

POLITECHNIKA WROCŁAWSKA

PROJEKT

UKŁADY CYFROWE I SYSTEMY WBUDOWANE 2

---

# **Obsługa komunikacji ze sterownikiem LCD na zestawie uruchomieniowym Spartan-3E**

---

*Authors:*

Rafał PIENIAŻEK  
Jakub POMYKAŁA

*Supervisor:*

Dr inż. Jarosław SUGIER

29 maja 2016

# **1 Temat**

## **1.1 Cel i zakres pracy**

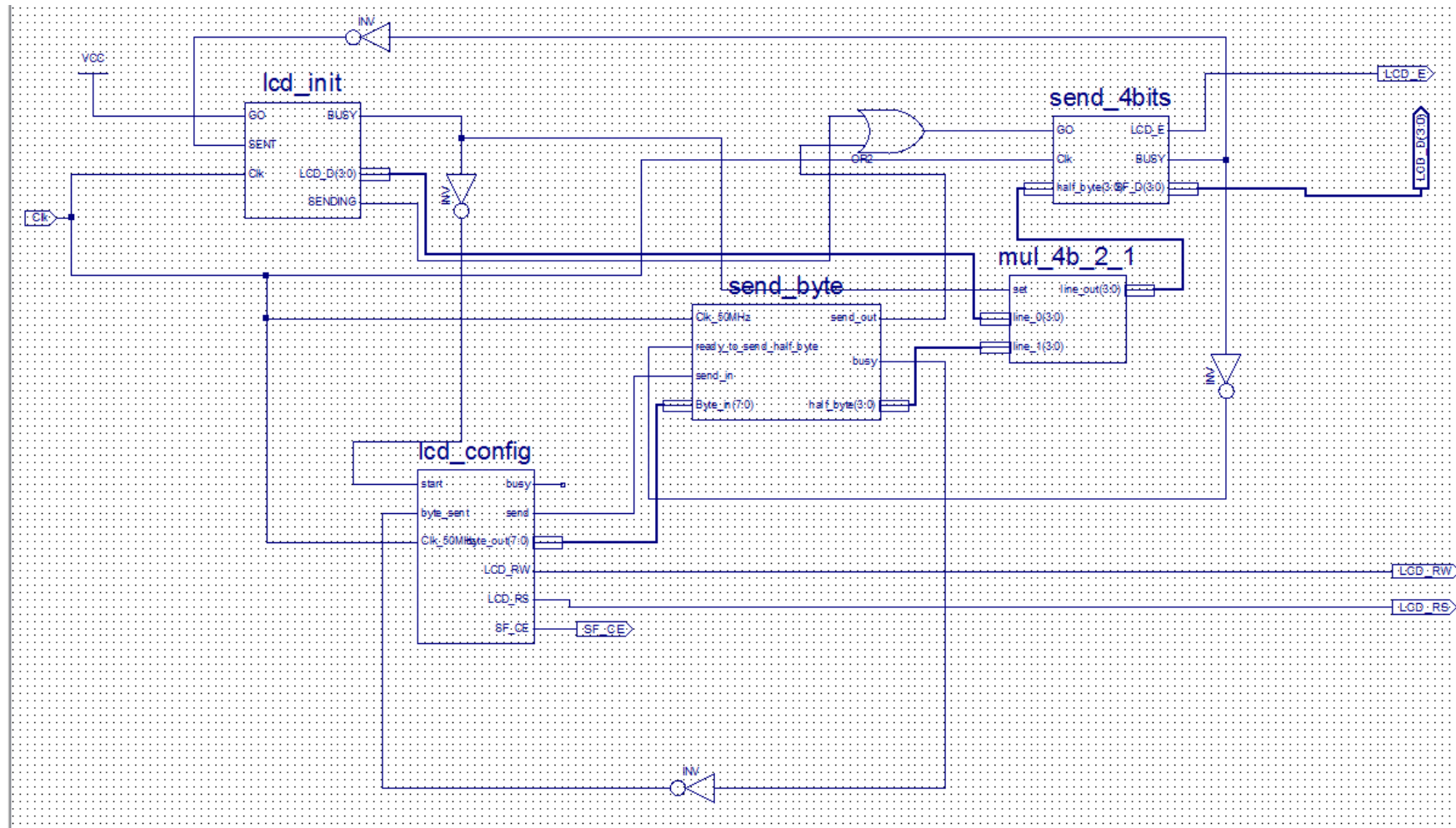
Celem projektu było zaimplementowanie modułu obsługi LCD podłączonego do układu Spartan-3E. W ramach projektu wykonano procedurę inicjalizacyjną i konfiguracyjną urządzenia.

## **1.2 Opis sprzętu**

Wyświetlacz podłączony do zestawu Spartan-3E posiada 2 linie 16 znakowe. W celu zminimalizowania ilości pinów wyjściowych, oraz zachowania kompatybilności sterownik do komunikacji korzysta z 4-bitowego interfejsu. Wyświetlacz LCD posiada zewnętrzny sterownik Sitronix ST7066U. Jest on zgodny funkcjonalnie z rodziną najbardziej popularnych wyświetlaczy z rodziny HD44780.

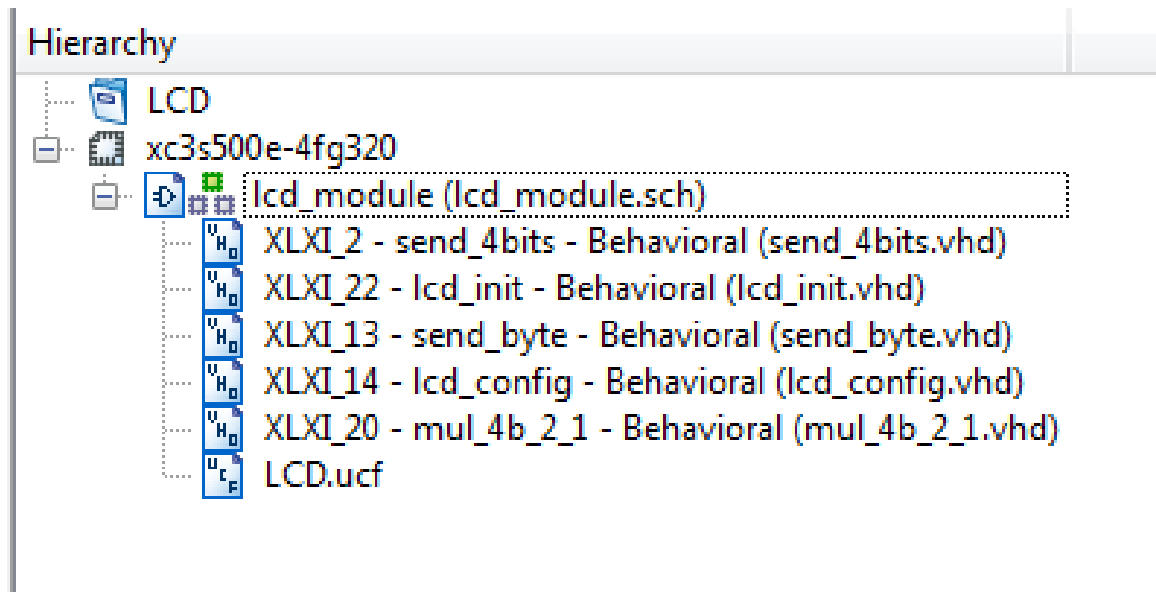
# **2 Opis projektu**

W projekcie zaimplementowano moduł do inicjalizacji i konfiguracji wyświetlacza LCD. Na poniższym schemacie przedstawiony został ogólny schemat ideowy projektu.



Rysunek 1: Schemat projektu

## 2.1 Hierarchia źródeł



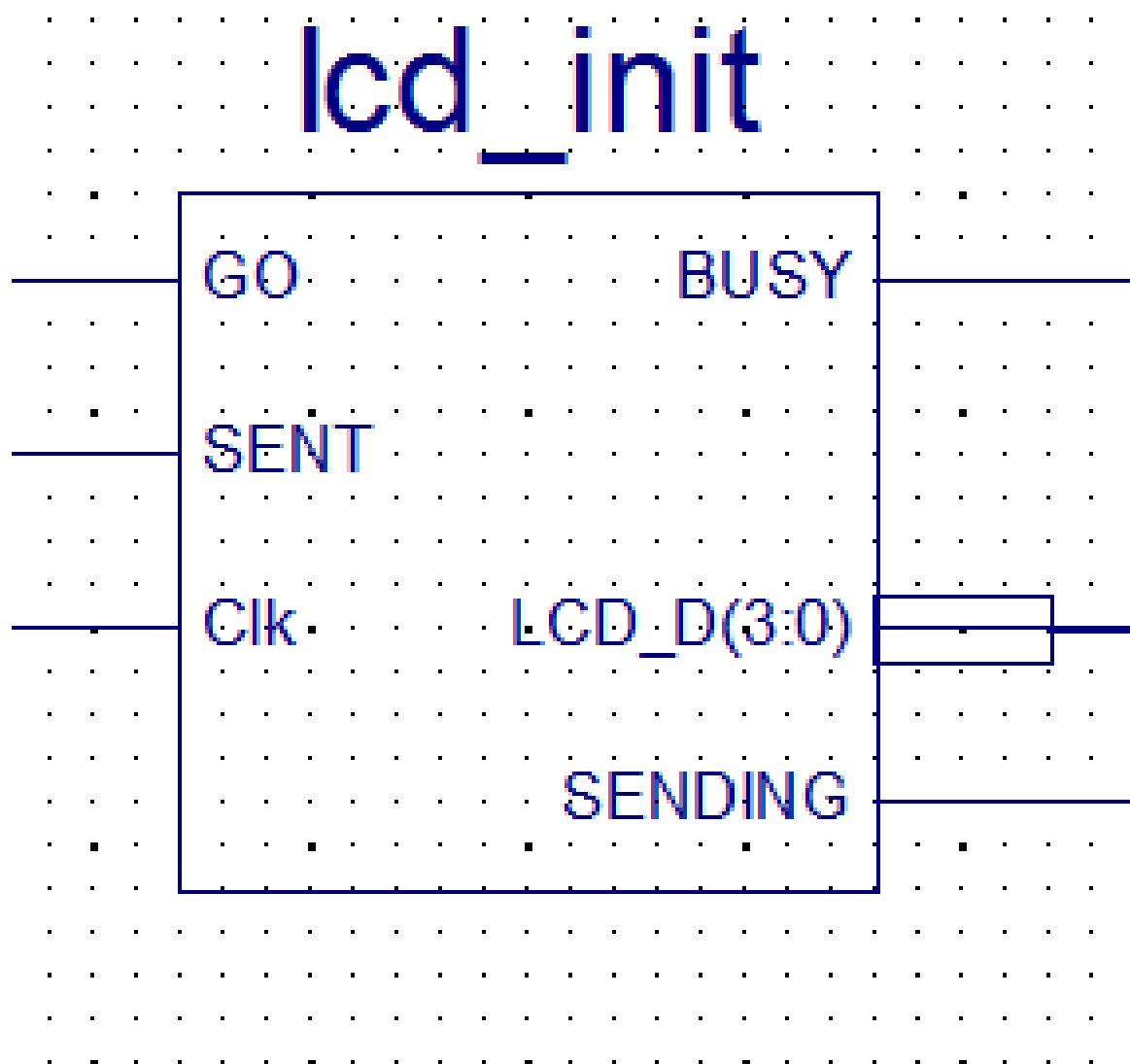
Rysunek 2: Hierarchia źródeł projektu

## 2.2 Opis modułów

Projekt został rozdzielony na poszczególne moduły. Wprowadzenie modularyzacji umożliwiło wyszczególnienie poszczególnych fragmentów, które służą do konkretnych celów. Poniżej opisano poszczególne moduły użyte w projekcie.

