Department of Electronic and Telecommunication Engineering

University of Moratuwa, Sri Lanka

EN4603 - Digital IC Design

Laboratory Experiment 1
RTL Synthesis

Laboratory Report

Submitted by

Thalagala B.P. 180631J

Submitted on

January 12, 2023

Contents

List of Figures

List of Tables

List of Abbreviations

ASIC	Application Specific Integrated Circuit
GPDK	Generic Process Design Kit
HDL	Hardware Description Language
RTL	Register-Transfer Level
TCL	Tool Command Language

Note:
All the materials related to the report can also be found at https://github.com/bimalka98/Digital-IC-Design

1 Introduction

1.1 Practical

In this practical, we will be using *Cadence Genus - Ver. 18.10* to synthesize an example RTL design, a transceiver. As inputs to Genus, we will provide

1. Source Verilog files
2. Technology libraries provided by the fabrication plant (here, 45 *nm* educational Generic Process Design Kit (GPDK) given by Cadence) : `(.lib, .lef, .tch)`
   - 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 characterize 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.
3. Timing constraints

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

1.2 RTL Synthesis

In the context of digital hardware design, RTL synthesis is the process of converting an RTL description of a digital circuit into an optimized gate-level design.

The RTL description of a digital circuit is written in a Hardware Description Language (HDL) such as Verilog or VHDL. It specifies the digital circuit in terms of the flow of digital signals between registers, and the logical operations that are performed on those signals as they are transferred between the registers.

During RTL synthesis, an RTL compiler reads the RTL description of the digital circuit and generates an optimized gate-level representation of the circuit. This gate-level representation is a description of the digital circuit in terms of gates and interconnections between them. Figure 1 illustrates an overview of an RTL synthesis flow.

