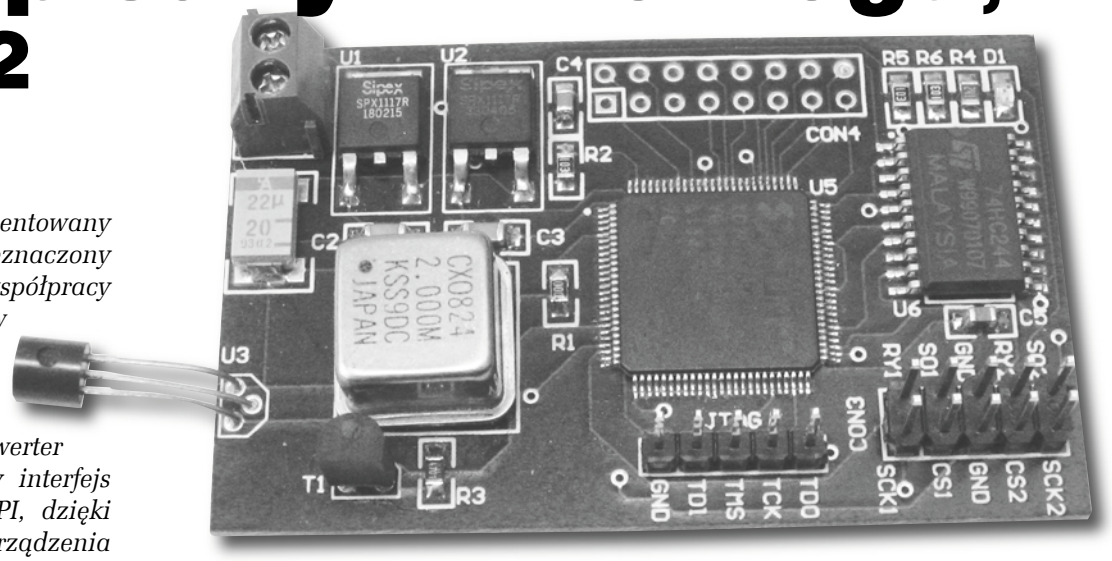
Konwerter 1–Wire –> SPI opisany w Verilogu, część 2

AVT–443



Konwerter prezentowany w artykule 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

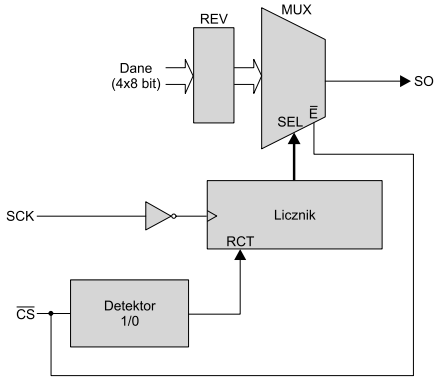
- Płytka o wymiarach 61 x 38 mm
- Zasilanie 5...8 V DC
- Jednokierunkowa transmisja danych odczytanych z interfejsu 1-wire do mikrokontrolera po SPI
- Kontrola CRC poprawności danych odebranych z 1-wire
- Liczba kanałów 1-wire: 1
- Liczba kanałów SPI: 1

Blok interfejsu SPI

Blok ten został zaimplementowany w oparciu o schemat blokowy pokazany na **rys. 3**. Opadające zbocze na wejściu CS zeruje 5–bitowy licznik, którego wyjście połączone jest z wejściem adresowym multipleksera. Wejście taktujące licznika sterowane jest poprzez inwerter z wejścia SCK modułu interfejsu SPI. Pojawiające się na wejściu SCK impulsy powodują zwiększanie zawartości licznika (podczas opadającego zbocza SCK) i tym samym połączenie odpowiedniego wejścia danych multipleksera z jego wyjściem, będącym równocześnie wyjściem szeregowym *SO* całego bloku SPI. Multiplekser został wyposażony dodatkowo w wejście zezwalające (aktywny poziom niski) połączone z wejściem wyboru CS bloku SPI. Poziom wysoki na tym wejściu powoduje, że niezależnie od stanu wejść danych i adresowych multipleksera, na jego wyjściu zawsze występuje stan niski. Ponieważ najczęściej implementowany standard interfejsu SPI zakłada, że bit najbardziej znaczący (MSB) przesyłany jest jako pierwszy (jest to również bardzo wygodne przy konstruowaniu programowej pętli obsługi interfejsu SPI dla mikrokontrolerów), dlatego też w przypadku prezentowanej tu aplikacji wymaga to odpowiedniego sposobu podłączenia poszczególnych bajtów danych wejściowych bloku SPI do wejścia danych multipleksera. Słowo wejściowe bloku SPI składa się w rzeczywistości z 4 niezależnych bajtów, stanowiących zawartość pamięci notatnikowej układu termometru cyfrowego. Transmisja za pomocą interfejsu SPI bitu najbardziej znaczącego jako pierwszego, przy konstrukcji modułu jak na rys. 3, wymaga odwrócenia bitów we wszystkich czterech bajtach, w taki sposób, aby bit najmniej znaczący stał się bitem najbardziej znaczącym, itd. Stąd na rys. 3 dodatkowy blok *REV* pośredniczący między wejściem bloku SPI a wejściem danych multipleksera i realizujący odwrócenie bitów.

