Department of Electronic and Telecommunication Engineering

University of Moratuwa, Sri Lanka

EN4603 - Digital IC Design

Laboratory Experiment 2
RTL Synthesis

Laboratory Report

Submitted by

Submitted by

C.S.Pallikkonda	180441C
R.M.A.S.Rathnayake	180534N
B.P.Thalagala	180631J

Submitted on

February 18, 2023

Contents

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

List of Figures

List of Tables

List of Abbreviations

ASIC Application Specific Integrated Circuit

ATPG Automatic Test Pattern Generation

DEF Design Exchange Format

DFT Design For Testability

GPDK Generic Process Design Kit

HDL Hardware Description Language

RTL Register-Transfer Level

RX Receiver

TCL Tool Command Language

TX Transmitter

UART Universal Asynchronous Receiver Transmitter

1 Introduction

1.1 Practical

In this practical, we will be using *Cadence Genus - Ver. 18.10* to perform DFT insertion into an example Register-Transfer Level (RTL) design, a UART transceiver (shown in Figure 1). 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 Scan-test compatible netlist (Verilog files), timing constrains (.sdc file), scanDesign Exchange Format (DEF) file and a set of files to be used as inputs to Cadence Modus for Automatic Test Pattern Generation (ATPG) as output. We will then analyze, compare and comment on the area and ports of the design at various stages of the Scan test insertion design flow.

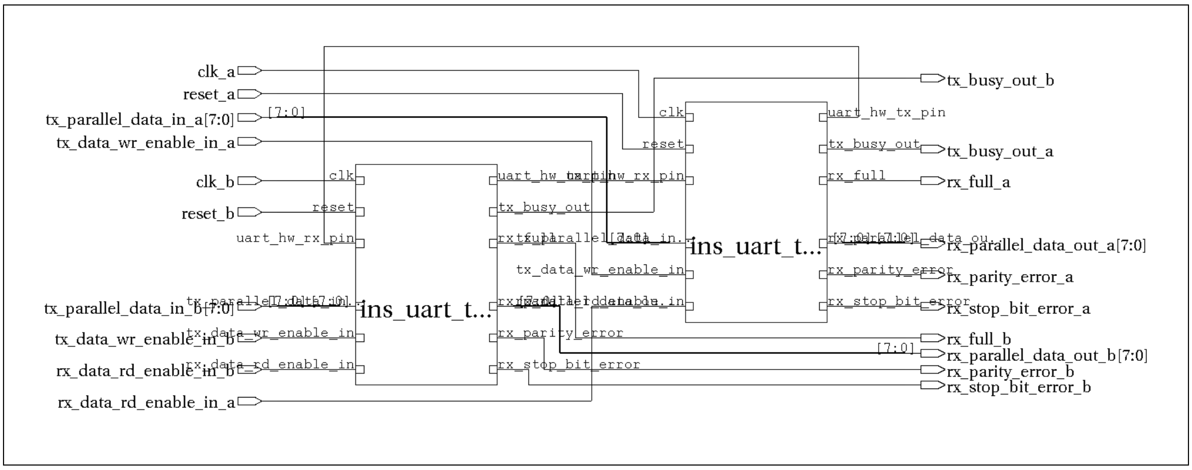


Figure 1: Top level diagram of the UART design

1.2 DFT Insertion

In the context of Application Specific Integrated Circuit (ASIC) design, DFT refers to the process of inserting additional logic to improve the testability of the design. Here the testability is defined as the ability to communicate to the internal nodes through the primary input and output ports. The goal in DFT is to cover a maximum number of faults with a minimum amount of additional logic, as the insertion of additional logic directly translates to additional cost per chip.Although there are various DFT techniques in digital IC design, in this experiment, we will be focusing on inserting the 'Scan Test' into an example design.

