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

Krzysztof Cabała 210047
Kinga Wilczek 210063

6 czerwca 2016

Spis treści

Spis rysunków

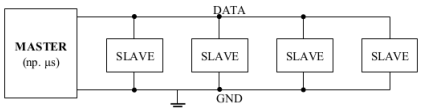
1 Założenia projektowe

Celem projektu było przygotowanie układu obsługującego sensor Dallas DS18S20 na platformie Xilinx Spartan-3E. W tym celu zaimplementowano obsługę magistrali One wire oraz interpretacje 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 lini danych, linii GND (konieczne odniesienie dla poprawnego rozpoznawania stanów logicznych) oraz zasialania Vcc. W ramach oszczędności przewodów ogranicza się połączenia do dwóch lini, wtedy układ zasilany jest pasożytniczo z lini danych. Przykładowe połacznie przedstawia schemat:



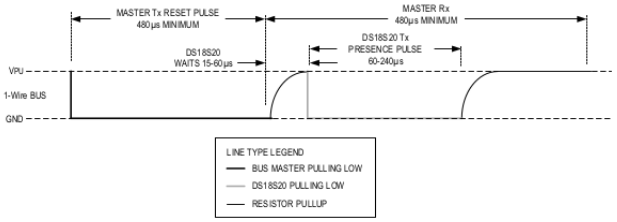
Rysunek 1: Przykładowe połączenie urządzeń

Wyróżnia się urządzania typu Master (najczęsciej 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 480μs. 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 lini i po upływie 15μs-60μs sygnalizuje swoją obecność poprzez wymuszenie stanu niskiego na okres 60μs-240μs.



Rysunek 2: Diagram czasowy dla procedury inicjalizacji

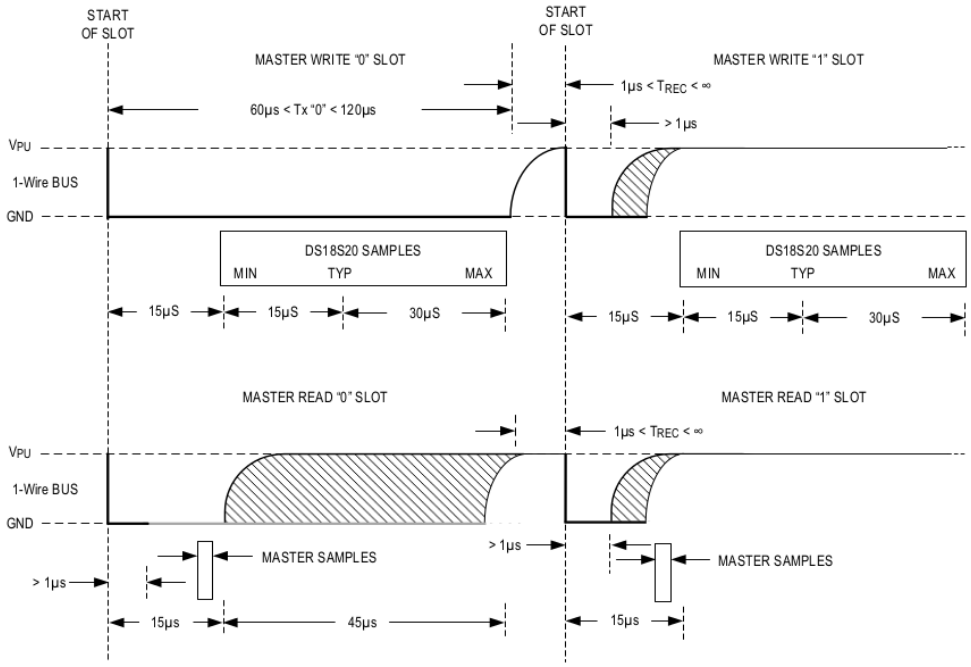
2.2.2 Zapis bitu

Operacje zapisu bitu realizowane są w ściśle określonych slotach czasowych. Długość jednego slotu wynosić zwykle 60μs. Próbkowanie dokonywane jest mniej więcej w środku slotu celem uodpornienia na błędy. Pomiędzy kolejnymi operacjami wymagana jest przynajmniej 1μs 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 15μs, ale nie krótszym niż 1μs zwolnić magistralęm tak aby w momencie próbkowania (po 30μs od zbocza opadającego) była w stanie wysokim.

2.2.3 Odczyt bitu

Odczyt bitu również wymaga 60μs slotu oraz 1μs przerwy. Odczyt rozpoczyna się wymuszeniem przez master stanu niskiego na linii danych na czas nie krótszy niż 1μs 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 maigistrala 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°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.