Układ sterujący

Układ sterujący został zaprojektowany również jako automat sekwencyjny, którego zadaniem jest koordynacja działań opisanych wy-



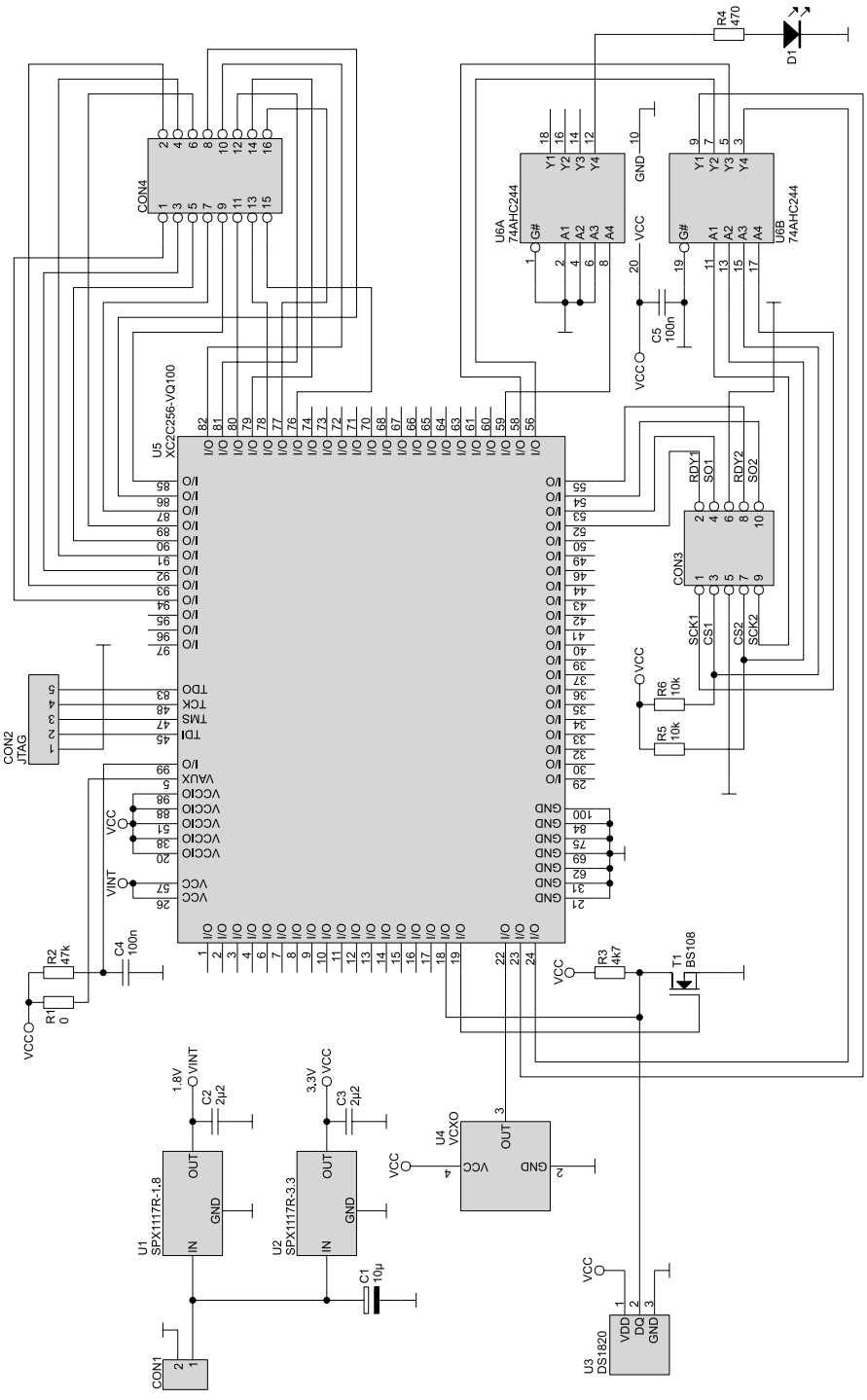
Rys. 3. Schemat blokowy interfejsu SPI

żej bloków, w taki sposób, aby możliwa była realizacja określonego algorytmu. Algorytm ten w przypadku prezentowanego układu konwertera 1–wire –> SPI obsługującego termometry cyfrowe firmy *Dallas/Maxim*, wygląda następująco:

- wygenerowanie impulsu zerującego dla układu termometru 1–wire,
- sprawdzenie impulsu obecności układu 1–wire,
- wysłanie do układu polecenia o kodzie 0CCh („pomiń ROM"),
- wysłanie do układu polecenia o kodzie 044h („zmierz temperaturę"),
- cykliczny odczyt danych z układu, aż do momentu, gdy odczytane bity będą miały wartość 1 (pomiar temperatury zakończony),
- wygenerowanie impulsu zerującego,
- sprawdzenie impulsu obecności,
- wysłanie do układu polecenia o kodzie 0CCh,
- wysłanie do układu polecenia o kodzie 0BEh („odczytaj pamięć notatnikową"),
- odczyt 9 bajtów danych (zapamiętywane są tylko 4 bajty o numerach 0, 1, 6 i 7),
- porównanie obliczonej wartości wielomianu CRC z otrzymanych danych z wartością odczytaną (dziewiąty bajt o numerze 8) i w przypadku zgodności ustawienie wyjść gotowości *RDY1* i *RDY2*,
- sprawdzenie stanu linii wyboru *CS1* i *CS2* interfejsu SPI – w przypadku, gdy na obu liniach występuje stan wysoki (brak transmisji danych za pomocą interfejsu) przepisanie na wyjście *data_out<31:0>* układu sterującego czterech bajtów danych odczytanych z pamięci notatnikowej układu termometru 1–wire,
- rozpoczęcie wykonywania algorytmu od początku.

