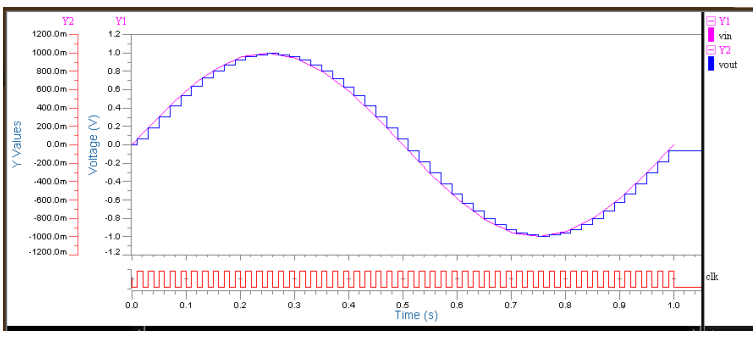
Sprawozdanie z laboratorium

Przedmiot	Modelowanie i Analiza Systemów
Temat laboratorium	Dyskretna transformata Z w VHDL AMS
Numer laboratorium	3
Imię i nazwisko	Maciej Stanek
Numer indeksu	122352
Data wykonania	10 maja 2018
Data sprawozdania	17 maja 2018

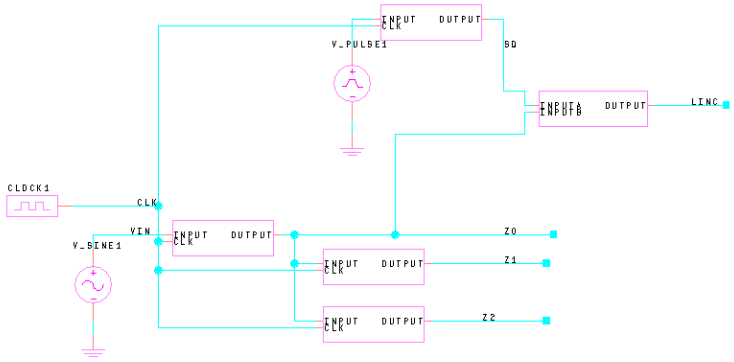
Zadanie 1: *Układ sample&hold ma próbkować napięciowy sygnał wejściowy podawany na terminal input. Próbkowanie ma następować przy narastającym zboczu zegara* `clk`.

Listing 1. Sample&hold.

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.electrical_systems.all;
use ieee.math_real.all;

entity sample_hold is
  generic(
    initial: real := 0.0;
    clk_edge: std_logic := '1');
  port(
    terminal input: electrical;
    clk: in std_logic;
    output: out real := initial);
end entity;

architecture default of sample_hold is
  quantity v_sampled across input;
begin
  sampling: process(clk)
  begin
    if clk'event and clk = clk_edge then
      output <= v_sampled;
    end if;
  end process;
end architecture;
```



Rysunek 1. Działanie układu sample&hold.

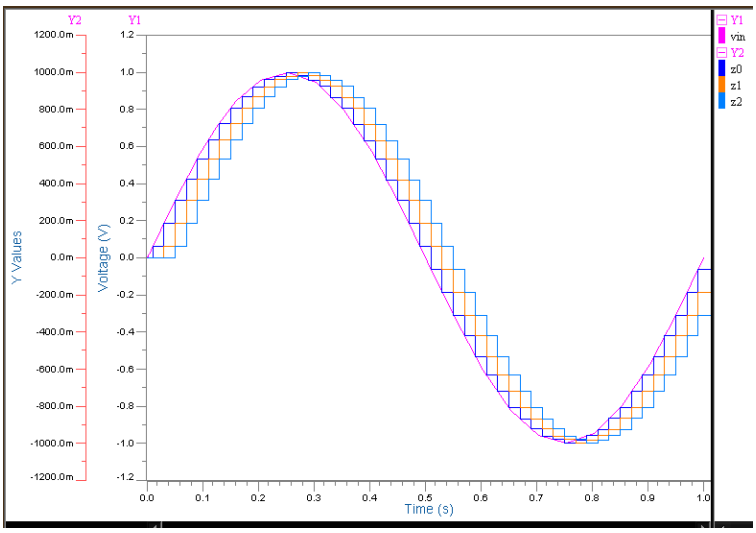


Rysunek 2. Układ testujący blok kombinacji liniowej, układ sample&hold oraz układy opóźniające.

Zadanie 2: *Układ opóźniający ma dwa porty wejściowe: sygnał zegarowy (std_logic) oraz sygnał wejściowy (real). Na wyjściu sygnał ma pojawić się z n-krotnym opóźnieniem. Opóźnienie zdefiniowane jest w formie parametru generic.*

Listing 2. Układ opóźniający Z^{-n}.

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.electrical_systems.all;
use ieee.math_real.all;

entity zdelay is
  generic(
    count : integer := 1;
    initial : real := 0.0;
    clk_edge : std_logic := '1');
  port(
    input: in real;
    clk: in std_logic;
    output: out real := initial);
end entity;

architecture default of zdelay is
  type holds_array is array (1 to count) of real;
  signal holds: holds_array := (others => initial);
begin
  process(clk)
  begin
    if clk'event and clk = clk_edge then
      if count > 1 then
        holds(2 to count) <= holds(1 to count-1);
      end if;
      holds(1) <= input;
    end if;
  end process;

  output <= holds(count);
end architecture;
```

