Department of Electronic & Telecommunication Engineering
University of Moratuwa

B.Sc. Eng. Semester 7

EN 4603 – Digital IC Design

Laboratory Experiment 1: RTL Synthesis

Objectives: (a) To understand ASIC synthesis flow using Verilog RTL
(b) To familiarize with Cadence Genus synthesis tool

Software Required: Cadence Genus - Ver. 18.10

In this practical, you will be using Cadence Genus to synthesize an example RTL design: a transceiver. As inputs to Genus, you will provide

1. Source Verilog files
2. Technology libraries provided by the fabrication plant (here, 35 nm educational GPDK given by cadence) : .lib, .lef, .tch
3. Timing constraints

and will obtain the synthesized netlist (Verilog files) and further timing constrains (.sdc) as output. You will then analyze the area, timing and power of the synthesized design.

Note: Unix commands are given as **unix:** and Genus commands are given as **genus:**

Lab report:

You need to submit a small report, documenting your observations in step 4 and 5 with screenshots and explanations. The format of the report is not a concern. You need to demonstrate that you understand every step of the process. Ask the junior staff for ANY clarification.

Step 1: Understand the UART Transceivers Reference Design

Figure 1 shows the example design that is being used in this Lab. It contains two UART Transceivers connected together as shown. All the codes are written in Verilog HDL.

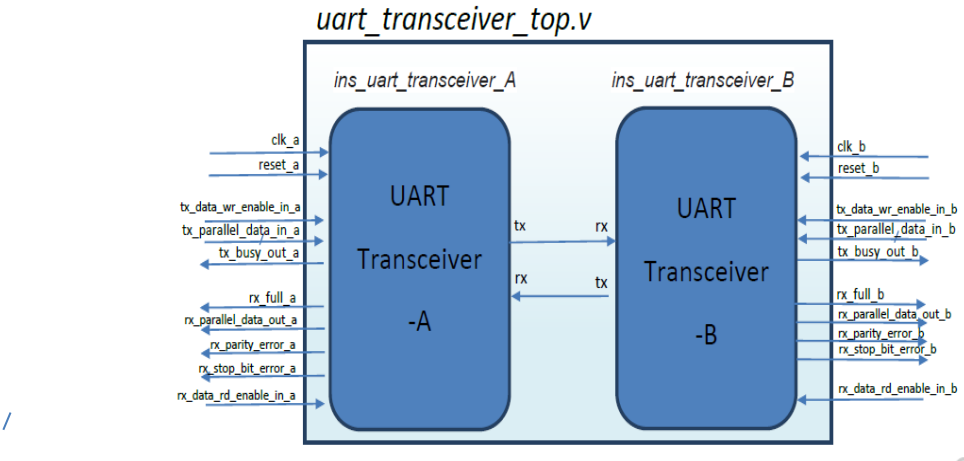


Figure 1

The design has two asynchronous clock signals and two reset signals. The clock signal of **Transceiver A** is ***clk_a*** and Transceiver B is ***clk_b***. The Transceivers A and B are having two reset signals, ***reset_a*** and ***reset_b*** respectively. Each Transceiver module sends and receives data according to the RS232 communication standard. User interface for reception and transmission is parallel.

The transmitter parallel user interface having three ports named:

1. *tx_data_wr_enable_in_x,*
2. *tx_parallel_data_in_x,* and
3. *tx_busy_out_x.*

The receiver interface ports are:

1. *rx_full_x,*
2. *rx_parallel_data_out_x,*
3. *rx_parity_error_x,*
4. *rx_stop_bit_error_x. and*
5. *rx_data_rd_enable_in_x.*

For more information about the design, go through its Verilog HDL files.

Step 2: Setup the Project Directories

1. Launch Terminal: In the start menu (application launcher) search and open: *Konsole*

2. Check if you are in the right directory:

   ```
   unix: pwd
   ```

   You should get /home/student/

3. Make a directory for your group.

   ```
   unix: mkdir entc17_dicd/<index_no>
   unix: cd entc17_dicd/<index_no>
   ```

4. Copy the lab files & examine the folder structure

   ```
   unix: cp -r ../../dicd_labs/lab1_synthesis ./
   unix: cd lab1_synthesis
   unix: ls
   ```

   lab1_synthesis:

   ```
   |--input/
       |--rtl              # Contains HDL source
       |--libs/            # Contains technology libraries (45nm GPDK)
       |--constraints.tcl  # Commands to set timing constraints
   |--log/                 # Stores generated log files
   |--output/              # Output Netlist & Constraints (SDC)
   |--report               # Generated reports
   |--work/                # This is the working directory of the project
   ```

5. Change directory to ***work***

Step 3: Using Genus

1. Start Genus (command line tool). Your terminal will change to the Genus command line.

```
unix: genus
```

2. Set the search path & include the technology libraries as follows. These libraries are provided by the foundry. However, for this practical we utilize educational 45nm Generic Libraries provided by Cadence. **This library is provided only for teaching digital IC design. It is strictly prohibited to use them for any kind of commercial purposes.**
   - Library Timing (.lib) files specify timing (cell delay, cell transition time, setup and hold time requirement) and power characteristics of standard cells. Slow and fast libraries characterise standard cells with maximum and minimum signal delays, which could occur from process variations.
   - Tch files are binary files that accurately characterize library elements, that include capacitance and resistance.
   - Library Exchange Format (LEF) specify design rules, metal capacitances, layer information…etc.

```
genus: set_db init_lib_search_path [list ../input/libs/gsclib045/lef
        ../input/libs/gsclib045/timing ../input/libs/gsclib045/qrc/qx]

genus: set_db library {slow_vdd1v0_basicCells.lib fast_vdd1v0_basicCells.lib}
genus: set_db lef_library {gsclib045_tech.lef gsclib045_macro.lef
        gsclib045_multibitsDFF.lef}
genus: set_db qrc_tech_file gpdk045.tch
```