### 2.2.1 Moduł inicjalizacji

Moduł inicjalizacji został wyposażony w wejścia GO i SENT, odpowiednio do rozpoczęcia działania maszyny stanów, oraz informowania modułu o zakończeniu wysyłania poszczególnej sekwencji. Wyjściami jest czterobitowa magistrala ustawiająca kolejne wartości inicjalizacyjne. Szczegółowy opis działania maszyny stanów zaimplementowanej w module został przedstawiony poniżej. Wyjście SENDING służy do poinformowania modułu przesyłającego czterobitowe dane do LCD o gotowości na wysłanie. Moduł informuje o zakończeniu działania poprzez opuszczenie stanu wysokiego wyjścia BUSY.

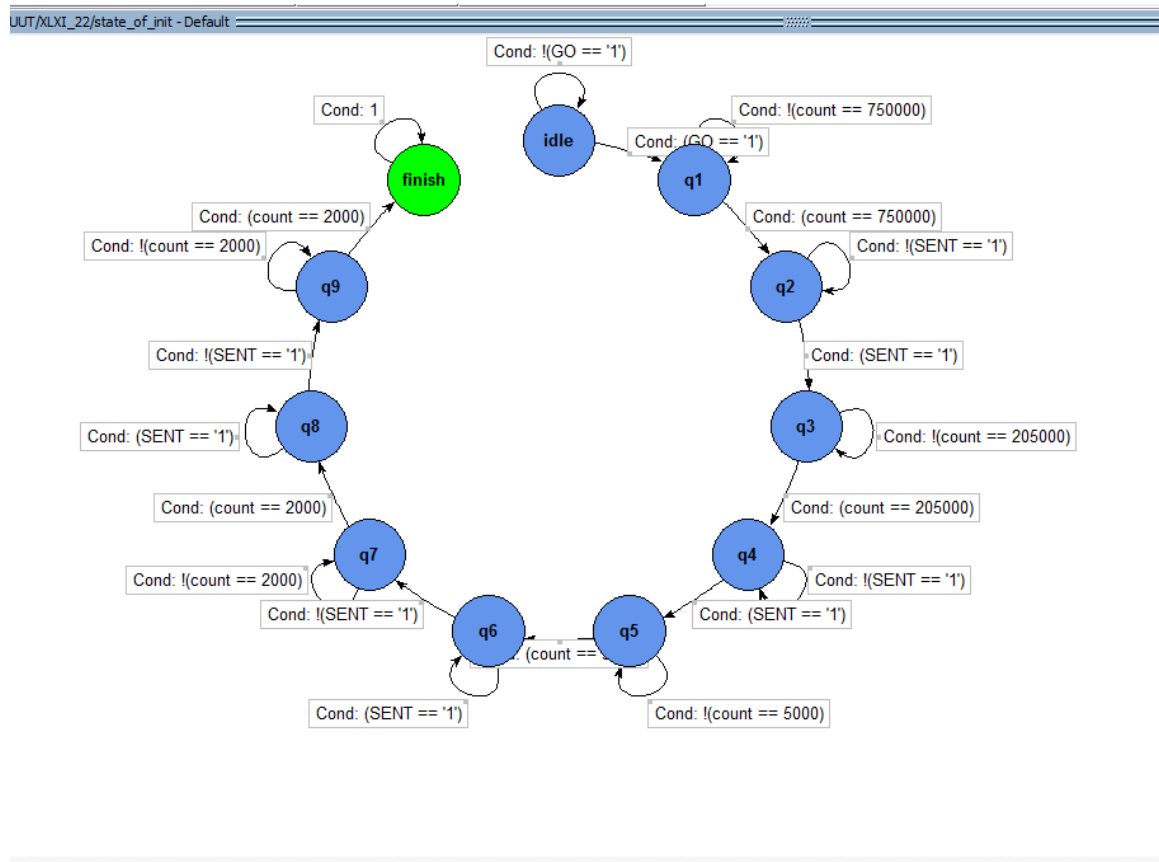


Rysunek 3: Symbol modułu inicjalizacji

Inicjalizacja układu składa z serii oczekiwań i wysłań odpowiednich wartości na magistralę SF\_D<11:8>. Ogólny schemat inicjalizacji został przedstawiony poniżej. Istotna jest linia LCD\_E ponieważ dopiero w przypadku kiedy jej stan jest wysoki, układ LCD przyjmuje dane z magistrali SF\_D. Zastosowanie procedury inicjalizacji jest konieczne do ustabilizowania połączenia interfejsu czterobitowego.

- Czekaj 15ms lub więcej
- Ustaw na SF\_D<11:8> wartość 0x3, ustaw stan LCD\_E na wysoki na 12 cykli
- Czekaj 4.1ms lub więcej
- Ustaw na SF\_D<11:8> wartość 0x3, ustaw stan LCD\_E na wysoki na 12 cykli
- Czekaj 0.1ms lub więcej
- Ustaw na SF\_D<11:8> wartość 0x3, ustaw stan LCD\_E na wysoki na 12 cykli
- Czekaj 0.04ms lub więcej
- Ustaw na SF\_D<11:8> wartość 0x2, ustaw stan LCD\_E na wysoki na 12 cykli
- Czekaj 0.04ms lub więcej

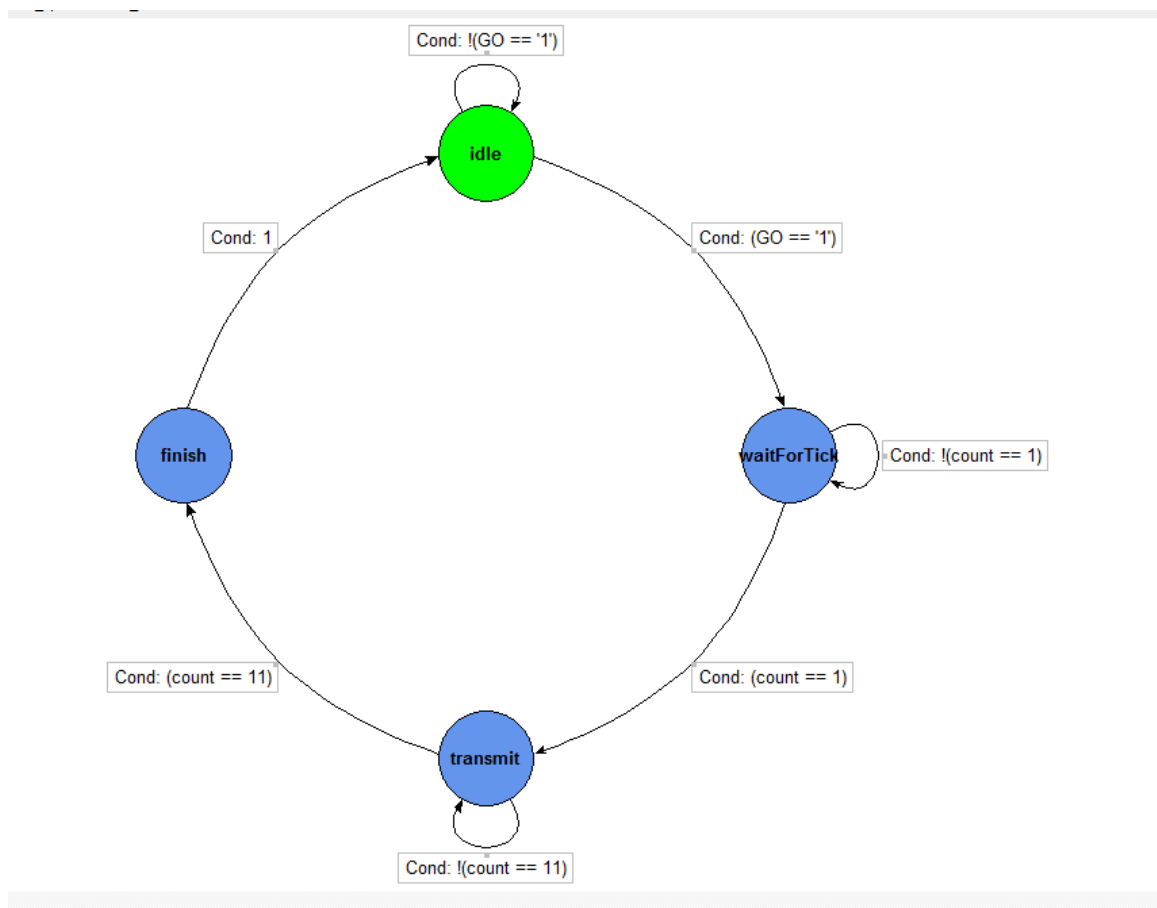
W projekcie został podłączony zegar 50MHz, dzięki czemu ustalone zostały minimalne ilości cykli jakie należy odczekać między kolejnymi wysyłaniami. Moduł inicjalizacyjny jest to maszyna stanów, która odmierza potrzebne wartości czasowe przy wykorzystaniu licznika.



Rysunek 4: Stany modułu inicjalizacyjnego

### 2.2.2 Wysyłanie bajtów i pół-bajtów

Sterownik LCD posiada czterobitowy interfejs. Wysyłanie danych musi odbywać się w odpowiedniej częstotliwości. Zgodnie z dokumentacją, aby wysłać 4 bitowy dane do LCD należy ustawić je na wyjściu SF\_D, oraz ustawić stan niski na LCD\_E na okres 40ns(równy dwóm cyklom zegara). Następnie należy podnieść na 12 cykli zegara wyjście LCD\_E, aby po tym czasie opuścić je na 1 cykl. Zaimplementowana została następująca maszyna stanów:



Rysunek 5: Maszyna stanów wysyłania danych na czterobitowy interfejs.

