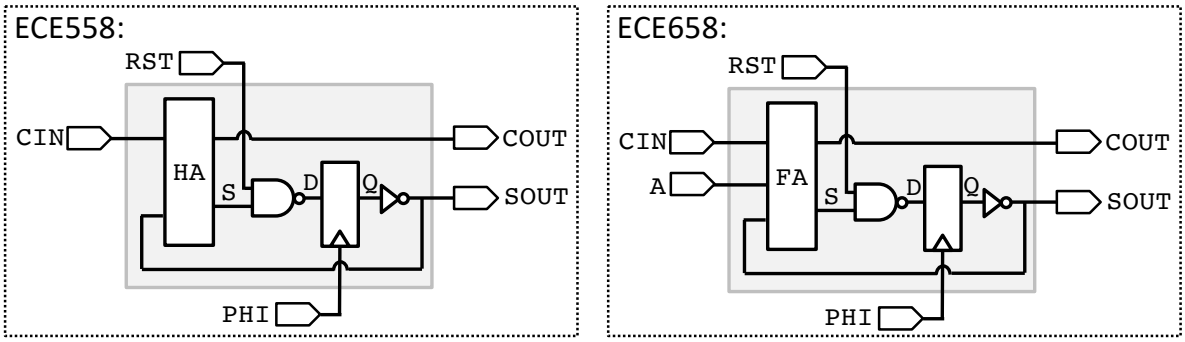
ECE 558/658 VLSI Design -- Fall 2021
Lab 2: Design of a 1-Bit Accumulator

I. General Instructions:

In this lab you will use the same tools from Lab 1 to design an accumulator bitslice. Bitslices are circuits that can be instantiated multiple times and connected together by abutment to produce a circuit that operates on multi-bit words. The bitslice accumulator you design can later be instantiated multiple times to form an n-bit accumulator. This circuit has many more transistors than the design from Lab 1, and it is a sequential circuit so you will need to deal with a clock. Your lab report should be submitted on Moodle as a pdf file with your name at the top and an indication of whether you are enrolled in ECE558 or ECE658. Some sections of the lab, as marked, are only required for students enrolled in ECE658. When you have questions on lab, you can see the TAs during office hours for help.

Students in ECE558 will create an accumulator bitslice using a half adder; n instances of this bitslice form a circuit that accumulates a sequence of 1-bit inputs into an n-bit sum. Students in ECE658 will create an accumulator bitslice using a full adder; n instances this bitslice form a circuit that accumulates a sequence of n-bit inputs into an n-bit sum.



II. Technical Background Information:

An accumulator consists of an adder and a resettable flip-flop. Its inputs are the clock PHI, an active-low reset RST, CIN, and (for ECE658) A. Its outputs are SOUT and COUT. The adder computes from its inputs a sum S and the carry COUT. The flip-flop, on rising edge of PHI, stores the value from its input D, and updates SOUT. Note that the NAND gate before the flip-flop and the inverter after the flip-flop cause offsetting inversions, so that S and SOUT have the same polarity.

The **flip-flop** must be a C^2MOS design based on the schematic in Figure 10.20c on page 394 of Weste & Harris:

- The flip-flop must sample its input on the RISING edge of PHI.
- The flip-flop should be synchronously reset by an active-low reset signal (i.e. RST=0 makes SOUT=0 on the next rising edge of PHI). This is accomplished by the NAND gate in the schematic above.
- The C^2MOS flip-flop from Fig. 10.20c has an inverter driving its output. This inverter is the one shown in the schematic above. The polarity of SOUT is therefore inverted relative to D and Q, but consistent with S.

The **full adder** (for ECE658 students) must be implemented in static CMOS according to the schematic in Figure 11.3 on page 431 of the textbook. You will need to create inverted versions of the inputs as indicated in Fig. 11.3.

