
Lab Module 8: Introduction to Sequential Logic: CMOS D-Latch/Flip-Flop Schematic and Simulation

Discipline: Digital VLSI Design

Module Title: Introduction to Sequential Logic: CMOS D-Latch/Flip-Flop Schematic and Simulation

Duration: 3.5 Hours (or two 105-minute sessions)

---

1. Aim:

The main goal of this lab is to help you understand and build basic memory circuits, which are super important for digital systems. You'll learn how to draw and test a CMOS D-Latch or a simple D-Flip-Flop. You'll also measure how quickly they respond and learn about special timing rules like setup time and hold time. We'll also touch upon a tricky issue called "metastability."

---

2. Theory:

Imagine your phone or computer. It doesn't just react to what you're doing right now; it also remembers what you did a moment ago. That's the power of **sequential logic circuits**. Unlike simpler "combinational" circuits (like the inverter from Lab 2), sequential circuits have memory. Their output depends on both what's coming in *now* and what they've *remembered* from the past. This memory is stored in special components called **latches** or **flip-flops**. All these memory parts usually work together, keeping time with a **clock signal**.

Latches vs. Flip-Flops: How They Listen to the Clock:

Think of a gate that opens and closes based on a signal.

- **Latches:** Are like a gate that stays open as long as the clock signal is at a certain level (e.g., high). While the gate is open, anything at the input immediately passes to the output. They are "transparent."
- **Flip-Flops:** Are smarter. They only "listen" and change their output at a very specific moment – a sudden change (an "edge") of the clock signal (e.g., when the clock goes from low to high, a "rising edge"). This "edge-triggering" makes them more predictable and is key for building reliable digital systems.

Building a CMOS D-Latch/Flip-Flop:

A basic D-Latch can be made using simple electronic switches called transmission gates (or "pass-transistor logic") along with our familiar inverters. When the clock "opens the gate," data flows through. When the clock "closes the gate," the latch holds the last piece of data it saw.

A **D-Flip-Flop** is usually built by connecting two D-Latches in a special way, called a **Master-Slave configuration**.

- The first latch (the "Master") captures the input data when the clock is active (e.g., clock is high).
- The second latch (the "Slave") then takes the data from the Master when the clock switches to its opposite state (e.g., clock goes low).
- This two-stage setup ensures that the flip-flop only changes its main output (Q) exactly when the clock signal makes a specific "edge" transition. It's like a person taking a snapshot of data at the precise click of a camera.

Key Timing Rules for Memory Circuits:

To make sure your memory circuits work perfectly in a fast system, you need to understand these critical timing rules:

- **Clock-to-Output Delay (t_CQ):** This is the time it takes for the flip-flop's output (Q) to change *after* the clock signal's active edge arrives. It's like the time from pressing a button to when a light turns on. A smaller t_CQ means a faster circuit.
- **Setup Time (t_setup):** Imagine a student rushing to get their work done *before* a deadline. Setup time is the *minimum time* that the data at the input (D) must be stable and ready *before* the active clock edge arrives. If the data changes too close to the clock edge, the flip-flop might get confused and capture the wrong value.
- **Hold Time (t_hold):** Now imagine a student needing to keep their work stable *after* the deadline, until it's collected. Hold time is the *minimum time* that the data at the input (D) must remain stable *after* the active clock edge has passed. If the data changes too soon after the clock edge, the flip-flop might accidentally let go of the value it just captured.
- **Metastability:** This is a tricky problem. If you violate setup time or hold time (meaning data changes exactly when the clock edge arrives), the flip-flop can get into a confused, undecided state. It's like a coin landing on its edge – not heads, not tails. It might stay in this "in-between" state for an unpredictable amount of time before finally deciding to be a '0' or '1'. If it takes too long to decide, your whole system could fail.

---

3. Pre-lab Questions:

Students must answer these questions before beginning the lab session.

