Stoper z możliwością zapamiętania 5 wyników. Funkcje Startu i Stopu realizowane jednym przyciskiem, natomiast funkcja zapamiętania wyniku drugim.

Układ za pomocą zewnętrznego zegara wytwarza swój własny zegarek 24-godzinny (z dokładnością do sekundy). Dzięki niemu odmierza czas, który może być zatrzymywany i zapisywany w pamięci mieszczącej 5 rekordów.

Układ ten podzielony jest na kilka modułów:

Zegarek – pod warunkiem, że jego praca jest zezwolona (sygnał *licz* = '1'), w takt podanego na wejściu (zbocza narastającego) zegara *zeg* (1 Hz) zlicza on i zwraca czas wyrażony 3 liczbami: godzin *zeg_g*, minut *zeg_m*, sekund *zeg_s*. Czas przedstawiony jest w formacie 24-godzinnym. Sekundę po godzinie 23:59:59 zegarek „zawraca" do 00:00:00.

```
23  -- Zegarek ------------------------------------------------------------------------
24  procedure zegarek
25     (signal zeg   : in std_logic; -- zegar zewnętrzny 1 Hz
26      signal licz  : in std_logic; -- wejście zezwalające na pracę zegarka
27      signal zeg_g : inout std_logic_vector (4 downto 0); -- liczba godzin
28      signal zeg_m : inout std_logic_vector (5 downto 0); -- liczba minut
29      signal zeg_s : inout std_logic_vector (5 downto 0)) is -- liczba sekund
30  begin
31     if (licz = '1') then
32         if (zeg'event and zeg = '1') then
33           if (zeg_s < 59) then
34              zeg_s <= zeg_s + 1;
35           else
36              zeg_s <= "000000";
37              if (zeg_m < 59) then
38                 zeg_m <= zeg_m + 1;
39              else
40                 zeg_m <= "000000";
41                 if (zeg_g < 23) then
42                    zeg_g <= zeg_g + 1;
43                 else
44                    zeg_g <= "00000";
45                 end if;
46              end if;
47           end if;
48        end if;
49     end if;
50  end zegarek;
51  ------------------------------------------------------------------------------------
```

Pamięć – jest to szczególny przypadek rozbudowanego rejestru, w którym zapamiętywane jest 5 rekordów, każdy reprezentowany 3 wektorami (dla rekordu pierwszego: *pam1_h*, *pam1_m*, *pam1_s*) – zawierających odpowiednio godzinę, minutę i sekundę. Sygnały *in_h*, *in_m*, *in_s* (odpowiednio godzina, minuta, sekunda) są wpisywane do rekordu pierwszego, a zapisane dotychczas są przesuwane do następnych rekordów (rekord ostatni – piąty – nie mieści się w pamięci i jest tracony).

```
53  -- Pamięć ------------------------------------------------------------------------
54  procedure pamiec
55     (signal in_h : inout std_logic_vector (4 downto 0); -- zapisywana godzina
56      signal in_m : inout std_logic_vector (5 downto 0); -- zapisywana minuta
57      signal in_s : inout std_logic_vector (5 downto 0); -- zapisywana sekunda
58      -- zapamiętane stany stopera:
59      signal pam1_h, pam2_h, pam3_h, pam4_h, pam5_h : inout std_logic_vector (4 downto 0);
60      signal pam1_m, pam2_m, pam3_m, pam4_m, pam5_m : inout std_logic_vector (5 downto 0);
61      signal pam1_s, pam2_s, pam3_s, pam4_s, pam5_s : inout std_logic_vector (5 downto 0)) is
62  begin
63     -- przesunięcie 4 najnowszych rekordów i "usunięcie" najstarszego
64     pam5_h <= pam4_h; pam5_m <= pam4_m; pam5_s <= pam4_s;
65     pam4_h <= pam3_h; pam4_m <= pam3_m; pam4_s <= pam3_s;
66     pam3_h <= pam2_h; pam3_m <= pam2_m; pam3_s <= pam2_s;
67     pam2_h <= pam1_h; pam2_m <= pam1_m; pam2_s <= pam1_s;
68     -- wpisanie nowego rekordu
69     pam1_h <= in_h; pam1_m <= in_m; pam1_s <= in_s;
70  end pamiec;
71  ---------------------------------------------------------------------------------------
```