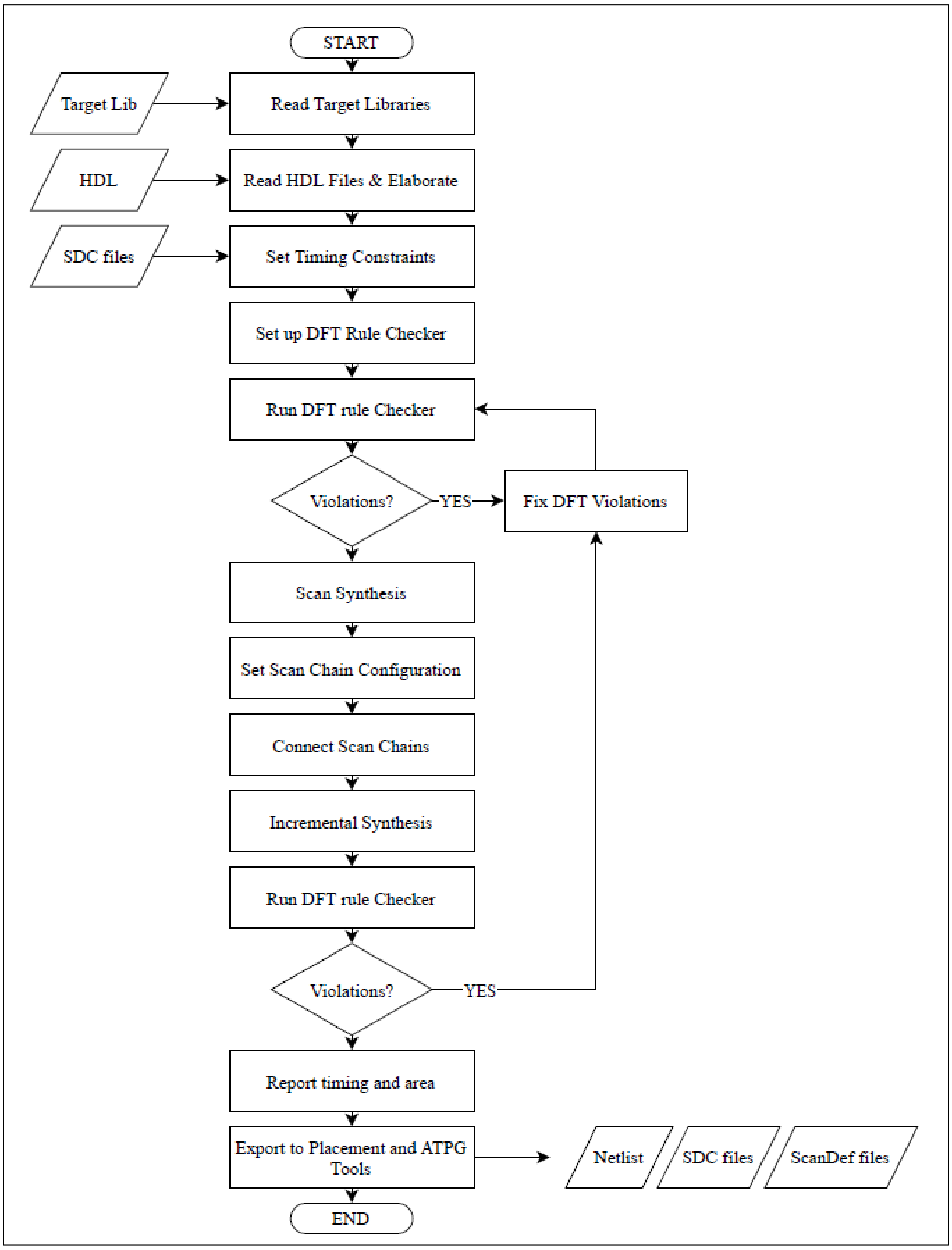


Figure 2: Overview of DFT insertion flow of 'Scan Test' in Cadence Genus software.