1. How is a circuit with memory (sequential) different from a circuit without memory (combinational)? Give an example of each.
2. Explain simply: What's the main difference in how a D-Latch and a D-Flip-Flop react to the clock signal? Why do we usually prefer D-Flip-Flops in digital systems?
3. Draw a simple diagram showing how you'd connect two D-Latches to make a D-Flip-Flop. Show the clock (CLK) and its inverted version (CLK_N).

4. Define t_setup, t_hold, and t_CQ. Why are these timing numbers so important for making sure memory circuits work correctly?
5. What does it mean for a flip-flop to enter a "metastable" state? When is this likely to happen?
6. If a flip-flop takes 100 picoseconds (t_CQ) to show its output after the clock edge, and your clock "ticks" every 1 nanosecond (1000 picoseconds), how much time is left for other logic circuits to do their work after this flip-flop?
7. Why is it important for the data going into a flip-flop to stay steady both *before* and *after* the clock signal changes?

---

4. Procedure:

Part A: Drawing the CMOS D-Latch/Flip-Flop Circuit

(Your instructor will tell you whether to design a D-Latch or a D-Flip-Flop for this lab. If you're up for a challenge, you can try both! The D-Flip-Flop gives you more to learn about timing.)

1. **Start Your Simulation Software:** Open the program you use for drawing and testing circuits (like Cadence Virtuoso, LTSpice, etc.).
2. **Make a New Project:** Create a new folder or file for this lab's designs.
3. **Basic D-Latch (Easier Option):**
    - **Parts:** You'll use our familiar nMOS and pMOS transistors, and also inverters (you can draw them using nMOS/pMOS, or use ready-made inverter blocks if your software has them). You might also use "transmission gates," which are good electronic switches.
    - **How to Build:** Draw a basic D-Latch. A common way is to use two inverters connected in a loop to remember the data, and then add switches (transmission gates or single transistors) at the input. These switches are controlled by the clock signal (CLK) and its opposite (CLK_N). When CLK is "on," data goes through; when CLK is "off," the latch holds its value.
    - **Names:** Label the data input as D, the clock input as CLK, and the data output as Q.
    - **Initial Sizes:** Give your transistors reasonable sizes (W/L ratios). For example, if your technology is 0.18µm, use nMOS: W=1µm, L=0.18µm; pMOS: W=2µm, L=0.18µm for inverters.
4. **Master-Slave D-Flip-Flop (More Detailed Option):**
    - **Parts:** You'll use nMOS, pMOS, and inverters. This flip-flop is built by connecting two D-Latches together.
    - **How to Build:**
        - Place two D-Latch circuits (either from the previous step, or draw them directly inside your flip-flop schematic).
        - Connect the output of the first latch (called the "Master") to the input of the second latch (called the "Slave").
        - Connect your main clock signal (CLK) to the control input of the Master latch.
        - Create an inverted clock signal (CLK_N) using an inverter, and connect CLK_N to the control input of the Slave latch.

        - Label your overall inputs D (data) and CLK (clock), and your overall output Q. You can add an inverted output Q_N if you want.
    - **Initial Sizes:** Keep the transistor sizes consistent for good performance.
5. **Create a Symbol:** Once your circuit is drawn and checked for errors, create a block symbol for your D-Latch/Flip-Flop. This makes it easier to use in other circuits.

