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 14, 2023

Contents

List of Figures 1

List of Tables 1

List of Abbreviations 1

1 Introduction 2
- 1.1 Practical . . . 2
- 1.2 RTL Synthesis . . . 3
- 1.3 Associated Genus Commands . . . 4

2 Exercise 5
- 2.1 System Clocks & Resets . . . 5
  - 2.1.1 System Clocks . . . 5
  - 2.1.2 System Resets . . . 5
  - 2.1.3 Derived Clocks . . . 6
- 2.2 Design Log Files . . . 6
  - 2.2.1 Area . . . 6
  - 2.2.2 Power . . . 6
  - 2.2.3 Timing . . . 7
  - 2.2.4 Ports . . . 8

Bibliography 9

List of Figures

1 Overview of Register-Transfer Level (RTL) synthesis flow of Cadence Genus software[1]. . . . 3
2 Area log file . . . 6
3 Power log file . . . 7
4 Timing log file . . . 8

List of Tables

1 Properties of the system clocks. (time unit = ns) . . . 5

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.

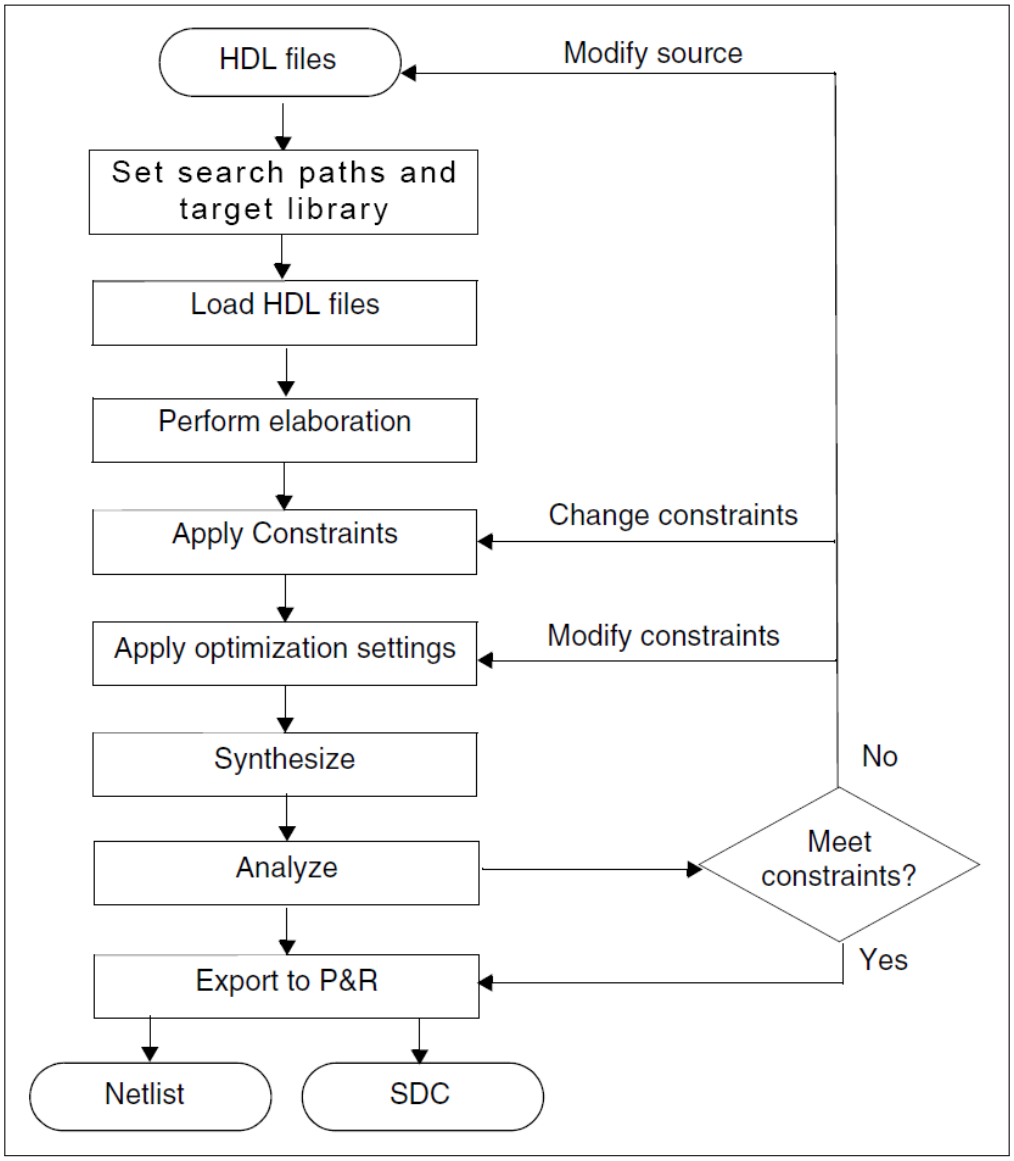

Figure 1: Overview of RTL synthesis flow of Cadence Genus software[1].

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

The `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

The `power.log` file in the `report` directory carries the details shown in the Figure 3. It provides the information related to the power consumption; however, the returned information depends on the current position in the design hierarchy and on the specified objects. If no objects are specified, the report is given for the design or instance at the current position in the design hierarchy[2].

i. **Leakage Power** refers to the power that is consumed by the circuit even when it is in a quiescent or standby state. This type of power consumption is caused by the leakage current flowing through the transistors and other components in the circuit.

ii. **Dynamic Power** refers to the power that is consumed by the circuit when it is actively switching or performing computations. This type of power consumption is caused by the charging and discharging of the load capacitance at the inputs and outputs of the transistors.

iii. **Total Power** is the sum of leakage power and dynamic power, it is the overall power consumed by the circuit.