The **half adder** (for ECE558 students) must be implemented in static CMOS. A half adder comprises an XOR gate to compute sum S, and a 2-input NOR gate (operating on inverted inputs) to compute COUT. You will need to create inverted inputs with inverters. You should use the schematic from Fig. 11.59d for the XOR gate and implement the NOR gate in the usual way.

Layout Requirements:

- The transistors must use minimum channel length. The technology that we use is denoted as 45nm technology node because 45nm is the effective channel length of the transistors. However, the drawn size of the channel length is L=50nm, and therefore λ=25nm.
- Use only M1 and M2 metal layers. As a rule, M1 should be laid horizontally and M2 vertically.
- You must add well ties (PTAP and NTAP) to make connections between the supply rails and the bodies of the respective MOS devices.
- The layout must fit into a standard row height as follows: cell height = 140λ, with 20λ-high rails in Metal 1 for VDD and GND on the top and bottom edges of the cell, respectively. This leaves 100λ of vertical space between the rails.
- We suggest, but do not require, that the aspect ratio of your entire accumulator layout should be approximately height:width = 4:1. When four copies of your layout are arrayed (in Lab 3), the result will then be a square piece of layout. Since your bitslice layout will not fit within a 140x35λ rectangle (which is one row with 4:1 aspect ratio), your layout will need to span several rows (probably 4 or 5, based on past implementations of this lab) with shared power rails in the middle.
- Keep your layout area small. Don't spend hours moving rectangles around, rotating transistors, and reshaping wires to try to squeeze your width by a few more λ. Do, however, move the features as close together as possible without violating design rules.
- The layout must be designed as a bitslice so that a multi-bit accumulator can be constructed by abutting multiple copies of your layout with no additional wiring. This means:
    - CIN must be accessible in M1 at the left edge of your layout. COUT must be accessible in M1 at the right edge of your layout, and the two must be vertically aligned.
    - The PHI and RST signals must run all the way across your layout in a horizontal strip of M1.
    - SOUT must be accessible in M2 at the top of your layout.
    - A (for ECE658 students) must be accessible in M2 at the top of your layout.
- Here's an example layout in 0.25um tech of a complete bit slice accumulator: http://www.ecs.umass.edu/ece558/TA_Page/layout/index.html. Note that this accumulator is different in several ways than what you are designing, but it does have some features which are relevant to this lab:
    - 20 λ-high power rails crossing the entire cell in M1. Note that they alternate between VDD and GND.
    - NMOS devices located above/below GND rails, PMOS above/below VDD rails.
    - Allow some space at left/right of your layout, so that when several instances of the layout are later placed side-by-side, there won't be any DRC violations due to transistors being too close together where the layout instances abut.
    - Wide transistors are fingered; i.e. laid out as several short transistors in parallel with shared drains. This is an especially useful technique for large inverters.
    - Minimal routing in poly, since poly has a high resistance (ohms/square) which will slow down your circuit. Route your poly with a tree topology; a piece of poly should end as soon as it crosses the diffusion and overhangs just enough to satisfy the DRC rule.
    - Give names to signals, even if they're not inputs or outputs, to make debugging easier

Assumptions:

- The nominal supply voltage is 1.1V.
- Use β = 2 (the P/N width ratio).
- Upsize appropriately for series transistors while ensuring that the smallest devices satisfy the process minimum rules (NMOS=90nm, PMOS=180nm).
- Unless stated otherwise, assume that inputs, including clock, have 30 ps rise time and fall times (t_r and t_f, using 20%-80% metric from pg 141 of book)

III. Procedure:

Be sure to include each required item (indicated by **SUBMIT**) in your report. You must also explain what you did and why; images alone are not sufficient.

Step 1: Truth Table [5pts]

Write the complete truth table for the combinational logic of your bitslice. The truth table should include all combinations of RST, Q, CIN, and (for ECE658) A. The truth table should show the value of D and COUT for every such combination. Note that the clock input is not part of the truth table. Although Q is not a controllable input to the circuit, Q is treated similarly to RST, CIN, and A when creating the truth table. In later steps we will have to deal with the fact that Q is not directly controllable when we test the circuit.