Sterowanie stoperem – odbywa się ono w 3 procesach:

I proces (start/stop odliczania) - wyzwalany przyciskiem zapisu *buton_start*. Ponieważ czas naciśnięcia przycisku trwa skończoną ilość czasu, układ reaguje nie na stan wysoki, tylko zbocze narastające. Wciśnięcie przycisku start/stop sprawia, że sygnał *counting* (zezwalający na odmierzanie czasu) zmienia stan na przeciwny – następuje start lub stop. W przypadku, gdy stoper startuje, jest automatycznie zerowany (nie można wznowić odmierzania czasu, można jedynie zacząć je od nowa).

II proces (praca stopera) – wyzwalany zegarem zewnętrznym *clk* lub zmianą zezwolenia na pracę *counting*. Odmierzanie czasu odbywa się z użyciem procedury *zegarek* (opisanej wyżej).

III proces (zapis rekordu) – wyzwalany przyciskiem zapisu *buton_save*. W przypadku naciśnięcia, wywoływana jest procedura *pamięć* (opisana wyżej).

```
75     -- reakcja układu na przycisk start/stop
76     process (button_start)
77     begin
78        if (button_start'event and button_start = '1') then
79           counting <= not counting;
80           if (counting = '0') then -- jeśli stoper startuje, to trzeba go wyzerować
81              time_h <= "00000";
82              time_m <= "000000";
83              time_s <= "000000";
84           end if;
85        end if;
86     end process;
87
88     -- praca stopera
89     process (Clk, counting)
90     begin
91        zegarek(Clk, counting, time_h, time_m, time_s);
92     end process;
93
94     -- reakcja układu na przycisk zapisu
95     process (button_save)
96     begin
97        if (button_save'event and button_save = '1') then
98           pamiec(time_h, time_m, time_s, mem1_h, mem2_h, mem3_h, mem4_h, mem5_h, mem1_m,
99                  mem2_m, mem3_m, mem4_m, mem5_m, mem1_s, mem2_s, mem3_s, mem4_s, mem5_s);
100       end if;
101    end process;
```

Opis całego układu w języku VHDL:

Układ ma wejścia 1-bitowe: *Clk* (zegarowe), *button_start* (przycisk start/stop), *button_save* (przycisku zapisu). Aktualny czas pokazywany przez stoper reprezentowany jest wektorami *time_h*, *time_m*, *time_s* (odpowiednio godzina, minuta, sekunda). Zezwolenie na pracę zegarka reprezentowane jest 1-bitowym *counting*. 5 rekordów zapisanych w pamięci stopera reprezentowanych jest w sumie 15 wektorami (5- i 6-bitowymi, w zależności od zakresu przechowywanej wartości – sekundy i minuty od 0 do 59, godziny od 0 do 23), których wartości są domyślnie wyzerowane.