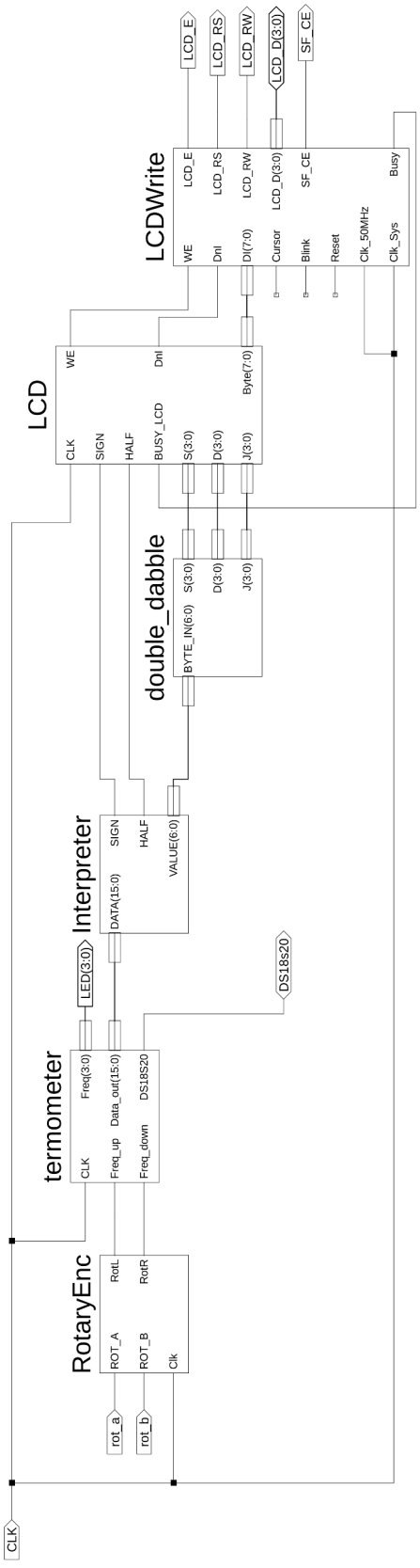
3.2.1 Controller

Moduł Controller (rysunek 8) odpowiada za przeprowadzenie poprawnej sekwencji operacji wymaganych do zmierzenia i odczytu temperatury.

Opis wyprowadzeń:

Wejścia

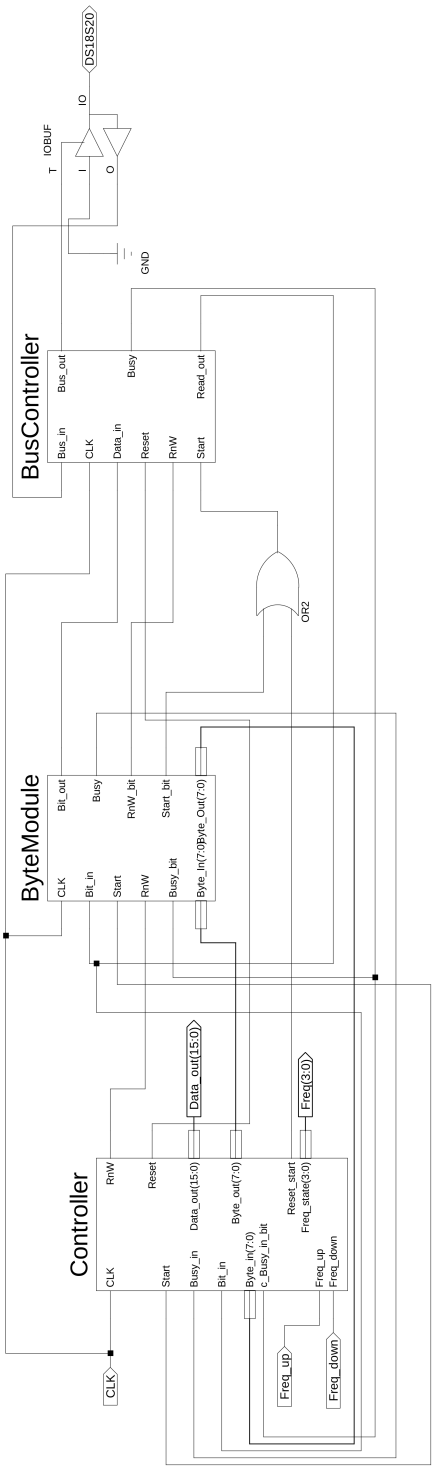
- CLK - Zegar;
- Freq_up - Impuls zwiększenia częstotliwości próbkowania;
- Freq_down - Impul zmniejszenia częstotliwości próbkowania;
- Byte_in - Bajt odebrany przez ByteModule;
- Bit_in - Bit odeberany przez BusController;
- Busy_in - Flaga zajętości ByteModule;



