

Laboratorium 4

Układ FPGA

Łukasz Kwinta, Kacper Kozubowski, Ida Ciepiela

maj 2024

Spis treści

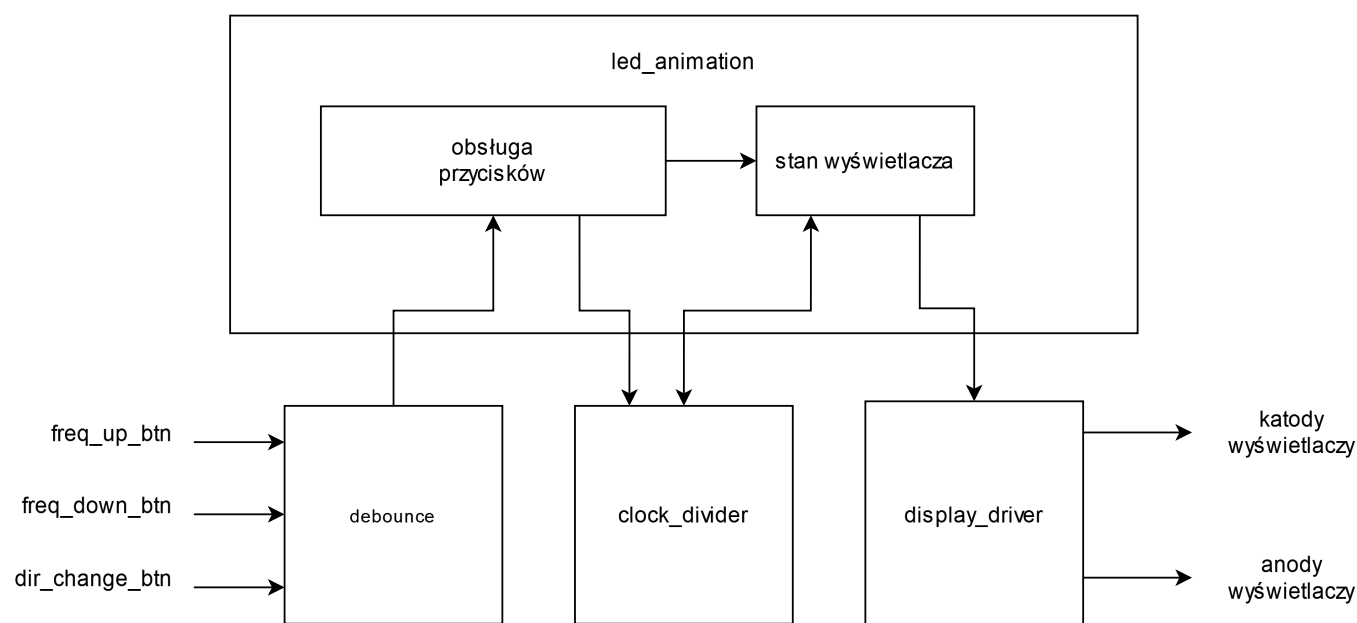
1	Cel zadania	3
2	Czym są układy FPGA?	3
3	Rozwiązanie	3
3.1	Moduł dzielnika częstotliwości	3
3.2	Moduł filtrujący przyciski	4
3.3	Moduł sterujący wyświetlaczami	5
3.4	Właściwy moduł generujący animację	7
4	Zastosowania układów FPGA	7
5	Wnioski	7

1 Cel zadania

Celem laboratorium było zaprogramowanie układu FPGA tak aby wyświetlał animację poruszających się segmentów na wyświetlaczach 7 segmentowych.

