Lab Module 9: ASIC Design Flow - Gate-Level Synthesis & First Look at Timing

Course: Digital Design Fundamentals / Introduction to VLSI

Module Title: Building Chips Automatically: From Code to Basic Gates & How Fast They Are

Duration: 4-5 Hours (includes pre-lab learning, using software, and writing your report)

---

1. Lab Goals

After this lab, you'll be able to:

- **Understand Chip Design Steps:** Get a clear picture of how we use computers to automatically design integrated circuits (ASICs), specifically how design code becomes a blueprint of basic gates.
- **Remember Design Languages (HDL):** Quickly recall what languages like Verilog or VHDL are for, and how they describe digital circuits.
- **Do Automatic Design (Synthesis):** Understand the steps involved in "synthesis," which is the process of converting your design code into a list of basic gates. You'll either do this in special software or learn how it's done.
- **Read Gate Blueprints (Netlist):** Look at the final "gate-level netlist" – a list of all the basic gates and how they're connected – and understand what it's telling you.
- **Understand Basic Timing Checks (STA):** Learn the main ideas behind "Static Timing Analysis" (STA), like finding the slowest path in your circuit and what "setup" and "hold" issues mean.
- **Read Simple Timing Reports:** Understand what the important numbers in a basic timing report tell you about how fast your circuit can run.

---

2. Before You Start (Pre-Lab Prep)

To get the most out of this lab, do these things first:

- **Review Your Notes:** Go over your class notes about:
    - **ASIC Design Steps:** The big picture of how chips are designed, from ideas to actual silicon.
    - **Hardware Description Languages (HDL):** How we write code to describe digital circuits (like Verilog or VHDL). Think about how you describe combinational logic (like an AND gate) and sequential logic (like a flip-flop).
    - **Standard Cells:** What these are (like pre-designed basic gates: AND, OR, flip-flops). Imagine them as LEGO bricks a computer uses to build your chip.
    - **Logic Synthesis:** How a computer program (the "synthesis tool") takes your code and picks the right LEGO bricks from a library to build your circuit.
    - **Basic Circuit Timing:** What a clock is, how flip-flops need data to be ready before the clock (setup time), and stay stable after the clock (hold time). Also, what "propagation delay" means for a gate.

- **Look at Example Code:** Your teacher might give you some simple Verilog or VHDL code for a circuit like an adder or a counter. Look at it to see how circuits are described.
- **Understand Inputs/Outputs:** Know what information goes *into* the synthesis software (your code, timing rules) and what comes *out* (the gate blueprint, performance reports).
- **High-Level Timing Idea:** Read a bit about why we use STA instead of just running simulations to check timing on big chips. Think about the idea of the "critical path" – the slowest path in your circuit.

---

3. What You'll Need (Tools & Materials)

- **Computer:** A good computer that can handle design software.
- **Chip Design Software (Synthesis Tool):**
    - **Professional Tools (Best, if available):** Software like Synopsys Design Compiler or Cadence Genus. These are used in real companies. *Often, only universities have licenses for these.*
    - **Free/Open-Source Tools (Good alternative):** Programs like Yosys (for synthesis) paired with a library of basic gates (like OSU_STDCELL or sky130_fd_sc_hd). This gives you a taste of the real process.
    - **"Learn by Looking" Option (If no software):** If you can't use the special software, this lab will be more about understanding pre-made results. Your teacher will give you the gate blueprints and timing reports, and you'll focus on learning what they mean. Your teacher will tell you which option you'll use.
- **Code Editor:** Any simple text editor (like Notepad++, VS Code) to write and view your design code.
- **Standard Cell Library Files:** Special files (`.lib` or `.db`) that describe all the basic gates your chosen technology has, including how fast they are. Your teacher will provide these.
- **Spreadsheet Program:** Like Microsoft Excel or Google Sheets, for organizing data and making graphs.

---

4. Lab Steps & Experiments

Important Tip: Always **save your work** as you go! Make sure your graphs and pictures are **clearly labeled** with titles and what the lines mean.