Poniżej przedstawiono kod maszyny stanów zaimplementowanej w module:

```

entity send_4bits is
Port ( half_byte : in  STD_LOGIC_VECTOR (3 downto 0);
      GO : in  STD_LOGIC;
      Clk: in  STD_LOGIC;
      SF_D : out  STD_LOGIC_VECTOR (3 downto 0);
      LCD_E : out  STD_LOGIC;
      BUSY : out  STD_LOGIC);
end send_4bits;

architecture Behavioral of send_4bits is
  type states_of_4bits_transmission is ( idle, waitForTick, transmit, finish);
  signal transmission_state : states_of_4bits_transmission := idle;

  signal count : integer range 0 to 11 :=0;
  signal data : STD_LOGIC_VECTOR (3 downto 0);
begin
  transmit_4bits: process (Clk, half_byte, GO)
  begin
    if(rising_edge(Clk)) then
      case transmission_state is
        --czekanie na GO
        when idle =>
          SF_D <= "0000";
          LCD_E <= '0';
      end case;
    end if;
  end process;
end Behavioral;

```

```

        BUSY <= '0';

    if(GO = '1') then
        data <= half_byte;
        BUSY <= '1';
        transmission_state <= waitForTick;
    else
        transmission_state <= idle;
    end if;

    --ustawienie data na LCD - 2 tick
    when waitForTick =>
        LCD_E <= '0';
        SF_D <= data;
        if(count = 1) then
            transmission_state <= transmit;
            count <= 0;
        else
            transmission_state <= waitForTick;
            count <= count+1;
        end if;

    --wyslanie polbajtu
    when transmit =>
        SF_D <= data;
        LCD_E <= '1';

        if(count = 11) then
            transmission_state <= finish;
            count <= 0;
        else
            transmission_state <= transmit;
            count <= count+1;
        end if;

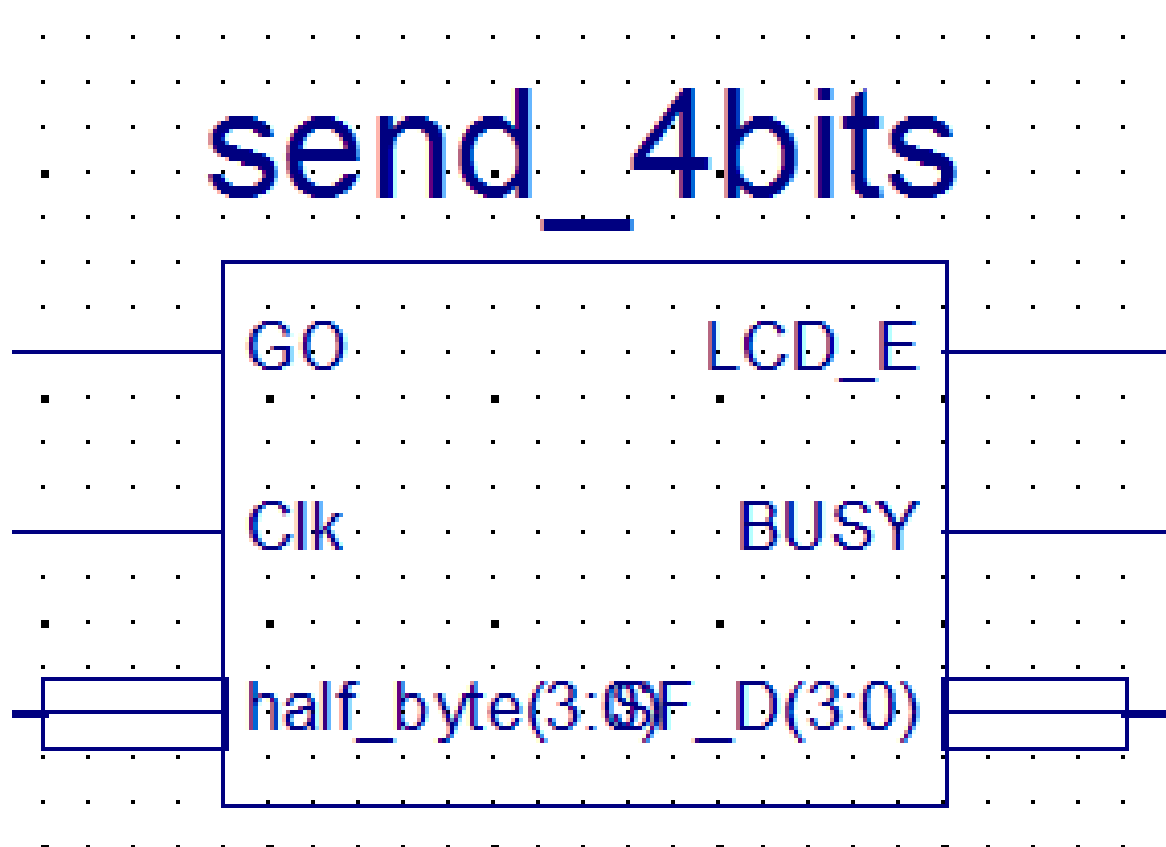
    -- zwolnienie BUSY - powrot do stanu poczatkowego
    when finish =>
        LCD_E <= '0';
        transmission_state <= idle;
    end case;
    end if;
end process transmit_4bits;

end Behavioral;

```

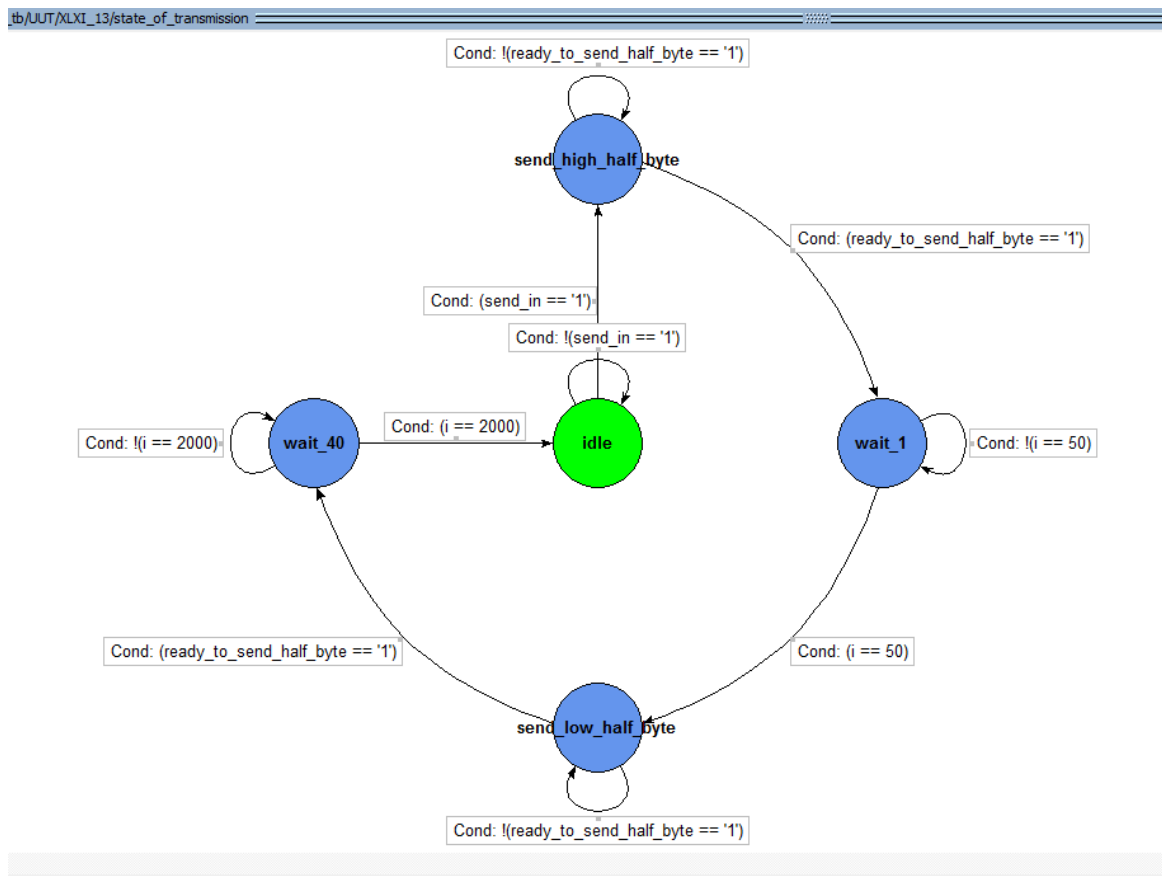
Dla modułu wysyłania czterobitowych danych wygenerowano symbol:



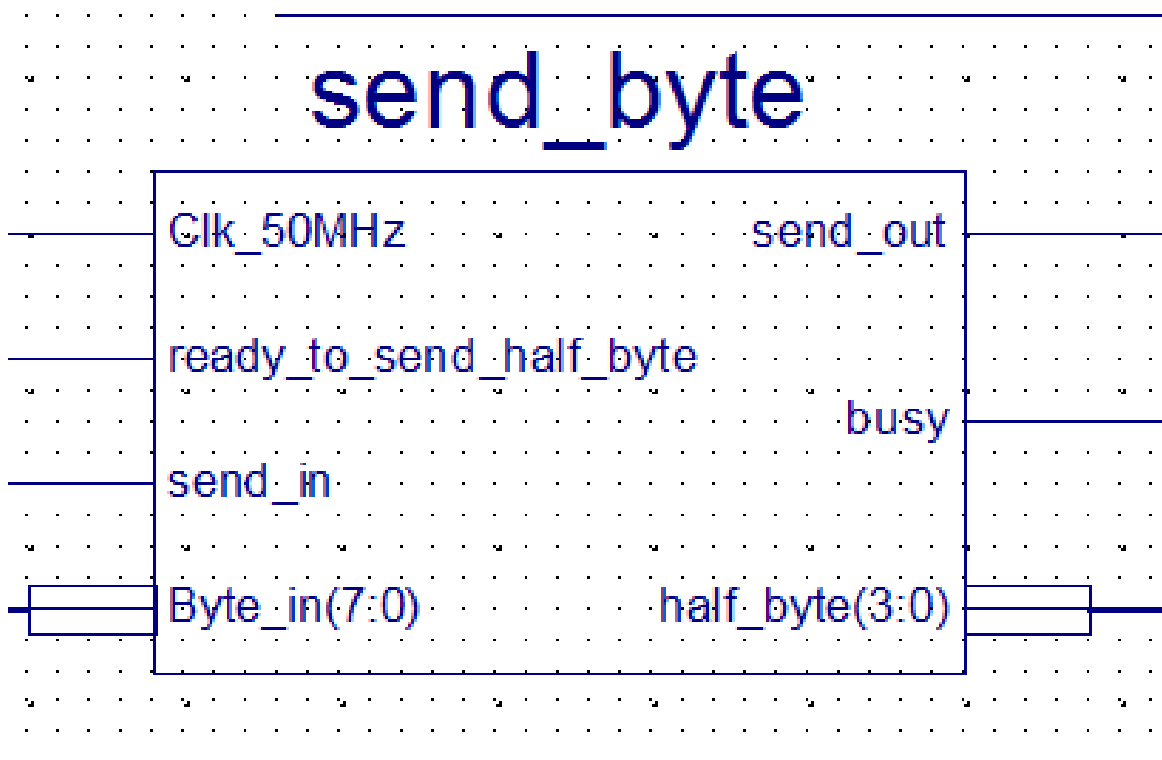


Rysunek 6: Symbol moduły wysyłającego cztery bity

Wysyłanie tylko danych czterobitowych jest jednak niewystarczające. Poprawna konfiguracja, czy wysyłanie danych programowych wymaga przesyłania pełnych bajtów. Problem ten jest opisany w dokumentacji sterownika LCD. Wysyłanie bajtu polega na wysłaniu najpierw jego górnej połówki, odczekaniu jednej mikrosekundy, następnie przesłaniu dolnej połówki bajtu. W tym celu zaimplementowano kolejną maszynę stanów. Korzysta ona z poprzednio opisanego modułu wysyłania czterobitowych danych.