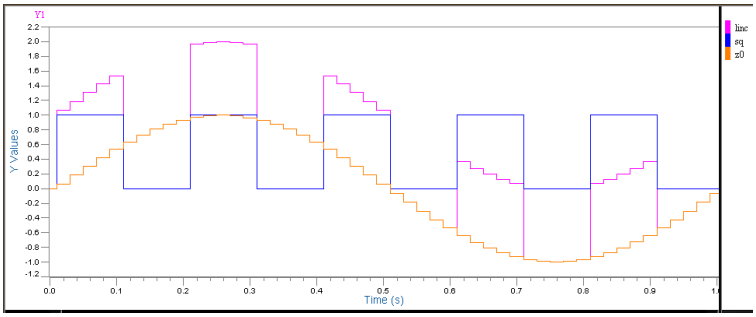


Rysunek 3. Działanie układu opóźniającego.

Zadanie 3: *Układ kombinacji liniowej na wejściu ma dwa sygnały typu real i sumuje je z odpowiednimi wagami (kombinacja liniowa). Współczynniki określone w postaci parametrów generic. Układ nie wymaga sygnału synchronizującego zegara. Zaprojektuj architekturę układu w sposób asynchroniczny.*

Listing 3. Układ kombinacji liniowej.

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.electrical_systems.all;
use ieee.math_real.all;

entity linearcombination is
  generic(
    initial : real := 0.0;
    coeffa : real := 1.0;
    coeffb : real := 1.0);
  port(
    inputa: in real;
    inputb: in real;
    output: out real := initial);
end entity;

architecture default of linearcombination is
begin
  output <= coeffa*inputa + coeffb*inputb;
end architecture;
```



Rysunek 4. Działanie układu kombinacji liniowej.

Zadanie 4: *Zaimplementuj układy integratorów według wyznaczonych na laboratorium schematów blokowych. Do ich implementacji należy wykorzystać wyłącznie układy kombinacji liniowej i opóźniające. Porty wejściowe integratora to sygnał typu real oraz sygnał zegarowy typu std_logic.*

Implementacji integratorów dokonano behawioralnie, bez wykorzystania gotowych bloków opóźnień.

Listing 4. Backward Euler Integrator.

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.electrical_systems.all;
use ieee.math_real.all;

entity backint is
  generic(
    initial : real := 0.0;
    clk_edge : std_logic := '1');
  port(
    input: in real;
    clk: in std_logic;
    output: out real := initial);
end entity;

architecture default of backint is
begin
  process(clk)
    variable integral : real := initial;
  begin
    if clk'event and clk = clk_edge then
      integral := integral + input;
      output <= integral;
    end if;
  end process;
end architecture;
```

Listing 5. Forward Euler Integrator.

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.electrical_systems.all;
use ieee.math_real.all;

entity forwardint is
  generic(
    initial : real := 0.0;
    clk_edge : std_logic := '1');
  port(
    input: in real;
    clk: in std_logic;
    output: out real := initial);
end entity;

architecture default of forwardint is
begin
  process(clk)
    variable integral : real := initial;
    variable last_input : real := initial;
  begin
    if clk'event and clk = clk_edge then
      integral := integral + last_input;
      last_input := input;
      output <= integral;
    end if;
  end process;
end architecture;
```

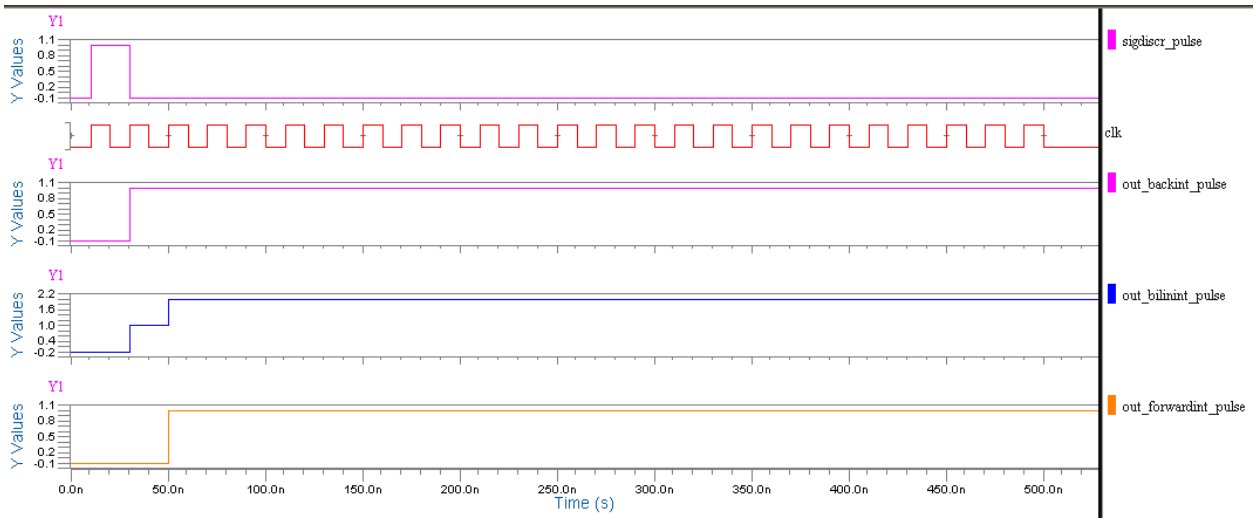
Listing 6. Bilinear Integrator.

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.electrical_systems.all;
use ieee.math_real.all;

entity bilinint is
  generic(
```