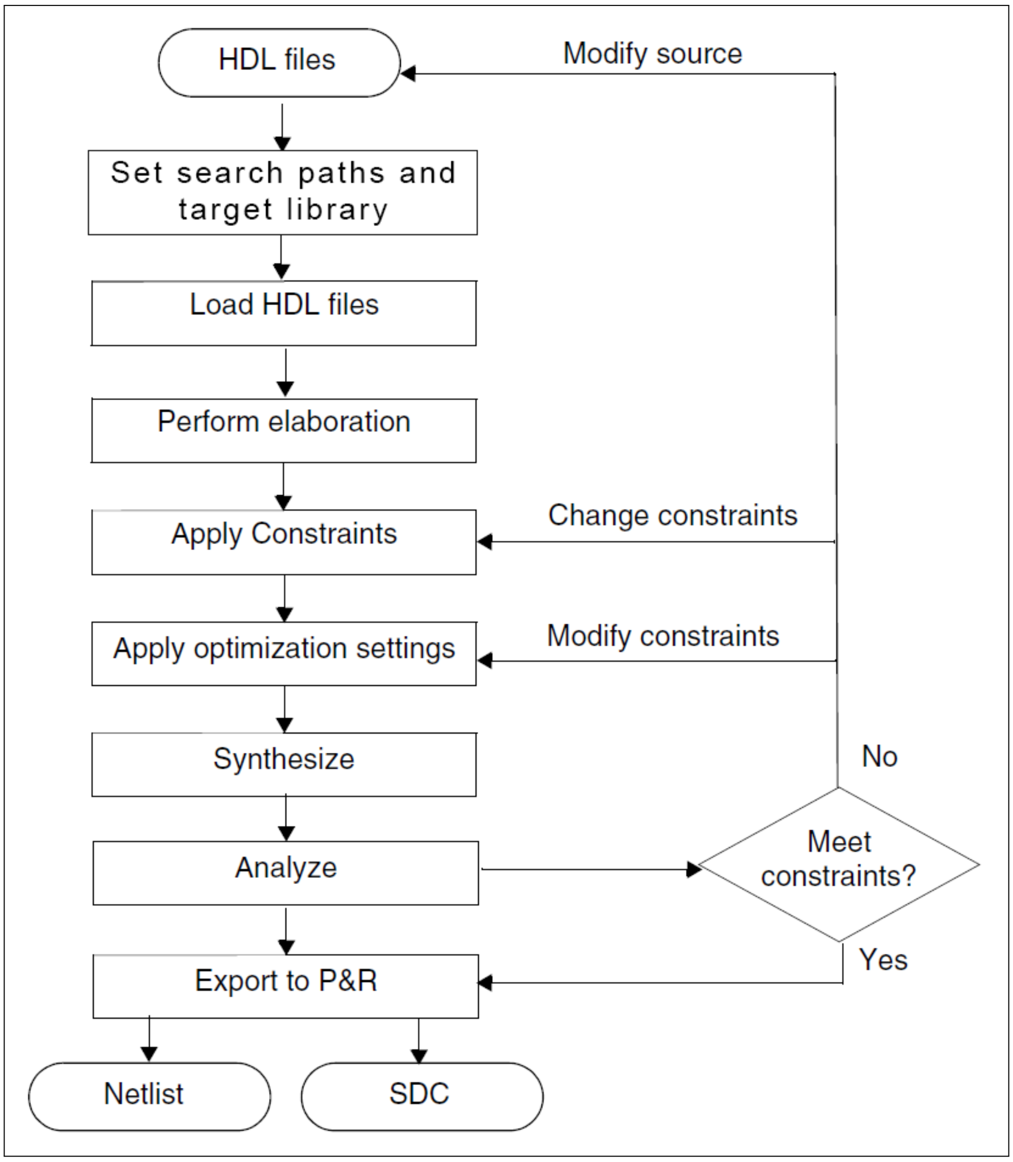


Figure 1: Overview of RTL synthesis flow of Cadence Genus software[1].

1.3 Associated Genus Commands

Note: *Genus is a Tool Command Language (TCL) based tool and therefore* `.tcl` *scripts can be created to execute a series of commands instead of typing each command individually. The entire interface of Genus is accessible through TCL, and true TCL syntax and semantics are supported.*

Followings are the commands used in this lab, and the entire script could be executed once using a single `.tcl` file. However, it was encouraged to execute the commands one-by-one in order to understand the process of synthesizing an RTL design.

```
# 1. Link Technology Library
set_db init_lib_search_path [list ../input/libs/gsclib045/lef
    ../input/libs/gsclib045/timing ../input/libs/gsclib045/qrc/qx]
set_db library {slow_vdd1v0_basicCells.lib fast_vdd1v0_basicCells.lib}
set_db lef_library {gsclib045_tech.lef gsclib045_macro.lef
    gsclib045_multibitsDFF.lef}
set_db qrc_tech_file gpdk045.tch

# 2. Read HDLs
read_hdl [glob ../input/rtl/*.v]

# 3. Elaborate the top module
elaborate uart_top

# 4. Check Design
check_design > ../log/check_design.log

# 5. Uniquifies the instances under the specified design or subdesign.
uniquify uart_top

# 6. Set constraints
source ../input/constraints.tcl

# 7. Synthesize
synthesize -to_mapped -effort m

# 8. Write netlist
write -mapped > ../output/uart_top.v
write_sdc > ../output/uart_top.sdc

# 9. Reports
report_area > ../report/area.log
report_timing  -nworst 10 > ../report/timing.log
report_constraint > ../report/constraint.log
report_port * > ../report/ports_final.log
report_power > ../report/power.log
```

2 Exercise

This section documents the observations made in step 4 and 5 of the practical guide, with screenshots and explanations.

2.1 System Clocks & Resets

2.1.1 System Clocks

The specifications related to the system clocks and other constraints are defined in the `constraints.tcl` script file, which can be executed once using the Genus software. The units of the clocks, and the other parameters related to timings are in nanoseconds (ns) scale, as defines in the Verilog source files by `timescale 1ns/1ps`.

