EE123 VLSI Design Lab: Experiment 3

Saurav Kumar Pandey
System on chip design
IIT Palakkad
152502019@smail.iitpkd.ac.in
October 2025

***Abstract*—This experiment provides a practical introduction to the Application-Specific Integrated Circuit (ASIC) synthesis workflow. The primary goal is to gain hands-on experience with synthesis tools, specifically focusing on interpreting and analyzing the crucial output reports for area, power, and timing. A key part of the lab involves learning how to influence and control the synthesis results by applying different timing constraints. The process will involve synthesizing Verilog RTL designs, examining the generated gate-level netlists, and understanding the trade-offs that arise when modifying clock period constraints to optimize for performance.**

I. Objective

The objectives of this experiment are as follows:

- To familiarize with ASIC synthesis tools.
- To interpret area, power and timing reports.
- To control the synthesis outputs through timing constraints.

II. Introduction to ASIC Synthesis

In the ASIC design flow, the Verilog Register Transfer Level (RTL) code serves to capture the architecture and functionality of the circuit. The process of ASIC synthesis is what translates this abstract RTL description into a physical gate-level netlist (GLN). This netlist is constructed using a standard cell library, which contains the actual circuit implementations for a specific technology (e.g., 180 nm CMOS).

Because the standard cell library contains real physical data, the synthesis tool can analyze the resulting GLN to estimate critical hardware metrics. These metrics include the design's total area, its power consumption, and its timing performance. A synthesis tool typically requires three main inputs: the RTL code, the technology library file, and a file specifying design constraints (like the target clock period). In return, it generates the gate-level netlist and detailed reports for area, power, and timing.

III. Tools and Library Used

- Tool: Synopsys Design Compiler (DC)
- Technology: 180 nm CMOS library from Semiconductor Laboratory (SCL)
- Simulation Tool: ModelSim / VCS

IV. Experimental Procedure

A. RTL Design

The following Verilog designs were synthesized:

1) 4-bit Fibonacci sequence generator.
2) 4-bit Fibonacci sequence generator implemented using a Ripple Carry Adder (RCA).
3) 32-bit Fibonacci sequence generator.

B. Synthesis Flow

A typical TCL script used inside Design Compiler is shown below:

```
set DESIGN fib4
set CLK_PERIOD 10
set LIB_PATH /home/SCL_PDK/.../tsl18fs120_scl_ss.db

read_file -format verilog ../rtl/${DESIGN}.v
set link_library "* $LIB_PATH"
set target_library $LIB_PATH
create_clock -period $CLK_PERIOD -name clk [
    get_ports clk]
compile_ultra

report_area > ${DESIGN}.area.rpt
report_power > ${DESIGN}.power.rpt
report_timing > ${DESIGN}.timing.rpt
write -format verilog -hierarchy -output ${DESIGN}
    .mapped.v
```

Listing 1. Example DC TCL flow

C. Timing Constraints

A base clock constraint of 10 ns was applied. The tool was directed to meet this constraint while minimizing total area and power. Reports were generated for every synthesized design.

D. Gate-Level Simulation

The synthesized netlists were simulated using ModelSim to confirm functional correctness. The SCL library simulation models were included for accurate gate-level behavior.

V. Synthesis Results

A. Area Report

Table I reports the total cell area for the 4-bit designs.

TABLE I
Area Summary for 4-bit Fibonacci Designs

Design	Total Cell Area (μm^2)
Fibonacci Sequence Generator	2126.08
Fibonacci with Ripple Carry Adder	1859.61

TABLE II
POWER SUMMARY FOR 4-BIT FIBONACCI DESIGNS

Design	Dynamic Power (µW)	Leakage Power (nW)
Fibonacci Sequence Generator	75.87	35.09
Fibonacci with Ripple Carry Adder	99.82	43.45

B. Power Report

C. Timing Analysis

Timing was evaluated under clock periods 10 ns, 15 ns and 20 ns. Slack values are reproduced in Table III.