Rysunek 7: Graf przejść maszyny stanów wysyłającej całe bajty

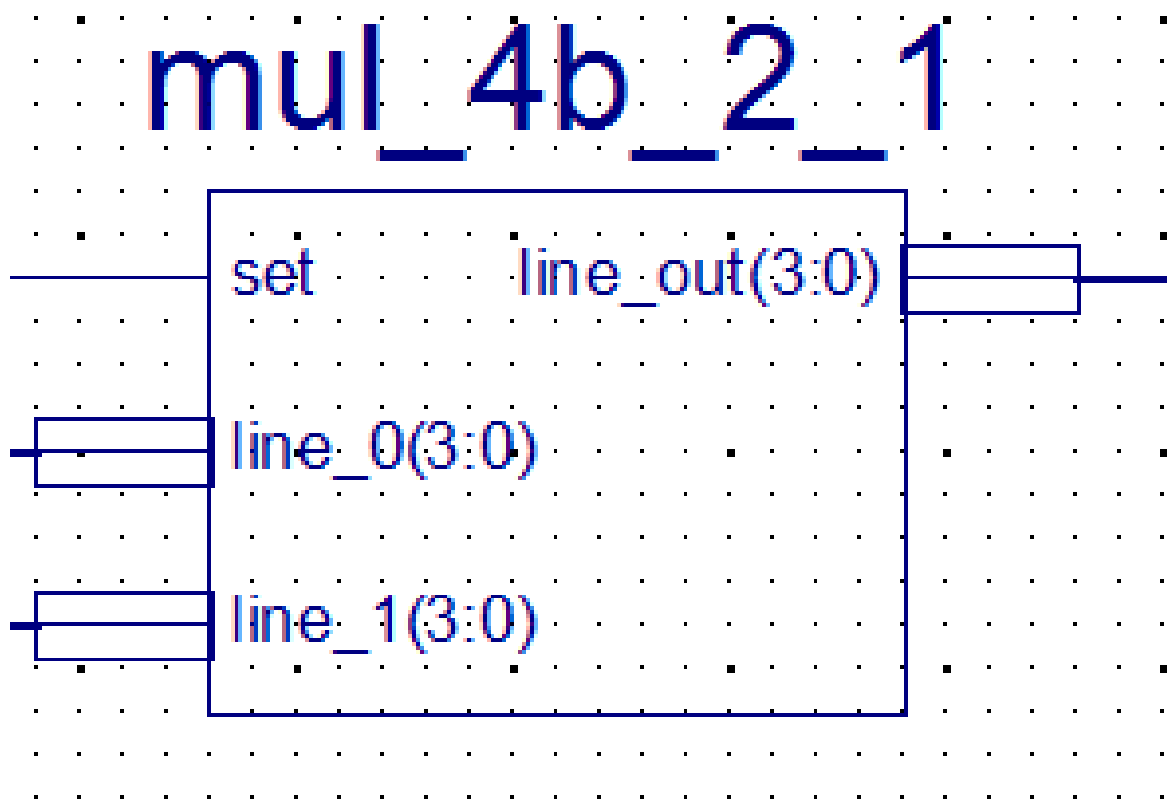


Rysunek 8: Symbol moduły wysyłającego całe bajty

### 2.2.3 Multiplexer 4-bitowy

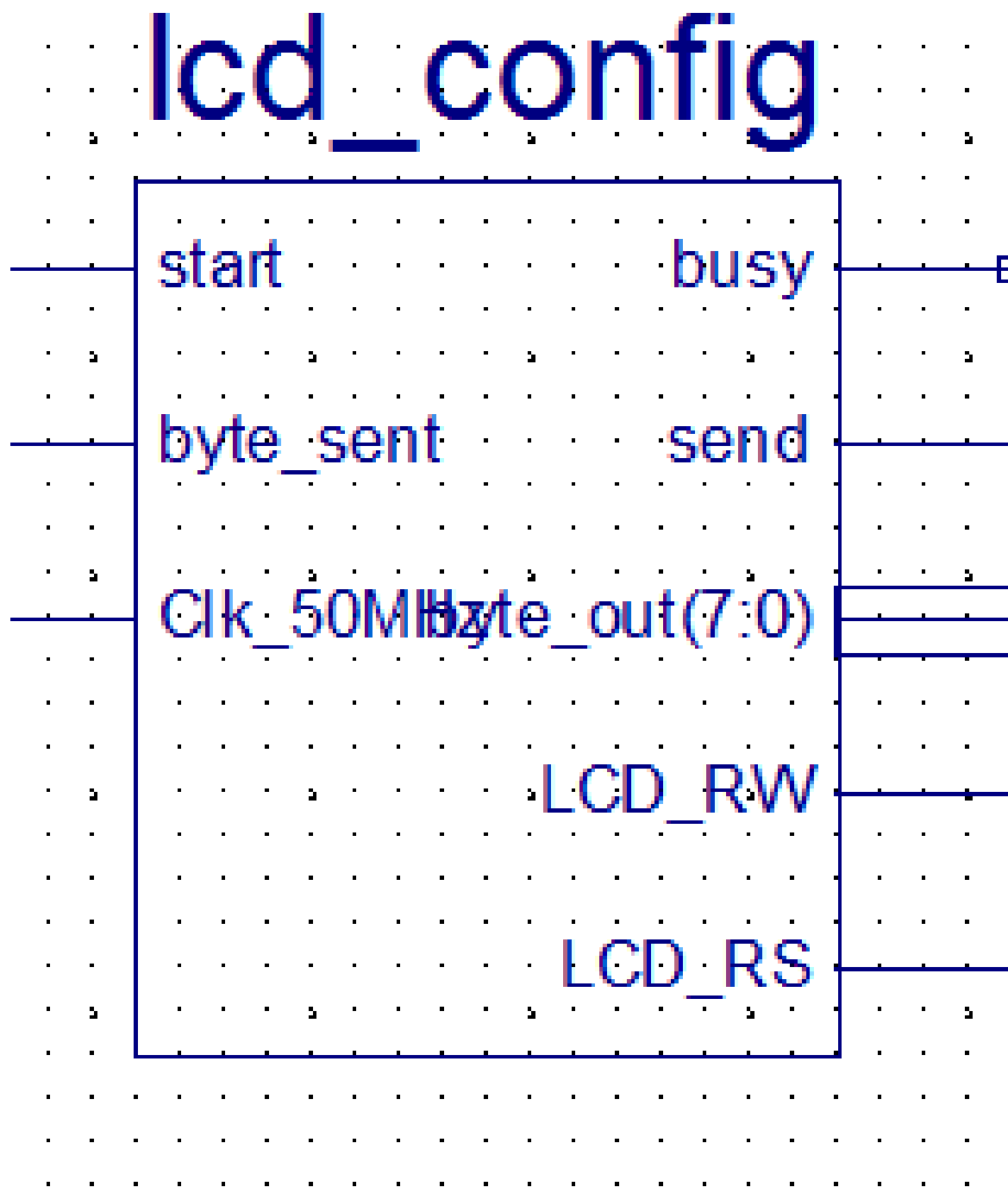
Po zakończeniu pracy przez moduł inicjalizacyjny należy rozpocząć pobieranie danych z modułu konfiguracji. Jednakże w odróżnieniu od pierwszego modułu, moduł konfiguracji wysyła dane konfiguracyjne w postaci pełnych bajtów. Wysyłanie całych bajtów odbywa się zgodnie z dokumentacją. Zatem konieczne jest zastosowanie multiplexera, aby zmienić źródło danych do wysłania. Wyjście multiplexera zależy od stanu linii set.

```
if(set = '1') then
  line_out <= line_0; -- moduł inicjalizacyjny
else
  line_out <= line_1; -- moduł konfiguracyjny
end if;
```



Rysunek 9: Symbol multiplexera

#### 2.2.4 Moduł konfiguracji



Rysunek 10: Symbol modułu konfiguracji

Linie LCD\_RS oraz LCD\_RW, które wchodzą w skład modułu są stale ustawione na stan niski. Linia SF\_CE jest odpowiedzialna za komunikację z pamięcią Intel StrataFlash. Ustawienie tej linii na 1 powoduje dezaktywację tej pamięci. W takim przypadku układ FPGA posiada pełną kontrolę na zapisem i odczytem do LCD.

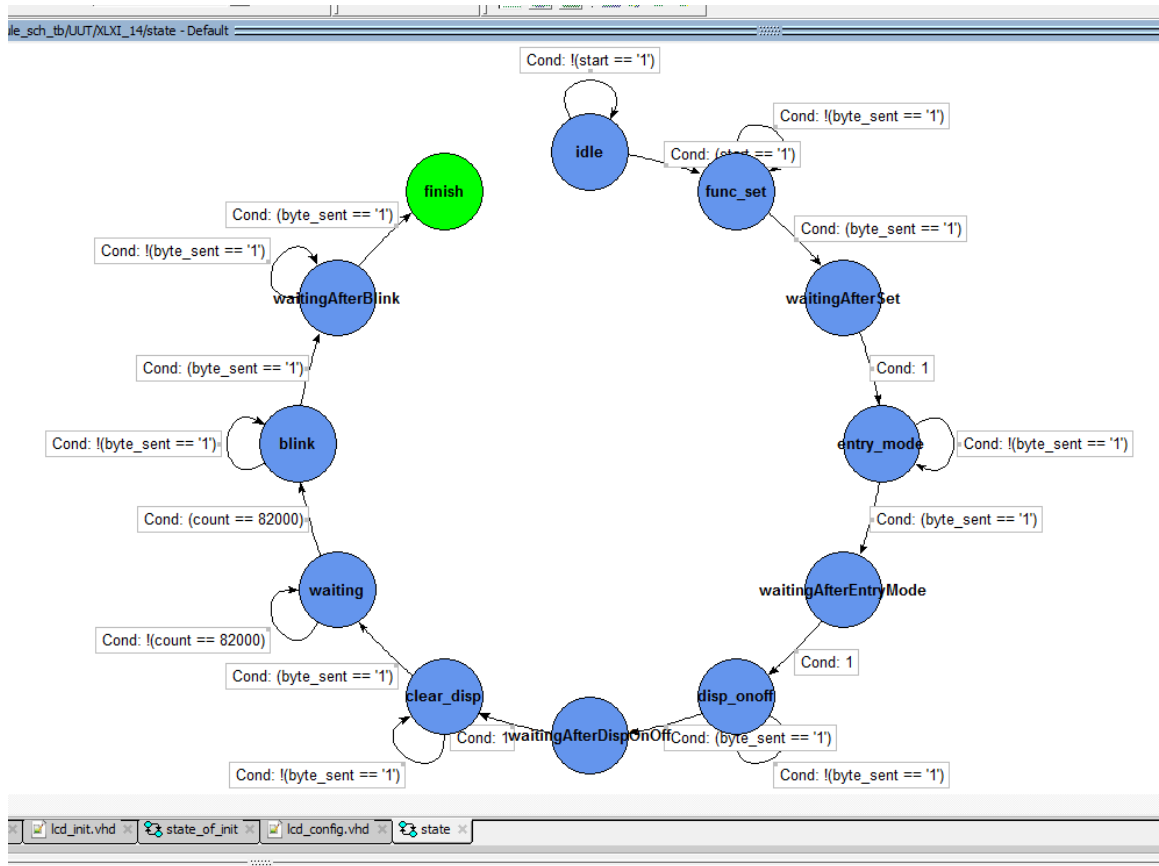
```
LCD_RW <= '0';  
LCD_RS <= '0';  
SF_CE <= '1';
```

Właściwa część konfiguracyjna składa się z wysłania czterech bajtów na magistralę SF\_C<11:8>.

