„Układy reprogramowalne i SoC”
„Testbenches. Symulacja sterowana zdarzeniami.”

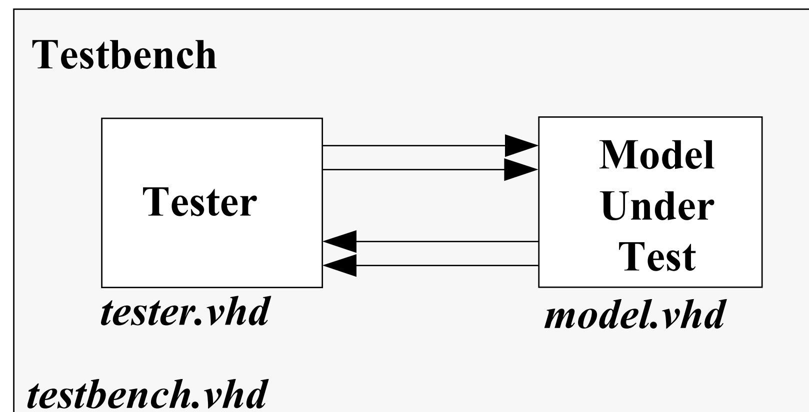
Prezentacja jest współfinansowana przez
Unię Europejską w ramach
Europejskiego Funduszu Społecznego w projekcie pt.

„Innowacyjna dydaktyka bez ograniczeń - zintegrowany rozwój Politechniki Łódzkiej - zarządzanie Uczelnią, nowoczesna oferta edukacyjna i wzmacniania zdolności do zatrudniania osób niepełnosprawnych”

Prezentacja dystrybuowana jest bezpłatnie

Testbenches

- W celu przetestowania zaprojektowanego kodu odpowiednią sekwencję testową wejść można wygenerować posługując się symulatorem
- Inną możliwością jest napisanie odpowiedniego kodu testującego w języku VHDL
  - Można w nim nie tylko wymuszać sygnały na wejściach testowanego układu, ale również odczytywać wyjścia i reagować na ich zmiany



KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI
Projekt współfinansowany przez Unię Europejską
w ramach Europejskiego Funduszu Społecznego
UNIA EUROPEJSKA
EUROPEJSKI FUNDUSZ SPOŁECZNY

Przykład: sumator

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

entity full_adder is
port (
       a, b, cin: in std_logic;
       s, cout: out std_logic
     );
end full_adder;
--------------------------------------
architecture dataflow of full_adder is
begin
   s <= a xor b xor cin;
   cout <= (a and b) or (a and cin) or
           (b and cin);
end dataflow;
```

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;
entity full_adder_tb is
end entity;
architecture a of full_adder_tb is
   component full_adder is
      port (
         a, b, cin: in std_logic;
         s, cout: out std_logic
      );
   end component;
signal a: unsigned(2 downto 0);
signal o: unsigned(1 downto 0);
begin
uut: full_adder port map (
        a => a(0),
        b => a(1),
        cin => a(2),
        s => o(0),
        cout => o(1)
     );

   test: process
   begin
      for i in 0 to 7 loop
         a <= to_unsigned(i,3);
         wait for 20 ns;
      end loop;
      wait;
   end process;
end architecture a;
```

- Testbench zazwyczaj nie ma żadnych wejść ani wyjść
- Mogą wystąpić problemy z oglądaniem wyjść, jeżeli nie są do niczego wykorzystywane
  - Optymalizacja może je usunąć
- Rozwiązanie: użyć opcji +acc programu vopt

```
vlib work
vcom adder.vhd
vcom adder_tb.vhd
vopt +acc+full_adder_tb full_adder_tb -o opt
vsim opt
do wave.do
run 200 ns
```

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Projekt współfinansowany przez Unię Europejską
w ramach Europejskiego Funduszu Społecznego

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Pliki

- Czasami zamiast oglądać przebiegi na ekranie, wolimy je wpisać do pliku
- Przy definicji typu plikowego podajemy, co plik przechowuje

```
type text is file of string;
type IntegerFileType is file of integer;
```