Experiment 1: Understanding Your Design Code (RTL)

1. **Goal:** See how a digital circuit is written in a Hardware Description Language (HDL) for the computer to understand.
2. **Steps:**
    - **Get the Code:** Your teacher will give you a Verilog (or VHDL) code file for a simple digital circuit. It might be something like:
        - A simple 4-bit addition circuit.
        - A basic counting circuit.

        - A simple "state machine" (a circuit that remembers a few different modes).
    - **Look at the Code:** Open this code file in your text editor.
        - Find where the main circuit block starts (like `module` in Verilog).
        - See where the inputs, outputs, and internal signals are listed.
        - If it's a calculating circuit, find how it describes adding or other operations.
        - If it's a memory circuit (like a counter), find how it uses clock signals to store information.
    - **Think About It:** Imagine how this written description will turn into actual basic gates.
    - **What to Write in Your Report:**
        - Briefly explain what the provided code does (e.g., "This code describes a 4-bit counter.").
        - Explain why this code is "synthesizable" – meaning a computer can automatically turn it into physical gates.

Experiment 2: The "Synthesis" Step - Turning Code into Gates

1. **Goal:** Learn how special software automatically converts your high-level design code into a detailed blueprint of basic gates.
2. **Steps:**
    - **How it Works (If no software access):** Your teacher will explain the conceptual steps of how the "synthesis tool" works:
        - **Reads Your Code:** It takes your Verilog/VHDL design.
        - **Applies Rules:** You give it rules, like "this circuit needs to run at 100 MHz" or "keep the circuit small."
        - **Loads Gate Library:** It looks at a special library file that lists all the available basic gates (AND, OR, flip-flops, etc.) and their characteristics (how fast they are, how much power they use).
        - **Optimizes & Maps:** It figures out the best way to build your circuit using these gates, making it as fast or small as possible based on your rules.
        - **Creates Gate Blueprint:** It generates a new file, the "gate-level netlist," which is the detailed list of gates and their connections.
    - **Using the Software (If you have access):**
        - **Start the Program:** Open your synthesis software (e.g., Synopsys Design Compiler).
        - **Load Gates:** Tell the program where your standard cell library files are.
        - **Load Your Design:** Load your Verilog/VHDL code into the program.
        - **Set Rules:** Tell the program about your clock speed and any other timing requirements. Your teacher will give you specific commands for this.
        - **Run Synthesis:** Give the command to "compile" or "synthesize" your design. This is where the magic happens!

        - **Save the Blueprint:** Save the new file that contains the gate-level netlist (usually a `.v` or `.vg` file).
    - **Initial Summary:** Look for a summary report from the synthesis tool. It usually tells you how many gates were used and the estimated size (area) of your circuit.
    - **What to Write in Your Report:**
        - Describe the main steps involved in synthesis, either from your experience with the software or from your teacher's explanation.
        - Write down the total number of basic gates and the estimated area that the tool reported for your circuit.

Experiment 3: Looking at the Gate Blueprint (Netlist)

1. **Goal:** Learn to read and understand the detailed list of basic gates that your design code became.
2. **Steps:**
    - **Open the File:** Open the newly generated gate-level netlist file in your text editor.
    - **Examine the Structure:**
        - Find the main block of your circuit in this file.
        - You'll see lines that represent individual basic gates (like `INV` for inverter, `NAND2X1` for a 2-input NAND gate). Each line will be an "instance" of a gate from the library.
        - **Gate Instances:** Notice the name of the gate (e.g., `INV`), a unique name for that specific copy of the gate (e.g., `U1`), and then connections to its inputs and outputs.
        - See how the original signals from your design code (inputs, outputs) are now connected to the inputs and outputs of these basic gates.
        - If your design had memory (flip-flops), find those instances (e.g., `DFF_X1`).
    - **Match Code to Gates:** Try to mentally follow a simple path in this gate list and connect it back to a line or a block of logic in your original design code. For example, if you designed an adder, see if you can spot the gates that make up one of the adder stages.
    - **What to Write in Your Report:**
        - Include a small piece (like 5-10 lines) of your gate-level netlist in your report.
        - Explain what each part of that snippet means (the gate type, its unique name, and what signals are connected to its inputs and outputs).
        - Discuss how this gate-level netlist is different from your original design code (e.g., it's a detailed list of physical parts, not just a description of behavior).