Part B: Testing How It Works (Functionality) and Measuring Clock-to-Output Delay

1. **Make a New Testbench:** Create a new circuit drawing specifically for testing your D-Latch/Flip-Flop.
2. **Place Your Memory Block:** Drag the symbol of your D-Latch/Flip-Flip into this testbench.
3. **Add Input Signals:**
    - **Clock (CLK):** Add a pulsating voltage source for the clock. Set its period (how often it "ticks," e.g., 10ns), its duty cycle (how long it's high vs. low, usually 50%), and how fast it rises and falls (e.g., 100ps rise/fall time).
    - **Data (D):** Add another pulsating voltage source for the data input. Make sure this data changes at specific times compared to the clock. For a first test, make D change about 2-3 nanoseconds *after* the clock's active edge. This ensures the data is stable when the clock samples it.
4. **Set Up the Simulation:**
    - Choose a "Transient Analysis" simulation.
    - **Stop Time:** Make it long enough to see several clock cycles and data changes (e.g., 50ns or 100ns).
    - **Time Step:** Set it small enough (e.g., 10ps or 100ps maximum step) to see quick changes accurately.
5. **Run Simulation:** Start the simulation.
6. **Look at the Graphs:** Display the waveforms for CLK, D, and Q on the graph viewer.
7. **Check if It Works Correctly:**
    - **For a D-Latch:** See if the Q output follows the D input when CLK is "on" (active), and if Q holds its value when CLK is "off" (inactive).
    - **For a D-Flip-Flop:** Watch closely! Does Q only change *exactly* when the active clock edge (e.g., rising edge) happens? Does it capture the value of D at that precise moment? This confirms it's edge-triggered.
8. **Measure Clock-to-Output Delay (t_CQ):**
    - Find a spot on the graph where CLK makes its active edge, and Q then changes.
    - Use the measurement tools on your graph (like cursors) to find the time difference from the 50% point of the active CLK edge to the 50% point of the corresponding Q output change.
    - Measure this for both Q going high (t_CQ_LH) and Q going low (t_CQ_HL).
    - Calculate the average t_CQ (add them up and divide by 2).

Part C: Exploring Setup Time (t_setup) and Hold Time (t_hold)

1. **Experimenting with Setup Time:**
    - Go back to your testbench and change the D input signal.

    - **The idea is to make D change closer and closer to the active clock edge.**
    - Start by making D change *very early* before the clock edge (e.g., 5ns before).
    - Then, step by step, move D's change time closer to the clock edge (e.g., 2ns, then 1ns, then 500ps, then 200ps, then 100ps *before* the clock edge).
    - **Your Goal:** Find the *smallest amount of time* (t_setup) that D needs to be stable *before* the active clock edge for Q to capture the data correctly. You'll know you've found it when Q starts to give a wrong answer or takes too long to respond.
    - Run the simulation for each change and carefully observe Q. Write down your findings.
2. **Experimenting with Hold Time:**
    - Go back to your testbench and change the D input signal again.
    - **This time, make D change closer and closer *after* the active clock edge.**
    - Start with D changing *very late* after the clock edge (e.g., 5ns after).
    - Then, step by step, move D's change time closer to the clock edge (e.g., 2ns, 1ns, 500ps, 200ps, 100ps *after* the clock edge). (Sometimes, hold time can even be "negative," meaning D can change a tiny bit *before* the clock without problems – if you see that, it's normal!).
    - **Your Goal:** Find the *smallest amount of time* (t_hold) that D needs to stay stable *after* the active clock edge for Q to hold the data it just captured. You'll know you've found it when Q shows a wrong value or a weird spike.
    - Run the simulation for each change and carefully observe Q. Write down your findings.

Part D: Trying to See Metastability (This can be tricky to simulate!)

1. **Triggering the Confusion:**
    - Try to make D change at the *exact same time* as the active clock edge, or within a few tiny picoseconds. This is hard to do precisely.
2. **What to Look For:**
    - Run the simulation.
    - Look for the Q output to go to a voltage level that's *neither a clear '0' nor a clear '1'* (like half VDD). It might stay at this confusing level for a while before eventually settling. This "stuck in the middle" state is metastability.
    - If you see it, take screenshots! If not, don't worry too much; it's often hard to make it happen reliably in simulations.

---

5. Observation/Results:

Record all your measurements, explanations, and screenshots clearly.

