

# Implementacja interfejsu 1wire w VHDL przy użyciu Spartan-3E oraz DS18S20

Krzysztof Cabała 210047

Kinga Wilczek 210063

24 maja 2016

# Spis treści

<b>1</b>	<b>Założenia projektowe</b>	<b>4</b>
<b>2</b>	<b>Wstęp teoretyczny</b>	<b>4</b>
2.1	Interfejs . . . . .	4
2.2	Komunikacja . . . . .	4
2.2.1	Inicjalizacja i reset . . . . .	4
2.2.2	Zapis bitu . . . . .	4
2.2.3	Odczyt bitu . . . . .	5
2.3	Konwersja i odczyt temperatury . . . . .	5
2.4	Double dabble . . . . .	5
<b>3</b>	<b>Implementacja</b>	<b>6</b>
3.1	Schemat główny . . . . .	6
3.2	Termometer . . . . .	6
3.2.1	Controller . . . . .	6
<b>4</b>	<b>Implementacja podstawowych operacji</b>	<b>10</b>
<b>5</b>	<b>Transmisja bajtu</b>	<b>11</b>
<b>6</b>	<b>Implementacja transmisji bajtu</b>	<b>11</b>
<b>7</b>	<b>Algorytm double dabble</b>	<b>11</b>
<b>8</b>	<b>Implementacja double dabble</b>	<b>11</b>

## Spis rysunków

1	Przykładowe połączenie urządzeń . . . . .	4
2	Diagram czasowy dla procedury inicjalizacji . . . . .	4
3	Diagramy czasowe dla procedury zapisu i odczytu . . . . .	5
4	Schemat ogólny projektu . . . . .	7
5	Moduł Termometer . . . . .	8
6	Moduł Controller . . . . .	9
7	Moduł obsługi podstawowych operacji bitowych . . . . .	10
8	Maszyna stanów BusController . . . . .	10
9	Symbol IOBuf . . . . .	11
10	Moduł transmisji bajtu . . . . .	11
11	Maszyna stanów ByteModule . . . . .	12

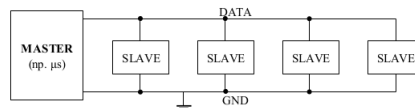
# 1 Założenia projektowe

Celem projektu było przygotowanie układu obsługującego sensor Dallas DS18S20 na platformie Xilinx Spartan-3E. W tym celu zaimplementowano obsługę magistrali One wire oraz interpretację wyniku. Efektem działania układu jest wyświetlona wartość temperatury na wyświetlaczu LCD zgodnym ze standardem HD44780.

## 2 Wstęp teoretyczny

### 2.1 Interfejs

Interfejs 1wire jest interfejsem opracowanym przez firmę Dallas Semiconductor do komunikacji między dwoma lub większą liczbą urządzeń przy wykorzystaniu zaledwie jednej linii danych, linii GND (konieczne odniesienie dla poprawnego rozpoznawania stanów logicznych) oraz zasilania Vcc. W ramach oszczędności przewodów ogranicza się połączenia do dwóch linii, wtedy układ zasilany jest pasywnie z linii danych. Przykładowe połączenie przedstawia schemat:



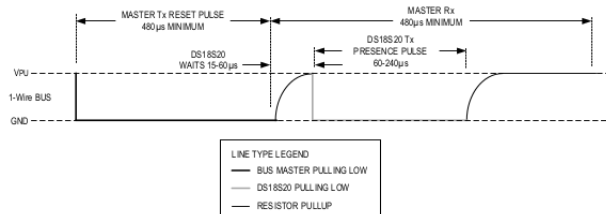
Rysunek 1: Przykładowe połączenie urządzeń

Wyróżnia się urządzenia typu Master (najczęściej mikrokontroler) oraz Slave (peryferia).