2 Czym są układy FPGA?

3 Rozwiązanie



Rysunek 3.1: Schemat blokowy rozwiązania

3.1 Moduł dzielnika częstotliwości

Aby w łatwy sposób zmieniać prędkość animacji, zdecydowaliśmy się na zastosowanie modułu dzielnika częstotliwości. Moduł przyjmuje na wejściu zegar systemowy oraz rejestr oznaczający obecny okres zegara wyjściowego. Następnie zlicza on takty zegara systemowego i gdy licznik dojdzie do zadanej wartości, zmienia stan zegara wyjściowego na przeciwny.

```

1 // moduł dzielący zegar
2 module clock_divider(
3     input integer clock_period,
4     input wire clk,
5
6     output reg divided_clock
7 );
8
9

```

```

10 initial
11     divided_clock <= 0;
12
13 longint counter_value = 0;
14
15 // zliczamy zadany okres zegara (ilość cykli zegara wejściowego), i gdy
16 // doliczymy do końca zmieniamy stan spowolnionego zegara na przeciwny
17 always@ (posedge clk)
18 begin
19     if (counter_value >= clock_period)
20     begin
21         divided_clock <= ~divided_clock;
22         counter_value <= 0;
23     end
24     else
25     begin
26         divided_clock <= divided_clock;
27         counter_value <= counter_value + 1;
28     end
29 end
30
31 endmodule

```

3.2 Moduł filtrujący przyciski

Aby uniknąć efektu drgania styków przycisków, zdecydowaliśmy się na zastosowanie modułu filtrującego wejście przycisków. Działa on na bardzo prostej zasadzie - zlicza on ilość cykli zegara systemowego w których przycisk jest w stanie wysokim - wciśnięty. Gdy ilość cykli przekroczy zadaną wartość, przycisk uznawany jest za wciśnięty. Każdy stan niski pomiędzy kolejnymi resetuje licznik. Długość odliczania można ustawić poprzez parametr DEBOUNCE_TIME przy instancjonowaniu modułu.

```

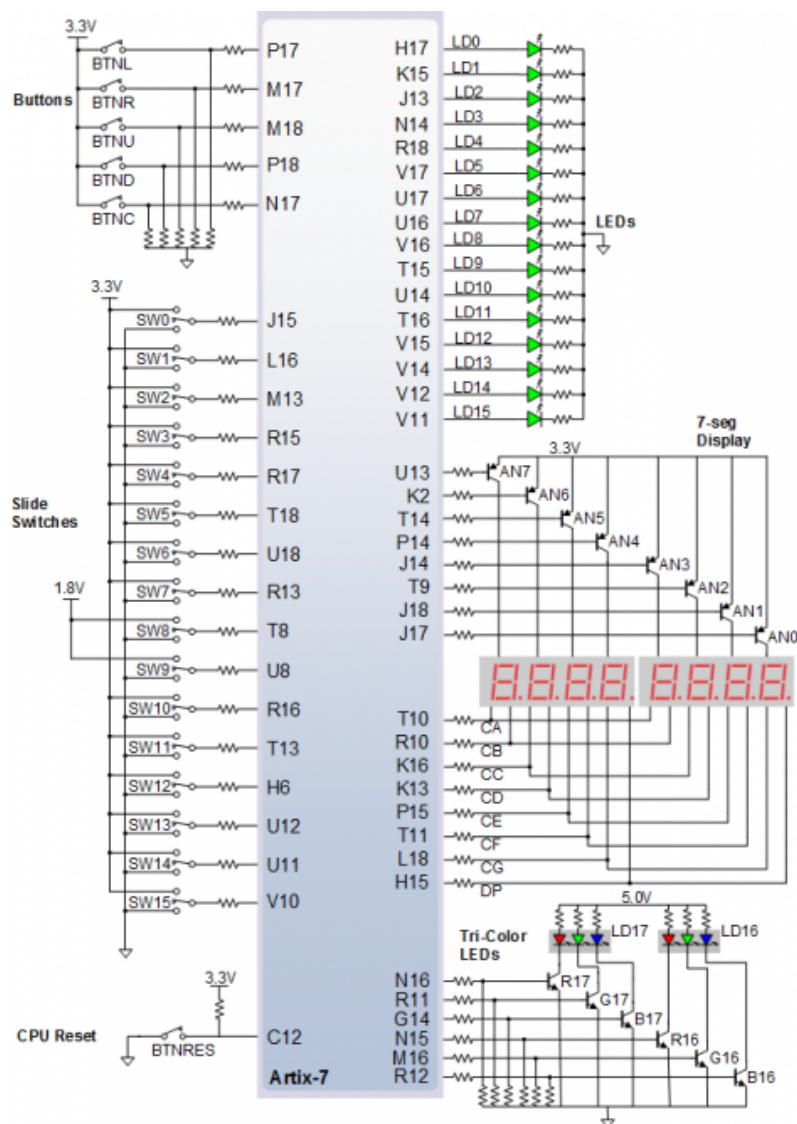
1 // moduł filtrujący przyciski
2 module debounce #(parameter DEBOUNCE_TIME = 1000 * 100) (
3     input wire clk,
4     input wire button_physical,
5
6     output reg button_active
7 );
8
9 // ustawiamy początkowy stan przycisku na 0
10 initial
11     button_active = 0;
12
13 integer btn_clock_cycles_counter = 0;
14
15 // zliczamy zadaną ilość cykli zegara
16 // jeśli w którymś cyklu przycisk będzie w stanie niskim,
17 // resetujemy licznik wartości
18 always@ (posedge clk)
19 begin
20     if (button_physical == 1)

