

# Laboratory to the subject: „Design of ASICs” Bachelor Level

## 1) Introduction.

It is intended to perform three exercises as a part of the laboratory. Completion of the exercises gives as the result design of simple digital IC core which is made using standard cells technique. As the design tool CADENCE RC Compiler and Encounter are used, as the simulator IRUN/Simvision are also used. Standard cells library used during labs is made as the result of one student master thesis. The design of cells were made with UMC 130nm CMOS process.

It is mandatory to complete all 3 exercises in the program lab. At the end of each of exercise student have to fill enclosed table and send it via email to the person conducting laboratory.

## 2) Computer accounts and logging in.

All students attending the labs will obtain account with the login name according to the following format:

**s\_xyyy**

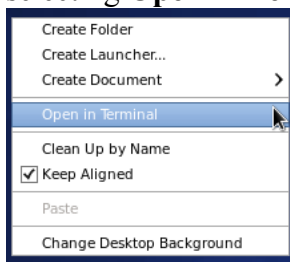
where: **x** – first letter of the name, **yyy** – full surname without national diacritics. Immediately after switching on of the computer in the lab (room EA308) it is necessary to choose LINUX as the starting OS (Win XP is the default one).

**NOTE: login name have to be typed in using only lowercase letters** otherwise system will allow to log in to the computer but rights to the network directories will not be granted properly (they are used for distribution of the CADENCE software as well as technology dependent files and libraries).

After first login it is mandatory to change the password from default to user given one. It can be performed by the command:

**passwd**

typed in LINUX terminal. LINUX terminal can be opened by right clicking in the desktop area and selecting **Open in Terminal** as it is shown at the figure below:



Account system used in the lab is common for all computers. Anyone can log in using the same login name to all machines. User file system is unique and different at all computers. Due to above reasons it is advised to take the same computer place in consecutive labs because there will be current working files on that particular machine. Unfortunately, there is sometimes need to copy files from one machine to another. In such a case one can use Linux scp command. For example if user named **s\_xyyy** wants to copy the directory named **lab1** recursively including all the files it includes from the computer named **fpgalab5** to the current directory in currently logged in machine should issue following command in the terminal:

```
scp -r s_xyyy@fpgalab5:/home2/s_xyyy/lab1 .
```

Please note dot at the end of the comment which mean current directory. Please also note that in the lab EA308 users directory is placed in **/home2** folder.

3) Main information regarding CADENCE software.

The CADENCE software is installed in the following directory:

**/cadence/cadence2012\_2013**. In order to use the software the number of environmental variables has to be set. There is a script prepared to run all necessary settings, please run the command (in the terminal):

```
source /cadence/cadence2012_2013/c64bit.csh
```

Hint: you can use **TAB** key to prompt for possible file and directory names during entering the above command. Next you should create new empty folder for your design and enter it below are some basic commands which can help with this task:

changes directory into user's home directory which in our case is **/home2/s\_xyyy**

```
cd
```

creates new directory named **new\_directory\_name**:

```
mkdir new_directory_name
```

enter into directory

```
cd new_directory_name
```

running a software, in below example running digital simulation console:

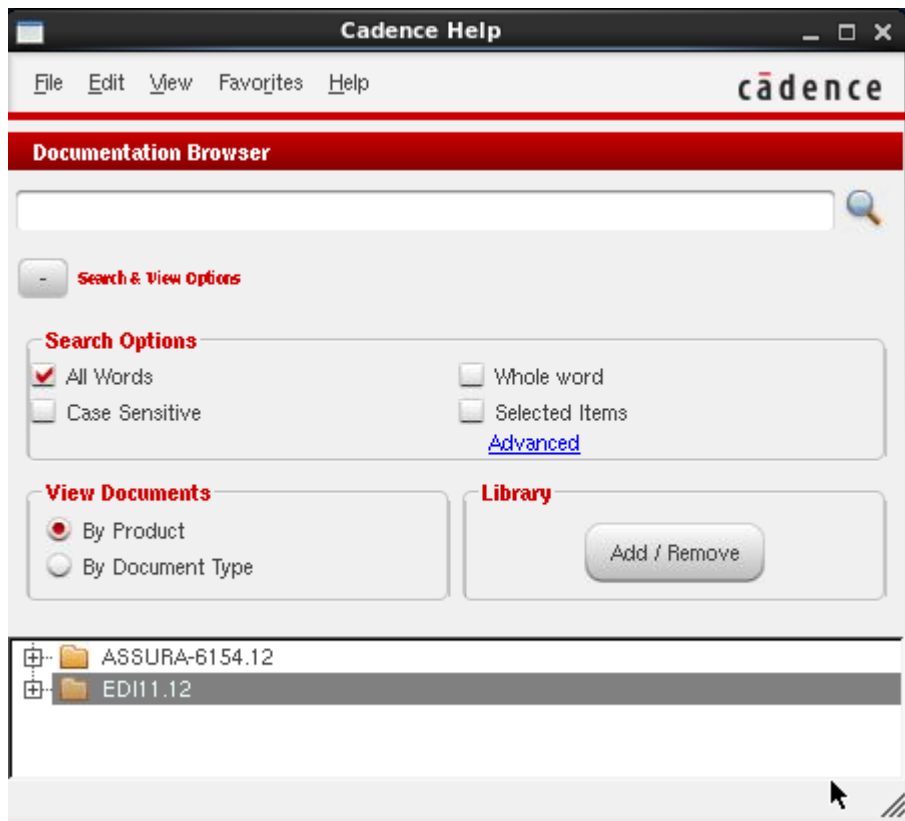
```
nclaunch -new &
```

Character **"&"** after command means that process will be run in the background and terminal will be released for entering another command.

Cadence software suite includes comprehensive support environment which can be started by entering command


```
cdnshelp &
```

After that help window opens as it is shown in the below figure.



There are possible help packets to be selected at the bottom of the **Cadence Help** window. In the above figure it is **ASSURA-6154.12** and **EDI11.12**. To add other necessary packets click button **Add/Remove** in the **Library** subwindow and next press **Add..** and then from the list of the available can be chosen needed one, please observe example explaining help packets location below:

**/cadence/cadence2009\_2010/(name of needed packet)/doc.**

After adding needed help suite there are possible options to sort according to document type or Product type. Alternatively, by clicking  on the task bar needed document can be viewed as PDF instead of hypertext. In case if there are difficulties in opening PDF documents it is necessary to configure Firefox as PDF reader. To do this click menu **Edit/Settings** next tab **Applications**, then click **Add** and next enter **pdf** in the column **Ext** and **firefox** in the column **Application**.

# Lab 1

## Simulation of simple digital block

**Optional but recommended:** There is tutorial included in Cadence Help explaining how to perform simulation. Please run command **cdnshelp** to open help window. Next choose **Incisive\_12.1** and then please click following buttons:

**View Documents - By Document Type**

expand sign "+" in documents given in **12.10**

expand sign "+" in documents entitled **Getting Started**

double click into **Verilog/VHDL Tutorial**

Going back to the lab exercise please prepare VHDL description of the digital circuit which should include following 4 blocks:

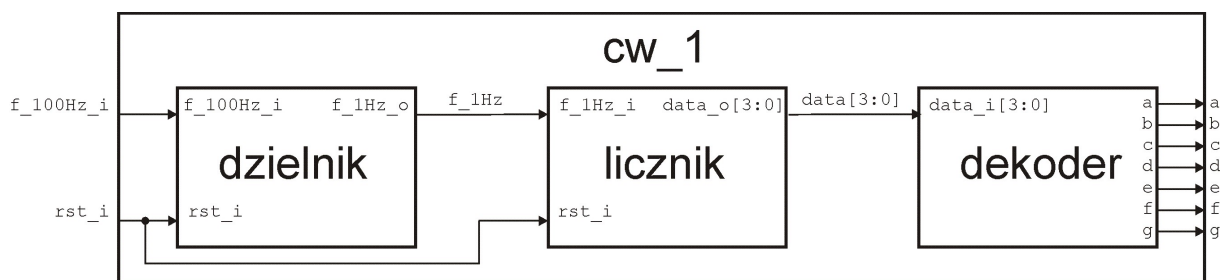
**cw\_1** – main block consisting of 3 remaining blocks,

**dzielnik** – synchronous frequency divider with asynchronous reset, constant divide ratio equal to 100,

**licznik** – synchronous counter with asynchronous reset, count range: 0-9,

**dekoder** – combinatorial block which should include BCD to 7 segment LED decoder for common cathode display.

All the blocks should be connected exactly as it is shown in the below figure. Used signal names should be also as in the figure. Additionally please prepare testbench file with the entity named **cw\_1\_testbench** which should instantiate **cw\_1** and generate input signals. Test signal **reset\_i** should be active until 35 ms, signal **f\_100Hz\_i** should be square wave of frequency equal to 100Hz. Each of blocks (VHDL entity) should be placed in separate VHDL file named as block name with the extension **.vhd1**. For making description you can use any text editor, for example: **gedit**, **vi** or **emacs**.



