie to będzie stosowane w dalszej części tekstu.

Ze względu na to, że SPI jest interfejsem synchronicznym, jego obsługa programowa np. za pomocą mikrokontrolera jest bardzo prosta, niewymagająca precyzyjnego odmierzania czasu, jak to ma miejsce w przypadku obsługi magistrali jednoprzewodowej. Oczywiście odbywa się to kosztem liczby fizycznych linii niezbędnych do realizacji transmisji danych.

Sprzętowa obsługa protokołu 1–wire

Na **rys. 2** przedstawiono ogólny schemat blokowy układu konwertera 1–Wire –> SPI. Można tu wyróżnić klasyczny podział na część operacyjną – odpowiedzialną za przetwarzanie danych i część sterującą zarządzającą pracą układu operacyjnego.

W skład części operacyjnej wchodzą również dwa, nie wyróżnione na rys. 2, bloki interfejsów: SPI1 i SPI2. Zadania pełnione przez układ operacyjny, dla lepszej przejrzystości, zgrupowano w czterech niezależnych blokach funkcjonalnych. Pierwszy z nich to blok układu czasowego (*timer*) odpowiedzialny za odmierzanie zadanych interwałów czasowych. Kolejne bloki to: blok generowania impulsu zerującego (*one_wire_reset*), blok realizujący zapis 8 bitów danych (*one_wire_write*) i blok odczytu 8 kolejnych bitów z magistrali (*one_wire_read*). Zarówno układ sterujący, jak i poszczególne bloki układu operacyjnego (z wyjątkiem timera) zrealizowane są jako odpowiednio zaprojektowane automaty sekwencyjne.

Układ sterujący komunikuje się z układem operacyjnym poprzez szereg sygnałów zwanych mikrorozkazami (sygnały kierowane od układu sterującego do układu operacyjnego, np. sygnał żądania wysłania impulsu zerującego na magistralę 1–Wire) oraz sygnałami predykatowymi (sygnały kierowane od układu operacyjnego do układu sterującego, np. sygnał potwier-

dzający wysłanie impulsu zerującego i obecność urządzeń dołączonych do magistrali). Ponieważ wszystkie trzy modułu układu operacyjnego: blok generowania impulsu zerującego oraz bloki zapisu i odczytu danych, wymagają obecności układów precyzyjnie odmierzających czas, a w danej chwili może być aktywny (może przejmować kontrolę nad magistralą 1–wire) tylko jeden z wymienionych bloków (np. nie jest możliwy jednoczesny zapis i odczyt danych z magistrali), dlatego też zastosowano jeden układ czasowy z odpowiednio sterowanymi, poprzez bramki OR, wejściem danych i wejściem wpisu równoległego. Blok, który jest w danej chwili nieaktywny zeruje swoje wyjście danych do układu czasowego (stan tego wyjścia określa zadany czas do odmierzenia przez układ czasowy; wyjście to w poszczególnych blokach oznaczone jest jako *time_c*) oraz wyjście wpisu równoległego (oznaczone jako *load*). Wyjścia te są następnie dołączane do wejść trójstanowych bramek OR, których wyjścia sterują wejściem danych i wejściem wpisu równoległego układu czasowego. W ten sposób blok aktywny może bezkolizyjnie z innymi blokami wpisać odpowiednią stałą czasową (dane) do układu czasowego.

Podobnie zorganizowane jest sterowanie linią danych magistrali jednoprzewodowej. Wyjścia danych magistrali bloków generowania impulsu zerującego oraz zapisu i odczytu danych, oznaczone *bus_out*, połączone są z wejściami bramki NAND, której wyjście steruje bramką tranzystora MOSFET bezpośrednio dołączonego do linii danych magistrali 1–wire. Blok w danej chwili nieaktywny utrzymuje swoje wyjście *bus_out* w stanie wysokim. Wystąpienie stanu niskiego na jednym z wyjść *bus_out*, spowoduje pojawienie się stanu wysokiego na wyjściu bramki NAND i załączenie tranzystora MOSFET wymuszającego poziom niski na magistrali jednoprzewodowej.

W dalszej części zostaną omówione poszczególne bloki układu operacyjnego oraz układ sterujący. Podana zostanie również implementacja tych bloków, opisana w języku Verilog (wirtualne komponenty).