- Operacje na plikach:
  - file_open(), file_close(), endfile(), read(), write()

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI
Projekt współfinansowany przez Unię Europejską
w ramach Europejskiego Funduszu Społecznego
UNIA EUROPEJSKA
EUROPEJSKI FUNDUSZ SPOŁECZNY

Przykład: zapis do pliku

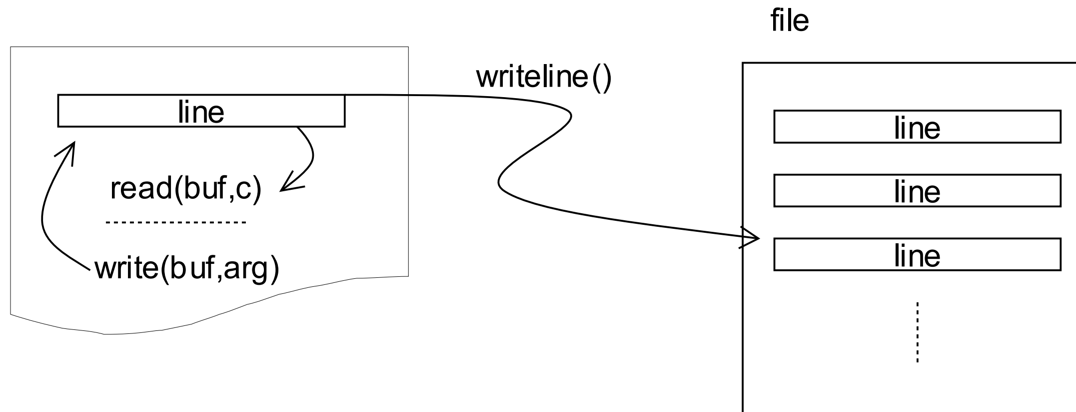
```
--
-- test of binary file I/O
--
entity io_write_test is
end io_write_test;

architecture behavioral of io_write_test is
begin
   process
      type IntegerFileType is file of integer;
      file dataout :IntegerFileType;
      variable check :integer :=0;
      variable fstatus : file_open_status;
   begin
      file_open(fstatus,dataout,"data.out",write_mode);
      for count in 1 to 10 loop
         check := check + 1;
         write(dataout, check);
      end loop;
      file_close(dataout);
      wait;
   end process;
end behavioral;
```

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Projekt współfinansowany przez Unię Europejską
w ramach Europejskiego Funduszu Społecznego

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Pliki tekstowe - pakiety TEXTIO i STD_LOGIC_TEXTIO



- Plik tekstowy jest zorganizowany jako sekwencja linii
- Procedury read() oraz write() operują na liniach
  - hread()/hwrite() do operacji w zapisie szesnastkowym
  - oread()/owrite() do operacji w zapisie ósemkowym
- Procedury readline() i writeline() przesyłają dane między liniami i plikami
- Procedury zawarte w pakiecie TEXTIO biblioteki STD oraz pakiecie STD_LOGIC_TEXTIO biblioteki IEEE
  - Umożliwiają zapis i odczyt standardowych typów do/z linii

KAPITAŁ LUDZKI NARODOWA STRATEGIA SPÓJNOŚCI
Projekt współfinansowany przez Unię Europejską w ramach Europejskiego Funduszu Społecznego
UNIA EUROPEJSKA EUROPEJSKI FUNDUSZ SPOŁECZNY

Przykład - test sumatora z zapisem do pliku

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

entity full_adder_tb is
end entity;

architecture a of full_adder_tb is
   component full_adder is
      port (
         a, b, cin: in std_logic;
         s, cout: out std_logic
      );
   end component;
signal a: unsigned(2 downto 0);
signal o: unsigned(1 downto 0);
begin

uut: full_adder port map (
        a => a(0),
        b => a(1),
        cin => a(2),
        s => o(0),
        cout => o(1)
     );
```