Układ działa sekwencyjnie, jego działanie jest opisane procesem wyzwalanym zegarem zewnętrznym *clk*, naciśnięciem przycisku startu/stopu *button_start* lub przycisku zapisu *buton_save*. Działanie poszczególnych części kodu wyjaśniono wyżej.

```
1  library IEEE;
2  use IEEE.STD_LOGIC_1164.ALL;
3  use IEEE.STD_LOGIC_UNSIGNED.ALL;
4  USE IEEE.NUMERIC_STD.ALL;
5
6  entity Lab4 is
7     port(Clk          : in std_logic; -- zegar zewnętrzny 1 Hz
8          button_start : in std_logic; -- przycisk start/stop
9          button_save  : in std_logic; -- przycisk zapisywania
10         time_h       : inout std_logic_vector (4 downto 0) := "00000"; -- aktualna godzina
11         time_m       : inout std_logic_vector (5 downto 0) := "000000"; -- aktualna minuta
12         time_s       : inout std_logic_vector (5 downto 0) := "000000"; -- aktualna sekunda
13         counting     : inout std_logic := '0'; -- czy zegarek pracuje? (zezwolenia na pracę zegarka)
14         -- zapamiętane stany stopera:
15         meml_h, mem2_h, mem3_h, mem4_h, mem5_h : inout std_logic_vector (4 downto 0) := "00000";
16         meml_m, mem2_m, mem3_m, mem4_m, mem5_m : inout std_logic_vector (5 downto 0) := "000000";
17         meml_s, mem2_s, mem3_s, mem4_s, mem5_s : inout std_logic_vector (5 downto 0) := "000000");
18
19 end Lab4;
20
21 architecture Behavioral of Lab4 is
```

```
22
23 -- Zegarek ------------------------------------------------------------------------------
24  procedure zegarek
25     (signal zeg   : in std_logic; -- zegar zewnętrzny 1 Hz
26      signal licz  : in std_logic; -- wejście zezwalające na pracę zegarka
27      signal zeg_g : inout std_logic_vector (4 downto 0); -- liczba godzin
28      signal zeg_m : inout std_logic_vector (5 downto 0); -- liczba minut
29      signal zeg_s : inout std_logic_vector (5 downto 0)) is -- liczba sekund
30  begin
31     if (licz = '1') then
32         if (zeg'event and zeg = '1') then
33           if (zeg_s < 59) then
34              zeg_s <= zeg_s + 1;
35           else
36              zeg_s <= "000000";
37              if (zeg_m < 59) then
38                 zeg_m <= zeg_m + 1;
39              else
40                 zeg_m <= "000000";
41                 if (zeg_g < 23) then
42                    zeg_g <= zeg_g + 1;
43                 else
44                    zeg_g <= "00000";
45                 end if;
46              end if;
47           end if;
48        end if;
49     end if;
50 end zegarek;
51 ------------------------------------------------------------------------------------------
```

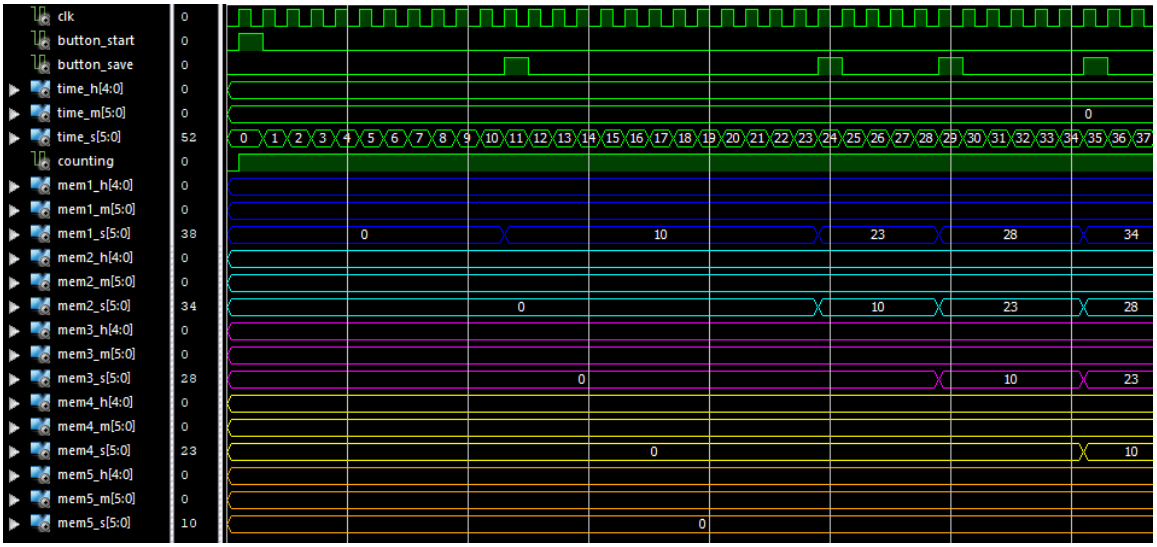
```
52
53  -- Pamięć ---------------------------------------------------------------------------
54  procedure pamiec
55     (signal in_h : inout std_logic_vector (4 downto 0); -- zapisywana godzina
56      signal in_m : inout std_logic_vector (5 downto 0); -- zapisywana minuta
57      signal in_s : inout std_logic_vector (5 downto 0); -- zapisywana sekunda
58      -- zapamiętane stany stopera:
59      signal pam1_h, pam2_h, pam3_h, pam4_h, pam5_h : inout std_logic_vector (4 downto 0);
60      signal pam1_m, pam2_m, pam3_m, pam4_m, pam5_m : inout std_logic_vector (5 downto 0);
61      signal pam1_s, pam2_s, pam3_s, pam4_s, pam5_s : inout std_logic_vector (5 downto 0)) is
62  begin
63     -- przesunięcie 4 najnowszych rekordów i "usunięcie" najstarszego
64     pam5_h <= pam4_h; pam5_m <= pam4_m; pam5_s <= pam4_s;
65     pam4_h <= pam3_h; pam4_m <= pam3_m; pam4_s <= pam3_s;
66     pam3_h <= pam2_h; pam3_m <= pam2_m; pam3_s <= pam2_s;
67     pam2_h <= pam1_h; pam2_m <= pam1_m; pam2_s <= pam1_s;
68     -- wpisanie nowego rekordu
69     pam1_h <= in_h; pam1_m <= in_m; pam1_s <= in_s;
70  end pamiec;
71  ------------------------------------------------------------------------------------
```

```
72
73  begin
74
75     -- reakcja układu na przycisk start/stop
76     process (button_start)
77     begin
78        if (button_start'event and button_start = '1') then
79           counting <= not counting;
80           if (counting = '0') then -- jeśli stoper startuje, to trzeba go wyzerować
81              time_h <= "00000";
82              time_m <= "000000";
83              time_s <= "000000";
84           end if;
85        end if;
86     end process;
87
88     -- praca stopera
89     process (Clk, counting)
90     begin
91        zegarek(Clk, counting, time_h, time_m, time_s);
92     end process;
93
94     -- reakcja układu na przycisk zapisu
95     process (button_save)
96     begin
97        if (button_save'event and button_save = '1') then
98           pamiec(time_h, time_m, time_s, mem1_h, mem2_h, mem3_h, mem4_h, mem5_h, mem1_m,
99                  mem2_m, mem3_m, mem4_m, mem5_m, mem1_s, mem2_s, mem3_s, mem4_s, mem5_s);
100       end if;
101    end process;
102
103 end Behavioral;
```

