Technika Cyfrowa - Ćwiczenie 4

Mateusz Furga

Maj 2022

1 Zadanie

Celem zadania jest zbudowanie prostego kalkulatora pozwalającego na dodanie i pomnożenie dwóch 4-bitowych liczb opartego na dowolnym układzie FPGA.

2 Wprowadzenie do układów FPGA

FPGA (field-programmable gate array) to programowalne układy logiczne składające się z tysięcy konfiguracyjnych bloków logicznych (configurable logic blocks), które są odpowiednio ze sobą połączone aby otrzymać pożądany układ. Sposób tych połączeń definiujemy w językach opisu sprzętu (HDL) takich jak: VHDL i Verilog. Stanowią one warstwę abstrakcji pomiędzy sprzętem oraz znacząco przyspieszają i ułatwiają projektowanie układu.

Najważniejsze zalety układów FPGA:

1. Pozwalają na budowę przeróżnych układów elektronicznych np. własnego procesora, układu kryptograficznego czy systemu czasu rzeczywistego.
2. Są o rzędy wielkości szybsze od mikrokontrolerów czy procesorów, ponieważ wykonują obliczenia równolegle oraz nie dekodują instrukcji kodu maszynowego. Sam "program"jest zapisany w postaci odpowiednich połączeń bramek logicznych.
3. Są bardzo elastyczne, ponieważ pozwalają na łatwą zmianę wewnętrznej konfiguracji układu FPGA, dzięki czemu wykonanie zmian w produkcie nie wymaga modyfikacji innych elementów oraz zmiany schematu.
4. Posiadają dużą liczbę portów I/O dlatego nowe układy są produkowane w technologii BGA, która pozwala ograniczyć ilość zajmowanego miejsca przez układ scalony.

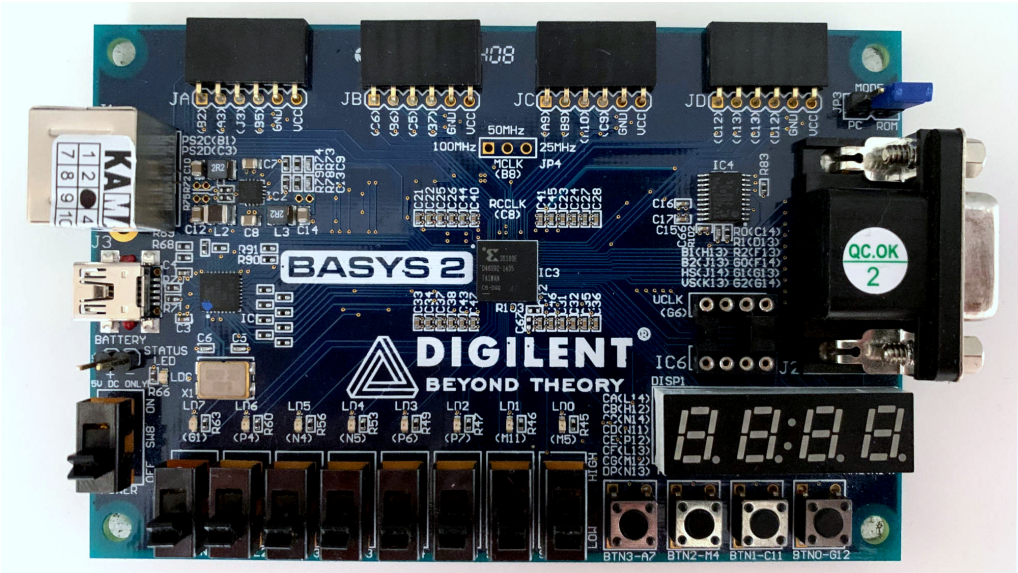
Układy FPGA są wykorzystywane m.in. w:

1. W przetwornikach cyfrowo-analogowych oraz przetwarzaniu sygnałów.
2. Wojsku oraz lotnictwie.
3. Projektowanieu układów ASIC. Gotowe układy pracujące na FPGA przekształca się na dedykowane wykonując tak zwane twarde kopie.
4. Astronomii i kosmonautyce np. układy FPGA były wykorzystane w misji na Marsa w łazikach Spirit i Opportunity.

3 Realizacja zadania

3.1 Wykorzystany układ FPGA

Zadanie zrealizowałem na płytce Basys 2, który bazuje na FPGA firmy Xilinx Spartan 3E XC3S100E.



Rysunek 1: Płytka Basys 2.