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 DFT insertion to an RTL design.

```
# 1. Link Technology Library
source ../scripts/setup.tcl

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

# 3. Elaborate the top module
elaborate uart_top

# Uniquify the top module
uniquify uart_top

# 5. Set timing constraints
source ../input/constraints.tcl
```

Now that we have set the target libraries, read the design files and set the design constrains as done in the Laboratory Experiment 1. We can now proceed to the DFT insertion steps.

```
# 7. Set the DFT scan style in order to configure the DFT rule checker
set_db dft_scan_style muxed_scan

# 8. Set the prefix for names of additional modules/ports
set_db dft_prefix dft_

# 9. Define shift_enable signal
define_shift_enable -name SE -active high -create_port SE

# 10. Run DFT rule checker
check_dft_rules
```

If all steps have been followed correctly up until this point, we should not see any DFT violations, and Genus should generate an output similar to that of Figure 3. The DFT rule checker checks all flip-flops to determine if clock pins to the flip-flops can be controlled and if asynchronous set/reset pins (if available) to the flip-flops can be held to their non-controlling value during scan-shift mode. It is essential that there are no DFT violations at this step as any flip-flops with DFT violations will not be affected by DFT insertion commands which will be executed in the subsequent steps.

```
@file(lab2.tcl) 38: check_dft_rules
  Checking DFT rules for 'uart_top' module under 'muxed_scan' style

  Checking DFT rules for clock pins
  Checking DFT rules for async. pins
  Checking DFT rules for shift registers.
Detected 0 DFT rule violation(s)
        Summary of check_dft_rules
        **************************
        Number of usable scan cells: 48
Clock Rule Violations:
---------------------
          Internally driven clock net: 0
              Tied constant clock net: 0
                   Undriven clock net: 0
        Conflicting async & clock net: 0
                      Misc. clock net: 0

Async. set/reset Rule Violations:
--------------------------------
        Internally driven async net: 0
              Tied active async net: 0
                 Undriven async net: 0
                    Misc. async net: 0

   Total number of DFT violations: 0

  Total number of Test Clock Domains: 2
  Number of user specified non-Scan registers:   0
      Number of registers that fail DFT rules:   0
      Number of registers that pass DFT rules: 176
  Percentage of total registers that are scannable: 100%
```

Figure 3: Genus log for the `check_dft_rules` command

Now the design is synthesized into the generic logic netlist, and then mapped to the technology library.

```
# 11.1 Synthesize to generic logic with medium effort
set_db syn_generic_effort medium
syn_generic

# 11.2 Map to technology library and re-synthesize with medium effort
set_db syn_map_effort medium
syn_map

# 12. Write the scan synthesized netlist as uart_top_1.v
write_hdl > ../output/uart_top_1.v

# 13. Generate the reports after scan synthesis.
report_area > ../report/after_scan_synthesis/area.log
report_timing -nworst 10 > ../report/after_scan_synthesis/timing.log
report_port * > ../report/after_scan_synthesis/ports.log
report_power > ../report/after_scan_synthesis/power.log
```

Then the Scan Configuration and Scan Stitching is done using the below commands. The genus log of those commands are shown in the Figure 4.

```
# 14. Set scan configuration
define_scan_chain -name top_chain_a -sdi scan_in_a -sdo scan_out_a
    -non_shared_output -create_ports -domain clk_a
define_scan_chain -name top_chain_b -sdi scan_in_b -sdo scan_out_b
    -non_shared_output -create_ports -domain clk_b

# 15. Preview the scan chains/Scan Stitching
connect_scan_chains -preview -auto_create_chains

# 16. Preview the scan chains/Scan Stitching
connect_scan_chains -auto_create_chains
```

```
@file(lab2.tcl) 64: define_scan_chain -name top_chain_a -sdi scan_in_a -sdo scan_out_a -non_shared_output -create_ports -domain clk_a
Pin or port 'scan_in_a' not found.
Pin or port 'scan_out_a' not found.
@file(lab2.tcl) 65: define_scan_chain -name top_chain_b -sdi scan_in_b -sdo scan_out_b -non_shared_output -create_ports -domain clk_b
Pin or port 'scan_in_b' not found.
Pin or port 'scan_out_b' not found.
@file(lab2.tcl) 68: connect_scan_chains -preview -auto_create_chains
  Starting DFT Scan Configuration for module 'uart_top' in 'normal' mode, with physical flow OFF
  Configuring 2 chains for 164 scan f/f
  Configured 1 chains for Domain: 'clk_a', edge: 'rising'
        top_chain_a (scan_in_a -> scan_out_a) has 82 registers; Domain:clk_a, edge: rising
  Configured 1 chains for Domain: 'clk_b', edge: 'rising'
        top_chain_b (scan_in_b -> scan_out_b) has 82 registers; Domain:clk_b, edge: rising
Processing 2 scan chains in 'muxed_scan' style.
Default shift enable signal is 'SE': 'port:uart_top/SE' active high.
@file(lab2.tcl) 71: connect_scan_chains -auto_create_chains
  Starting DFT Scan Configuration for module 'uart_top' in 'normal' mode, with physical flow OFF
  Configuring 2 chains for 164 scan f/f
  Configured 1 chains for Domain: 'clk_a', edge: 'rising'
        top_chain_a (scan_in_a -> scan_out_a) has 82 registers; Domain:clk_a, edge: rising
  Configured 1 chains for Domain: 'clk_b', edge: 'rising'
        top_chain_b (scan_in_b -> scan_out_b) has 82 registers; Domain:clk_b, edge: rising
  Processing 2 scan chains in 'muxed_scan' style.
  Default shift enable signal is 'SE': 'port:uart_top/SE' active high.
Mapping DFT logic introduced by scan chain connection...

Mapping DFT logic done.
```