### 2.2 Komunikacja

#### 2.2.1 Inicjalizacja i reset

Każda próba komunikacji urządzeń master i slave musi zacząć się od sekwencji składającej się z sygnału reset, wysyłanego przez master, po którym następuje sygnał obecności układu slave. Sygnał reset to wymuszony stan 0 trwający przynajmniej  $480\mu s$ . Następnie master oczekuje na sygnał obecności innego urządzenia na linii. Następuje wówczas zwolnienie magistrali, co powoduje podciągnięcie jej do stanu wysokiego przez rezystor pull-up. Urządzenie slave wykrywa wówczas narastające zbocze linii i po upływie  $15\mu s$ - $60\mu s$  sygnalizuje swoją obecność poprzez wymuszenie stanu niskiego na okres  $60\mu s$ - $240\mu s$ .



Rysunek 2: Diagram czasowy dla procedury inicjalizacji

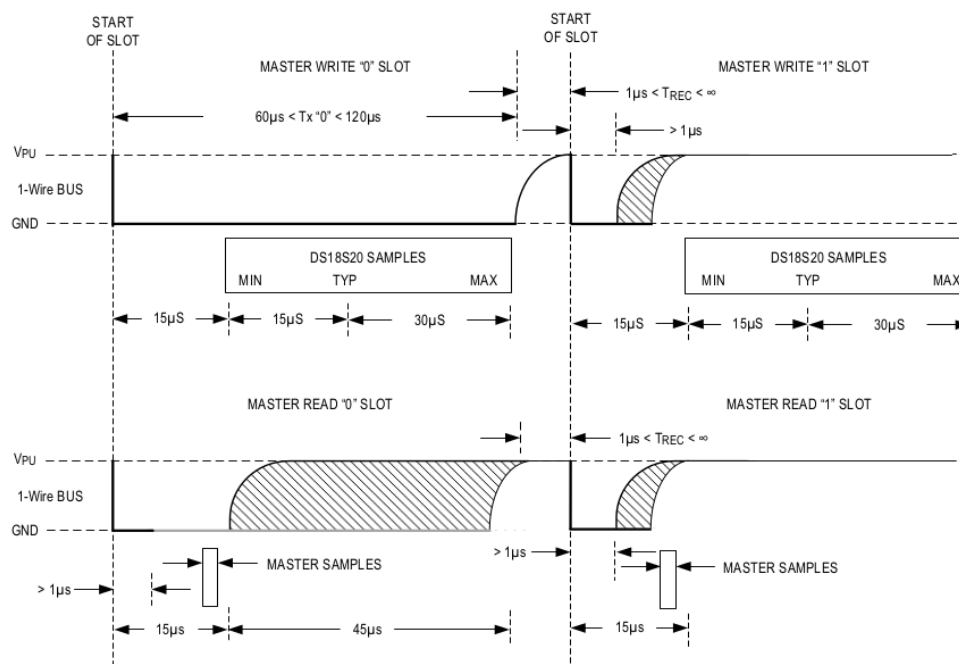
#### 2.2.2 Zapis bitu

Operacje zapisu bitu realizowane są w ściśle określonych slotach czasowych. Długość jednego slotu wynosić zwykle  $60\mu s$ . Próbkowanie dokonywane jest mniej więcej w środku slotu celem uodpornienia na błędy. Pomiedzy kolejnymi operacjami wymagana jest przynajmniej  $1\mu s$  odstępu.

Zapis rozpoczyna się wysterowaniem linii danych przez master na poziom niski. Zapis 0 wymaga utrzymania jej w tym stanie przez cały slot. Zapis 1 jest nieco bardziej skomplikowany. Master musi w czasie nie dłuższym niż  $15\mu\text{s}$ , ale nie krótszym niż  $1\mu\text{s}$  zwolnić magistralę tak aby w momencie próbkowania (po  $30\mu\text{s}$  od zbocza opadającego) była w stanie wysokim.