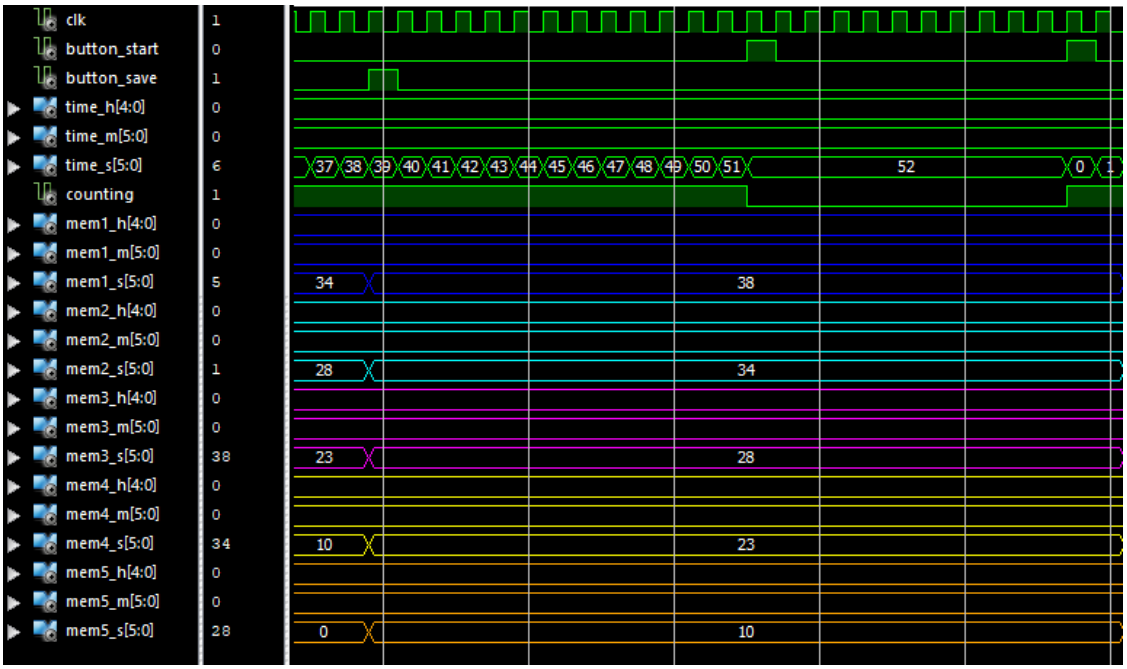
Przebiegi symulacyjne:

Dla poprawy czytelności przebiegów, wartości reprezentowane wektorami przedstawiono w postaci liczb dziesiętnych. Użyto kolorów do odróżnienia rekordów pamięci.

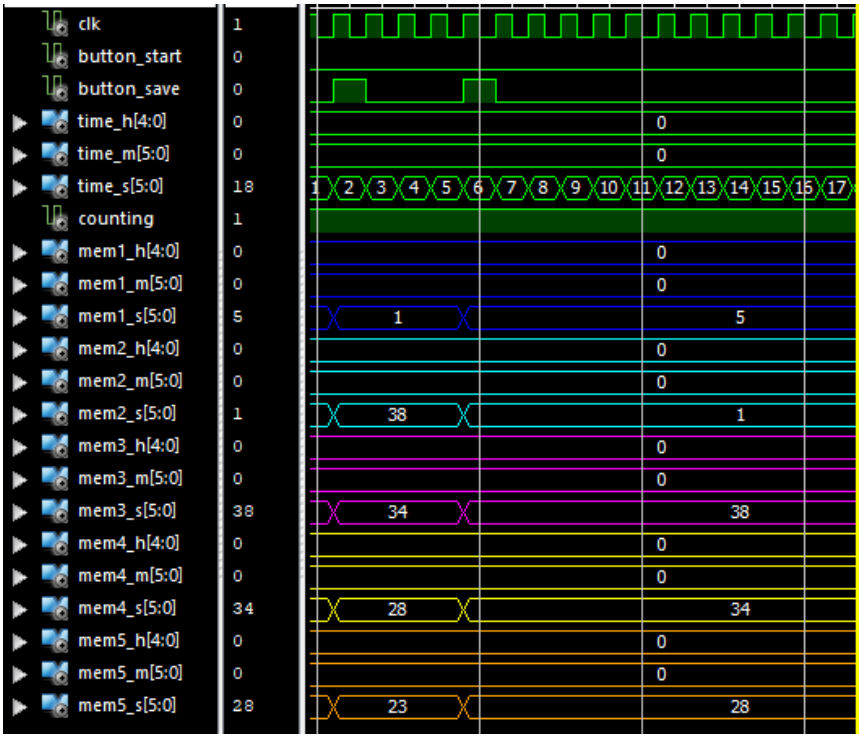
Pierwotnie ustalony czas to 00:00:00. Gdy nadchodzi pierwsze zbocze narastające *buton_start*, sygnał *counting* zmienia stan na wysoki – dozwolone jest odmierzanie czasu. W takt (zbocza narastającego) zegara wejściowego *clk* inkrementuje się *time_s*. Gdy pojawia się zbocze narastające *buton_save* (zostaje wciśnięty przycisk zapisu), w wektorach *mem1_* zostaje zapisany aktualny czas. W przypadku kolejnych naciśnięć przycisku, rekord *mem1_* zostaje przesunięty do rekordu *mem2_*, a do rekordu *mem1_* wpisany zostaje aktualny. Za każdym kolejnym razem, zapisane rekordy „przesuwają się" w pamięci.



Gdy występuje zbocze narastające *button_start* (zostaje naciśnięty przycisk start/stop), sygnał *counting* zmienia stan na przeciwny – niski. Stoper przestaje zliczać czas. Przy wystąpieniu kolejnego zbocza narastającego, stoper zostaje wyzerowany i zaczyna mierzyć czas od początku.



Do pamięci wpisywane są kolejne rekordy.



Zapisanie godziny 23:59:55 w pamięci również przebiega poprawnie. Gdy stoper przekracza swój zakres, a więc sekundę po 23:59:59, jego stan „zawraca” do 00:00:00. Następuje mierzenie czasu od zera.