Figure 4: Genus log for the scan configuration and scan stitching commands

After scan chain connecting, incremental synthesis is performed to generate th netlist of the scan connected design. Then DFT rules are checked again to identify any potential violations. The Genus log of that is shown in the Figure 5.

```
# 17. Perform incremental synthesis to generate the netlist
# of the scan connected design
syn_opt -incr

# 18. Perform DFT rule check after scan connecting
check_dft_rules
```

```
@file(lab2.tcl) 77: check_dft_rules
  Checking DFT rules for 'uart_top' module under 'muxed_scan' style

  Checking DFT rules for clock pins
  Checking DFT rules for async. pins
  Checking DFT rules for shift registers.
Detected 0 DFT rule violation(s)
        Summary of check_dft_rules
        **************************

        Number of usable scan cells: 48
Clock Rule Violations:
---------------------
          Internally driven clock net: 0
              Tied constant clock net: 0
                   Undriven clock net: 0
        Conflicting async & clock net: 0
                       Misc. clock net: 0

Async. set/reset Rule Violations:
--------------------------------
        Internally driven async net: 0
              Tied active async net: 0
                 Undriven async net: 0
                    Misc. async net: 0

   Total number of DFT violations: 0

  Total number of Test Clock Domains: 2
  Number of user specified non-Scan registers:   0
      Number of registers that fail DFT rules:   0
      Number of registers that pass DFT rules: 164
  Percentage of total registers that are scannable: 100%
```

Figure 5: Genus log for the `check_dft_rules` command after scan chain connecting

Once the above mentioned steps are completed, the below commands are used to finalize the DFT insertion process. They generate the Scan-test compatible netlist (Verilog files), further timing constrains (.sdc file), scanDEF file and a set of files to be used as inputs to Cadence Modus for ATPG as output.

```
# 19. Report scan setup and scan chain information
report_scan_setup > ../report/scan_setup.log
report_scan_chains > ../report/scan_chains.log

# 20. Write the DFT (scan test) inserted netlist and constrains.
write_hdl > ../output/uart_top_2.v
write_sdc > ../output/uart_top_2.sdc

# 21. Write the scanDEF file
write_scandef > ../output/uart_top_2_scanDEF.scandef

# 22. Generate the reports after scan connect
report_area > ../report/after_scan_connect/area.log
report_timing -nworst 10 > ../report/after_scan_connect/timing.log
report_port * > ../report/after_scan_connect/ports.log
report_power > ../report/after_scan_connect/power.log

# 23. Write the scripts required for the ATPG tool
write_dft_atpg -library
    ../input/libs/gsclib045/timing/slow_vdd1v0_basicCells.lib
```