List. 2. Opis bloku generowania sygnału zerującego

```
module one_wire_reset(reset,clk,time_c,load,wire1_rst,zero,done,bus_out,bus_in);
input clk,reset,wire1_rst,zero,bus_in;
output done,bus_out,load;
output [6:0] time_c;
reg  done,wire1_out,ld;
reg [2:0] state;
reg [6:0] tc;

always @(posedge clk or negedge reset)
begin
  if(~reset) state=0; else
  begin
    case (state)
  3'd0: begin // Stan początkowy
    wire1_out=1'b1; done=1'b0; ld=1'b0; tc=7'd0;
    if(wire1_rst==1'b1) state=3'd1;
  end
       3'd1: begin
        tc=7'd120; wire1_out=0; ld=1'b1;
        if(zero) state=3'd2;
        //poziom niski na magistrali 1-wire przez 600us
  end
  3'd2: begin
     ld=1'b0; wire1_out=1; state=3'd3;
           //zwolnienie magistali
  end
  3'd3: begin
   tc=7'd18; ld=1'b1;
        if(zero) state=3'd4;
        //odczekanie 90us
  end
  3'd4: begin
    ld=1'b0; //próbkowanie stanu magistali
    if(bus_in==1'b0) state=3'd5; // impuls obecności OK
    else state=3'd7; //brak odpowiedzi 1-wire
  end
  3'd5: begin
    tc=7'd80; ld=1'b1;
    if(zero) state=3'd6;
          //odczekanie jeszcze 400us
        end
  3'd6: begin
    ld=1'b0; done=1; state=3'd0;
          //ustawienie wyjścia done
  end
  3'd7: begin
    tc=7'd127; ld=1'b1;
    if(zero) state=3'd0;
    //gdy brak odpowiedzi 1-wire odczekanie maksymalnego czasu i powrót do stanu
początkowego
  end
    endcase
  end
end
 //przypisania ciągłe związane z portami wyjściowymi modułu
 assign time_c=tc;
 assign load=ld;
 assign bus_out=wire1_out;
endmodule
```

Blok układu czasowego

Układ czasowy (timer) jest w rzeczywistości licznikiem z wpisem równoległym, liczącym wstecz. W przypadku prezentowanej implementacji, timer musi być taktowany sygnałem o częstotliwości 1 MHz. Zegar ten jest następnie dzielony przez 5, w wyniku czego licznik odmierza czas z rozdzielczością 5 µs, co jest wartością zupełnie wystarczającą do celów obsługi magistrali 1–Wire. Timer posiada 7–bitowe wejście danych, którego stan przepisywany jest do licznika w momencie wystąpienia narastającego zbocza na wejściu *load* wpisu równoległego. Osiągnięcie przez licznik stanu 0 sygnalizowane jest ustawieniem przez jeden okres zegara (1 µs) wyjścia *zero*.

Opis behawioralny w języku Verilog (wirtualny komponent) modułu układu czasowego pokazano na **list. 1**.

Blok generowania sygnału zerującego

Blok ten aktywowany jest w momencie, gdy układ sterujący ustawi w stan wysoki wejście *wire1_rst* bloku. Na **list. 2** pokazano opis w języku Verilog automatu sekwencyjnego realizującego funkcję bloku generowania impulsu zerującego.

Automat znajduje się w stanie początkowym 0 (zapis w języku Verilog *n*–bitowej liczby w kodzie dziesiętnym to *n'dw*, gdzie *d* jest symbolem oznaczającym podstawę dziesiętną,

WYKAZ ELEMENTÓW

Rezystory (0805)
R1: 0 Ω
R2: 47 kΩ
R3: 4,7 kΩ
R4: 470 Ω
R5, R6: 10 kΩ

Kondensatory
C1 (2220): 10 µF/16 V
C2, C3 (1206): 2.2 µF
C4, C5 (0805): 100 nF

Półprzewodniki
U1: SPX1117R–1.8 (TO252)
U2: SPX1117R–3.3 (TO252)
U3: DS1820 (TO92)
U4: generator 2 MHz
U5: XC2C256–VQ100
U6: 74AHC244 (SO–G20)
T1: BS108 (TO92)
D1: LED (1206)

Inne
CON1: ARK2 3.5 mm
CON2: goldpin 1x5
CON3: goldpin 2x5
CON4: goldpin 2x8

a *w* jest wartością liczby, np. *4'd10* oznacza liczbę 10 zapisaną na 4 bitach) do momentu, gdy wystąpi stan wysoki na wejściu *wire1_rst*. Wówczas stanem następnym jest stan 1. W tym stanie na 7–bitowym wyjściu *time_c* bloku pojawia się odpowiednia wartość stałej czasowej dla układu czasowego, ustawiane jest wyjście *load* wpisu równoległego do timera oraz ustawiany jest poziom wysoki na wyjściu *bus_out* (czyli na magistrali 1–Wire zostanie wymuszony poziom niski). Automat pozostanie w stanie 1 aż do momentu zakończenia odmierzania czasu przez timer, czyli do momentu, gdy na wejściu *zero* bloku pojawi się stan wysoki. Wówczas automat przejdzie do stanu 2, w którym zeruje wyjście wpisu równoległego *load*, zwalnia magistralę (wyjście *bus_out* w stanie niskim) i przechodzi do stanu 3. Po zwolnieniu magistrali, w kolejnych stanach, automat odczekuje jeszcze pewien czas (co realizowane jest poprzez komunikację z układem czasowym, w sposób analogiczny jak opisano to wyżej) i próbkuje stan magistrali. Jeżeli urządzenie dołączone do magistrali, sygnalizując swoją obecność, ustawi magistralę w stan niski, wówczas po odczekaniu dodatkowego czasu, automat ustawia wyjście *done* bloku, informując układ sterujący o pomyślnie zakończonej procedurze zerowania i wraca do stanu początkowego. W przypadku, gdy w czasie próbkowania na magistrali będzie utrzymywany poziom wysoki, wówczas automat po odczekaniu pewnego czasu również powraca do stanu początkowego, jednak nie ustawiając wyjścia *done*.