Instance	Cells	Leakage Power(nW)	Dynamic Power(nW)	Total Power(nW)
uart_top	772	60.819	172490.052	172550.870
ins_uart_transceiver_B	386	30.415	87250.682	87281.098
ins_rx_wrapper	197	19.437	60900.810	60920.247
ins_rx_buffer	67	11.235	44449.732	44460.967
ins_rx_fsm	74	5.135	11368.789	11373.924
ins_sampli..ick_generator	56	3.067	5082.288	5085.355
ins_tx_wrapper	189	10.979	26349.873	26360.851
ins_tx_fsm	94	5.124	14188.375	14193.500
ins_tx_buffer	53	3.228	7822.657	7825.885
ins_sampli..ick_generator	42	2.626	4338.840	4341.466
ins_uart_transceiver_A	386	30.403	80046.763	80077.167
ins_rx_wrapper	197	19.422	54041.240	54060.663
ins_rx_buffer	67	11.223	37609.104	37620.327
ins_rx_fsm	74	5.146	12300.830	12305.977
ins_sampli..ick_generator	56	3.053	4131.306	4134.359
ins_tx_wrapper	189	10.981	26005.523	26016.504
ins_tx_fsm	94	5.117	13676.204	13681.321
ins_tx_buffer	53	3.232	8467.156	8470.388
ins_sampli..ick_generator	42	2.632	3862.163	3864.795

Figure 3: Power log file

2.2.3 Timing

The `timing.log` file in the `report` directory carries the details shown in the Figure 4. The command given below was used to generate this timing report. '`-nworst 10`' argument in the command specifies that, the maximum number of paths to report to each endpoint is 10.

```
report_timing  -nworst 10 > ../report/timing.log

Warning : Possible timing problems have been detected in this design. [TIM-11]
: The design is 'uart_top'.
: Use 'report timing -lint' for more information.
```

In addition, below constraints are defined in the `constraints.tcl` to be used for the synthesis. Here the input delay is constrained to 6 $ns = 6000$ ps, and this information is also available in the timing report shown in the Figure 4.

```
set_input_delay  6 -clock clk_a  $design_inputs_a
set_input_delay  6 -clock clk_b  $design_inputs_b
set_output_delay 6 -clock clk_a $design_outputs_a
set_output_delay 6 -clock clk_b $design_outputs_b
```

```
Path 1: MET (2539 ps) Setup Check with Pin ins_uart_transceiver_B/ins_rx_wrapper/ins_rx_buffer/buffer_full_counter_reg[2]/CK->D
          Group: clk_b
     Startpoint: (F) rx_data_rd_enable_in_b
          Clock: (R) clk_b
       Endpoint: (R) ins_uart_transceiver_B/ins_rx_wrapper/ins_rx_buffer/buffer_full_counter_reg[2]/D
          Clock: (R) clk_b

                    Capture       Launch
        Clock Edge:+   10000            0
        Drv Adjust:+       0            0
       Src Latency:+       0            0
       Net Latency:+       0 (I)        0 (I)
           Arrival:=   10000            0

             Setup:-     168
       Uncertainty:-     500
     Required Time:=    9332
      Launch Clock:-       0
       Input Delay:-    6000
         Data Path:-     793
             Slack:=    2539

Exceptions/Constraints:
  input_delay             6000            in_del_19_1

#-------------------------------------------------------------------------------------------------------------------------------------------
#                                Timing Point                                 Flags   Arc    Edge   Cell      Fanout Load Trans Delay Arrival Instance
#                                                                                                               (fF)  (ps)  (ps)   (ps)  Location
#-------------------------------------------------------------------------------------------------------------------------------------------
  rx_data_rd_enable_in_b                                                      -       -      F     (arrival)       9 11.4     0     0    6000    (-,-)
  ins_uart_transceiver_B/ins_rx_wrapper/ins_rx_buffer/g598/Y                  -       B->Y   R     NOR2BX1         3  4.4   216   132    6132    (-,-)
  ins_uart_transceiver_B/ins_rx_wrapper/ins_rx_buffer/g593/Y                  -       B->Y   F     NOR2X1          2  2.9   123   181    6312    (-,-)
  ins_uart_transceiver_B/ins_rx_wrapper/ins_rx_buffer/g574/Y                  -       B0->Y  R     OAI21X1         1  2.0   144   104    6416    (-,-)
  ins_uart_transceiver_B/ins_rx_wrapper/ins_rx_buffer/g560/Y                  -       A1->Y  F     AOI22X1         1  1.9   190   206    6623    (-,-)
  ins_uart_transceiver_B/ins_rx_wrapper/ins_rx_buffer/g555/Y                  -       B->Y   R     NOR2X1          1  1.9   127   171    6793    (-,-)
  ins_uart_transceiver_B/ins_rx_wrapper/ins_rx_buffer/buffer_full_counter_reg[2]/D -       -      R     DFFHQX1         1    -     -     0    6793    (-,-)
#-------------------------------------------------------------------------------------------------------------------------------------------
```

Figure 4: Timing log file

According to the information found on the timing log file, **the most critical path** (Path 1 in Figure 4) has a positive slack time of 2539 *ps*. This is the least slack time of the given design. This indicates that, there is no timing violations in the design.

Note: Slack is a measure of the timing margin available in a design. It is the difference between the *Required Time* and the *Arrival Time* of a signal at a specific endpoint. Positive slack indicates that the design meets the timing constraints and there is extra time available, while negative slack indicates that the design does not meet the timing constraints and the path is considered to be in violation. **The paths with the least slack are the most critical paths**. A design with a high slack value has more margin for manufacturing variations, temperature changes, and other factors that can affect the timing of the system.

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.*