1.4 Top Level Design before and after DFT Insertion

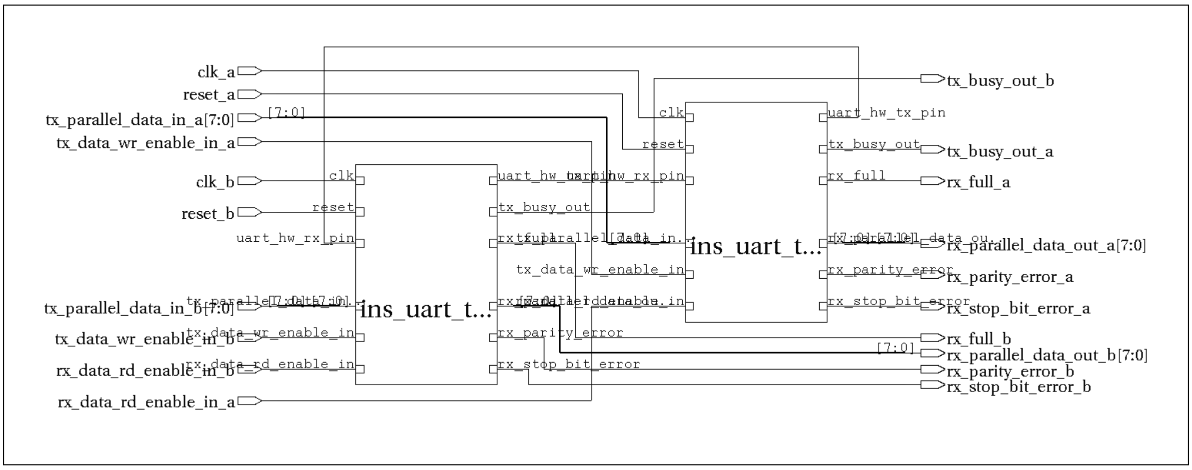


Figure 6: Initial top level diagram of the UART design, before DFT insertion

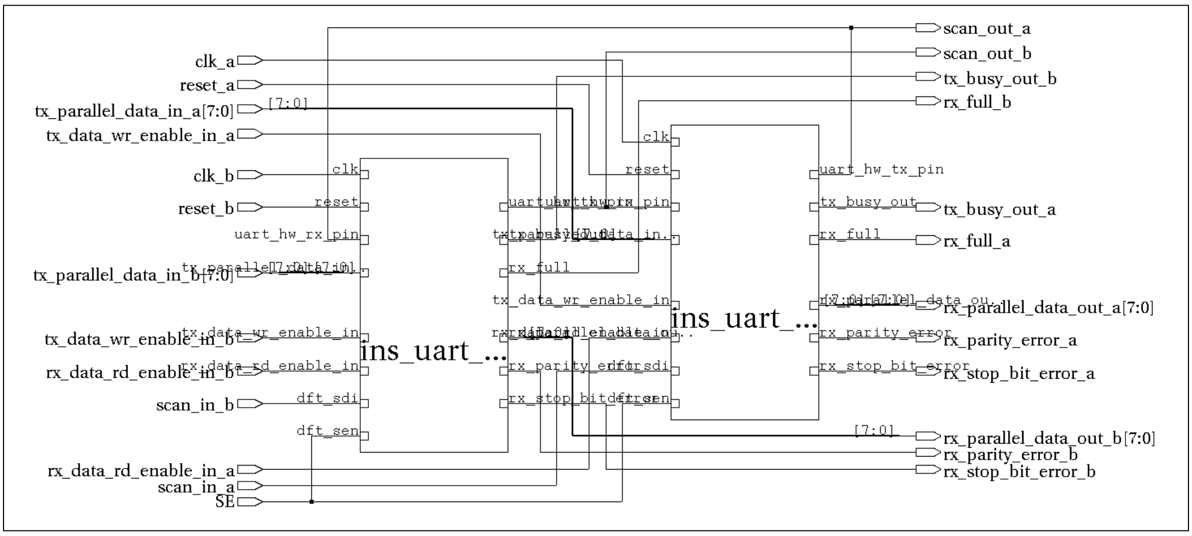


Figure 7: Final top level diagram of the UART design, after DFT insertion

By comparing the Figures 6 and 7, it can be observed that, additional ports have been created to facilitate the Scan Test. Those are `scan_in_a`, `scan_in_b`, `SE`, `scan_out_a` and `scan_out_b`. In total there are two scan chains in the design, each having 82 registers in a chain.

2 Exercise

This section documents the answers to the questions at the end of the practical guide, with screenshots and explanations.