### 2.2.3 Odczyt bitu

Odczyt bitu również wymaga  $60\mu\text{s}$  slotu oraz  $1\mu\text{s}$  przerwy. Odczyt rozpoczyna się wymuszeniem przez master stanu niskiego na linii danych na czas nie krótszy niż  $1\mu\text{s}$  i zwolnienie jej (powrót do stanu wysokiego). Po tym sygnale sterowanie linią przejmuje urządzenie slave, wysyłające bit 0 lub 1. Slave po wykryciu zbocza opadającego wymusza stan niski (dla 0) lub utrzymuje wysoki (dla 1) linii danych. Sygnał musi być wtedy spróbkowany przez master. Przed upłynięciem czasu końca slotu magistrala zostaje zwolniona przez slave, co powoduje jej powrót do stanu wysokiego.



Rysunek 3: Diagramy czasowe dla procedury zapisu i odczytu

## 2.3 Konwersja i odczyt temperatury

Aktualny odczyt temperatury zapisany jest w dwóch pierwszych bajtach pamięci Scratchpad. Po poprawnej inicjalizacji czujnika DS18S20 ich zawartość odpowiada temperaturze  $+85^{\circ}\text{C}$ . W celu zmierzenia aktualnej temperatury należy wysłać do termometru komendę konwersji, zresetować układ i odczytać pamięć. W pamięci urządzenia jest 9 bajtów wyniku. Tylko dwa pierwsze są istotne w punktu wyniku. Pierwszy uzyskany bajt określa znak odczytu. Natomiast ostatni bit drugiego bajtu stanowi o wystąpieniu cyfry po przecinku. Pozostałe bity wyniku to wartość odczytanej temperatury. Wartość ta następnie podlega konwersji na kod BCD za pomocą algorytmu double dabble.

## 2.4 Double dabble

Double dabble jest powszechnie wykorzystywanym algorytmem do konwersji liczb binarnych na kod BCD (Binary-Coded Decimal - system dziesiętny zakodowany dwójkowo).

Algorytm polega na wykonaniu  $n$  iteracji (w zależności od długości ciągu bitów). Początkowo wynikowy kod BCD jest zainicjalizowany jako ciąg zer podzielonych na grupy po 4 bity. Podczas każdej iteracji wykonywane jest przesunięcie o jeden bit w lewo, a na koniec doklejany jest jeden bit ciągu wejściowego. Jeżeli przed kolejnym przesunięciem wartość w jednej z grup w kodzie BCD jest wyższa niż 4 to następuje dodanie binarnej 3. Po wykonaniu odpowiedniej ilości iteracji algorytm kończy swoje działanie. Ostatecznie wynikowy ciąg jest podzielony po cztery bity, które odpowiadają kolejnym cyfrom wyniku.

## 3 Implementacja

### 3.1 Schemat główny

Funkcjonalność projektu opiera się na 6 podstawowych modułach widocznych na rysunku 4:

- Termometer - obsługa sensora DS18S20;
- RotaryEnc- sterowanie częstotliwością pomiaru;
- Interpreter - interpretacja 2 bajtowego wyniku;
- double\_dabble - zamiana liczby binarnej bez znaku na kod BCD;
- LCD - generowanie zawartości wyświetlacza LCD;
- LCDWrite - sterownik wyświetlacza.

### 3.2 Termometer

Moduł Termometer jest najbardziej rozbudowanym modułem składającym się z 4 układów (rysunek 5):

- Controller - obsługa sekwencji komunikacji i konwersji temperatury;
- ByteModule - obsługa komunikacji na poziomie bajtu (instrukcji);
- BusController - obsługa magistrali Onewire;
- IOBuf - ustala kierunek komunikacji.

