VHDL seria druga

Adrian Jałoszewski

1 Blok entity

```
entity dynamiczny is
    Port (
        CLK : in STD_LOGIC;
        DIS : out STD_LOGIC_VECTOR(6 downto 0);
        AN : out STD_LOGIC_VECTOR(7 downto 0)
    );
end dynamiczny;
```

2 Funkcje pomocnicze

Funkcja przyjmująca liczbę jako argument i zwracająca zakodowaną cyfrę z wyświetlacza.

```
function selectNumberToDisplay ( x : integer range 0 to 9)
    return STD_LOGIC_VECTOR is
    variable temp : std_logic_vector(7 downto 0);
begin
    case x is
        when 0 => temp:= not x"3f";
        when 1 => temp:= not x"06";
        when 2 => temp:= not x"5b";
        when 3 => temp:= not x"4f";
        when 4 => temp:= not x"66";
        when 5 => temp:= not x"6d";
        when 6 => temp:= not x"7d";
        when 7 => temp:= not x"07";
        when 8 => temp:= not x"7f";
        when 9 => temp:= not x"6f";
        when others => temp:= not x"3f";
    end case;
    return temp(6 downto 0);
end selectNumberToDisplay;
```

Funkcja przyjmująca jako argument numer wyświetlacze i zwracająca wartość zapalającą go:

```
function selectDisplay(x : integer range 1 to 8)
    return std_logic_vector is
begin
    case x is
        when 1 => return not x"01";
        when 2 => return not x"02";
        when 3 => return not x"04";
        when 4 => return not x"08";
        when 5 => return not x"10";
        when 6 => return not x"20";
        when 7 => return not x"40";
        when 8 => return not x"80";
        when others => return not x"00";
    end case;
end selectDisplay;
```

3 Proces odpowiadający za wyświetlanie ośmiu cyfr na wyświetlaczu

```
process(CLK)
    variable CNT : integer :=0;
    variable oneDisplayTime : integer := 250000;
    variable selected_display : integer := 1;
begin
    if rising_edge(CLK) then
        if CNT < oneDisplayTime then
            CNT:=CNT+1;
        else
            selected_display := selected_display + 1;
            if (selected_display > 8) then
                selected_display := 1;
            end if;
            CNT:=0;
        end if;
        DIS <= selectNumberToDisplay(selected_display);
        AN <= selectDisplay(selected_display);
    end if;
end process;
    \section{title}
```

4 Wyświetlanie liczb podanych na przełącznikach na poszczególnych wyświetlaczach

```
process(CLK)
    variable CNT : integer :=0;
    variable oneDisplayTime : integer := 500000;
    variable selected_display : integer := 1;
    variable first : integer;
    variable second : integer;
    variable third : integer;
    variable fourth : integer;
begin
    if rising_edge(CLK) then
        first := to_integer(unsigned(sw(3 downto 0)));
        second := to_integer(unsigned(sw(7 downto 4)));
        third := to_integer(unsigned(sw(11 downto 8)));
        fourth := to_integer(unsigned(sw(15 downto 12)));

        if CNT < oneDisplayTime then
            CNT:=CNT+1;
        else
            selected_display := selected_display + 1;
            if (selected_display > 4) then
                selected_display := 1;
            end if;
            CNT:=0;
        end if;
        case selected_display is
            when 1 =>
                DIS <= selectNumberToDisplay(first);
                AN <= selectDisplay(selected_display);
            when 2 =>
                DIS <= selectNumberToDisplay(second);
                AN <= selectDisplay(selected_display);
            when 3 =>
                DIS <= selectNumberToDisplay(third);
                AN <= selectDisplay(selected_display);
            when 4 =>
                DIS <= selectNumberToDisplay(fourth);
                AN <= selectDisplay(selected_display);
            when others =>
                DIS <= selectNumberToDisplay(0);
                AN <= selectDisplay(selected_display);
        end case;
    end if;
end process;
```

5 Licznik wyświetlający na czterech wyświetlaczach, inkrementujący co sekundę

Dodatkowe sygnały dodane na potrzeby kodu:

```
    signal one_second_counter : integer := 0;
    type INT_ARRAY is array (integer range <>) of integer;
    signal display_values : INT_ARRAY(4 downto 1);
```

Proces inkrementujący licznik wewnątrz co sekundę:

```
every_second:process(CLK)
    variable counter : integer := 0;
    constant one_second : integer := 100_000_000;
begin
    if rising_edge(CLK) then
        counter := counter + 1;
        if counter >= one_second then
            counter := 0;
            one_second_counter <= one_second_counter + 1;
        end if;
    end if;
end process;
```

Proces wyświetlający poszczególne cyfry:

```
display:process(CLK)
    variable CNT : integer :=0;
    constant oneDisplayTime : integer := 500000;
    variable selected_display : integer := 1;
begin
    if rising_edge(CLK) then
        if CNT < oneDisplayTime then
            CNT:=CNT+1;
        else
            selected_display := selected_display + 1;
            if (selected_display > 4) then
                selected_display := 1;
            end if;
            CNT:=0;
        end if;
    DIS <= decode_digit_to_ssd(display_values(selected_display));
        AN <=  select_display(selected_display);
    end if;
end process;
```

Przypisywanie wartości do wyświetlania odbywa się w części asynchronicznej programu:

```
display_values(1) <= one_second_counter mod 10;
display_values(2) <= one_second_counter / 10 mod 10;
display_values(3) <= one_second_counter / 100 mod 10;
display_values(4) <= one_second_counter / 1000 mod 10;
```

6 Zegarek ze wskaźnikiem minut i sekund

Aby z powyższego kodu otrzymać czasomierz należy zmienić wartości przypisywanie wartości do wyświetlania na:

```
display_values(1) <= one_second_counter mod 10;
display_values(2) <= one_second_counter / 10 mod 6;
display_values(3) <= one_second_counter / 60 mod 10;
display_values(4) <= one_second_counter / 600 mod 6;
```