```
   test: process
      file outfile: text;
      variable fstatus: file_open_status;
      variable buf: line;
   begin
      file_open(fstatus,outfile,"myfile.txt",write_mode);
      write(buf, string'("Simulation of full adder"));
      writeline(outfile,buf);
      for i in 0 to 7 loop
         a <= to_unsigned(i,3);
         wait for 20 ns;
         write(buf, std_logic_vector(a));
         write(buf, string'(" --> "));
         write(buf, to_integer(o));
         writeline(outfile,buf);
      end loop;
      file_close(outfile);
      wait;
   end process;

end architecture a;
```

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Projekt współfinansowany przez Unię Europejską
w ramach Europejskiego Funduszu Społecznego

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Przykład - weryfikacja sumatora na podstawie pliku

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

entity full_adder_tb is
end entity;

architecture a of full_adder_tb is
   component full_adder is
      port (
         a, b, cin: in std_logic;
         s, cout: out std_logic
      );
   end component;
signal a: std_logic_vector(2 downto 0);
signal o: std_logic_vector(1 downto 0);
begin

uut: full_adder port map (
        a => a(0),
        b => a(1),
        cin => a(2),
        s => o(0),
        cout => o(1)
     );
```

```
   test: process
      file infile: text;
      variable fstatus: file_open_status;
      variable buf: line;
      variable stimulus: std_logic_vector(2 downto 0);
      variable expected_o: integer;
      variable vector_valid: boolean;
   begin
      file_open(fstatus,infile,"infile.txt",read_mode);
      while not endfile(infile) loop
         readline(infile, buf);
         read(buf, stimulus, good => vector_valid);
         next when not vector_valid;
         read(buf, expected_o);
         a <= stimulus;
         wait for 20 ns;
         assert to_integer(unsigned(o)) = expected_o
           report "Simulation mismatch!";
      end loop;
      file_close(infile);
      assert false
         report "Simulation OK!";
      wait;
   end process;

end architecture a;
```

```
# test vectors
000  0
001  1
010  1
011  2
100  1
101  2
110  2
111  3
```

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI
Projekt współfinansowany przez Unię Europejską
w ramach Europejskiego Funduszu Społecznego
UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Przykład: test nadajnika RS232

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

use work.rs_test_interface.all;

entity rs_tb is
end entity rs_tb;

architecture a of rs_tb is

component TxUnit is
  port (
     Clk          : in  Std_Logic;  -- Clock signal
     Reset_n      : in  Std_Logic;  -- Reset input, active low
     Load         : in  Std_Logic;  -- Load transmit data
     TxD          : out Std_Logic;  -- RS-232 data output
     TRegE_n      : out Std_Logic;  -- Tx register empty, active low
     TBufE_n      : out Std_Logic;  -- Tx buffer empty, active low
     DataO        : in  Std_Logic_Vector(7 downto 0));
end component;

signal clk: std_logic := '0';
signal reset_n: std_logic;
signal load: std_logic;
signal txd: std_logic;
signal buf_empty_n: std_logic;
signal data: std_logic_vector (7 downto 0);
signal treg_empty_n: std_logic;
begin
```

```
tx: TxUnit port map (
       Clk => clk,
       Reset_n => reset_n,
       Load => load,
       TxD => txd,
       TRegE_n => treg_empty_n,
       TBufE_n => buf_empty_n,
       DataO => data
    );

clkgen: process
   begin
      wait for 1000 ms / 9600 / 2;
      clk <= not clk;
   end process;

test: process
   begin
      reset_n <= '0';
      load <= '0';
      wait until clk'event and clk='1';
      reset_n <= '1';
      PROC_RS_WRITE_BYTE (x"AA", buf_empty_n, clk, load, data);
      PROC_RS_WRITE_BYTE (x"66", buf_empty_n, clk, load, data);
      wait until treg_empty_n = '0' and buf_empty_n = '0';
      assert false
         report "Simulation finished"
         severity failure;
end process;