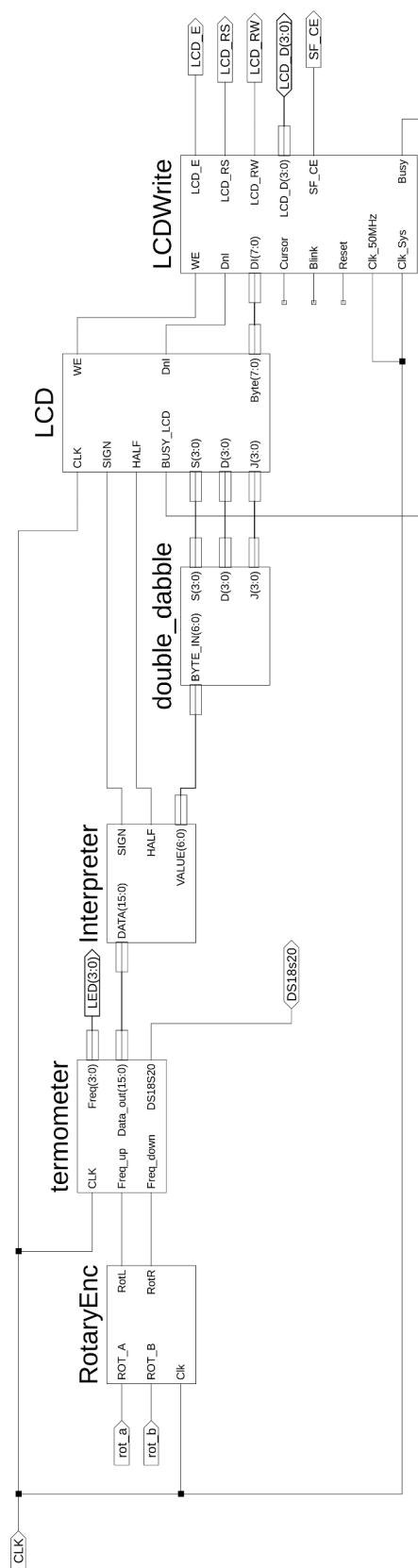
#### 3.2.1 Controller

Moduł Controller (rysunek 6) odpowiada za przeprowadzenie poprawnej sekwencji operacji wymaganych do zmierzenia i odczytu temperatury.

Opis wyprowadzeń:

Wejścia

- CLK - Zegar;
- Freq-up - Impuls zwiększenia częstotliwości próbkowania;
- Freq-down - Impul zmniejszenia częstotliwości próbkowania;
- Byte\_in - Bajt odebrany przez ByteModule;
- Bit\_in - Bit odeberany przez BusController;
- Busy\_in - Flaga zajętości ByteModule;



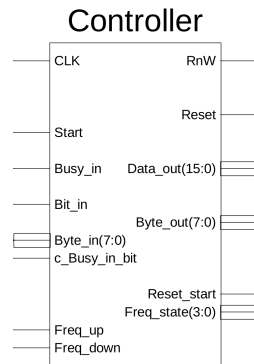
Rysunek 4: Schemat ogólny projektu

- **c\_Busy\_in** - Flaga zajętości BusController;

Wyjścia:







Rysunek 6: Moduł Controller

- Reset\_start - Sygnał start dla BusControllera.

W celu ograniczenia częstotliwości odczytu (co powoduje nadmierne nagrzewanie się czujnika) dodano możliwość sterowania odstępem między pomiarami. Moduł umożliwia odczyt 4, 2, 1 sekundę lub ciągle. Zmiana odbywa się w procesie freq-process.

```

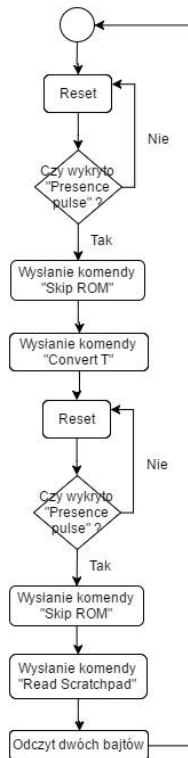
freq-process: process (CLK, freq-up, freq-down)
begin
    if rising-edge(clk) then
        if freq-up = '1' then
            frequency <= frequency(2 downto 0) & frequency (3);
        elsif freq-down = '1' then
            frequency <= frequency (0) & frequency(3 downto 1);
        end if;
    end if;
end process;
  
```

Moduł jest automatem skończonym (fsm). Zmiana stanów odbywa się w procesie state-service.

```

state-service: process (CLK, next_state)
begin
    if (rising-edge (CLK)) then
        state <= next_state;
    end if;
end process;
  
