ECE 429 Laboratory 9
Standard Cell Based ASIC Design Flow

Announcements:

1. **Teaching Assistants**

- Victoria Heekyung Kim (hkim104@hawk.iit.edu), Office Hours: 10:00 – 11:00 am at Collaborate Ultra on Thursday

2. **Time Frame for Lab-02**

- In-Class students
  - ECE-429-01 (every Wednesday 8:35 - 11:15 a.m.) -11/08/2023
- Internet Students & India International Students
  - See the Collaborate Ultra session recordings on the Blackboard.

Report Due
11:59 PM 11/15/2023. Use the blackboard submission link.

3. **Account Administrator**

If you have account issues or remote access problems, please contact Mr. Upendra Gandhi and Mr. Kamran (support@ece.iit.edu, ugandhi@ece.iit.edu, bkamran@iit.edu).

1. Introduction

In this lab we will try to answer that question that has been plaguing your mind for the whole semester: “Can I create circuit layouts automatically?” The answer is “yes!” The art of changing a circuit description, usually written in a hardware description language into a layout is a complex procedure that involves many steps, which are usually called collectively as a design flow. You will be introduced to one of the most popular design flows, the standard cell-based ASIC design flow, in this lab. We will use tools and libraries from various vendors including OSU standard cell library, Synopsys Design Compiler, Synopsys Formality, Cadence Encounter Digital Implementation System, and Cadence Virtuoso.

2. Standard Cell Based ASIC Design Flow

The standard cell-based ASIC design flow automatically synthesizes a chip layout from a register-transfer-level (RTL) description of the desired chip functionality. In other words, the chip functionality should be specified as a sequential circuit consisting of flip-flops that store the state (register) and combinational circuits that compute outputs and the next state (transfer). The design flow will utilize a standard cell library, which is usually provided by the fab, to synthesize a chip layout according to design constraints including cost, performance, power consumption, etc. While you may treat this design flow as a push-button solution to chip design, in practice the tools seldom generate a satisfactory chip design at the very beginning - it is a must for the designers to understand the various steps of the design flow in order to guide the tools through multiple iterations until a solution is found.

Roughly speaking, the standard cell-based ASIC design flow is consisting of two steps: logic synthesis and physical design. Before we look into the details of these two steps, let's first study what are standard cells.

2.1 Standard Cell Library

A standard cell is a logic gate with a layout designed for a specific fabrication process. For a particular fabrication process, standard cells for different kinds of logic gates or different sizes of a same logic gate are usually grouped into a standard cell library. Cells from the same library are most likely with the same height - it simplifies both the non-overlapping placement of cells and the routing of VDD/GND rails plus the wells during the chip physical design.

Since the layout of a standard cell is known, the characteristics of the cell, e.g. delay and power consumption, can be obtained, e.g. by SPICE simulations, well before the chip design stage. Such data usually accompany the standard cell library so that it is not necessary for the tools to go through the lengthy process to generate them again. Therefore, to fully understand a standard cell library, we need to understand how those characteristics participate various steps of the design flow.

2.2 Logic Synthesis

The purpose of logic synthesis is to transform your RTL description of chip functionality into a netlist consisting of standard cells that will be implemented on your chip.

Nowadays, it is very common to have Datapath/arithmetic operations, e.g. addition, in an RTL description. Therefore, the logic synthesis tool should first implement such operations as Boolean logics. Obviously, there are design efforts to make such implementations possible. In our lab, the logic synthesis tool Synopsys Design Compiler

will utilize the Design Ware IP (intellectual property) library from the file 'dw foundation.sldb' to complete this task.

After synthesizing the RTL description into logic gates, the logic synthesis tool will perform generic logic optimizations to optimize the design without using any information from the standard cell library. This could include usual code optimizing techniques like constant propagation and dead code elimination or more powerful techniques like multi-output multi-level logic minimization.

Finally, the logic synthesis tool will create a netlist consisting of standard cells that has the same Boolean functionality as the one generated by generic logic optimizations. The tool will need a description of the functionality of the standard cells so that it can match them with the logic gates. It is also necessary for the tool to understand the cost (area, delay, power consumption, etc.) of the cells so that it can pick up the best one when there are many choices. In our lab, all such data are stored in the file 'gscl45nm.db', which is part of the standard cell library. Note that the file contains NO layout information of the standard cells as they are not necessary for logic synthesis.

2.3 Physical Design

The purpose of physical design is to create a physical implementation of the netlist consisting of standard cells, i.e. the chip layout. Obviously, since every standard cell will consume the silicon surface to form the transistors, no two standard cells should overlap. Therefore, the most straight forward method of physical design is to simply place the standard cells row by row (since they have the same height!) and then use metal layers to route the wires. However, as interconnect delay becomes critical and it may require more metal layers to route the wires than available, physical design is a very challenging step that may need extensive guidance from designers.

In our lab, we will use Cadence Encounter Digital Implementation System for physical design. The file 'gscl45nm.tlf' will provide timing of the standard cells and the interconnects. The file 'gscl45nm.lef' will provide the physical dimensions of the standard cells including height, width, pin location, etc., and define the routing resources including the layout rules for metal layers and obstacles in the standard cells. Note that the files contain NO exact layout information of the standard cells as they are not necessary for a typical physical design flow.

3. Preparing for the Laboratory (Pre-Lab)

Please familiarize yourself with Tutorial IV: Standard Cell Based ASIC Design Flow before you come to lab, especially for the following topics.

- Verilog code and testbench for the 8-bit accumulator
- Steps to setup logic synthesis and place & route

- Equivalence checking

4. Lab Instructions

Follow Tutorial IV to create the layout of the accumulator and to verify its correctness using equivalence checking.

5. Deliverables

The requirement of the lab/project reports with a template can be found on the Blackboard. *NEVER share your writings/screenshots with others.* We prefer to receive reports electronically as either .pdf or .doc files through the Assignments section on the Blackboard.

Follow the template for lab/project reports posted on the Blackboard to structure your lab report. Briefly describe the tasks you have accomplished. Report the measurements of chip area, critical path delay, and power consumption. At various stages of the design flow. Discuss your lab by addressing at least the following questions.

- What is a standard cell?
- What are the differences among ‘accu.v’, ‘accu.vh’, and ‘final.v’ ?
- How does the area of your design change after place & route?
- How does the timing of your design change after place & route?
- Why do we need to use Virtuoso to generate the final layout for fabrication? What information is available from ‘final.gds2’?

Finally, attach screenshots of the below list:

1. RTL simulation result
2. Logic synthesis result
3. Place and route result
4. Power report Result
5. Area report result
6. Timing report result
7. ESP result
8. Layout printing result

Keep the report format for the lab report, including the introduction, theory/pre-lab, implementation, and conclusion.