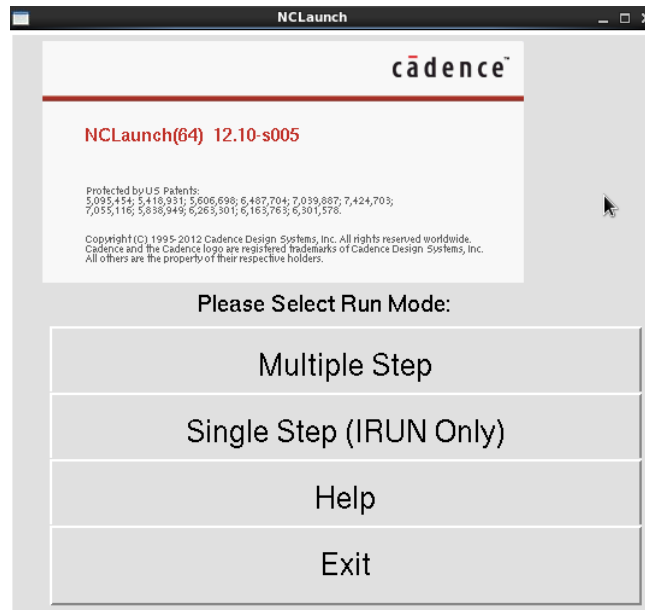
To check VHDL syntax use below command:

**ncvhd1 -messages -V93 name\_of\_the\_file.vhd1**

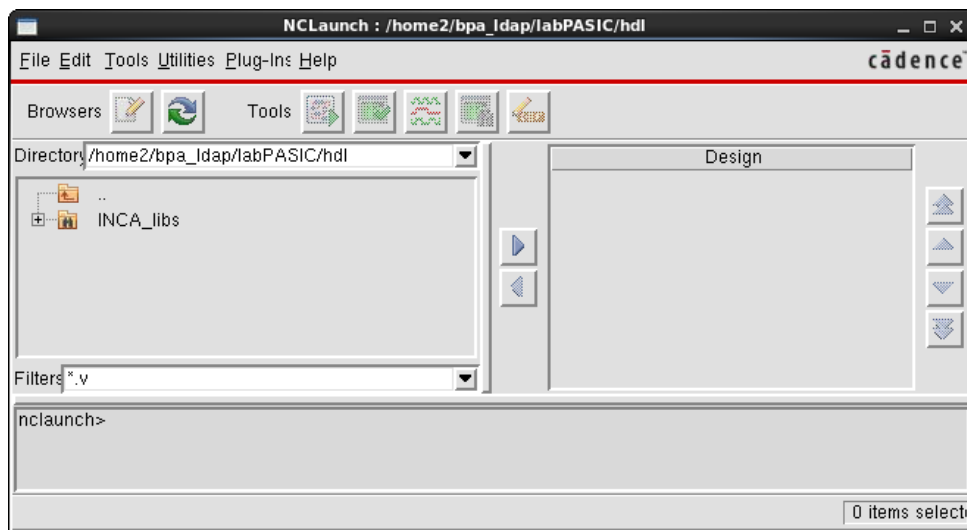
In the case there are errors there will be information given with the line and column in which an error occurred. If there are many errors please **start correction with first error!** After checking all the files please start the simulation with the command:


**nclaunch -new &**

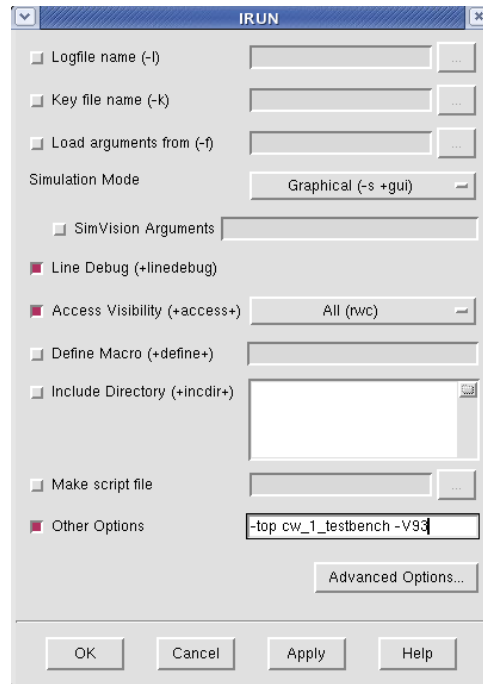
The window should be raised:



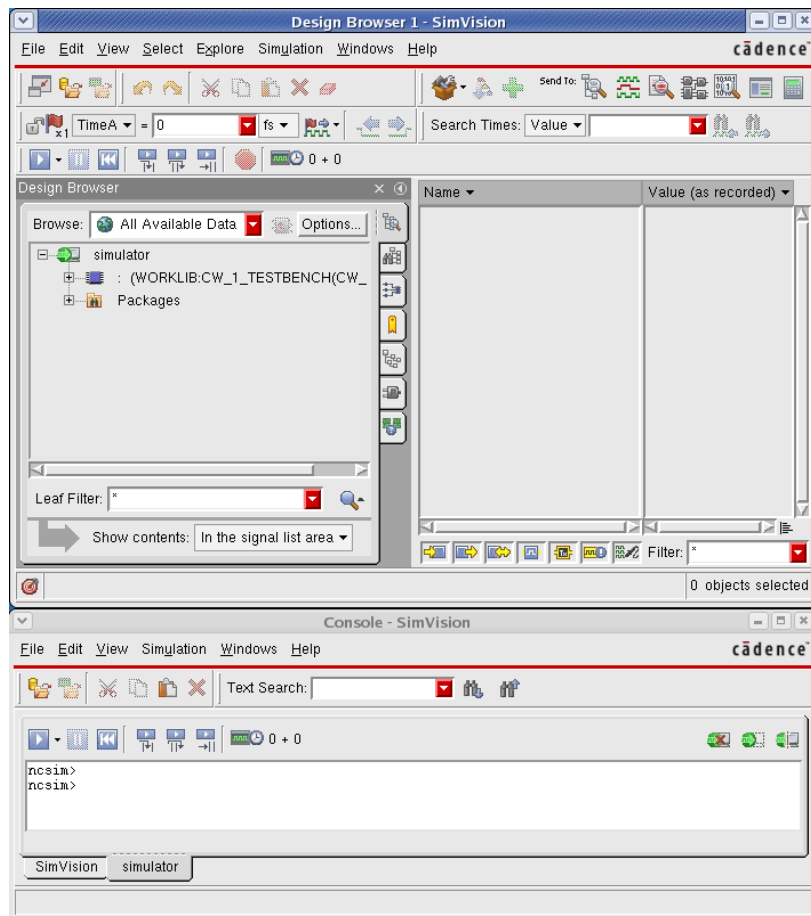
There are 2 modes of simulation possible but we use single step one. Please press button **Single Step** and below shown window should open:




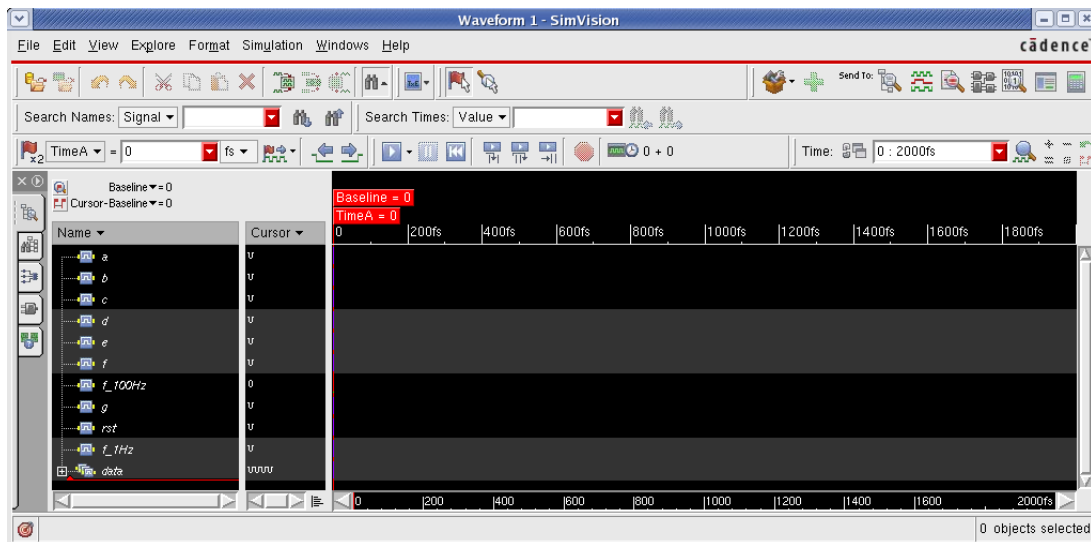
Default there is Verilog file filter set. To browse VHDL files enter **.vhd1** in **Filters** field. Next add all the files which were previously checked against VHDL syntax correctness. First browse for the file and next press  button. Then choose menu **Tools/ IRUN** and check field **Other Options** and enter text: **-top cw\_1\_testbench -V93**. It is necessary in order to indicate which block is of the highest hierarchy. After that window should look like:







Please click **OK** and if all the files are VHDL syntax correct **SimVtion** simulator should automatically start. **SimVtion** starts with 2 windows named: **Design Browser** and **Console** as it is shown below:



To make simulation and see results on the waveform signals should be selected and “send” to waveform. Please select signals in the **Design Browser** and press . Clicking „+” just next to name of design will expand hierarchy, all signals from any point of the hierarchy can be selected and then send to waveform. For example, in the figure below, all signals of block **cw\_1\_tesbench** and internal **data** and **f\_1Hz** are added to the waveform.




Simulation starts after clicking right arrow , but it is worth to limit end time before simulation starts (by clicking down arrow). Time unit can be changed using button: . Simulation restart can be made through . Time scale can be changed by dragging time cursor given at the bottom of waveform or by clicking one of the buttons  which are situated at the right high side. In the case when some VHDL files should be changed simulation restart can be performed using menu: **Simulation/ Reinvoke Simulator.../ Yes**.

**SimVision** simulator has many additional functions and only two of them are described below:



### **Finding places where signal is set/changed**

We want to find place in the source code where signal **f\_1Hz** changes value from 1 to 0, in order to make it is necessary to:

- made simulation containing signal **f\_1Hz** in the waveform,
- click on the signal **f\_1Hz**,
- click on the symbol , which starts searching for signal changes, it is necessary to click as many times as needed to find interesting time point,
- click menu **Explore/Go To/Cause, Source Browser** window should rise up in which it is possible to choose signal control cases (left side), at the same time right part shows source code with the arrow pointing to the code line,
- window including source code is used only for code browsing, if some code changes are to be made one can run text editor through menu **Edit/Edit File**, default system editor is **vi**, to change default to any other can choose menu **Edit/ Preferences/ Source Browser/ Editor Command**, and next enter command, for example command **xterm -e gedit %F** runs **gedit**,
- after source code changes simulation has to be restarted through menu **Simulation/ Reinvoke Simulator.../ Yes**.

### **Browsing schematics**

Schematic browsing can be made through selecting block to be viewed and then „sending” it to the **Schematic Tracer**. For example we want to see full schematic of our design then we should:

- a) in the window **Design Browser** select block **WORKLIB: CW\_1\_TESTBENCH**,
- b) send selection to the schematic tracer clicking ,
- c) **Schematic Tracer** window should open, selecting object and clicking  one can change current hierarchy point.

Annex: Model protocol:

<b>LAB 1, Design of ASICs</b>		
<b>No</b>	<b>Description</b>	<b>Value</b>
1	Date of completion	
2	Name and surname	
3	Screen of simulation window, all signals of the top block should be included. Simulation time 15 s.	
4	File: cw_1_testbench.vhdl	
5	File: cw_1.vhdl	
6	File: dzielnik.vhdl	
7	File: licznik.vhdl	
8	File: dekodier.vhdl	

*Bogdan Pankiewicz, Gdańsk, September 2010  
Amendments: Oct. 2011, Sep. 2014*



# LAB2

## Logic synthesis and simulation after synthesis

*Help for EDI11.12 - synthesis:* PDF version of the help to *RTL Compiler* is available: */cadence/cadence2012\_2013/edi\_11.1/doc/rc\_start/rc\_start.pdf* and *rc\_user/rc\_user.pdf*. Alternatively, it is also possible to run command *cdnshelp &* and next choose library EDI11.12/ RTL Compiler.