```

```
21     begin
22         btn_clock_cycles_counter <= btn_clock_cycles_counter + 1;
23         if (btn_clock_cycles_counter >= DEBOUNCE_TIME)
24             button_active <= 1;
25         end
26     else
27         begin
28             btn_clock_cycles_counter <= 0;
29             button_active <= 0;
30         end
31     end
32 end
33
34 endmodule
```

3.3 Moduł sterujący wyświetlaczami

Aby wyświetlać wiele segmentów wielu wyświetlaczach 7 segmentowych na raz, musieliśmy zaimplementować moduł sterujący wyświetlaczami. Moduł ten odświeża wyświetlacze z zadaną częstotliwością po kolei, tak aby stworzyć wrażenie, że wiele wyświetlaczy aktywnych jest w jednym czasie.



Rysunek 3.2: Schemat podpięcia wyświetlaczy do układu FPGA

Zabieg ten musieliśmy zastosować gdyż wszystkie wyświetlacze mają wspólne katody segmentów co oznacza, że przy aktywacji anody wielu wyświetlaczy w jednym czasie, będą one wyświetlać te same segmenty.

```

1 module displays_driver #(parameter REFERESH_PERIOD = 100 * 1000)(
2     input wire clk,
3
4     // rejestr wejściowy określający stan wszystkich wyświetlaczy
5     input reg [7:0] display [7:0],
6
7     // wyjścia sterujące wyświetlaczami
8     // stan niski na danym indeksie aktywuje dany wyświetlacz
9     output reg [7:0] sseg_anodes,
10
11     // | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
12     // | DP | CG | CF | CE | CD | CC | CB | CA |
13     // stan niski na danym indeksie aktywuje dany segment na wszystkich aktywnych wyświetlaczach

```

```
14     output reg [7:0] sseg_cathodes
15 );
16
17 initial
18 begin
19     sseg_anodes <= '1;
20     sseg_cathodes <= '1;
21 end
22
23 reg refresh_clk; //1 khz refresh clock
24 clock_divider clk_div (
25     .clock_period(REFERESH_PERIOD),
26     .clk(clk),
27     .divided_clock(refresh_clk)
28 );
29
30 reg [3:0] display_number = 0;
31
32 always@ (posedge refresh_clk)
33 begin
34     sseg_anodes = '1;
35     sseg_anodes[display_number] <= 0;
36     sseg_cathodes <= 8'(~display[display_number]);
37
38     display_number = display_number + 1;
39     if (display_number >= `DISPLAY_COUNT)
40         display_number <= 0;
41 end
42
43 endmodule
```

3.4 Właściwy moduł generujący animację

1

4 Zastosowania układów FPGA

5 Wnioski