Sprawozdanie z laboratorium

Przedmiot	Modelowanie i Analiza Systemów
Temat laboratorium	Testbench układu decymacyjnego
Numer laboratorium	2
Imię i nazwisko	Maciej Stanek
Numer indeksu	122352
Data wykonania	16 marca 2018
Data sprawozdania	30 marca 2018

Zadanie 1: *Opis koncepcji układu decymacyjnego i wyjaśnienie jego działania.*

Układ decymacyjny to urządzenie zliczające ilość zer i jedynek w kolejnych segmentach ciągu bitowego o stałej szerokości OSR. Licznik rozpoczyna zliczanie na wartości OSR i inkrementuje bądź dekrementuje jego wartość po otrzymaniu wartości odpowiednio 1 lub 0. Po analizie OSR bitów, układ wystawia na wyjściu wyznaczoną wartość końcową (w przedziale od 0 do 2OSR), resetuje się do wartości OSR i rozpoczyna kolejny cykl.

Zadanie 2: *Opis koncepcji działania całego układu testbenchu — zależności czasowe taktowania.*

Testbench instancjonuje decymator, po czym ładuje z zewnętrznego pliku segmenty ciągu bitowego. Dla każdego takie segmentu wprowadza go on synchronicznie na wejście decymatora i wypisuje do pliku wyjściowego wartość wynikową.

Zadanie 3: *Wyjaśnienie znaczenia/funkcji wszystkich portów (we/wy) oraz parametrów (generic). W kodzie należy nadać wartości domyślne (default) wszystkim parametrom.*

- Port `clk` — pozwala na synchroniczne wprowadzanie ciągu bitowego.
- Port `data_in` — wejście synchroniczne ciągu bitowego.
- Port `data_out` — port wyjściowy decymatora.
- Parametr `osr` — ilość bitów w jednym segmencie.

Zadanie 4: *Kody źródłowe VHDL — testbenchu.*

Listing 1. Testbench układu decymacyjnego.

```
library ieee;
use ieee.std_logic_1164.all;
library std;
use std.textio.all;

entity decim_tb is
  generic(
    din: string := "0.in.txt";
    dout: string := "0.out.txt");
end entity decim_tb;

architecture default of decim_tb is
  file infile: text open read_mode is din;
  file outfile: text open write_mode is dout;
  constant clk_period: time := 20 ns;
  signal clk_i, data_in_i: std_logic;
  signal data_out_i: integer;
begin
  uut: entity work.decim(default)
    generic map(osr => 32)
    port map(
      clk => clk_i,
```

```
      data_in => data_in_i,
      data_out => data_out_i);

  process
    variable text_in, text_out: line;
    variable value_out: integer := 0;
  begin
    clk_i <= '0';
    wait for clk_period;
    while not endfile(infile) loop
      readline(infile, text_in);
      if text_in(1) = '#' then
        -- Do not analyze comments, just pass them through.
        writeline(outfile, text_in);
      else
        for i in text_in'range loop
          if text_in(i) = '1' then
            data_in_i <= '1';
          elsif text_in(i) = '0' then
            data_in_i <= '0';
          end if;
          clk_i <= '1';
          wait for clk_period;
          clk_i <= '0';
          wait for clk_period;
        end loop;
        write(text_out, data_out_i);
        writeline(outfile, text_out);
      end if;
    end loop;
    assert false severity error;
  end process;
end architecture;
```

Listing 2. Skrypt uruchamiający testbench.

```
set din "0.in.txt"
set dout "0.out.txt"
set d decim
vcom ${d}.vhd
vcom ${d}_tb.vhd

vsim work.${d}_tb -Gdin=$din -Gdout=$dout
foreach s {clk data_in data_out} {
    add wave -position insertpoint sim:/${d}_tb/${s}_i
}
run -all
```

Zadanie 5: *Kod źródłowy układu decymacyjnego.*

Listing 3. Układ decymacyjny.

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.std_logic_unsigned.all;
use ieee.std_logic_arith.all;

entity decim is
  generic(osr: integer := 32);
  port(clk: in std_logic;
    data_in: in std_logic;
    data_out: out integer);
end entity;

architecture default of decim is
begin
  process(clk)
    variable progress: integer := 0;
    variable result: integer := osr;
  begin
    if rising_edge(clk) then
      if data_in = '1' then
        result := result + 1;
      end if;
```

```
      if data_in = '0' then
        result := result - 1;
      end if;

      progress := progress + 1;
      if progress = osr then
        progress := 0;
        data_out <= result;
        result := osr;
      end if;
    end if;
  end process;
end architecture;
```

Zadanie 6: *Plik wejściowy (z opisem sposobu jego generacji i wyjaśnienie co jest w nim zakodowane i z jakimi parametrami OSR).*

Składnia pliku wejściowego wymaga podania ciągu wejściowego w postaci słów złożonych z OCR zer i jedynek, zakończonych znakiem nowej linii. Testbench dopuszcza komentarze w pliku wejściowym (zgodnie z zaleceniami dra Śniatały). Komentarzem jest każda linijka rozpoczynająca się od symbolu kratki.

Listing 4. Plik wejściowy z komentarzami.

```
# Description: Example input
# Creation time: 2018-03-16 18:00:00
# OCR: 32
00000000000000000000000000000000
01010101010101010101010101010101
# THIS IS A COMMENT
11111111111111111111111111111111
00000000000000001111111111111111
00000000000000000000000011111111
00000000000000000000000000001111
00000000000000000000000000000011
00000000000000000000000000000001
00000000000000000000000000000000
```

Zadanie 7: *Plik wyjściowy z wyjaśnieniem zawartości.*

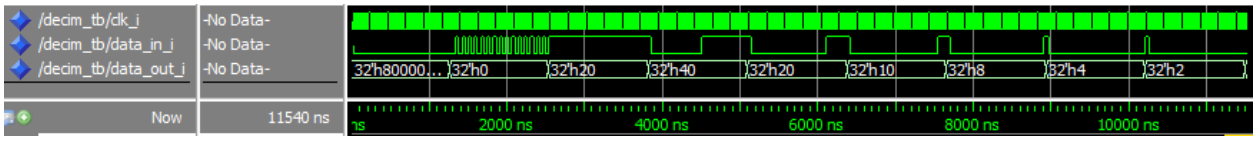
Plik wyjściowy zawiera liczby, będące wynikiem działania decymatora dla poszczególnych segmentów z pliku wejściowego. Testbench przepisuje komentarze z pliku wejściowego do pliku wyjściowego.

Listing 5. Plik wyjściowy z przepisanymi komentarzami.

```
# Description: Example input
# Creation time: 2018-03-16 18:00:00
# OCR: 32
0
32
# THIS IS A COMMENT
64
32
16
8
4
2
0
```

Zadanie 8: *Zrzut ekranu przedstawiający działania układu, który zawiera co najmniej następujące przebiegi (sygnały):*

- *Zegar bazowy (CLK),*
- *Bitstream wejściowy (Data_In) — przed decymacją,*
- *Ciąg wyjściowy liczb całkowitych reprezentujący sygnał wejściowy (Data_Out).*



Rysunek 1. Przebiegi sygnałów wejściowych i wyjściowych.