Experiment 4: First Look at Static Timing Analysis (STA) - The Big Picture

1. **Goal:** Understand the main ideas behind STA, which is how we automatically check if our chip will run fast enough.
2. **Steps (Mostly concepts, using examples from your teacher):**
    - **Why STA is Needed:** Think about why just simulating your circuit is not enough for huge chips (it's too slow to test every possible way data can move). STA mathematically checks all paths, which is much faster.
    - **Circuit Paths:** Understand the different ways data can travel through your circuit:
        - **From Input to Flip-Flop:** Data comes from outside the chip to a flip-flop.
        - **From Flip-Flop to Flip-Flop:** Data moves from one flip-flop to another.
        - **From Flip-Flop to Output:** Data goes from a flip-flop out of the chip.
        - **From Input to Output:** Data goes directly from an input to an output, only through basic gates (no flip-flops).
    - **Clock Speed Rule:** The most important rule is the clock period (Tclk) – how fast your clock "ticks."
    - **Setup Time Explained:**
        - **What it means:** Data needs to be stable and "set up" at the input of a flip-flop *before* the clock edge arrives.
        - **Problem:** If data arrives too late, it's a "setup violation." The flip-flop might not store the correct value.
        - **The Check:** The time it takes for data to arrive must be less than the clock period minus the setup time needed by the flip-flop.
    - **Hold Time Explained:**
        - **What it means:** Data needs to "hold" stable at the input of a flip-flop *after* the clock edge arrives.
        - **Problem:** If data changes too quickly (arrives too early), it's a "hold violation." The flip-flop might also get confused.
        - **The Check:** The time it takes for data to arrive must be *greater* than the hold time needed by the flip-flop.
    - **The Slowest Path (Critical Path):**
        - STA finds the "longest" or slowest path in your circuit. This path determines the absolute fastest clock speed your circuit can handle. This is called the **critical path** (for setup time).
        - STA also finds the "shortest" or fastest path, which is important for hold time.
    - **"Slack" (Room to Breathe):** Slack is simply the difference between when data *needs* to be there and when it *actually* gets there.
        - **Positive Slack:** Good! Your timing rules are met, you have extra time.
        - **Negative Slack:** Bad! Your timing rules are broken, data is either too late or too early.
    - **What to Write in Your Report:**
        - In your own words, explain what "setup time" and "hold time" are for a flip-flop.
        - Describe what a "critical path" is and why finding it is so important for chip performance.

        - Explain "slack" – what does positive slack mean, and what does negative slack mean?

Experiment 5: Reading a Basic Timing Report from STA

1. **Goal:** Learn how to look at a report from an STA tool and find the important numbers about circuit speed.
2. **Steps (Mostly by looking at reports provided by your teacher):**
    - **Get the Report:** Your teacher will give you a simplified STA report (or parts of one) for your synthesized design.
    - **Scan the Report:** Look for sections like:
        - **Design Info:** Your circuit's name, the basic gate library used.
        - **Clock Info:** Details about your clock signal.
        - **Summary:** A quick overview of the worst timing issues (the biggest negative slack).
        - **Detailed Path Reports:** This is the most important part! It shows you step-by-step information for specific paths.
    - **Focus on a Critical Path:** Find a detailed report for a "critical path" (the one with the worst slack).
        - **Starting Point:** Where the data path begins (e.g., an input pin or a flip-flop's output).
        - **Ending Point:** Where the data path ends (e.g., a flip-flop's input or an output pin).
        - **Clock Delay:** How long it takes the clock signal to reach the endpoint's clock pin.
        - **Data Delay:** How long it takes the data to travel through all the gates and wires from the starting point to the ending point.
        - **Time Needed (Required Time):** The latest the data *should* arrive at the endpoint to meet the timing rule.
        - **Time Arrived (Arrival Time):** The actual time the data *does* arrive at the endpoint.
        - **Slack:** The difference between "Time Needed" and "Time Arrived."
        - **Gate List:** See the list of individual gates and wires along this critical path, and their individual delays.
    - **What to Write in Your Report:**
        - Briefly describe the main parts of an STA report.
        - Show a small piece (or draw a simple diagram) of a critical path you found in the report.
        - For *your* critical path, clearly list its: Starting Point, Ending Point, Clock Delay, Data Delay, Time Needed, Time Arrived, and Slack.
        - Tell us if the timing rule (setup) was met (positive slack) or broken (negative slack). Explain what that means for your circuit's speed.
        - Explain how looking at this report helps chip designers find and fix slow parts of their circuit.