end architecture a;
```

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Projekt współfinansowany przez Unię Europejską
w ramach Europejskiego Funduszu Społecznego

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Przykład: test nadajnika RS232 (c.d.)

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

package rs_test_interface is
procedure PROC_RS_WRITE_BYTE (
  val          : in std_logic_vector (7 downto 0);
  signal buf_empty_n : in std_logic;
  signal clk   : in std_logic;
  signal load : out std_logic;
  signal data : out std_logic_vector (7 downto 0)
);
end package rs_test_interface;

-- Package Body
package body rs_test_interface is

procedure PROC_RS_WRITE_BYTE (
  val          : in std_logic_vector (7 downto 0);
  signal buf_empty_n : in std_logic;
  signal clk   : in std_logic;
  signal load : out std_logic;
  signal data : out std_logic_vector (7 downto 0)
) is
begin
   while true loop
      wait until clk'event and clk = '1';
      if buf_empty_n = '0' then
         load <= '1';
         data <= val;
         wait until clk'event and clk = '1';
         load <= '0';
         exit;
      end if;
   end loop;
end PROC_RS_WRITE_BYTE;

end package body rs_test_interface;
```

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Projekt współfinansowany przez Unię Europejską
w ramach Europejskiego Funduszu Społecznego

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

Opóźnienia w języku VHDL

- W języku VHDL występują dwa rodzaje opóźnień
  - Inercyjne
    - Nie uwzględnia impulsów krótszych niż podane opóźnienie

```
C <= A and B after 5ns;
```

```
C <= inertial A and B after 5ns;
```

  - Transportowe
    - Uwzględnia wszystkie impulsy

```
C <= transport A and B after 5ns;
```

Opóźnienia w języku VHDL

- Kolejna konstrukcja umożliwia wyeliminowanie krótkich impulsów

```
Z3 <= reject 2 ns inertial X after 5 ns;
```

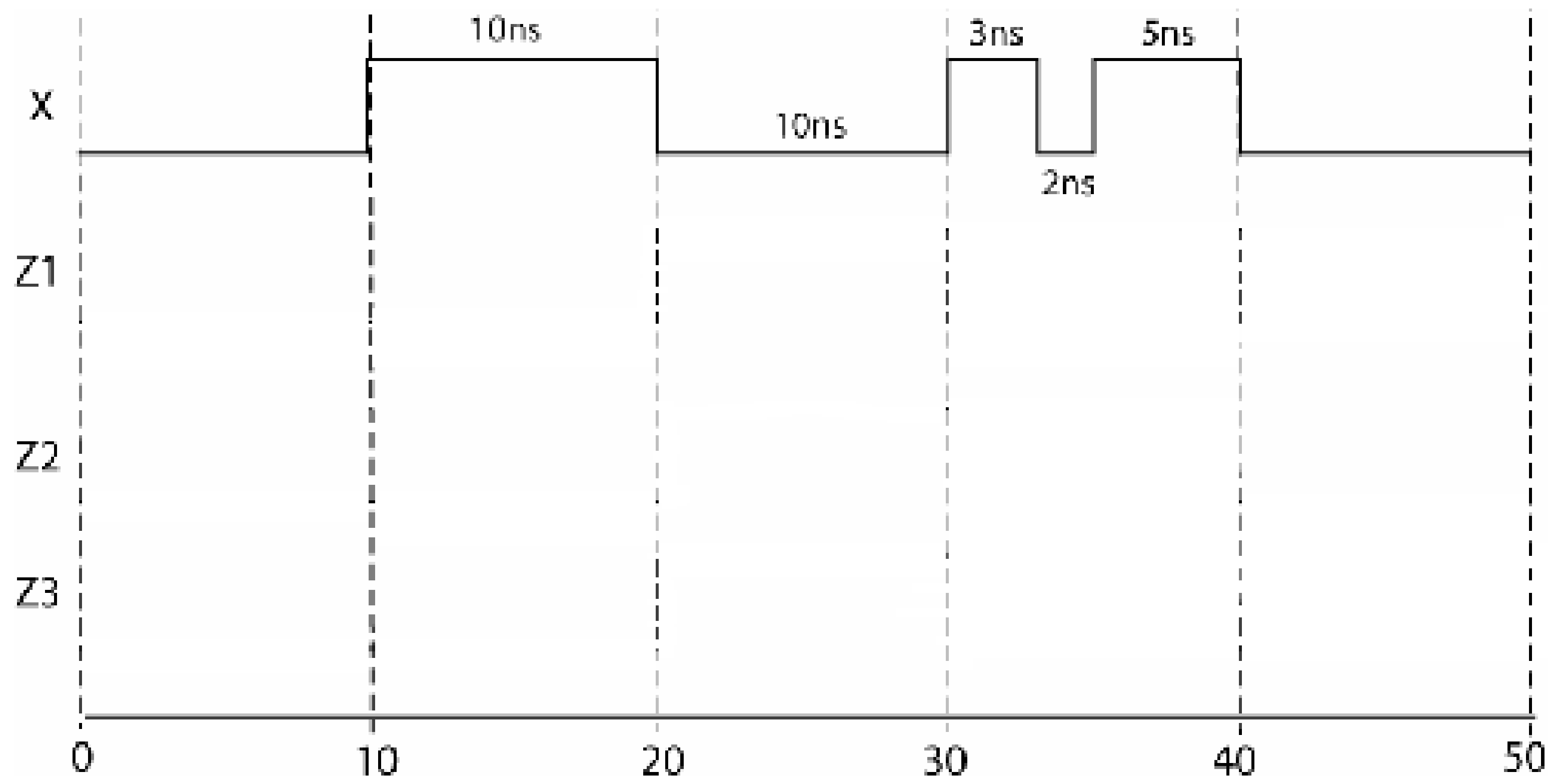
  – równoważne

```
Zm <= X after 2 ns;
Z3 <= transport Zm after 3 ns;
```