1. **Your D-Latch/Flip-Flop Circuit:**
    - Paste a clear picture of the circuit you drew (your schematic). Make sure all the inputs (D, CLK) and outputs (Q) are labeled.
2. **Proof It Works! (Waveforms):**

    - Include a screenshot of your simulation graph (CLK, D, Q) that clearly shows your circuit doing its job correctly (e.g., for a flip-flop, Q changes only on the rising clock edge, capturing the D value).
    - Add notes on the graph to point out how the clock edge leads to the output change.
3. **Clock-to-Output Delay (t_CQ) Results:**

Delay Measurement	Value (in ps or ns)
t_CQ_LH (Low-to-High)	
t_CQ_HL (High-to-Low)	
t_CQ (Average)	

4. **Setup Time (t_setup) Results:**
    - Briefly describe how you found this number (e.g., "I started with D changing 1ns before CLK, then moved it closer...").
    - Write down the specific t_setup value you measured (the smallest time D had to be stable *before* the clock).
    - Show two screenshots: one where D changes just fine (t_setup met) and the output is correct, and another where D changes too close (t_setup violated) and the output is wrong or delayed.
5. **Hold Time (t_hold) Results:**
    - Briefly describe how you found this number.
    - Write down the specific t_hold value you measured (the smallest time D had to be stable *after* the clock).
    - Show two screenshots: one where D changes just fine (t_hold met) and the output is correct, and another where D changes too close (t_hold violated) and the output is wrong or has glitches.
6. **Metastability Observation (If you saw it):**
    - If you successfully made your Q output go into a "confused" state, include a screenshot of that graph.
    - Explain what specific timing of D and CLK made this happen. If you couldn't see it clearly, just mention that it's hard to simulate precisely.

---

6. Analysis and Discussion:

Write a clear explanation of what you observed and why it matters.

1. **How Your Memory Circuit Works:**
   - Explain, based on your simulation, how your D-Latch or D-Flip-Flop successfully "remembers" data.
   - If you built a D-Flip-Flop, explain how its two-latch (Master-Slave) setup makes it respond only to the clock's *edge*, not its level.
2. **Understanding Clock-to-Output Delay:**
   - Discuss the t_CQ values you measured. Were the times for Q going high and Q going low similar? If not, why might one be faster or slower than the other (think about the strength of nMOS vs. pMOS transistors)?
   - Explain why knowing t_CQ is important for deciding how fast you can make your whole digital system run.
3. **The Importance of Setup and Hold Times:**
   - Look at your setup time experiment results. Why does Q get confused if D changes too late (violating setup time)? (Hint: The flip-flop needs time to prepare to capture the data).
   - Look at your hold time experiment results. Why does Q get confused if D changes too early (violating hold time)? (Hint: The flip-flop needs time to securely lock in the captured data).
   - Imagine a big digital chip. What could go wrong if different parts of the chip have signals that violate these setup or hold times?
4. **Discussing Metastability:**
   - If you saw metastability, describe what the Q waveform looked like when it was "confused." What caused it to enter this unstable state?
   - If you didn't see it, explain why it's so difficult to get a simulator to show it consistently. Even if it's hard to simulate, why do chip designers still worry so much about metastability in real chips?
   - Think about ways designers can try to *avoid* metastability in their designs (e.g., making sure signals don't change at bad times).
5. **Why Sequential Logic is Key:** In your own words, summarize why learning about these memory circuits and their timing rules is so important for designing reliable and fast digital electronics, like those in your phone or computer.

---

7. Post-lab Questions:

Answer these questions after completing the lab and analyzing your results.

1. Imagine a digital system needs to work at a speed where the clock "ticks" every 1 nanosecond (1 ns). If your D-Flip-Flop has a t_CQ of 80 picoseconds (ps), a t_setup of 70 ps, and a t_hold of 20 ps, how much time is left for all the calculation (combinational logic) to happen *between* two flip-flops in that clock period?
2. What is a "race condition" in digital circuits? How are setup and hold time problems related to this?
3. Some modern, very fast flip-flops have very tiny, or even "negative," hold times. What's the benefit of having a negative hold time?
4. If you increase the power supply voltage (VDD) to your CMOS D-Flip-Flop, how would you expect its t_CQ and t_setup/t_hold values to generally change? Why?

5. If your flip-flop is too slow (its t_CQ is too high), how could you change the sizes of the transistors (W/L ratios) in its design to make it faster? What might be the downsides of doing this?
6. Sometimes, data needs to move between different parts of a chip that are running on different clock signals (e.g., a fast part and a slow part). Why is this a common place for metastability to become a problem, and what special circuits do engineers use to prevent it?