2.1 Comments on the Area

The `area.log` files in the `report` directories, provide 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[1].

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'

Figures 8 and 9 illustrates the log files, before and after the scan synthesis process respectively.

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 8: Area of the UART design, before scan synthesis (Lab 1)

Instance	Module	Cell Count	Cell Area	Net Area	Total Area
uart_top		764	2570.472	987.659	3558.131
ins_uart_transceiver_B	uart_transceiver_CLOCK_IN_MHZ100_TX_WORD_LENGTH8_T	382	1285.236	443.179	1728.415
ins_rx_wrapper	rx_wrapper_NO_OF_WORS_IN_BUFFER1_NO_OF_DATA_BITS8_	191	717.516	214.481	931.997
ins_rx_buffer	rx_buffer_WORD_SIZE8_NO_OF_WORDS1_1	70	336.186	69.809	405.995
ins_rx_fsm	rx_fsm_NO_OF_DATA_BITS8_PARITY_ENABLED32h54525545_	75	214.776	81.136	295.912
ins_sampling_tick_generator	sampling_tick_generator_BAUD115200_CLOCK_IN_MHZ100	46	166.554	52.608	219.162
ins_tx_wrapper	tx_wrapper_NO_OF_WORDS_IN_BUFFER1_NO_OF_DATA_BITS8	191	567.720	228.698	796.418
ins_tx_fsm	tx_fsm_NO_OF_DATA_BITS8_PARITY_ENABLED32h54525545_	96	250.002	108.999	359.001
ins_tx_buffer	tx_buffer_WORD_SIZE8_NO_OF_WORDS1_1	53	162.108	56.580	218.688
ins_sampling_tick_generator	sampling_tick_generator_BAUD115200_CLOCK_IN_MHZ100	42	155.610	50.023	205.633
ins_uart_transceiver_A	uart_transceiver_CLOCK_IN_MHZ100_TX_WORD_LENGTH8_T	382	1285.236	443.179	1728.415
ins_rx_wrapper	rx_wrapper_NO_OF_WORS_IN_BUFFER1_NO_OF_DATA_BITS8_	191	717.516	214.481	931.997
ins_rx_buffer	rx_buffer_WORD_SIZE8_NO_OF_WORDS1	70	336.186	69.809	405.995
ins_rx_fsm	rx_fsm_NO_OF_DATA_BITS8_PARITY_ENABLED32h54525545_	75	214.776	81.136	295.912
ins_sampling_tick_generator	sampling_tick_generator_BAUD115200_CLOCK_IN_MHZ100	46	166.554	52.608	219.162
ins_tx_wrapper	tx_wrapper_NO_OF_WORDS_IN_BUFFER1_NO_OF_DATA_BITS8	191	567.720	228.698	796.418
ins_tx_fsm	tx_fsm_NO_OF_DATA_BITS8_PARITY_ENABLED32h54525545_	96	250.002	108.999	359.001
ins_tx_buffer	tx_buffer_WORD_SIZE8_NO_OF_WORDS1	53	162.108	56.580	218.688
ins_sampling_tick_generator	sampling_tick_generator_BAUD115200_CLOCK_IN_MHZ100	42	155.610	50.023	205.633

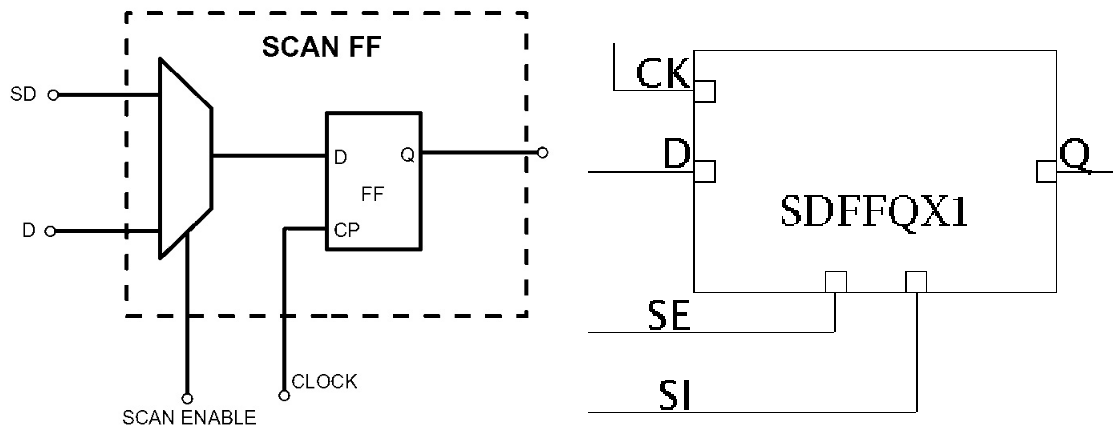
Figure 9: Area of the UART design, after scan synthesis