During the LAB2 logic synthesis of the circuit prepared as the result of the LAB1 is made. After that simulation of synthesized circuit is also made.

### NOTE:

If someone wants to use given in LAB2 commands directly, without need of paths and file names changes, should prepare working directory as follows:

- create directory *lab2* and enter it,
- create subdirectories: *hdl*, *rpt* and *out*,
- to perform above mentioned steps it is necessary to:  
open terminal and run following commands:

```
cd
mkdir lab2
cd lab2
mkdir hdl
mkdir rpt
mkdir out
```

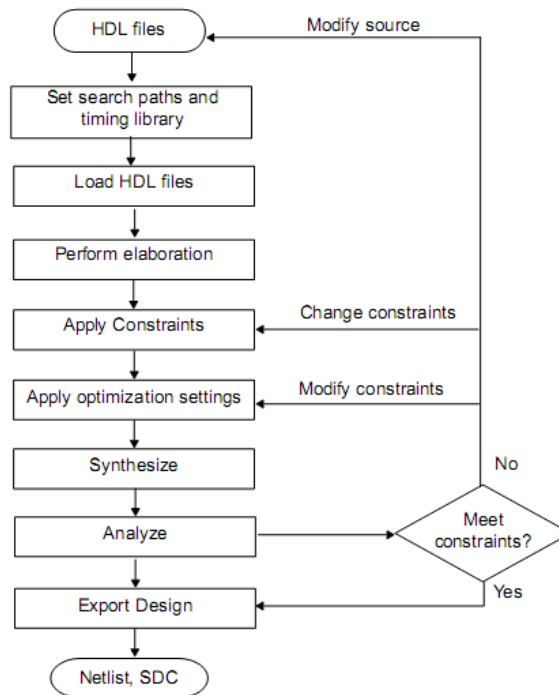
- next, into subdirectory *hdl* prepared during Lab1 VHDL files should be copied,
- running of the *RC Compiler* should be performed from *lab2* directory.

Cadence software for logic synthesis is *RTL Compiler*. Software starting is made from the terminal through command:

```
rc or rc -gui
```

if we want to also rise up graphical window. After starting up the software commands are issued using text mode. General help can be obtained by command *help*. Help regarding certain specific command can be obtained by command *help name\_of\_command*.

Typical design flow is presented in the below figure. This figure is taken from the *RTL Compiler* help and it is basic flow which does not contain advanced possibilities such as: DFT, clock gating, low power and many others.



During this lab we also will go through basic flow, if someone wants to extend please find appropriate chapter in the **RTL Compiler** manual. First step during the synthesis is setting of paths and libraries. All the settings are made through attribute setting. In order to set directory which will be searched for HDL files **hdl\_serch\_path** attribute should be set. For example if we want our HDL files to be searched in hdl subdirectory following command should be run:

```
set_attribute hdl_search_path ./hdl /
```

Character / at the end of the command means setting attribute to the root of the design. Setting path for library searching:

```
set_attribute lib_search_path /techfiles_ldap/umc_ksmi_lab/lib /
```

To check value of the attribute following command can be used:

```
get_attribute checked_attribute_name
```

At the computers in the lab EA308 directory /techfiles\_ldap contains various libraries divided for various manufacturers and technologies. During this lab libraries for 130nm CMOS UMC process are used and are located at /techfiles\_ldap/umc\_ksmi\_lab. Some of the files are confidential and users have access only to public files. Above mentioned directory includes following subdirectories:

- **lib** libraries of the standard cells in the **Liberty** format, three types of cells are given: typical, fast and slow,
- **techLEF** containing general LEF file (description of supported metal layers and vias),
- **cellsFEF** containing shapes and pinout of the cells,
- **verilog** and **VHDL** containing descriptions of the standard cells in Verilog and VHDL formats, can be used for simulations and are also available in three speed grades.