The command `create_clock`[2] is used to define the clocks, and the necessary parameters related to them.

```
create_clock -name clk_a -period 10 [get_ports clk_a] -waveform {0 5}
create_clock -name clk_b -period 10 [get_ports clk_b] -waveform {0 5}
```

The clocks specified in the constraints file has the properties described in the Table 1. In addition to those properties, constraints related to the clock network uncertainty is also defined in the same file. In practical Application Specific Integrated Circuit (ASIC) design, ideal clock networks do not exist and clock signal arrival time may differ from cell to cell. In order to facilitate this, a parameter known as ***clock uncertainty*** is defined. It takes into account all the possible variations of the clock signal such as 1. ***jitter***, which is caused by the physical properties of the clock source, and 2. ***skew***, which is due to the routing length variations.

The `set_clock_uncertainty`[2] command is used to define the clock uncertainty as below.

```
set_clock_uncertainty 0.5 [all_clocks]
```

TABLE 1
PROPERTIES OF THE SYSTEM CLOCKS. (TIME UNIT = ns)

Name	Period	Rise Time	Fall Time	Clock Uncertainty
`clk_a`	10	0	5	0.5
`clk_b`	10	0	5	0.5

2.1.2 System Resets

Two system reset signals are defined in the top module `uart_top.v` Verilog source file, as `reset_a` and `reset_b`. Both resets are synchronous and active high.

```
always@(posedge clk) begin
    if (reset) begin
      some statements;
    end
end
```

2.1.3 Derived Clocks

A *derived clock* (a new clock signal) can be created from the clock waveform of a given pin in the design using the command `create_generated_clock`[2]. Howevr, the given design for this lab does not in cooperate any derived clocks.

2.2 Design Log Files

2.2.1 Area

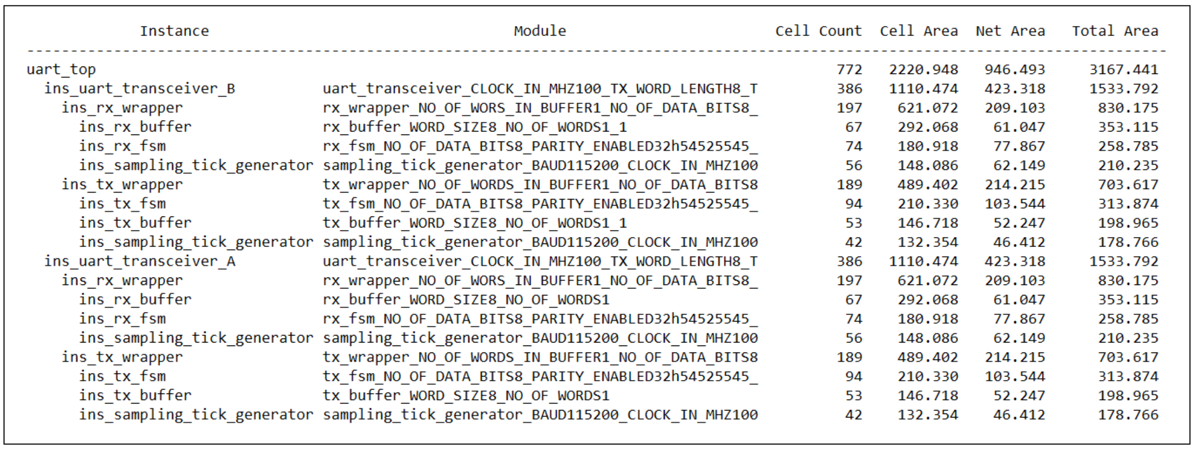
`area.log` file in the `report` directory carries the information shown in the Figure 2. It provides a breakdown of the area usage by design hierarchy and by instance, which can be helpful in identifying specific modules that are contributing to the overall area of the design. Specifically it provides the below information[2].