TABLE III
TIMING SLACK COMPARISON FOR FIBONACCI DESIGNS

Clock Period (ns)	Fibonacci (Behavioral) Slack (ns)	Fibonacci with RCA Slack (ns)
10	5.0	6.0
15	9.7	13.5
20	14.8	16.0

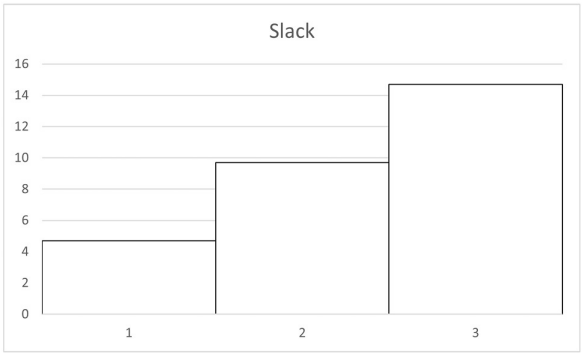

Fig. 1. Timing slack comparison visualization (10 ns clock period).

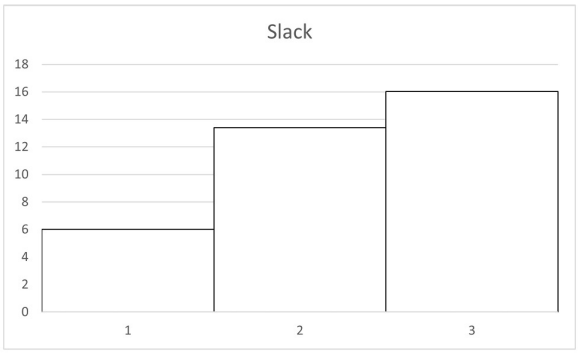

Fig. 2. Timing slack comparison visualization (multiple clock periods).

VI. SIMULATION AND SYNTHESIS VERIFICATION

Functional verification of both the RTL and synthesized netlists was performed using ModelSim. The testbench generated Fibonacci sequence values on every positive clock edge following reset deassertion.

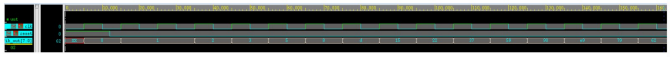

Fig. 3. RTL simulation waveform of the 4-bit Fibonacci with Ripple Carry Adder.

Fig. 4. Gate-level netlist waveform of the 4-bit Fibonacci with Ripple Carry Adder.

VII. 32-BIT FIBONACCI SYNTHESIS RESULTS

A 32-bit generator was synthesized to evaluate scaling effects.

A. Area Report

TABLE IV
AREA SUMMARY FOR 32-BIT FIBONACCI DESIGN

Design	Total Cell Area (μm^2)
32-bit Fibonacci Sequence Generator	3142.18

B. Power Report

TABLE V
POWER SUMMARY FOR 32-BIT FIBONACCI DESIGN

Design	Dynamic Power (µW)	Leakage Power (nW)
32-bit Fibonacci Sequence Generator	124.23	70.32

C. Timing Report

No timing violations were observed for the 32-bit design; it met a 10 ns clock period with positive slack.

TABLE VI
COMPARISON BETWEEN 4-BIT AND 32-BIT FIBONACCI GENERATORS

Design	Area (μm^2)	Dynamic Power (µW)	Leakage (nW)
4-bit Fibonacci (RCA)	1859.61	99.82	43.45
32-bit Fibonacci	3142.18	124.23	70.32

VIII. CONCLUSION

This experiment provided a practical introduction to the ASIC synthesis workflow using Synopsys Design Compiler and an 180 nm SCL library. We successfully synthesized multiple Verilog designs, including 4-bit and 32-bit Fibonacci generators, and learned to interpret the resulting area, power, and timing reports. A key part of the lab involved manipulating timing constraints to observe the direct trade-offs between clock speed, area utilization, and power consumption. Finally, the functionality of the synthesized gate-level netlists was verified using simulation, confirming the correctness of the flow from RTL to GLN. Both 4-bit and 32-bit Fibonacci sequence

generators were successfully synthesized and analyzed using Synopsys Design Compiler. The 4-bit design with the Ripple Carry Adder used less area, while the 32-bit design scaled well and met the 10 ns timing requirement without issues.