- Function Set - 0x28

- Entry Set - 0x06 w celu inkrementacji pozycji wskaźnika po każdym wpisaniu znaku
- Display On/Off - 0x0E - włącznie migającego wskaźnika
- Clear Display - 1.64ms

Istotną informacją jest, że wykonanie każdej z tych instrukcji zajmuje pewien okres czasu. Dlatego też po wysłaniu każdej komendy występuje stan który odmierza czas między kolejnymi wysłaniami bajtów. Czasy poszczególnych operacji znajdują się w dokumentacji układu Spartan-3E [1].

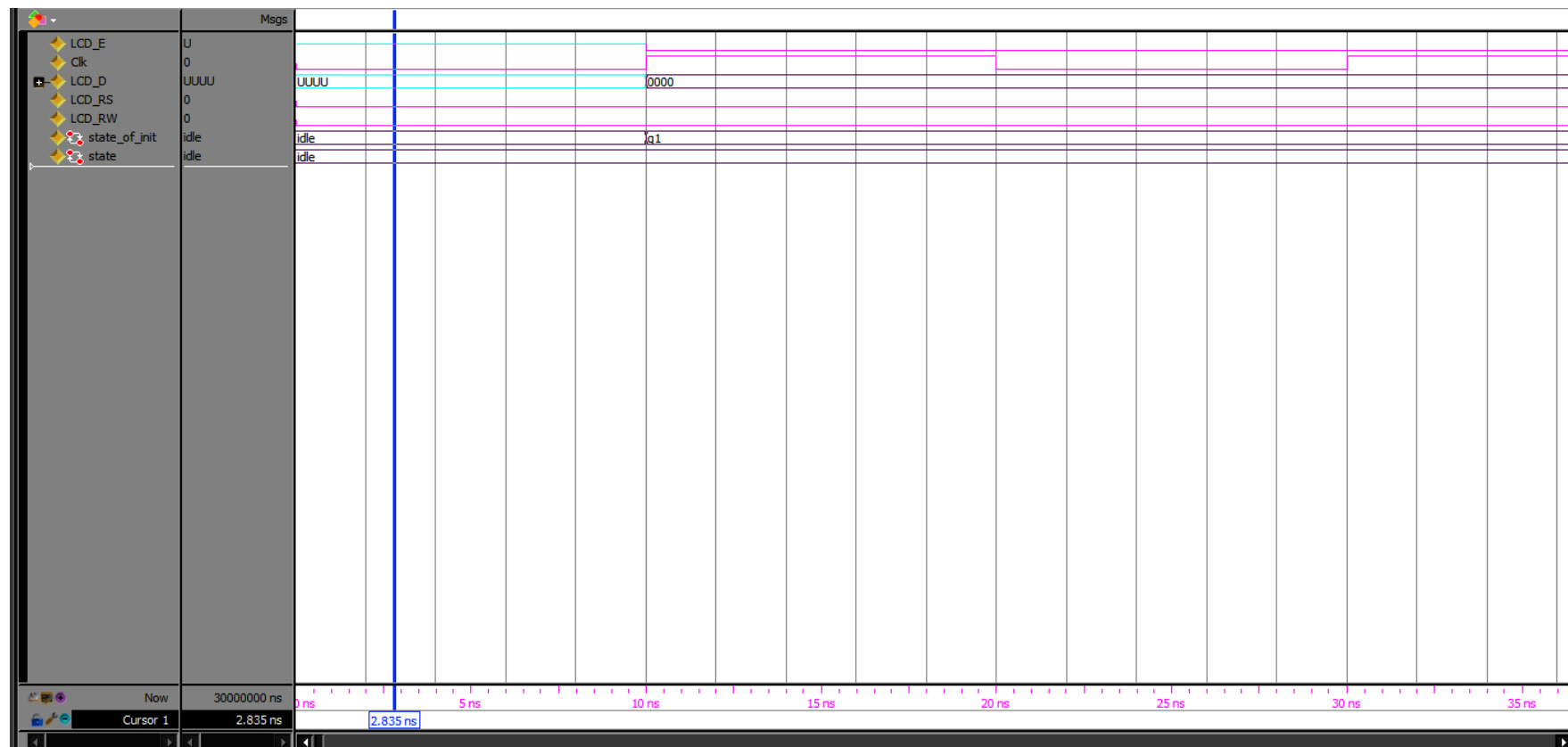


Rysunek 11: Stany modułu konfiguracyjnego

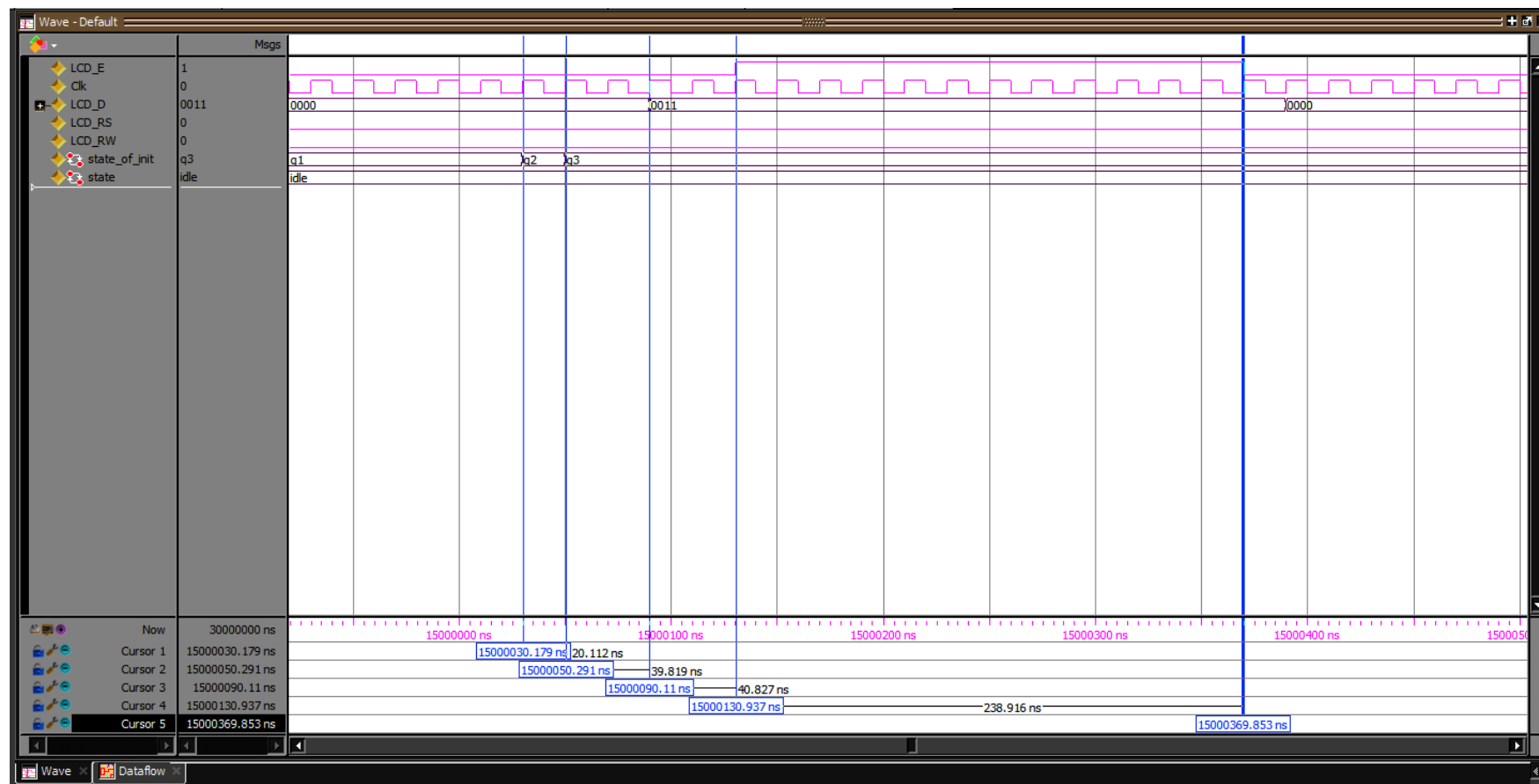
Po stworzeniu każdego modułu, były przeprowadzane testy behawioralne, które miały na celu sprawdzić poprawność działania. W przeciwnym wypadku szukanie potencjalnych błędów byłoby bardzo utrudnione. Moduł stworzony w ramach projektu jest odpowiedzialny za prawidłową inicjalizację i konfigurację wyświetlacza. Jest to bezpośrednio związane z odpowiednim wysyłaniem danych oraz synchronizacją wszystkich procesów. Zastosowane rozwiązania związane są z powiadomieniem zwrotnym *callback*. Oznacza to, że przygotowany moduł, rozwiązujący pojedyncze kwestie implementacji, informował moduł z niego korzystający (kliencki) o zakończeniu pracy. Zastosowanie tego rozwiązania pozwala na uniknięcie problemów z odpowiednią częstotliwością wysyłania danych.