---

6. Lab Report Guidelines

Your lab report should be a clear, easy-to-understand, and professional document that shows what you learned about automated chip design and basic timing checks.

Report Structure:

1. **Title Page:**
    - Lab Module Name and Number
    - Your Name, Student ID
    - Course Name, Date
    - Teacher's Name
2. **1. Lab Goals:** (Copy this directly from Section 1 above).
3. **2. What I Did Before Lab:** Briefly describe how you prepared, like reviewing notes and looking at code examples. Mention the specific circuit code your teacher gave you (e.g., "I worked with a Verilog code for a 4-bit adder.").
4. **3. Tools Used:**
    - Name the software you used for synthesis (or state "I analyzed pre-generated files").
    - Mention if you used Verilog or VHDL.
    - Name the standard cell library used.
5. **4. Steps and Results:**
    - For each Experiment (1 through 5), create a separate section.
    - **Inside each section:**
        - **Purpose:** Briefly state the goal of that experiment in your own words.
        - **Setup:** Briefly explain how you set up the experiment (e.g., "I used the counter code and a 50MHz clock rule.").
        - **Results:**
            - **Experiment 1 (Code):** Include the provided HDL code (or a small, important part if it's very long).
            - **Experiment 2 (Synthesis):** State the number of basic gates used and the estimated size of your circuit.
            - **Experiment 3 (Netlist):** Show a small example (5-10 lines) from your gate-level netlist.
            - **Experiment 5 (Timing):** Include a simplified example of the critical path timing report.
        - **What I Learned/Why it Matters:**
            - **Experiment 1:** Explain what your code does and why the computer can turn it into hardware.
            - **Experiment 2:** Describe how the computer program takes your code and "builds" the circuit using basic gates.
            - **Experiment 3:** Explain what each part of the gate blueprint means and how it's different from your original code.
            - **Experiment 4 (Concepts):** Explain in your own words what "setup time," "hold time," and the "critical path" are. Explain what "slack" tells you about your circuit's speed.
            - **Experiment 5:** For your critical path example, explain what the numbers for "Time Needed," "Time Arrived," and "Slack" mean.

Tell us if your circuit met the timing rules or if it was too slow/fast. Explain how this report helps designers find problems.

6. **5. Conclusion:**
    - Summarize the main things you learned about automated chip design and checking circuit speed.
    - Confirm if you achieved all the lab goals.
    - Talk about any difficulties you faced during the lab and how you figured them out.
    - Suggest what you'd like to learn next about chip design (e.g., how to fix timing problems, how much power the gates use, how they're physically placed on the chip).

Making Your Report Clear:

- **Simple Language:** Use easy words. Don't use fancy terms unless you really need to, and explain them.
- **Be Professional:** Write clearly and correctly.
- **Number Everything:** All your pictures (code, graphs) and tables should have numbers and a short description.
- **Refer to Figures:** Always mention your pictures and tables in your writing (e.g., "As Figure 3 shows...").
- **Use Correct Units:** Always include units like nanoseconds (ns) for time, micrometers (µm) for size, etc.
- **Check Spelling/Grammar:** Read your report carefully to catch any mistakes.