Dodatkowo układ sterujący, przy każdym kolejnym cyklu obiegu wyżej opisanego algorytmu, zmienia stan wyjścia *LED* na przeciwny. Do tego wyjścia może być dołączona dioda LED, której cykliczne miganie (w przypadku pozostawienia wejść *CS1* i *CS2* w stanie wysokim) oznacza poprawną pracę całego układu.

Układ sterujący powoduje odczytanie całej zawartości pamięci notatnikowej układu termometru cyfrowego, jednak na swoim wyjściu (wejściu interfejsu SPI) udostępnia tylko 4 bajty tej pamięci (bajty 0, 1, 6, 7). W przypadku podłączenia do konwertera termometrów precyzyjnych typu DS18B20, bajty 6 i 7 można zupełnie pominąć, gdyż kompletna informacja o temperaturze zawarta jest w dwóch pierwszych bajtach 0 i 1. W przypadku podłączenia termometrów np. typu DS1820 bajty 6 i 7 można wykorzystać do bardziej precyzyjnego obliczenia wartości zmierzonej temperatury.



Rys. 4. Schemat ideowy układu konwertera 1–wire –> SPI

Warto również w tym miejscu dodać, że zmiana algorytmu pracy prezentowanego tutaj konwertera (np. obsługa termometru 1–wire w trybie z tzw. zasilaniem pasożytniczym) wymaga jedynie dokonania zmian w strukturze automatu sekwencyjnego w układzie sterującym, a wirtualne komponenty układu operacyjnego pozostają niezmienione.

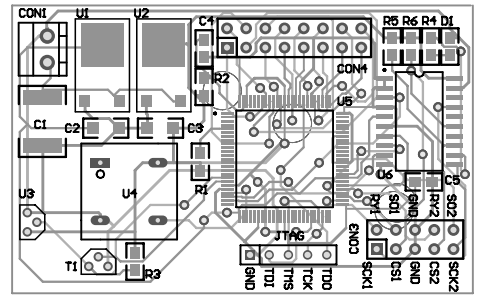
Przykładowa aplikacja

Schemat ideowy przykładowej aplikacji układu konwertera 1–wire<–>SPI zaimplementowanego

z wykorzystaniem wyżej opisanych wirtualnych komponentów, pokazano na **rys. 4**. Jako układ programowalny wybrano XC2C256 z serii *CoolRunner II* CPLD firmy *Xilinx*. Charakteryzuje się on m. in. bardzo niskim poborem prądu (w stosunku do większości układów klasy CPLD), krótkimi opóźnieniami typu *pin–to–pin* oraz bardzo szybkim programowaniem w systemie (ISP) za pomocą interfejsu JTAG.