### 2.3 Wyniki symulacji

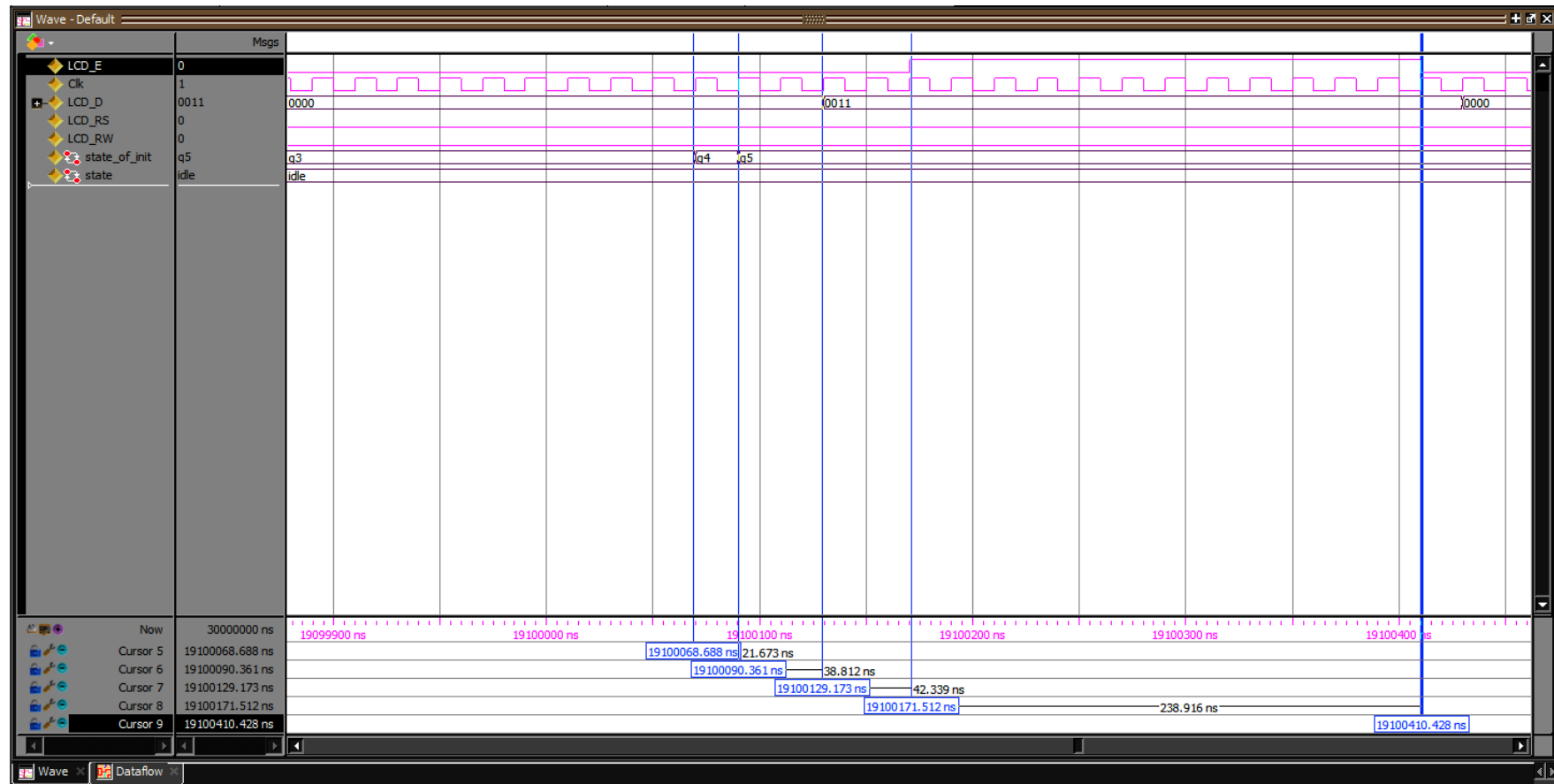
Z racji synchronicznego i stałego charakteru projektu zdecydowano się na testowanie całości modułu w programie ModelSim. Poniżej przedstawiono kluczowe elementy w symulacji.