Projekt współfinansowany przez Unię Europejską
w ramach Europejskiego Funduszu Społecznego

Opóźnienia w języku VHDL

```
Z1 <= transport X after 10 ns;
Z2 <= X after 10 ns;
Z3 <= reject 4 ns inertial X after 10 ns;
```



Symulacja

- Proces symulacji składa się z dwóch faz
  - Faza inicjalizacji - przypisanie wartości początkowych
  - Faza symulacji właściwej
- VHDL używa modelu symulacji sterowanego zdarzeniami
  - Czas jest podzielony na dyskretne kroki
  - Wykonanie instrukcji powoduje, że w pewnym czasie w przyszłości szeregowana jest transakcja
    - Transakcja niekoniecznie zmienia wartość sygnału
    - Jeżeli transakcja zmienia wartość sygnału, zwana jest zdarzeniem

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI
Projekt współfinansowany przez Unię Europejską
w ramach Europejskiego Funduszu Społecznego
UNIA EUROPEJSKA
EUROPEJSKI FUNDUSZ SPOŁECZNY

Symulacja

- Inicjalizacja
  - Wartości początkowe podawane są w kodzie VHDL
    - Jeżeli ich nie podano, symulator przyjmie wartości domyślne, zależnie od typu
    - Może to spowodować niezgodność z wynikami syntezy
  - Czas symulacji ustawiony na zero
  - Aktywowane wszystkie procesy

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI
Projekt współfinansowany przez Unię Europejską
w ramach Europejskiego Funduszu Społecznego
UNIA EUROPEJSKA
EUROPEJSKI FUNDUSZ SPOŁECZNY

Przykład

```
A <= B or C after 2 ns;
D <= A after 3 ns;
process (A)
begin
    E <= A or B after 1 ns;
end process;
process
begin
    C <= not C after 3 ns;
    wait on D;
    B <= E after 3 ns;
end process;
```

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI
Projekt współfinansowany przez Unię Europejską
w ramach Europejskiego Funduszu Społecznego
UNIA EUROPEJSKA
EUROPEJSKI FUNDUSZ SPOŁECZNY

Delta delay

- Jeżeli instrukcja ma opóźnienie zerowe
  - Symulator używa nieskończenie małego opóźnienia, Δ (delta)
  - Czas symulacji nie ulega zmianie
    - Używane tylko do umieszczania transakcji w kolejce
    - Licznik opóźnień delta kasowany, gdy nastąpi zmiana czasu o skończoną wartość