Rysunek 4: Schemat ogólny projektu

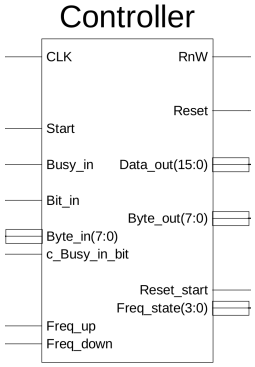
- c_Busy_in - Flaga zajętości BusController;

Wyjścia:



Rysunek 5: Moduł Termometer

- Start - Sygnał start dla ByteControllera;
- Data_out - Odebrany ciag reprezentujący temperaturę (2 bajty);
- Byte_out - Dane (kod instrukcji) dla ByteContorllera;
- Reser - Sygnał reset;
- Freq_state - Wektor reprezentujący aktualną częstotliwość próbkowania;



Rysunek 6: Moduł Controller

- Reset_start - Sygnał start dla BusControllera.

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

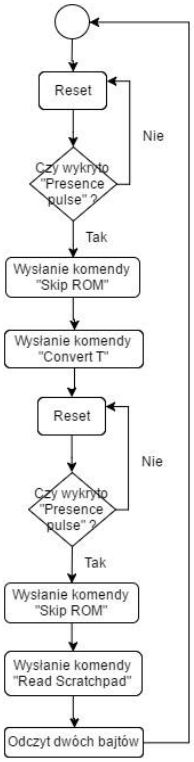
```
freq_process: process (CLK, freq_up, freq_down)
begin
        if rising_edge(clk) then
                if freq_up = '1' then
                        frequence <= frequence(2 downto 0) & frequence (3);
                elsif freq_down = '1' then