Rysunek 12: Symulacja - stan początkowy

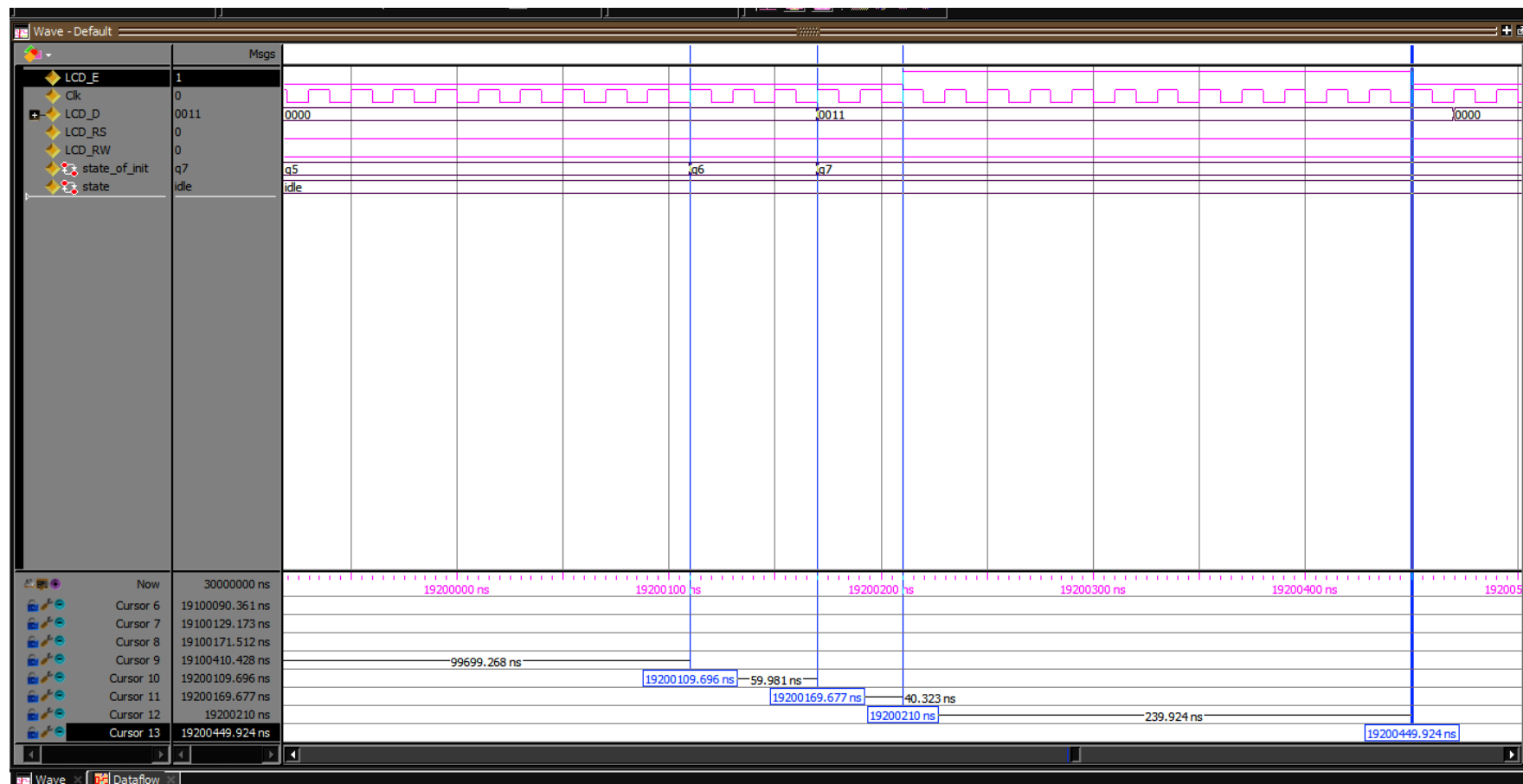


Rysunek 13: Symulacja - działanie modułu inicjalizującego

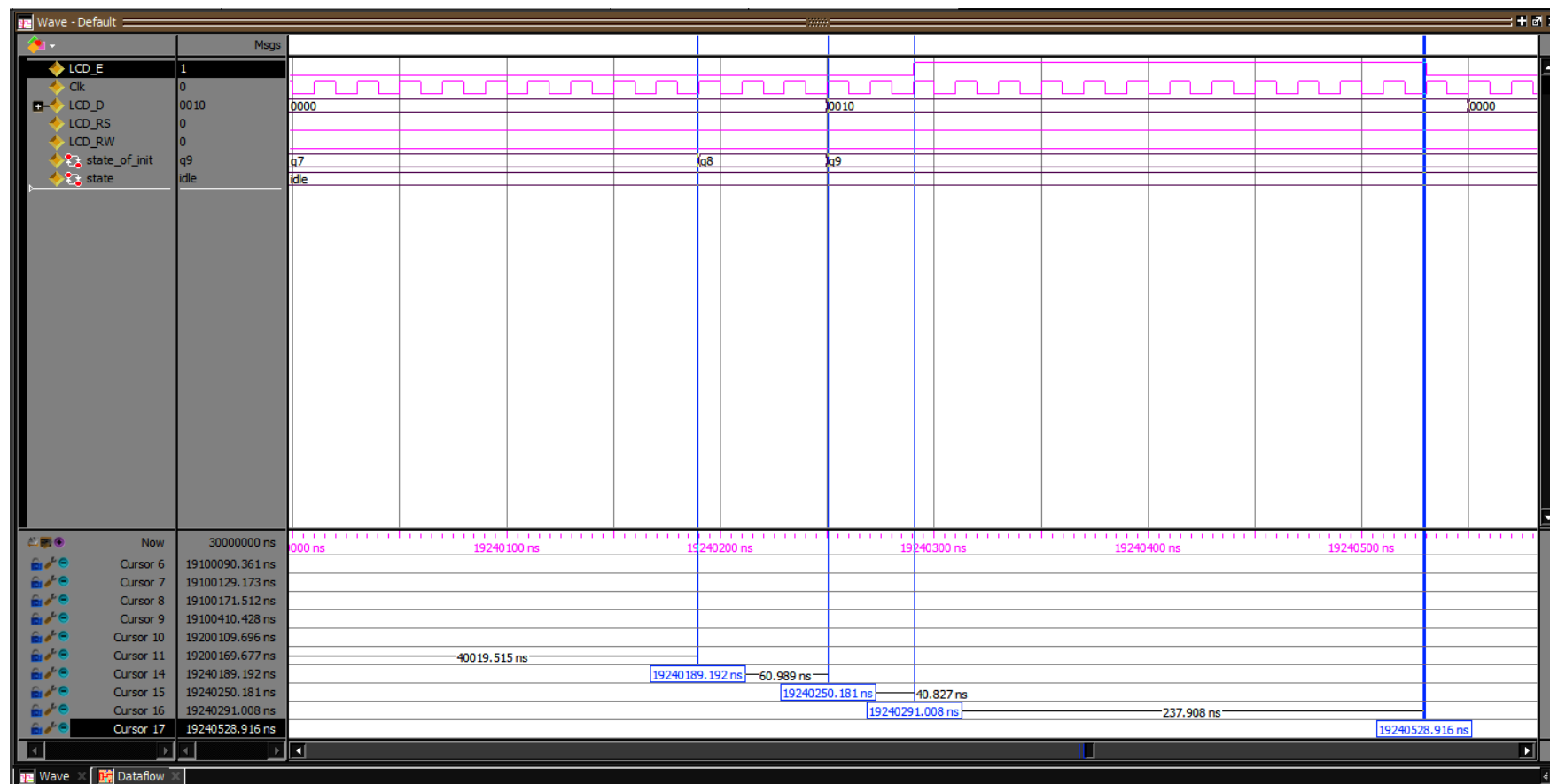


Rysunek 14: Symulacja - działanie modułu inicjalizującego

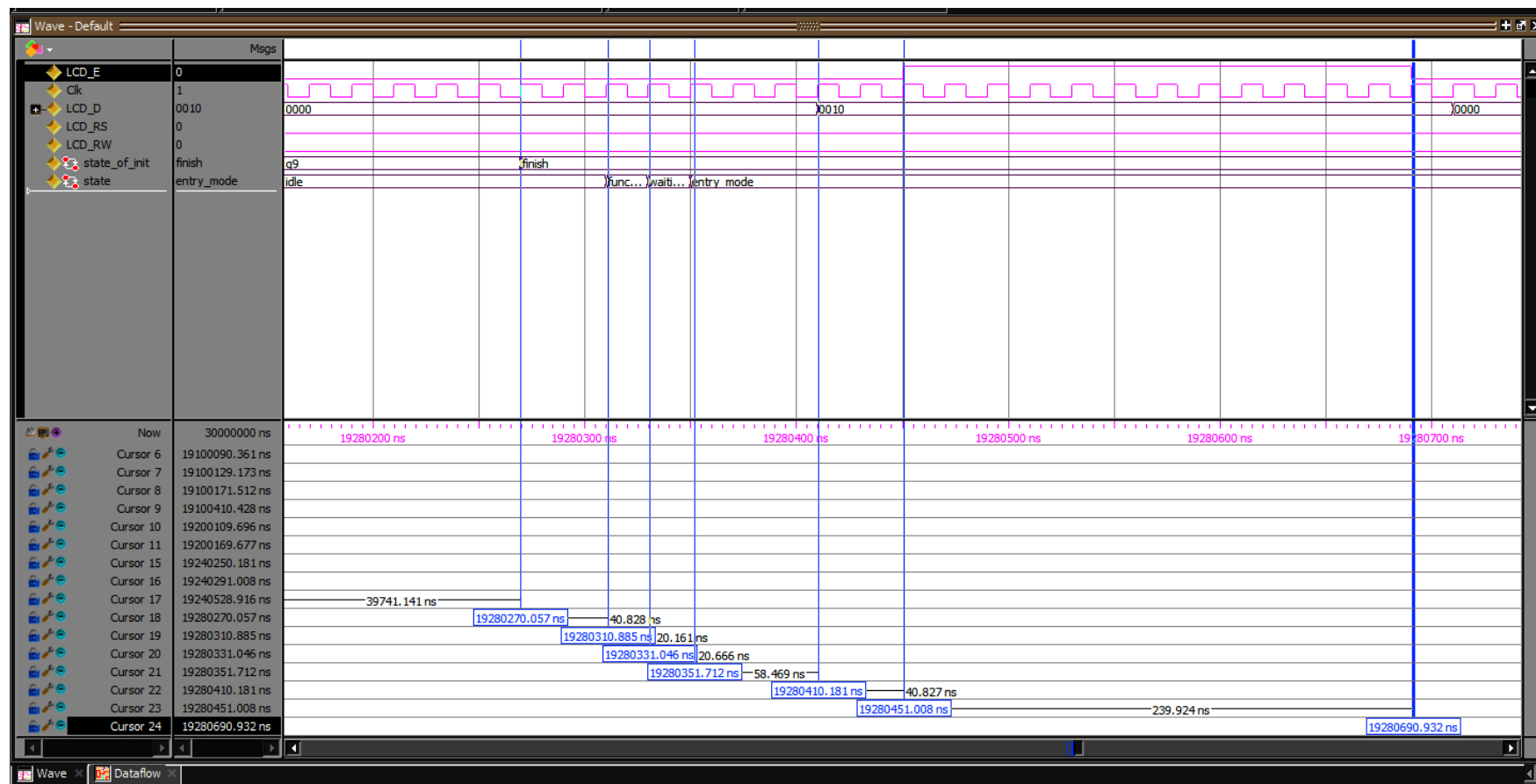




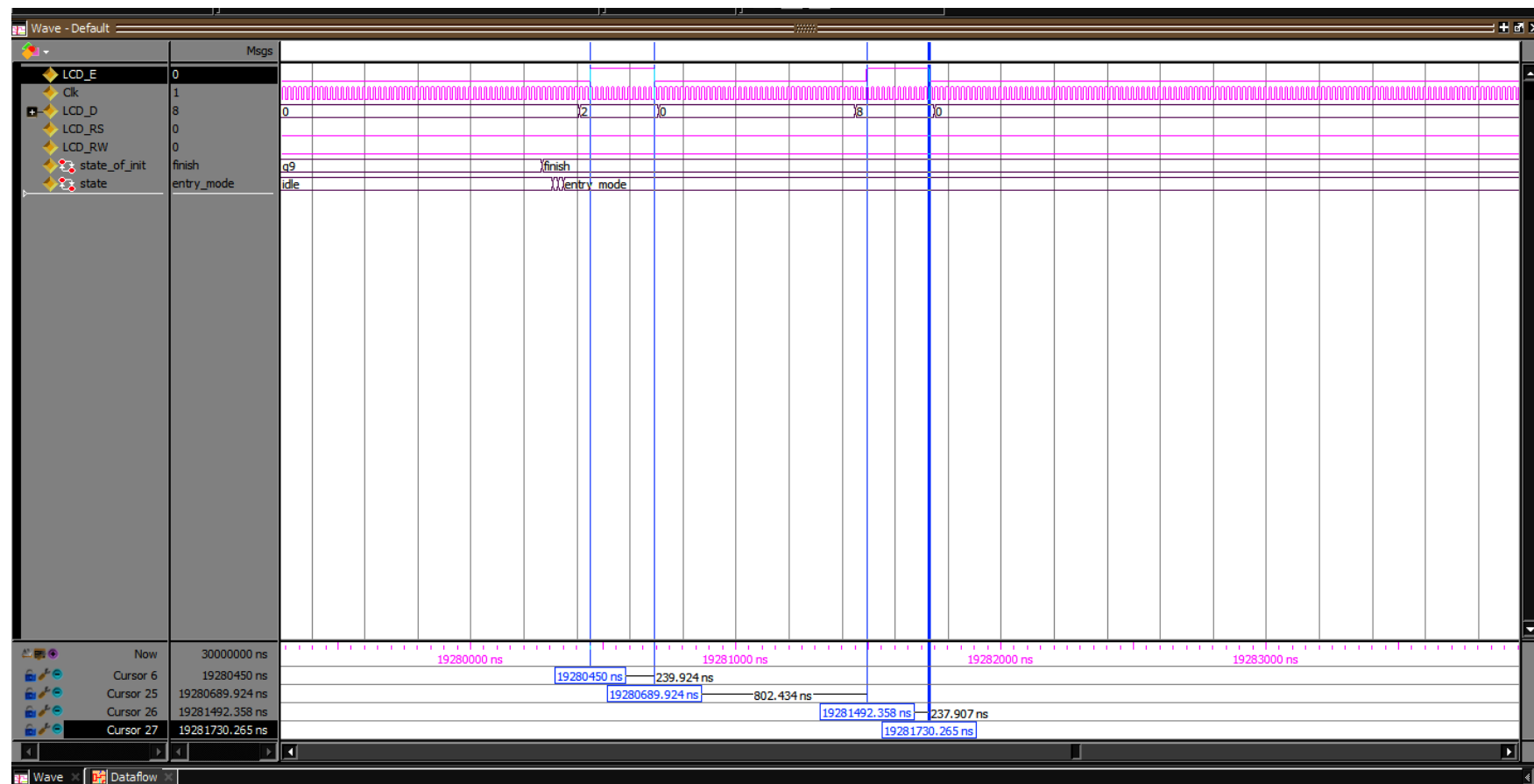
Rysunek 15: Symulacja - działanie modułu inicjalizującego



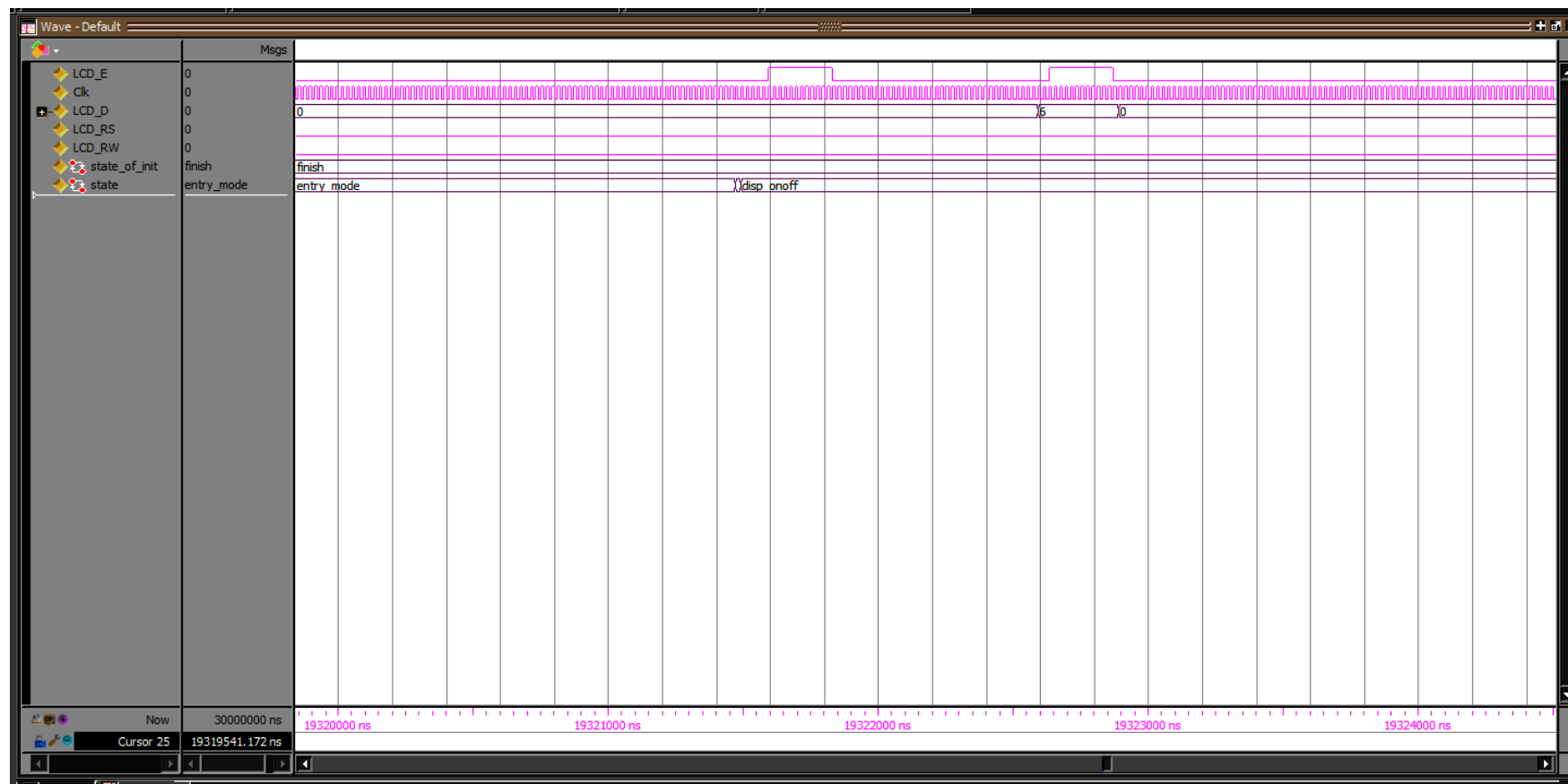
Rysunek 16: Symulacja - działanie modułu inicjalizującego