By comparing the area details shown in the Figures 8 and 9, it can be concluded that after scan synthesis, the area of the design has increased. A summary of area comparison is given in the Table 1.

Table 1
Area comparison of the design before and after scan synthesis

Design	Cell Count	Cell Area	Net Area	Total Area
Before Scan Synthesis (Lab 1)	772	2220.948	946.493	3167.441
After Scan Synthesis	764	2570.472	987.659	3558.131

The reason behind this area increment is the additional logic that is introduced by the Scannable Flip-Flops. they introduce additional multiplexer to each generic D-Flip-Flop. The structure of such Flip-Flop is shown in the Figure 10.



(a) Logical structure of a Scannable Flip-Flops (b) Schematic view of a Scannable Flip-Flops

Figure 10: Logical and schematic view of a Scannable flip-flop

2.2 Comments on the Ports

The `ports.log` files in the `report` directories carry the details related to the ports of the current design. By default, the report gives information on port direction, external delays, exception objects and their types, driver, slew, fanout load, pin capacitance and wire capacitance for the ports[1]. Figures 11 and 12 illustrates a part of the 'External Delays & Exceptions' section of the ports log files, before and after the scan synthesis process respectively.

```
clk_b                      in   clk_b       0.0  no_value  create_clock_delay_domain_1_clk_b_R_0      N/A
                                clk_b  no_value       0.0  create_clock_delay_domain_1_clk_b_F_0
reset_b                    in   N/A         N/A       N/A  N/A                                        N/A
tx_parallel_data_in_b[7]   in   clk_b    6000.0    6000.0  in_del_10_1                                N/A
tx_parallel_data_in_b[6]   in   clk_b    6000.0    6000.0  in_del_11_1                                N/A
tx_parallel_data_in_b[5]   in   clk_b    6000.0    6000.0  in_del_12_1                                N/A
tx_parallel_data_in_b[4]   in   clk_b    6000.0    6000.0  in_del_13_1                                N/A
tx_parallel_data_in_b[3]   in   clk_b    6000.0    6000.0  in_del_14_1                                N/A
tx_parallel_data_in_b[2]   in   clk_b    6000.0    6000.0  in_del_15_1                                N/A
tx_parallel_data_in_b[1]   in   clk_b    6000.0    6000.0  in_del_16_1                                N/A
tx_parallel_data_in_b[0]   in   clk_b    6000.0    6000.0  in_del_17_1                                N/A
tx_data_wr_enable_in_b     in   clk_b    6000.0    6000.0  in_del_18_1                                N/A
rx_data_rd_enable_in_b     in   clk_b    6000.0    6000.0  in_del_19_1                                N/A
tx_busy_out_a              out  clk_a    6000.0    6000.0  ou_del                                     N/A
rx_full_a                  out  clk_b    6000.0    6000.0  ou_del_32_1                                N/A
rx_parallel_data_out_a[7]  out  clk_a    6000.0    6000.0  ou_del_21_1                                N/A
rx_parallel_data_out_a[6]  out  clk_a    6000.0    6000.0  ou_del_22_1                                N/A
rx_parallel_data_out_a[5]  out  clk_a    6000.0    6000.0  ou_del_23_1                                N/A
rx_parallel_data_out_a[4]  out  clk_a    6000.0    6000.0  ou_del_24_1                                N/A
rx_parallel_data_out_a[3]  out  clk_a    6000.0    6000.0  ou_del_25_1                                N/A
rx_parallel_data_out_a[2]  out  clk_a    6000.0    6000.0  ou_del_26_1                                N/A
rx_parallel_data_out_a[1]  out  clk_a    6000.0    6000.0  ou_del_27_1                                N/A
rx_parallel_data_out_a[0]  out  clk_a    6000.0    6000.0  ou_del_28_1                                N/A
rx_parity_error_a          out  clk_a    6000.0    6000.0  ou_del_29_1                                N/A
rx_stop_bit_error_a        out  clk_a    6000.0    6000.0  ou_del_30_1                                N/A
tx_busy_out_b              out  clk_b    6000.0    6000.0  ou_del_31_1                                N/A
rx_full_b                  out  N/A         N/A       N/A  N/A                                        N/A
```