Wybrany układ programowalny wymaga stosowania dwóch napięć zasilających, oddzielnego napięcia do zasilania rdzenia (1,8 V) i oddzielnego do zasilania buforów wejścia – wyjścia (o wartości zależnej od wybranego standardu pracy, tu wybrano standard LVTTL wymagający napięcia 3,3 V). Dlatego też zastosowano dwa, stosunkowo łatwo dostępne, stabilizatory LDO (*Low Drop Output*) U1 i U2 typu SPX1117 firmy Sipex.

Pewną wadą układów CPLD z serii *CoolRunnerII* jest to, że ich komórki wejścia – wyjścia nie tolerują napięć wyższych niż napięcie zasilania tych komórek. Oznacza to brak kompatybilności z wciąż szeroko stosowanym, pięciowoltowym, standardem TTL. W celu zapewnienia możliwości współpracy układu konwertera (poprzez interfejs SPI) z innymi układami pracującymi w standardzie TTL, zastosowano dodatkowy bufor magistrali U6 z serii 74AHC, zapewniający konwersję napięć wejściowych 5 V do poziomu 3,3 V. Bufor ten jest zbędny w przypadku współpracy z urządzeniami pracującymi w standardzie LVTTL. Ostatecznie można go również zastąpić zwykłymi rezystorami o wartości kilkuset omów, łączącymi odpowiednie końcówki złącza CON3 i układu U5. Wyjścia układu CPLD (U5) nie są buforowane.



Rys. 5. Schemat montażowy układu konwertera 1–wire –> SPI

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],cr-
c[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
```

Tranzystor T1 spełnia rolę typowego elementu sprzęgającego z magistralą jednoprzewodową. Choć istnieje możliwość skonfigurowania komórek wejścia – wyjścia układu XC2C256 do pracy w trybie otwarty dren, jednak ze względu na większą elastyczność zastosowań opisywanych tu wirtualnych komponentów (brak zależności od właściwości wybranej struktury układu PLD), zdecydowano się na zastosowanie zewnętrznego tranzystora MOSFET.

Elementy RC: R2 i C4 spełniają rolę prostego układu zerującego, utrzymującego przez pewien czas po włączeniu napięcia zasilania, poziom niski na wybranej końcówce układu U5. Rezystor R1 o wartości zero omów (zwora) zasila interfejs JTAG wewnątrz układu CPLD i może być usunięty w przypadku, gdy nie przewiduje się przeprogramowania (zmiany konfiguracji) układu CPLD. Jako generator zegara zastosowano typowy zintegrowany (w metalowej obudowie) generator o wartości 2 MHz. Na złączu CON2 dostępne są linie interfejsu JTAG do konfigurowania układu PLD. Złącze CON4 jest opcjonalne, przydatne w fazie testowania.

Po zmontowaniu układu i włączeniu napięcia zasilania, przy nie podłączonym złączu CON3 interfejsu SPI, układ powinien sygnalizować prawidłową pracę miganiem diody LED D1. Wymuszenie poziomu niskiego na linii *CS1* lub *CS2* złącza CON3 powinno spowodować zaprzestania migania diody. Miganie diody sygnalizuje wykonywanie kolejnych cykli konwersji i odczytu temperatury z układu termometru cyfrowego. Informacja o temperaturze w postaci czterech bajtów (0,

1, 6 i 7) stanowiących zawartość pamięci notatnikowej układu termometru dostępna jest na podwójnym złączu SPI (CON3). Bajty te, podczas taktowania linii SCK1 lub SCK2 interfejsu SPI przesyłane są w wymienionej wyżej kolejności bitem najbardziej znaczącym jako pierwszym. Dane dostępne poprzez podwójny interfejs SPI mogą być odczytywane w dowolnym momencie, w pełni równolegle przez dwa niezależne urządzenia (np. mikrokontrolery). Oprócz standardowych linii interfejsu takich jak SCK, CS, SO na złączu CON3 dostępne są także dwie linie gotowości RDY1 i RDY2. Na liniach tych pojawia się stan wysoki, wówczas, gdy dostępna jest nowa, wartość temperatury zmierzona przez termometr i gotowa do odczytania poprzez interfejs SPI. Linie gotowości są zerowane, gdy zostanie przeprowadzony odczyt danych za pośrednictwem korespondującego z daną linią gotowości, kanału interfejsu SPI (SPI1 lub SPI2).

Możliwości modyfikacji

Opisany tu wirtualny komponent konwertera 1–wire –> SPI, przeznaczony do współpracy z termometrami cyfrowymi z magistralą 1–wire, może być w sposób nieskomplikowany poddany wielu modyfikacjom zwiększającym jego funkcjonalność. Na przykład poprzez stosunkowo niewielką zmianę automatu w układzie sterującym (dodanie kilku stanów) i uzupełnienie układu elektrycznego o dodatkowy tranzystor MOSFET realizujący silne podciąganie do plusa zasilania, można zapewnić pracę termometru z tzw. zasilaniem pasożytniczym, umożliwiającym pełne wykorzystanie zalet magistrali jednoprzewodowej. Nieco bardziej skomplikowana modyfikacja układu sterującego i bloku interfejsu SPI mogłaby, na przykład, umożliwić dwukierunkową konwersją 1–wire <–> SPI, pozwalającą np. na zapis (poprzez interfejs SPI) wybranych bajtów do pamięci notatnikowej termometru cyfrowego, itp.

List. 4. Opis bloku zapisu danych