                        frequence <= frequence (0) & frequence(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 Rysunek 7. Po każdym ze stanów (poza presence) występuje dodatkowy stan oczekiwania na zwolnienie flagi busy (zajętości kontrolerów niższego rzędu). Stany te zostały pominięte na schemacie ze względu na jego czytelność. Należy zaznaczyć, że stany w których wykonywane są kolejne komendy są jednotaktowe, podczas gdy oczekiwanie na ich wykonanie zajmuje stosunkowo dużo więcej czasu. Zauważyć to można na symulacji pokazanej na rysunku 8

Zaznaczona na symulacji pozyjca A pokazuje moment zwolenia magistrali przez master (stan wysokiej impedancji). Zaraz po jej zwolnieniu kontrolę przejmuje slave, który wymuszając stan niski sygnalizuje swoją obecność (moment B). Przybliżony fragment (czerwona ramka) obrazuje przejścia stanów pomiędzy reset_slave_b a skip_b. Widać, że stany presence oraz skip są jednotaktowe.



Rysunek 7: Maszyna stanów Controller

==========

4 Implementacja podstawowych operacji

Za realizację podstawowych operacji odpowiada moduł BusController.

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 impednacji (zwalnia magistralę). Pin O służy do odczytu stanu magistrali.

5 Transmisja bajtu

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

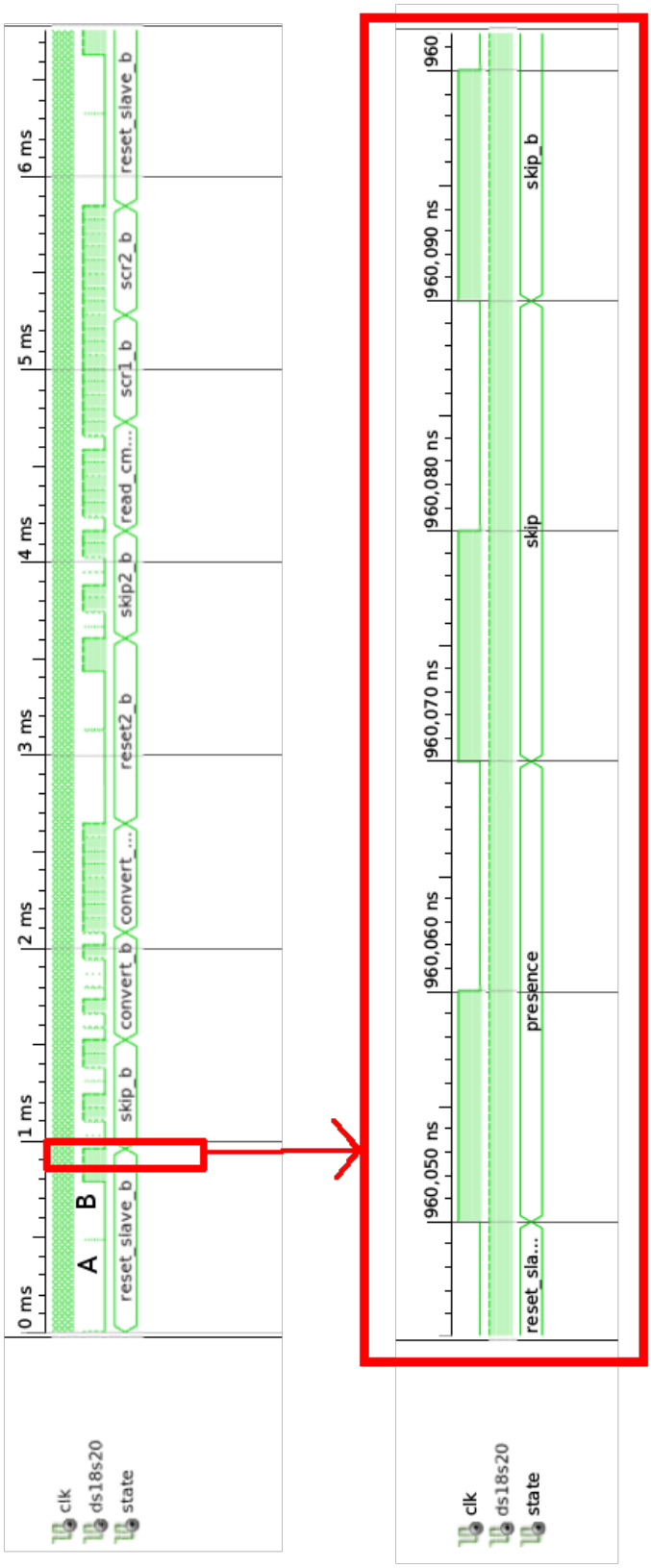
6 Implementacja transmisji bajtu

Transmisja bajtu realizowana jest poprzez następny moduł - ByteModule.

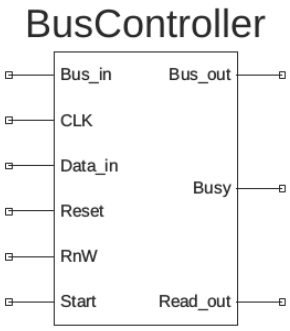
Automat zawiera po 4 stany na odczyt i zapis oraz jeden stan oczekiwania na sygnał startu.

7 Algorytm double dabble

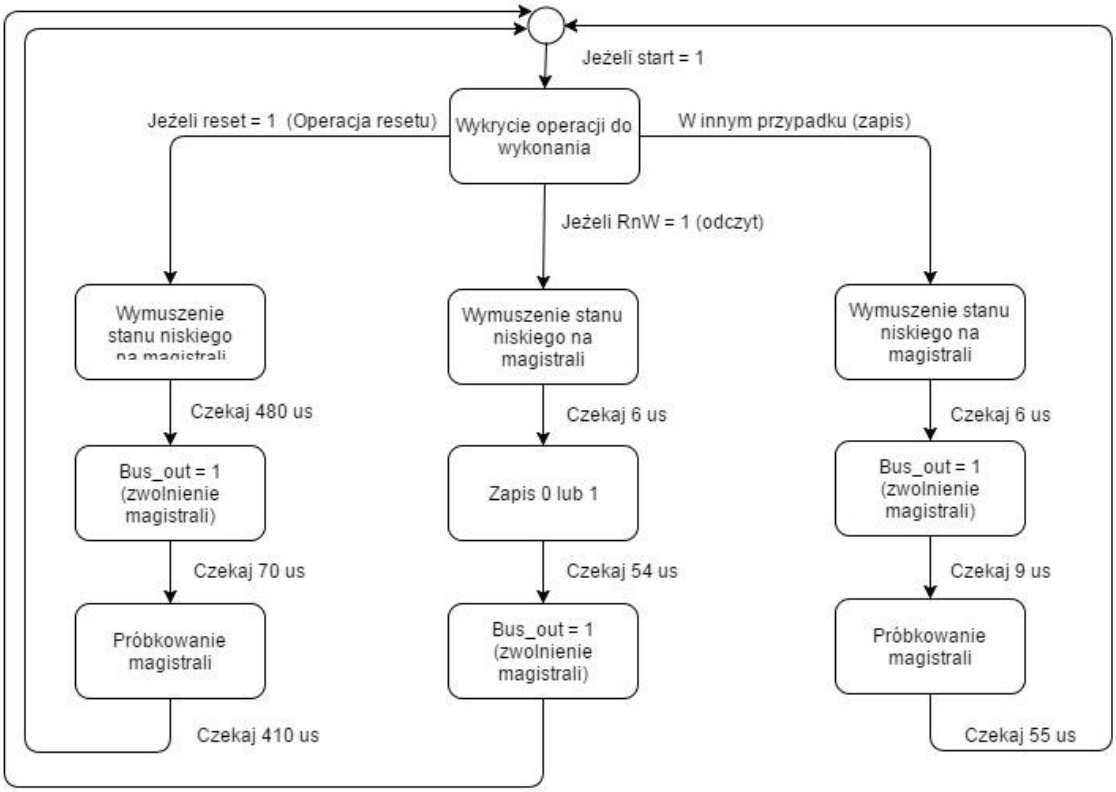
8 Implementacja double dabble



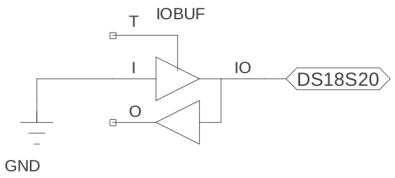
Rysunek 8: Symulacja automatu Controller



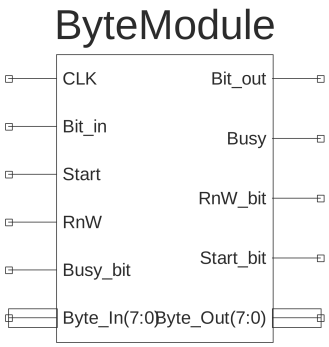
Rysunek 9: Moduł obsługi podstawowych operacji bitowych



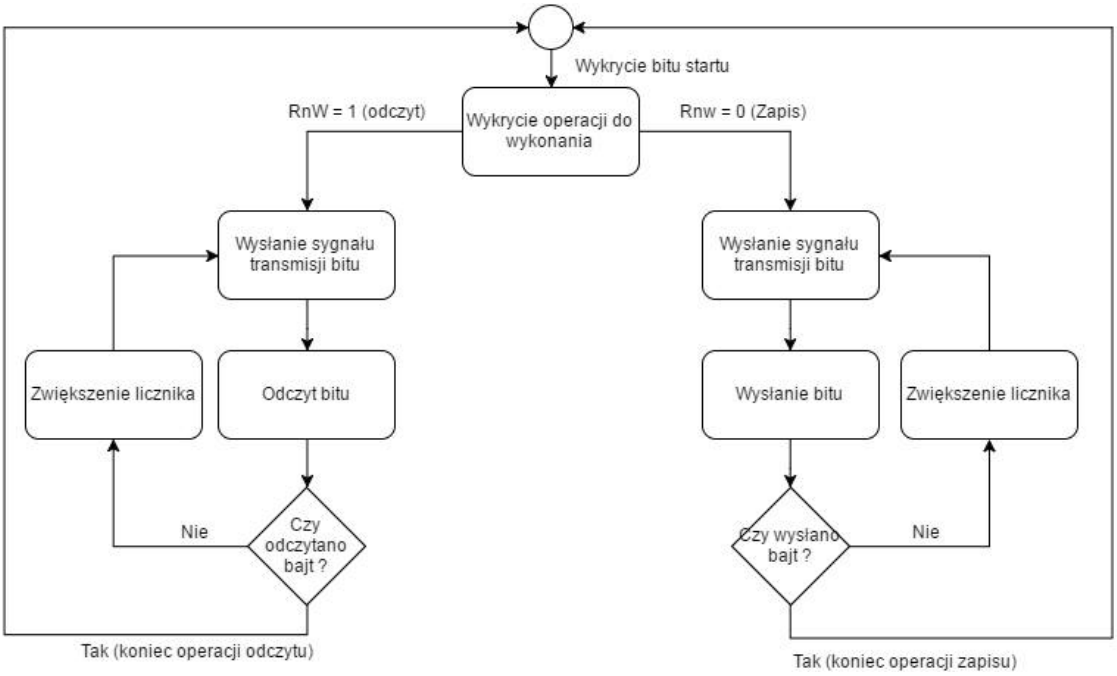
Rysunek 10: Maszyna stanów BusController



Rysunek 11: Symbol IOBuf



Rysunek 12: Moduł transmisji bajtu



Rysunek 13: Maszyna stanów ByteModule