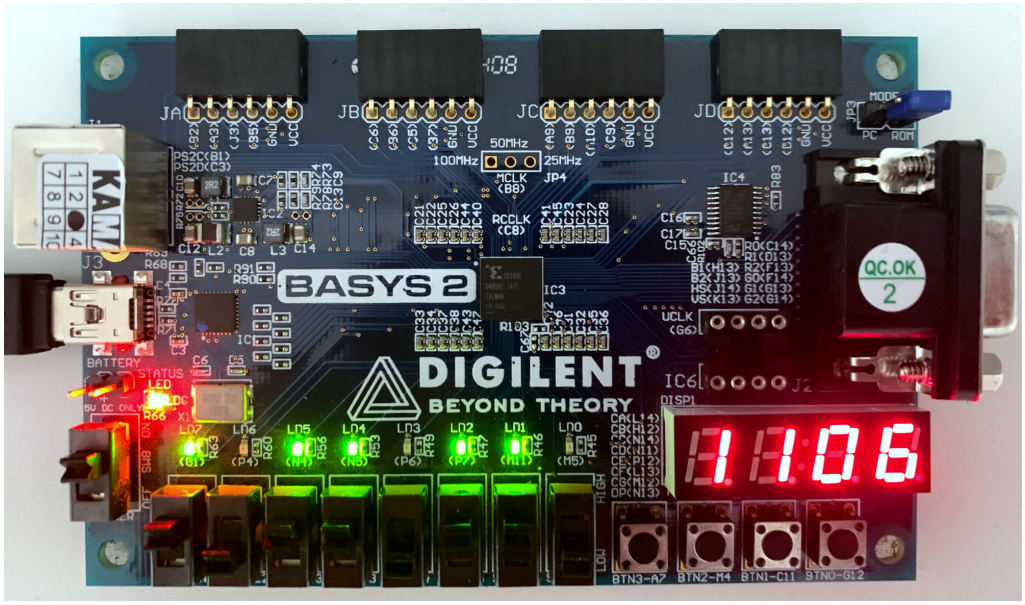
W skład płytki Basys 2 wchodzą następujące elementy:

1. Xilinx Spartan 3E XC3S100E FPGA
2. Atmel AT90USB2 USB2 - dostarczający zasilanie oraz interfejs programistyczny
3. Xilinx Platform Flash ROM - pamięć Flash, w której znajduje się konfiguracja FPGA wgrywana przy starcie działania
4. 8 LEDs, 4 wyświetlacze siedmiosegmentowe, 4 przyciski (push-button), 8 przełączników (switchy)
5. Port PS/2 oraz 8 bitowy port VGA
6. Regulowany zegar o częstotliwości taktowania 25Mhz, 50Mhz i 100Mhz

Sam Xilinx Spartan 3E to niedrogie FPGA posiadające 240 układów CLB oraz 108 portów I/O.

3.2 Opis rozwiązania

Aby wprowadzić 2 liczby do kalkulatora korzystamy z 8 switchy, po 4 dla każdej z liczb co daje zakres 0 - 15 możliwych wartości. Wprowadzone liczy wyświetlają się w postaci dziesiętnej na 4 wyświetlaczach siedmiosegmentowych po 2 dla każdej z liczby.



Rysunek 2: Wprowadzenie 2 liczb: 11 i 6 oraz jednoczesne wyświetlenie ich na 4 wyświetlaczach siedmiosegmentowych.

Wprowadzone liczby l i r są przechowywane w 11-bitowym rejestrze *bin* w postaci: $100l + r$. Następnie przy pomocy modułu *bin2bcd* liczba *bin* jest konwertowana na 16 bitową w reprezentacji BCD, która potem jest wyświetlana przez moduł *seg7* na 4 wyświetlaczach siedmiosegmentowych.

W celu dodania dwóch liczb do siebie należy przytrzymać prawy przycisk (G12), co spowoduje że do rejestru *bin* powędruje suma: $l + r$. Natomiast aby pomnożyć 2 liczby należy przytrzymać lewy przycisk (C11), wtedy w rejestrze *bin* znajdzie się ich iloczyn: $l \cdot r$. Każda zmiana rejestru *bin* jest od razu widoczna na wyświetlaczach siedmiosegmentowych.

Moduł *bin2bcd* konwertuję 11-bitową liczbę w postaci binarnej na 16-bitową w postaci BCD wykorzystując algorytm *double dabble*.

Moduł *seg7* jest odpowiedzialny za kontrolę nad wyświetlaczem. Każdy z 4 wyświetlaczy jest podłączony do wspólnych 7 katod, które stanowią input do układu. Skutkiem tego jest powielanie tej samej cyfry na każdym z nich. Aby wyświetlić różne cyfry, moduł *seg7* przełącza cyklicznie każdy z 4 wyświetlaczy, tak aby w danym momencie działał tylko jeden i wyświetlał odpowiednią cyfrę. Częstotliwość przełączania wyświetlaczy jest większa niż reakcja ludzkiego oka wynikiem czego na wyświetlaczu widzimy jednocześnie 4 cyfry.