List. 3. Opis bloku odczytu danych

```
module one_wire_read(clk,reset,data,rx_activate,reset_crc,zero,
                     load,done,time_c,bus_out,bus_in,crc,crc_enable);
input clk,reset,rx_activate,zero,bus_in,reset_crc,crc_enable;
output done,bus_out,load;
output [7:0] crc;
output [7:0] data;
reg [7:0] data;
output [6:0] time_c;

reg  done,wire1_out,ld;
reg [2:0] state;
reg [6:0] tc;
reg [2:0] cntr;
reg [7:0] crc;
wire in_crc;

assign in_crc=bus_in^crc[0];
//pomocniczy sygnał do obliczania CRC

always@(posedge clk or negedge reset)
begin
   if(~reset)  state=0; else
   begin
     case(state) //realizacja automatu
  3'd0:
  begin //Stan początkowy
          done=0; tc=7'd0; ld=1'b0;
    cntr=0; wire1_out=1'b1;
    if(rx_activate) state=3'd1;
    if(reset_crc) crc=8'd0;
  end
  3'd1: begin
          wire1_out=1'b0;
    tc=7'd0; ld=1'b1;
    if(zero) state=3'd2;
          //wymuszenie poziomu nieskiego na 1-wire przez 5us
  end
  3'd2: begin
          wire1_out=1'b1; //zwolnienie magistrali
    state=3'd3; ld=1'b0;
  end
  3'd3: begin
          tc=7'd0; ld=1'b1;
    if(zero) state=3'd5; //Odczekanie 5us
  end
  3'd5: begin
    ld=1'b0; state=3'd6;
    data={bus_in,data[7:1]}; //próbkowanie magistrali - rejest przesuwny
    if(crc_enable)
          crc={in_crc,crc[7],crc[6],crc[5],crc[4]^in_crc,crc[3]^in_
crc,crc[2],crc[1]};
      //powyżej realizacja obliczenia wielomianu CRC
        end
  3'd6: begin
    tc=7'd17; ld=1'b1; //odczekanie 90us
    if(zero) state=3'd7;
  end
  3'd7: begin
    ld=1'b0;
    if(cntr!=3'd7)  begin
     cntr=cntr+1; state=3'd1; end
    else begin done=1'b1; state=3'd0; end
          //jeżeli odczytano juz 8 bajtów ustaw done i przejdź do stanu początko-
wego
  end
   endcase
   end
end
 assign time_c=tc;
 assign load=ld;
 assign bus_out=wire1_out;
endmodule
```

Blok zapisu danych

Blok zapisu danych aktywowany jest poprzez ustawienie w stan wysoki linii *trx_activate*. Na magistralę jednoprzewodową wysyłanych jest wówczas kolejno 8 szczelin czasowych zapisu poszczególnych bitów odpowiadających stanowi wejścia danych *data<7:0>* bloku. Zakończenie procesu zapisu całego bajtu (8 bitów) sygnalizowane jest ustawieniem wyjścia *done* bloku w stan wysoki przez jeden okres zegara. Sposób komunikacji bloku zapisu z modułem układu czasowego jest identyczny jak opisano to przy okazji bloku generowania impulsu zerującego.

Wirtualny komponent opisujący automat sekwencyjny bloku zapisu danych podobnie jak pozostałe wirtualne komponenty bloków odczytu danych, interfejsu SPI oraz układu sterującego zamieszczono w materiałach dodatkowych.

Blok odczytu danych

Blok odczytu danych został również zintegrowany ze sprzętowym układem obliczania wielomianu kontrolnego CRC (**list. 3**), typowego dla układów *Dallas/Maxim* pracujących z magistralą jednoprzewodową. Dlatego też blok ten posiada więcej sygnałów sterujących niż np. blok zapisu danych.

Sygnały związane z obsługą układu obliczania wartości wielomianu kontrolnego to: *reset_crc*, *crc_enable* oraz *crc<7:0>*. Poziom wysoki na pierwszym z wymienionych sygnałów powoduje wyzerowanie wewnętrznych rejestrów bloku przechowujących wartość wielomianu CRC. Sygnał *crc_enable* służy do określania czy poszczególne bity odczytywane z magistrali jednoprzewodowej mają być wykorzystane do obliczania wielomianu (poziom wysoki na tym wejściu) czy też nie (poziom niski). Ostatni sygnał *crc<7:0>* to 8–bitowe wyjście bloku, na którym dostępna jest bieżąca wartość wielomianu CRC.

Rozpoczęcie odczytu danych z magistrali (wygenerowanie kolejno 8 szczelin czasowych odczytu) inicjowane jest poprzez ustawienie w stan wysoki wejścia *rx_activate*. Zakończenie procesu odczytu danych sygnalizowane jest ustawieniem w stan wysoki, przez jeden okres zegara, wyjścia *done* bloku. Jednocześnie na 8–bitowym wyjściu *data<7:0>* bloku dostępne są odczytane dane.

Zbigniew Hajduk