```

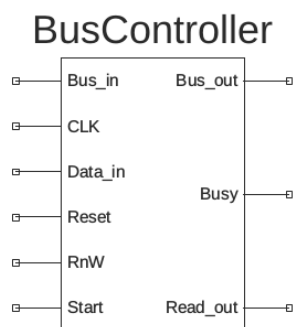
Kolejne stany maszyny pokazuje schemat automatu:



Rysunek 7: Maszyna stanów Controller

## 4 Implementacja podstawowych operacji

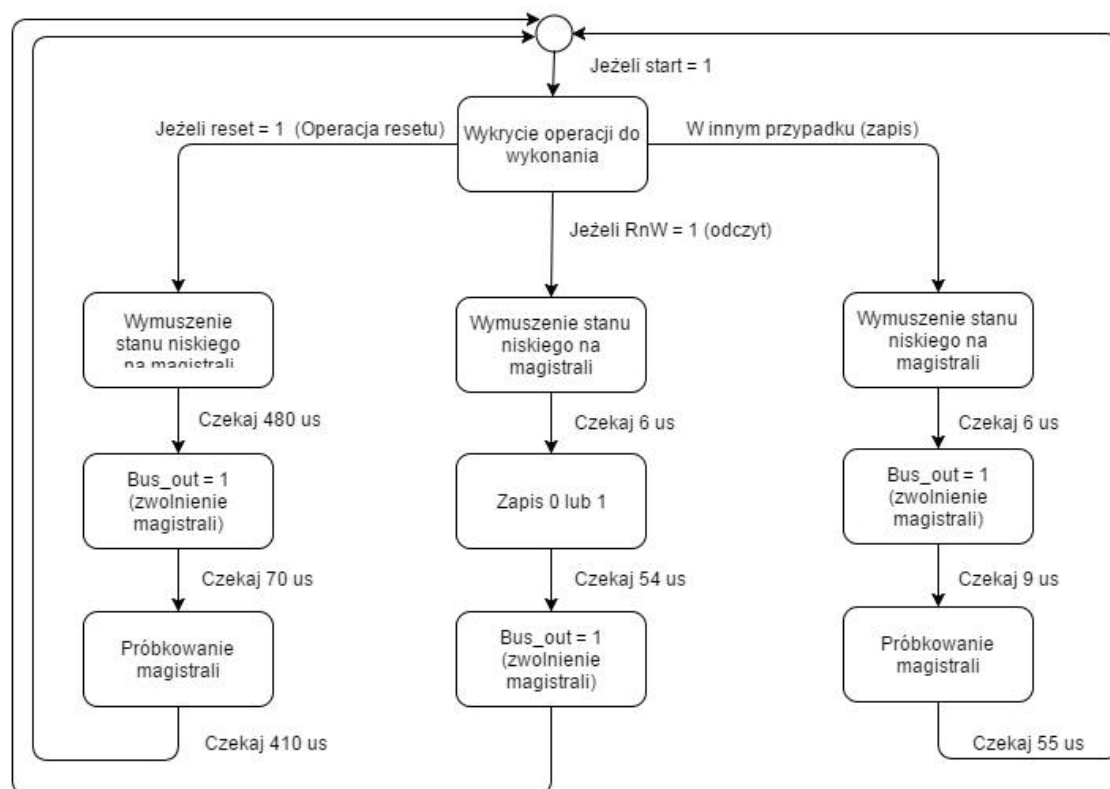
Za realizację podstawowych operacji odpowiada moduł BusController.



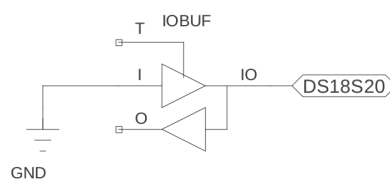
Rysunek 8: Moduł obsługi podstawowych operacji bitowych

Poniżej przedstawiono schemat blokowy tego automatu.