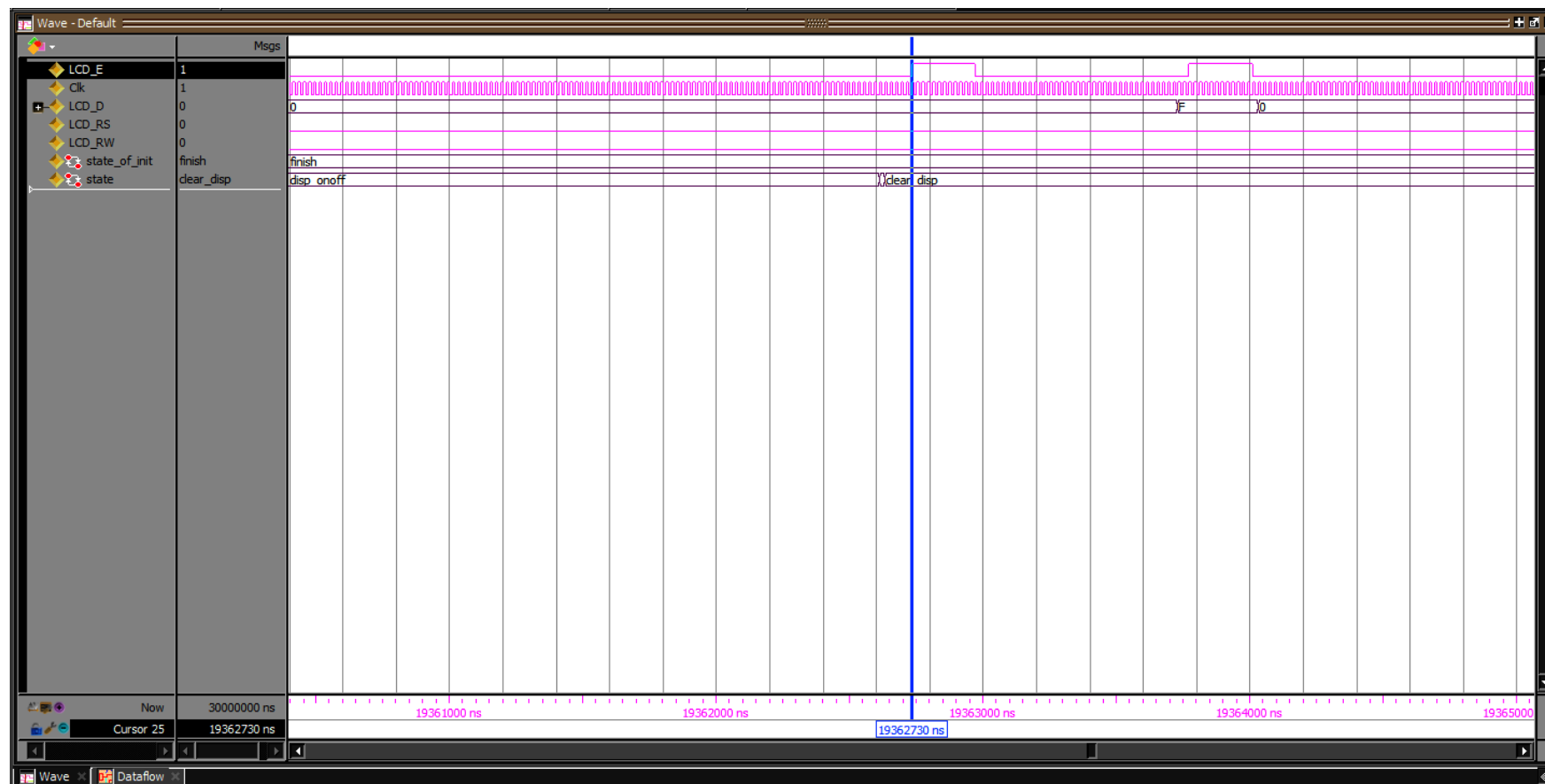
Rysunek 17: Przejście z modułu inicjalizującego do konfiguracji



Rysunek 18: Wysłanie danych 0x28 przez moduł konfiguracji



Rysunek 19: Wysłanie danych 0x06 przez moduł konfiguracji



Rysunek 20: Wysłanie danych 0x0F przez moduł konfiguracji - włączenie kursora

### 3 Implementacja

Projekt został zaimplementowany w środowisku Xlinx w wersji 14.7. Poprawność działania poszczególnych modułów, oraz w fazie końcowej całości projektu była testowana w środowisku ModelSim. Do poprawnej pracy układu niezbędna jest konfiguracja pliku LCD.ucf oraz GenIO.ucf, których listingi zostały przedstawione poniżej.

```
NET "LCD_E" LOC = "M18" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
NET "LCD_RS" LOC = "L18" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
NET "LCD_RW" LOC = "L17" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
NET "LCD_D<0>" LOC = "R15" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
NET "LCD_D<1>" LOC = "R16" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
NET "LCD_D<2>" LOC = "P17" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
NET "LCD_D<3>" LOC = "M15" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;
NET "SF_CE" LOC = "D16" | IOSTANDARD = LVCMOS33 | DRIVE = 4 | SLEW = SLOW ;

NET "Clk" LOC = "C9" | IOSTANDARD = LVTTTL;
NET "Clk" PERIOD = 20.0ns HIGH 50%;
```

#### 3.1 Rozmiar układu

Poniżej przedstawiono raporty wygenerowane przez środowisko Xlinx podczas generowania implementacji. Widać z niej, że projekt wymaga jedynie 168 LUT spośród 9312 dostępnych. Ponadto do implementacji zużyto zaledwie 1% dostępnych przerzutników.

lcd_module Project Status				
Project File:	LCD.xise	Parser Errors:	No Errors	
Module Name:	lcd_module	Implementation State:	Synthesized	
Target Device:	xc3s500e-4fg320	• Errors:		
Product Version:	ISE 14.7	• Warnings:		
Design Goal:	Balanced	• Routing Results:	<a href="#">All Signals Completely Routed</a>	
Design Strategy:	<a href="#">Xilinx Default (unlocked)</a>	• Timing Constraints:	<a href="#">All Constraints Met</a>	
Environment:	<a href="#">System Settings</a>	• Final Timing Score:	0 ( <a href="#">Timing Report</a> )	

Device Utilization Summary					[+]
Logic Utilization	Used	Available	Utilization	Note(s)	
Number of Slice Flip Flops	83	9,312	1%		
Number of 4 input LUTs	138	9,312	1%		
Number of occupied Slices	95	4,656	2%		
Number of Slices containing only related logic	95	95	100%		
Number of Slices containing unrelated logic	0	95	0%		
Total Number of 4 input LUTs	168	9,312	1%		
Number used as logic	138				
Number used as a route-thru	30				
Number of bonded <a href="#">IOBs</a>	9	232	3%		
Number of BUFGMUXs	1	24	4%		
Average Fanout of Non-Clock Nets	3.07				

Performance Summary				[+]
Final Timing Score:	0 (Setup: 0, Hold: 0, Component Switching Limit: 0)	Pinout Data:	<a href="#">Pinout Report</a>	
Routing Results:	<a href="#">All Signals Completely Routed</a>	Clock Data:	<a href="#">Clock Report</a>	
Timing Constraints:	<a href="#">All Constraints Met</a>			

Rysunek 21: Raport generowania implementacji

Detailed Reports						[+]
Report Name	Status	Generated	Errors	Warnings	Infos	
<a href="#">Synthesis Report</a>	Current	Sat May 28 14:55:40 2016	0	<a href="#">30 Warnings (13 new)</a>	<a href="#">2 Infos (0 new)</a>	
<a href="#">Translation Report</a>	Current	Sat May 28 14:55:53 2016	0	0	0	
<a href="#">Map Report</a>	Current	Sat May 28 14:56:03 2016	0	0	<a href="#">2 Infos (0 new)</a>	
<a href="#">Place and Route Report</a>	Current	Sat May 28 14:56:37 2016	0	0	<a href="#">2 Infos (2 new)</a>	
Power Report						
<a href="#">Post-PAR Static Timing Report</a>	Current	Sat May 28 14:56:42 2016	0	0	<a href="#">6 Infos (1 new)</a>	
<a href="#">Bitgen Report</a>	Current	Sat May 28 14:56:52 2016	0	0	0	

Secondary Reports			[+]
Report Name	Status	Generated	
<a href="#">WebTalk Report</a>	Current	Sat May 28 14:56:53 2016	
<a href="#">WebTalk Log File</a>	Current	Sat May 28 14:56:55 2016	

Date Generated: 05/28/2016 - 14:57:31

Rysunek 22: Raport generowania implementacji, część 2

### 3.2 Instrukcja obsługi urządzenia

Z racji charakteru projektu nie wymaga on działania zewnętrznego. Procedura inicjalizacyjna i konfiguracyjna rozpocznie się automatycznie po podłączeniu zasilania.

## 4 Podsumowanie

Projekt zawiera poprawną inicjalizację oraz podstawową konfigurację całego układu LCD, co pozwala na dalszą modyfikację w postaci obsługi wejścia danych od użytkownika. Nie został zrealizowany moduł, który pozwalałby na wyświetlanie znaków wprowadzonych przez użytkownika. Taka funkcjonalność mogłaby zostać zrealizowana poprzez rozszerzenie modułu konfiguracyjnego w postaci dodania dodatkowych stanów. Lepszą opcją byłby jednak stworzenie osobnego modułu który odbierałby bajt ze znakiem od użytkownika i przesyłał go dalej do układu LCD w odpowiednich odstępach czasowych.

## Literatura

- [1] [http://www.xilinx.com/support/documentation/boards\\_and\\_kits/ug230.pdf](http://www.xilinx.com/support/documentation/boards_and_kits/ug230.pdf), dokumentacja układu Szpartan-3E.
- [2] <http://www.sitronix.com.tw/sitronix/product.nsf/Doc/ST7066U?OpenDocument>, dokumentacja sterownika Sitronix ST7066U.