

Instrukcja realizacji zadań projektowych w środowisku SystemC w trybie zdalnym.

1. Sposób podłączenia do maszyny oraz zasady oceniania zadań realizowanych w formie zdalnej opisano w instrukcjach dostępnych pod poniższymi linkami:
http://www.ue.eti.pg.gda.pl/fpgalab/hdl/remote_lab-instrukcja.pdf
http://www.ue.eti.pg.gda.pl/fpgalab/hdl/remote_lab-zasady-oceniania.pdf
2. SystemC jest dostępny za pomocą środowiska Cygwin
 - a. Skrót do terminala znajduje się na pulpicie;
 - b. Terminal automatycznie startuje w katalogu domowym użytkownika lab1, tj. `C:\Users\lab1\cygwin` (dostępny także w menadżerze plików Windows z zakładki Quick access jako *cygwin*). **UWAGA: katalog domowy jest automatycznie czyszczony przy każdym nowym logowaniu!** Proszę robić kopie zapasowe opracowywanych kodów
 - c. Pliki źródłowe do realizacji zadania można edytować na przykład za pomocą edytora Notepad++ (zainstalowany na zdalnej maszynie);
3. Kompilację opracowanych kodów realizuje się przy użyciu odpowiednio spreparowanego pliku Makefile (przykład na końcu instrukcji) komendą *make*;
4. Skompilowany kod uruchamiamy przez wykonanie w katalogu programu polecenia `./<nazwa_pliku>.exe`
5. W przypadku braku błędów program wygeneruje plik o rozszerzeniu `<nazwa_pliku>.vcd`, który możemy obejrzeć przy użyciu GTKWave poprzez wywołanie komendy `gtkwave <nazwa_pliku>.vcd`
6. Przykładowy projekt w SystemC przystosowany do kompilacji i uruchomienia za pomocą środowiska Cygwin zawarto w załączniku *sysc_example_cygwin.zip*