3. Read the RTL design files into Genus:

```
genus: read_hdl [glob ../input/rtl/*.v]
```

4. Elaborate the design. During elaboration, the tool reads through your design, reports syntax errors, creates design hierarchy, applies parameters and connects signals. You need to provide the name of the top module (uart_top here)

```
genus: elaborate uart_top
```

5. Check design and output the log to a file. This reports any bugs: undriven pins, unloaded ports, unresolved references, empty modules…etc.

```
genus: check_design > ../log/check_design.log
```

6. Uniquify the top module. This eliminates sharing of subdesigns between instances

```
genus: uniquify uart_top
```

7. Set timing constraints in Genus. Open the constraints.tcl file with vim (in a separate terminal) and read it (then quit vim with [esc], !q, [enter]) . You can observe the period, duty cycle of two clocks, input delay of each input ports…etc.

```
unix: vim ../input/constraints.tcl
genus: source ../input/constraints.tcl
```

8. Finally, you can synthesize. to_mapped specifies the design to stop after optimizing RTL and mapping to the given technology. We are using medium effort. With high effort, you can get better results, but it runs for longer time.

```
genus: synthesize -to_mapped -effort m
```

9. Write netlist & constraints. Netlist is another Verilog file (open and read it!) where your behavioural source code has been mapped into the standard cells provided by the foundry. SDC file contains constraints needed by the place & route tools.

```
genus: write -mapped > ../output/uart_top.v
genus: write_sdc > ../output/uart_top.sdc

unix: vim ../output/uart_top.v
unix: vim ../output/uart_top.sdc
```

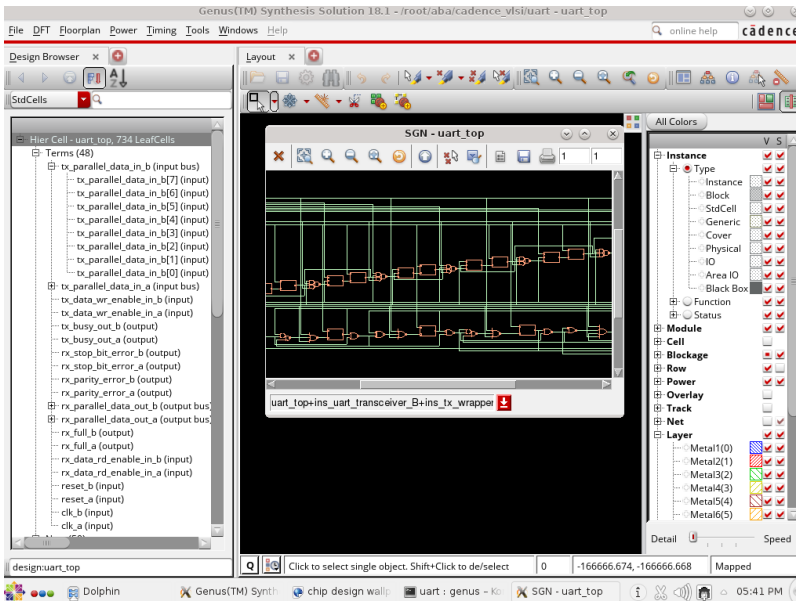
10. Now you can generate the reports and read them. You can see the area estimate (mm^2), power estimate, any timing violations…etc.:

```
genus: report_area > ../report/area.log
genus: report_timing  -nworst 10 > ../report/timing.log
genus: report_port * > ../report/ports_final.log
genus: report_power > ../report/power.log

unix: vim ../report/area.log
unix: vim ../report/timing.log
unix: vim ../report/ports_final.log
unix: vim ../report/power.log
```

11. You can observe the synthesized design in GUI too. Try to select the designs in hierarchy (left), right click, schematic view, in New. You can also report power, timing…etc.

```
genus: gui_show
```



Step 4: Exercise

1. Identify the system clocks, system resets and their properties.
2. If there are any derived clocks, identify their properties.
4. Observe the *area.log* and *constraints.log* on the *report* directory.
5. Change the top level designs parameters;

   TX_WORD_LENGTH =32 and RX_WORD_LENGTH =32.

   Next, elaborate and synthesize the design again.
6. Observe the area report and document your results.
7. Observe the timing report, *timing.log* on *report* directory.
8. Identify timing violations if there is any.
9. Without modifying the HDL source files, do the necessary changes to constraints.tcl and synthesize the design to meet the below criteria.
   a) Change the system clocks of both A and B transceivers to operate at 50MHz.
   b) Change the duty cycle of both clocks to 40%
   c) Introduce 180^{o} phase shift to clock B
   d) Increase the pin load by 5pF and check the timing violations