Przykład delta delay

```
A <= B or C;
D <= A;
process (A)
begin
    E <= A or B;
end process;
process
begin
    C <= not C after 3 ns;
    wait on C;
    B <= E;
end process;
```

Symulacja wielu procesów

- Jeżeli model zawiera wiele procesów, wszystkie są wykonywane współbieżnie
- Instrukcje współbieżne poza procesami są również wykonywane współbieżnie
- Instrukcje wewnątrz procesu wykonywane sekwencyjnie
- Proces wykonuje się w zerowym czasie
  - chyba że zawiera instrukcję wait
    - wait for 10 ns; wait on E;
- Jeżeli nie podano opóźnienia, wartości sygnałów zostaną uaktualnione po czasie delta

KAPITAŁ LUDZKI NARODOWA STRATEGIA SPÓJNOŚCI
Projekt współfinansowany przez Unię Europejską w ramach Europejskiego Funduszu Społecznego
UNIA EUROPEJSKA EUROPEJSKI FUNDUSZ SPOŁECZNY

Symulacja sterowana zdarzeniami

- Zdarzenie - zmiana wartości sygnału
- Po każdym zdarzeniu
  - Wszystkie procesy czekające na zdarzenie wykonują się natychmiast, w zerowym czasie
  - Zmiany sygnałów z tego wynikające zostaną umieszczone w kolejce zdarzeń w jakiejś przyszłej chwili czasowej
- Po zakończeniu wykonywania wszystkich aktywnych procesów
  - Czas symulacji zmienia się na czas najbliższego zdarzenia w kolejce
  - Symulator przetwarza to zdarzenie
- Proces kontynuowany dopóki
  - Są zdarzenia w kolejce
  - Nie przekroczono zadanego czasu symulacji

KAPITAŁ LUDZKI NARODOWA STRATEGIA SPÓJNOŚCI
Projekt współfinansowany przez Unię Europejską w ramach Europejskiego Funduszu Społecznego
UNIA EUROPEJSKA EUROPEJSKI FUNDUSZ SPOŁECZNY

Perfidny błąd

Nie wykonywać przypisań sygnałów zegarowych!

```
library ieee;
use ieee.std_logic_1164.all;
use ieee.numeric_std.all;

entity ff_tb is
end entity;

architecture a of ff_tb is
   component ff is
   port (
       d, clk: in std_logic;
       q: out std_logic
    );
    end component;
signal a,b,c,d: std_logic := '0';
signal clk,clk2: std_logic := '0';
begin

uut1: ff port map (
        d => a,
        q => b,
        clk => clk
     );
```

```
uut2: ff port map (
        d => b,
        q => c,
        clk => clk
     );

uut3: ff port map (
        d => b,
        q => d,
        clk => clk2
     );

clk2 <= clk;

   gena: process
   begin
         a <= not a;
         wait for 7 ns;
   end process;

   genclk: process
   begin
         clk <= not clk;
         wait for 10 ns;
   end process;

end architecture a;
```

KAPITAŁ LUDZKI
NARODOWA STRATEGIA SPÓJNOŚCI

Projekt współfinansowany przez Unię Europejską
w ramach Europejskiego Funduszu Społecznego

UNIA EUROPEJSKA
EUROPEJSKI
FUNDUSZ SPOŁECZNY

„Układy reprogramowalne i SoC"
„Testbenches. Symulacja sterowana zdarzeniami."

Prezentacja jest współfinansowana przez
Unię Europejską w ramach
Europejskiego Funduszu Społecznego w projekcie pt.

„Innowacyjna dydaktyka bez ograniczeń - zintegrowany rozwój Politechniki Łódzkiej - zarządzanie Uczelnią, nowoczesna oferta edukacyjna i wzmacniania zdolności do zatrudniania osób niepełnosprawnych"

Prezentacja dystrybuowana jest bezpłatnie