```
module one_wire_write(clk,reset,data,trx_activate,zero,load,done,time_c,bus_out);
input clk,reset,trx_activate,zero;
output done,bus_out,load;
input [7:0] data;
output [6:0] time_c;

reg  done,wire1_out,ld;
reg [2:0] state;
reg [6:0] tc;
reg [2:0] mux_addr;

wire tx_bit;

//poniżej opis behawioralny multipleksera
function mux81;
input [7:0] data;
input [2:0] addr;
integer i;
begin
    for(i=0;i<8;i=i+1)
      if(i==addr) mux81=data[i];
end
endfunction

assign tx_bit=mux81(data,mux_addr);
//tx_bit = transmitowany bit

always@(posedge clk or negedge reset)
begin
   if(~reset) state=0; else
   begin
    case(state) //realizacja autoamtu
  3'd0: begin //stan początkowy
          mux_addr=0; done=0; tc=7'd0; ld=1'b0;
    wire1_out=1'b1;
    if(trx_activate) state=3'd1;
  end
  3'd1:begin
          wire1_out=1'b0; //wymuszenie 0 na 1-wire przez 5us
     tc=7'd0; ld=1'b1;
    if(zero) state=3'd2;
  end
  3'd2: begin
    ld=1'b0; state=3'd3;
    if(tx_bit==1'b1) wire1_out=1'b1;
          //jeżeli tx_bit=1 zwolnij magistralę
  end
  3'd3: begin
    tc=7'd17; ld=1'b1; //odczekanie 90us
    if(zero) state=3'd4;
  end
  3'd4: begin
    ld=1'b0; wire1_out=1'b1;
          //ustaw stan wysoki na 1-wire
    state=3'd5;
  end
  3'd5: begin
    tc=7'd0; ld=1'b1; //odczekaj 5us
    if(zero) state=3'd6;
  end
  3'd6: begin
    ld=1'b0;
          if(mux_addr==3'd7) state=3'd7;
    else
    begin mux_addr=mux_addr+1; state=3'd1; end
          //jeżeli nie wysłano jeszcze 8 bitów wróć do stanu 1
  end
  3'd7: begin
    done=1'b1; state=3'd0;
          //zapis zakończony, ustaw done i przejdź do stanu początkowego.
  end
   endcase
   end
end
 assign time_c=tc;
 assign load=ld;
 assign bus_out=wire1_out;
endmodule
```

W przedstawionej wersji konwerter zajmuje 186 makrokomórek spośród 256 dostępnych w układzie XC2C256, pozostaje wiec jeszcze trochę zasobów do wykorzystania. Możliwa jest jednak pewna optymalizacja budowy części operacyjnej konwertera, poprzez zintegrowanie funkcji bloków generowania sygnału zerującego oraz zapisu i odczytu w jednym module, opisanym jednym automatem, przez co można uzyskać mniejsze zużycie zasobów danego układu programowalnego.

Zbigniew Hajduk