Skrót do środowiska Cygwin

Skrót do katalogu home dostępnego dla Cygwin

```
testbench.cpp
//
// Testbench for the 4-bit up-counter
// Example from www.asic-world.com
//
#include "systemc.h"
#include "design.h"

int sc_main (int argc, char* argv)
{
    sc_signal<bool> clock;
    sc_signal<bool> reset;
    sc_signal<sc_uint<8> > counter;
    int i = 0;
    // Connect the DUT
    johnson_counter counter("COUNTER",
        counter.clock(clock);
        counter.reset(reset);
        counter.counter_out(counter_out));
}

SC_CNS;

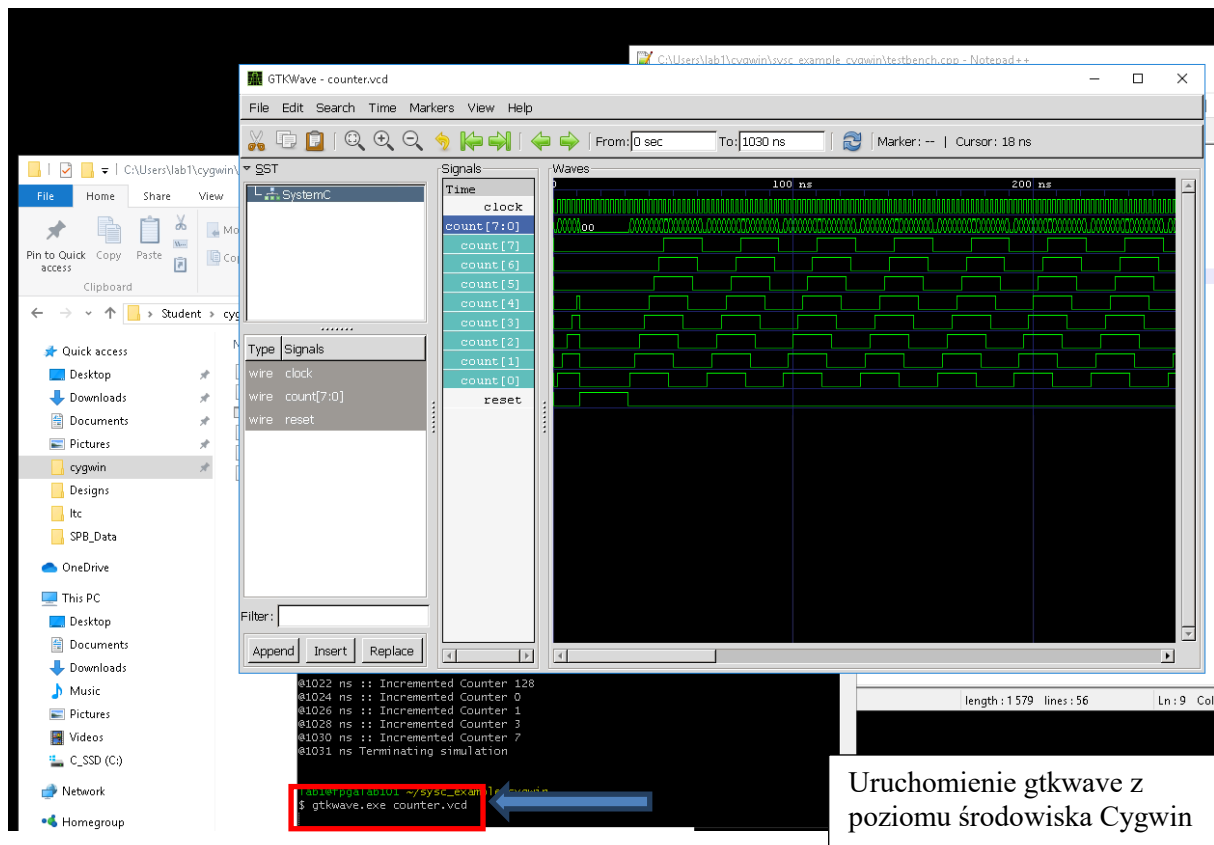
file
*uf = sc_create_v
desired signals
clock, "clock");
reset, "reset");
counter_out, "count

all variables
// initial val
i++) {

SC_NS;

SC_NS;
```

```
lab1@fpgalab101 ~/sysc_example_cygwin
$ cd ..
lab1@fpgalab101 ~
$ ls -al
total 49
drwxr-xr-x+ 1 lab1 None 0 Apr 14 15:04 .
drwxr-xr-x+ 1 SYSTEM SYSTEM 0 Apr 14 12:50 ..
-rw-r-----+ 1 lab1 None 3 Apr 14 12:50 .bash_history
-rwxr-xr-x 1 lab1 None 1494 Apr 14 10:46 .bash_profile
-rwxr-xr-x 1 lab1 None 5645 Apr 14 10:46 .bashrc
-rwxr-xr-x 1 lab1 None 1919 Apr 14 10:46 .inputrc
-rwxr-xr-x 1 lab1 None 1236 Apr 14 10:46 .profile
drwxr-xr-x+ 1 lab1 None 0 Apr 14 15:06 sysc_example_cygwin
lab1@fpgalab101 ~
$
```



Przykładowy Makefile dla programów w SystemC do kompilacji z poziomu Cygwin'a (na czerwono zaznaczono zmiany w stosunku do pliku kompilowanego pod Linux'em):

```
# Simple Makefile
TARGET = johnson
SYSTEMC = /usr/local/systemc-2.2
INCDIR = -I. -I/usr/local/systemc-2.2/include
LIBDIR = -L. -L/usr/local/systemc-2.2/lib-cygwin
LIBS = -lsystemc -lm

CC = g++
CFLAGS = -g -Wno-deprecated -Wall -std=c++98 -O0 -fpermissive

OBJS = testbench.o

EXE = $(TARGET)

all: $(EXE)

$(EXE): $(OBJS)
    $(CC) $(CFLAGS) $(INCDIR) $(LIBDIR) -o $@ $(OBJS) $(LIBS) 2>&1

testbench.o: testbench.cpp design.h
    $(CC) $(CFLAGS) $(INCDIR) -c $<

clean:
    rm -f $(OBJS) $(EXE) *.vcd
```