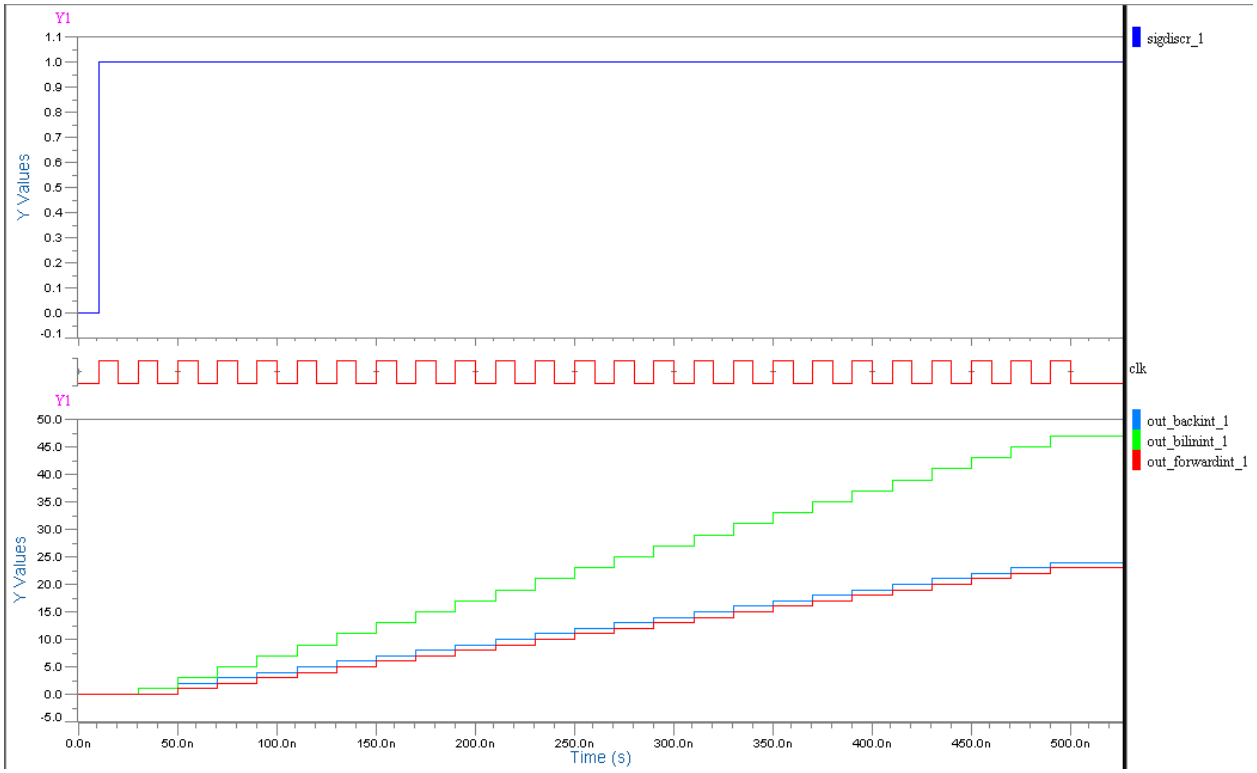
```
    initial : real := 0.0;
    clk_edge : std_logic := '1');
  port(
    input: in real;
    clk: in std_logic;
    output: out real := initial);
end entity;

architecture default of bilinint is
begin
  process(clk)
    variable integral : real := initial;
    variable last_input : real := initial;
  begin
    if clk'event and clk = clk_edge then
      integral := integral + input + last_input;
      last_input := input;
      output <= integral;
    end if;
  end process;
end architecture;
```

Zadanie 5: *Korzystając z układów integratorów z poprzedniego zadania przetestuj i porównaj działanie każdego z układów całkujących. Porównaj odpowiedzi układów na wymuszenia impulsem Kroneckera i skokiem jednostkowym z obliczeniami z laboratorium. Wykorzystaj dany układ test–bench do przetestowania układu.*

Zaprojektowane układy zachowują się tak samo, jak układy wzorcowe zaprezentowane na przykładowych wykresach odpowiedzi skokowych i impulsowych.



Rysunek 5. Odpowiedź trzech zaprojektowanych układów całkujących na impuls jednostkowy.



Rysunek 6. Odpowiedź trzech zaprojektowanych układów całkujących na skok jednostkowy.