3.3 Implementacja

Układ zaimplementowałem w języku Verilog. W celu wygenerowania pliku .bit użyłem programu Xilinx ISE 14.7 oraz djtgcfg firmy Digilent do zaprogramowania pamięci Flash.

```
module bin2bcd
#(parameter BUS_WIDTH = 11)(
  input wire [BUS_WIDTH:0] bin,
  output reg [15:0] bcd
);

  integer i;

  always @ (bin) begin
    bcd = 0;
    for (i = 0; i <= BUS_WIDTH; i = i + 1) begin
      if (bcd[3:0] >= 5) bcd[3:0] = bcd[3:0] + 3;
      if (bcd[7:4] >= 5) bcd[7:4] = bcd[7:4] + 3;
      if (bcd[11:8] >= 5) bcd[11:8] = bcd[11:8] + 3;
      if (bcd[15:12] >= 5) bcd[15:12] = bcd[15:12] + 3;
      bcd = { bcd[14:0], bin[BUS_WIDTH - i] };
    end
  end
endmodule
```

Listing 1: Kod modułu bin2bcd w Verilogu.

```
module seg7(
  input wire clk_25mhz,
  input wire [15:0] bcd,

  output reg [3:0] an_led,
  output reg [6:0] seg_led
);

  function [6:0] num_to_seg (
    input [3:0] num
  );
    case (num)
      4'h0: num_to_seg = 7'b1000000;
      4'h1: num_to_seg = 7'b1111001;
      4'h2: num_to_seg = 7'b0100100;
      4'h3: num_to_seg = 7'b0110000;
      4'h4: num_to_seg = 7'b0011001;
      4'h5: num_to_seg = 7'b0010010;
      4'h6: num_to_seg = 7'b0000010;
      4'h7: num_to_seg = 7'b1111000;
      4'h8: num_to_seg = 7'b0000000;
      4'h9: num_to_seg = 7'b0010000;
      4'ha: num_to_seg = 7'b0001000;
      4'hb: num_to_seg = 7'b0000011;
      4'hc: num_to_seg = 7'b1000110;
      4'hd: num_to_seg = 7'b0100001;
      4'he: num_to_seg = 7'b0000110;
      4'hf: num_to_seg = 7'b0001110;
    endcase
  endfunction

  reg [16:0] counter;
  wire [1:0] selector;

  // Switch each segment at frequency 25Mhz / 2^15 ~ 763Hz.
```

```
  assign selector = counter[16:15];

  always @ (posedge clk_25mhz) begin
    counter <= counter + 1;

    case (selector)
      2'b00: begin
        an_led <= 4'b0111;
        seg_led <= num_to_seg(bcd[15:12]);
      end
      2'b01: begin
        an_led <= 4'b1011;
        seg_led <= num_to_seg(bcd[11:8]);
      end
      2'b10: begin
        an_led <= 4'b1101;
        seg_led <= num_to_seg(bcd[7:4]);
      end
      2'b11: begin
        seg_led <= num_to_seg(bcd[3:0]);
        an_led <= 4'b1110;
      end
    endcase
  end
endmodule
```

Listing 2: Kod modułu seg7 w Verilogu.

```
module main (
  input wire CLK_25MHZ,
  input wire [7:0] SW,
  input wire [1:0] BTN,

  output wire [7:0] LED,
  output wire [3:0] AN_LED,
  output wire [6:0] SEG_LED
);

  reg [11:0] bin = 0;
  wire [15:0] bcd;

  assign LED = SW;

  bin2bcd bin2bcd(
    .bin(bin),
    .bcd(bcd)
  );

  seg7 seg7(
    .clk_25mhz(CLK_25MHZ),
    .an_led(AN_LED),
    .seg_led(SEG_LED),
    .bcd(bcd)
  );

  always @ (posedge CLK_25MHZ) begin
    if (BTN[0]) begin
      bin <= SW[7:4] + SW[3:0];
    end else if (BTN[1]) begin
      bin <= SW[7:4] * SW[3:0];
    end else begin
      bin <= SW[7:4] * 100 + SW[3:0];
```

```
    end
  end
endmodule
```

Listing 3: Kod głównego modułu w Verilogu.

```
// Basys 2 pins configuration.

NET "CLK_25MHZ" LOC = "B8";

// NET "BTN<3>" LOC = "A7";
// NET "BTN<2>" LOC = "M4";
NET "BTN<1>" LOC = "C11";
NET "BTN<0>" LOC = "G12";

NET "SW<7>" LOC = "N3";
NET "SW<6>" LOC = "E2";
NET "SW<5>" LOC = "F3";
NET "SW<4>" LOC = "G3";
NET "SW<3>" LOC = "B4";
NET "SW<2>" LOC = "K3";
NET "SW<1>" LOC = "L3";
NET "SW<0>" LOC = "P11";

NET "LED<7>" LOC = "G1" ;
NET "LED<6>" LOC = "P4" ;
NET "LED<5>" LOC = "N4" ;
NET "LED<4>" LOC = "N5" ;
NET "LED<3>" LOC = "P6" ;
NET "LED<2>" LOC = "P7" ;
NET "LED<1>" LOC = "M11";
NET "LED<0>" LOC = "M5" ;

NET "SEG_LED<0>" LOC = "L14";
NET "SEG_LED<1>" LOC = "H12";
NET "SEG_LED<2>" LOC = "N14";
NET "SEG_LED<3>" LOC = "N11";
NET "SEG_LED<4>" LOC = "P12";
NET "SEG_LED<5>" LOC = "L13";
NET "SEG_LED<6>" LOC = "M12";

NET "AN_LED<3>" LOC = "K14";
NET "AN_LED<2>" LOC = "M13";
NET "AN_LED<1>" LOC = "J12";
NET "AN_LED<0>" LOC = "F12";
```

Listing 4: Konfiguracja pinów płytki Basys 2.