Figure 11: Ports of the UART design, before scan synthesis (Lab 1)

```
clk_b                      in   clk_b       0.0  no_value  create_clock_delay_domain_1_clk_b_R_0      N/A
                                clk_b  no_value       0.0  create_clock_delay_domain_1_clk_b_F_0
reset_b                    in   N/A         N/A       N/A  N/A                                        N/A
tx_parallel_data_in_b[7]   in   clk_b    6000.0    6000.0  in_del_10_1                                N/A
tx_parallel_data_in_b[6]   in   clk_b    6000.0    6000.0  in_del_11_1                                N/A
tx_parallel_data_in_b[5]   in   clk_b    6000.0    6000.0  in_del_12_1                                N/A
tx_parallel_data_in_b[4]   in   clk_b    6000.0    6000.0  in_del_13_1                                N/A
tx_parallel_data_in_b[3]   in   clk_b    6000.0    6000.0  in_del_14_1                                N/A
tx_parallel_data_in_b[2]   in   clk_b    6000.0    6000.0  in_del_15_1                                N/A
tx_parallel_data_in_b[1]   in   clk_b    6000.0    6000.0  in_del_16_1                                N/A
tx_parallel_data_in_b[0]   in   clk_b    6000.0    6000.0  in_del_17_1                                N/A
tx_data_wr_enable_in_b     in   clk_b    6000.0    6000.0  in_del_18_1                                N/A
rx_data_rd_enable_in_b     in   clk_b    6000.0    6000.0  in_del_19_1                                N/A
SE                         in   N/A         N/A       N/A  N/A                                        N/A
tx_busy_out_a              out  clk_a    6000.0    6000.0  ou_del                                     N/A
rx_full_a                  out  clk_b    6000.0    6000.0  ou_del_32_1                                N/A
rx_parallel_data_out_a[7]  out  clk_a    6000.0    6000.0  ou_del_21_1                                N/A
rx_parallel_data_out_a[6]  out  clk_a    6000.0    6000.0  ou_del_22_1                                N/A
rx_parallel_data_out_a[5]  out  clk_a    6000.0    6000.0  ou_del_23_1                                N/A
rx_parallel_data_out_a[4]  out  clk_a    6000.0    6000.0  ou_del_24_1                                N/A
rx_parallel_data_out_a[3]  out  clk_a    6000.0    6000.0  ou_del_25_1                                N/A
rx_parallel_data_out_a[2]  out  clk_a    6000.0    6000.0  ou_del_26_1                                N/A
rx_parallel_data_out_a[1]  out  clk_a    6000.0    6000.0  ou_del_27_1                                N/A
rx_parallel_data_out_a[0]  out  clk_a    6000.0    6000.0  ou_del_28_1                                N/A
rx_parity_error_a          out  clk_a    6000.0    6000.0  ou_del_29_1                                N/A
rx_stop_bit_error_a        out  clk_a    6000.0    6000.0  ou_del_30_1                                N/A
tx_busy_out_b              out  clk_b    6000.0    6000.0  ou_del_31_1                                N/A
rx_full_b                  out  N/A         N/A       N/A  N/A                                        N/A
```

Figure 12: Ports of the UART design, after scan synthesis

It can be observed that after the scan synthesis, number of ports has been increased by one, due to the addition of the `Scan Enable (SE)` port. Before the scan synthesis number of ports was 48, and after the scan synthesis it is 49. In addition to that there is no other change to the ports.

Bibliography

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