There is also possible to prepare script and run in **RC Compiler**. In order to do it include command can be used. **RC Compiler** during work logs all issued commands into default file named **rc.cmd**, logs also all console infos into file **rc.log**. To rerun previously made commands one can use command:

```
include rc.cmdxx where xx is the last number of log.
```

Going back to the lab, all the HDL files have to be placed in the **hdl** directory and the search path should be set as in above examples. Next read in libraries by command:

```
set_attribute library {cells_typical.lib cells_best.lib cells_worst.lib } /
```

In this point LEF files can also be read in. It is not necessary but improve delay calculation.

```
set_attribute      lef_library      {../techLEF/tech8m2t/tech8m2t.lef
../cellsLEF/cells.lef} /
```

Reading in of single file can be made without {} parenthesis, for many it is obligatory. In the next step HDL files are read in:

```
read_hdl -vhdl {cw_1.vhdl dekoderek.vhdl dzielnik.vhdl licznik.vhdl}
```

Source files should have UNIX line terminations otherwise errors may occur (Windows files have to be converted first). Parameter `-vhdl` means VHDL files, defaults RTL Compiler reads in Verilog files.

Next step is elaboration which builds general structure of the design and checks semantics correctness.

```
elaborate
```

After elaboration, if **RC Compiler** was run with `-gui` option, in the graphical window block diagram should appear. It can be browsed and one can go deeper through expanding "+" character in the **Hierarchy** tab. Next step is adding constraints. All the constraint can be divided into 3 main categories:

- work conditions,
- clock signals,
- I/O timing constraints.

Constraints can be set using following methods:

- manual entering in the RTL Compiler program,
- reading in constraint file,
- reading in constraint file in SDC format.

In this lab only timing constraints will be set. First clock with the period 3500ps and name CLK will be defined:

```
define_clock -name CLK -period 3500 [find / -port f_*]
```

Additionally I/O paths are supposed to have extra delay 100ps:

```
external_delay -output 100 -clock CLK [find / -port a]
external_delay -output 100 -clock CLK [find / -port b]
external_delay -output 100 -clock CLK [find / -port c]
external_delay -output 100 -clock CLK [find / -port d]
external_delay -output 100 -clock CLK [find / -port e]
external_delay -output 100 -clock CLK [find / -port f]
external_delay -output 100 -clock CLK [find / -port g]
external_delay -input 100 -clock CLK [find / -port rs*]
```

Command **find** works on the entire design hierarchy and can be used in scripts.

Next step is logic synthesis:

```
synthesize -to_mapped
```

After synthesis as the result netlist using standard cells is obtained. Graphical window of the program shows detailed schematic of resulting circuit.

Last step is reports generation final netlist generation:

```
report timing
report area
report timing > ./rpt/timing_umc.rpt
report area > ./rpt/area_umc.rpt
write_hdl > ./out/cw_1_umc.v
write_sdf > ./out/cw_1_umc.sdf
write_sdc > ./out/cw_1_umc.sdc
```

As the result of above commands, in directories **rpt** and **out**, reports and final netlist are placed. Additionally in the **out** directory constraint file in SDC format is placed. It is necessary for next lab – implementation.

Second part of the lab is timing simulation of the circuit after synthesis. Simulation is made identical as in Lab1, the only differences are that different files are to be read in. After launching simulation:

***nclaunch -new &***

Please choose Single step and add files:

***/techfiles\_ldap/umc\_ksmi\_lab/verilog/cells\_typical\_corrected.v*** - which is standard cells library in Verilog format,

***./out/cw1\_umc.v*** - which is netlist resulting from synthesis process and

***./hdl/cw\_1\_testbench.vhdl*** – which is previously prepared testbench for functional simulation

Before reading in testbench file please change frequency of the clock to 100MHz and fasten release of the reset. Simulation waveform should include delays caused by gates and flip-flops used in the circuit.

Annex: Model protocol:

<b>LAB 2, Design of ASICs</b>		
<b>No</b>	<b>Description</b>	<b>Value</b>
1	Date of completion	
2	Name and surname	
3	Snapshot of the synthesis graphical window – top hierarchy <b><i>cw_1</i></b>	
4	Snapshot of the synthesis graphical window – expanded block top hierarchy <b><i>licznik</i></b>	
5	Snapshot of the synthesis graphical window – expanded block <b><i>dzielnik</i></b>	
6	Snapshot of the synthesis graphical window – expanded block <b><i>dekoder</i></b>	
7	Snapshot of the simulation window which will show delays of signals	
8	Delay of the signal <b><i>f_1Hz</i></b> relative to <b><i>f_100Hz</i></b>	

LAB 3 – in translation process