i. ***Cell Count*** : The total count of cells mapped against the hierarchical blocks in the current design.

ii. ***Cell Area*** : The combined cell area in each of the blocks and the top level design (hierarchical breakup)

iii. ***Net Area*** : The estimated post-route net area, which is based on the minimum wire widths defined in the LEF and capacitance table files and the area of the design blocks.

iv. ***Total Area*** : Simply combines the 'Cell Area' and the 'Net Area'



```
            Instance                                   Module                    Cell Count  Cell Area  Net Area   Total Area
----------------------------------------------------------------------------------------------------------------------------------
uart_top                                                                                772   2220.948   946.493     3167.441
  ins_uart_transceiver_B         uart_transceiver_CLOCK_IN_MHZ100_TX_WORD_LENGTH8_T    386   1110.474   423.318     1533.792
    ins_rx_wrapper               rx_wrapper_NO_OF_WORS_IN_BUFFER1_NO_OF_DATA_BITS8_    197    621.072   209.103      830.175
      ins_rx_buffer              rx_buffer_WORD_SIZE8_NO_OF_WORDS1_1                     67    292.068    61.047      353.115
      ins_rx_fsm                 rx_fsm_NO_OF_DATA_BITS8_PARITY_ENABLED32h54525545_      74    180.918    77.867      258.785
      ins_sampling_tick_generator sampling_tick_generator_BAUD115200_CLOCK_IN_MHZ100     56    148.086    62.149      210.235
    ins_tx_wrapper               tx_wrapper_NO_OF_WORDS_IN_BUFFER1_NO_OF_DATA_BITS8    189    489.402   214.215      703.617
      ins_tx_fsm                 tx_fsm_NO_OF_DATA_BITS8_PARITY_ENABLED32h54525545_      94    210.330   103.544      313.874
      ins_tx_buffer              tx_buffer_WORD_SIZE8_NO_OF_WORDS1_1                     53    146.718    52.247      198.965
      ins_sampling_tick_generator sampling_tick_generator_BAUD115200_CLOCK_IN_MHZ100     42    132.354    46.412      178.766
  ins_uart_transceiver_A         uart_transceiver_CLOCK_IN_MHZ100_TX_WORD_LENGTH8_T    386   1110.474   423.318     1533.792
    ins_rx_wrapper               rx_wrapper_NO_OF_WORS_IN_BUFFER1_NO_OF_DATA_BITS8_    197    621.072   209.103      830.175
      ins_rx_buffer              rx_buffer_WORD_SIZE8_NO_OF_WORDS1                       67    292.068    61.047      353.115
      ins_rx_fsm                 rx_fsm_NO_OF_DATA_BITS8_PARITY_ENABLED32h54525545_      74    180.918    77.867      258.785
      ins_sampling_tick_generator sampling_tick_generator_BAUD115200_CLOCK_IN_MHZ100     56    148.086    62.149      210.235
    ins_tx_wrapper               tx_wrapper_NO_OF_WORDS_IN_BUFFER1_NO_OF_DATA_BITS8    189    489.402   214.215      703.617
      ins_tx_fsm                 tx_fsm_NO_OF_DATA_BITS8_PARITY_ENABLED32h54525545_      94    210.330   103.544      313.874
      ins_tx_buffer              tx_buffer_WORD_SIZE8_NO_OF_WORDS1                       53    146.718    52.247      198.965
      ins_sampling_tick_generator sampling_tick_generator_BAUD115200_CLOCK_IN_MHZ100     42    132.354    46.412      178.766
```

Figure 2: Area log file

2.2.2 Power

2.2.3 Timing

2.2.4 Ports

Bibliography

[1] "Genus User Guide for Legacy UI," version 19.1, November 2019, published by *Cadence Design Systems, Inc.*

[2] "Genus Command Reference," version 19.1, November 2019, published by *Cadence Design Systems, Inc.*