Poprawna obsługa magistrali wymaga jej zwalniania poprzez ustawienie w stan wysokiej impedancji (podciągnięcie do Vcc przez rezystor pull-up). Służy do tego element IOBuf. Podanie logicznego zera na wejście T otwiera bufor wyjściowy i przekazuje sygnał podawany na pin I (GND - Logiczne 0). Logiczne 1 na wejściu T ustawia linie w stanie wysokiej impedancji (zwalnia magistralę). Pin O służy do odczytu stanu magistrali.



Rysunek 9: Maszyna stanów BusController



Rysunek 10: Symbol IOBuf

## 5 Transmisja bajtu

Komunikacja master - slave jest dwukierunkowa. Dane przesyłane są w formie bajtów, kolejność bitów określa zasada najpierw najmłodszy.

## 6 Implementacja transmisji bajtu

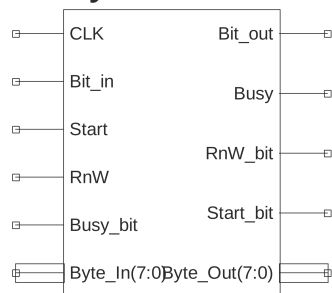
Transmisja bajtu realizowana jest poprzez następny moduł - ByteModule.

Automat zawiera po 4 stany na odczyt i zapis oraz jeden stan oczekiwania na sygnał startu.

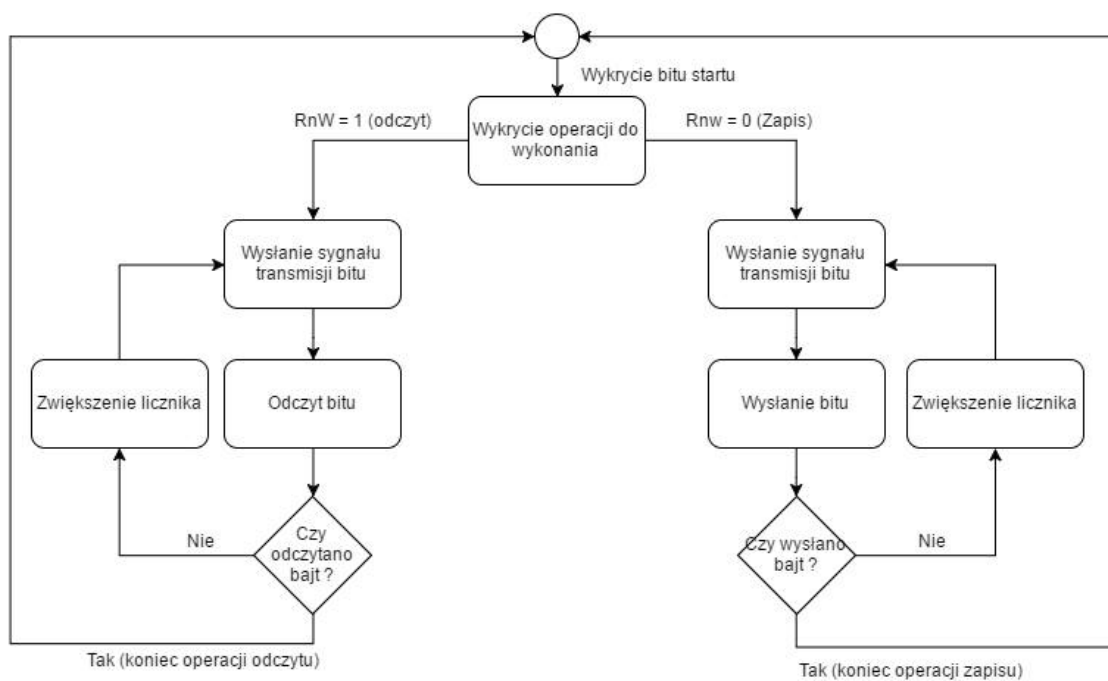
## 7 Algorytm double dabble

## 8 Implementacja double dabble

## ByteModule



Rysunek 11: Moduł transmisji bajtu



Rysunek 12: Maszyna stanów ByteModule