SUBMIT: Truth table. Each line of the table should give the value of CIN, RST, Q, A (ECE658), D, and COUT.

Step 2: Create Schematic [10pts]

Design on paper your modified flip-flop that includes the required reset functionality. Then use the schematic editor tool (Cadence Schematic L) to draw a schematic for the entire accumulator which implements the adder, flip-flop, and reset circuit. Size all transistors appropriately for P/N ratio and series transistors. We recommend first creating transistor level schematics for gates and then composing them to create the full design. Capture screen images of the schematic editor window.

SUBMIT: Image of your transistor-level schematics, with sizes of all transistors visible. Also include the gate-level schematic.

Step 3: Check Functionality of Schematic [15pts]

Extract a SPICE netlist and simulate it using the HSPICE digital vector generation feature (.vec) to verify that it correctly implements the truth table. The process of checking whether a given implementation (e.g. a schematic) conforms to a given higher-level specification (e.g. an accumulator) is called validation. You will validate your schematic by simulation, which means that you must create an input sequence to apply to your accumulator's primary inputs and an expected response sequence which you can compare with your accumulator's internal nodes (D) and primary outputs (SOUT and COUT).

You should apply each row of your truth table to the circuit in sequence, with the inputs changing each clock cycle. Because Q is not directly controllable (it is the value of D from prior cycle), you will need to insert 1 or 2 additional vectors at various positions in the sequence to propagate the value of Q that is needed for the next test. For example, if Q =1 is needed for testing a row from the truth table, it can be induced by inserting a preceding vector with RST = 0, which then propagates to D=1, and after the next clock edge Q=1. Remember that, at the beginning of your simulation, the flip-flop is in an undefined state so you cannot assume an initial value of Q until you reset it.

SUBMIT: Validation test sequence in the form of a table. Each line of the table represents 1 clock cycle and should give the value of CIN, RST, Q, A (ECE658), D, COUT for that clock cycle. The value of Q should always be the value of D from the prior clock cycle. Indicate which rows of the table correspond to the rows of the truth table, and which have been added.
SUBMIT: The waveforms showing the simulator output as the tests are applied. The simulator outputs should be annotated to match the table, and in the same order as the table.

Step 4: Determine Maximum Clock Frequency [15pts]

Use trial and error to determine maximum clock frequency. Apply your test vectors from step 3 while sweeping the clock frequency and check at each frequency whether the results are correct. When the clock frequency is too high, the flip-flop will capture incorrect values in one or more cycles of the test. Note that the timing of the inputs must change appropriately in accordance with clock frequency. Rise and fall times of all inputs, including the clock, must not exceed 10% of clock period as the clock period is changed. Otherwise, the clock waveform may become triangular when the frequency is high, and this is not acceptable. Therefore, when applying a clock period below 300ps, the rise/fall time should be 10% of clock period, instead of being 30ps.

SUBMIT: State the maximum clock frequency.
SUBMIT: Give image of the simulator output showing correct functionality for the validation sequence at the maximum clock frequency. Annotate the maximum clock frequency on the simulation waveforms.
SUBMIT: Give another image of the simulator output failing while applying the same sequence at 10% higher frequency, and show on the waveforms where the incorrect result is captured into state.

Step 5: Power [10pts]

Apply the test vectors from step 3 and measure average power at three or more clock frequencies. The time period over which you compute the average power should be from the start of the first vector to the end of the last vector, as any extra time included will interfere with the measurement. Plot average power vs frequency. Calculate switching power in units of µW/MHz. Does your calculation show the power from the three frequencies to be consistent?

Extrapolate from the µW/MHz number you calculated to find what the dynamic power would be at your maximum operating frequency. Use µW/MHz to also estimate static power by extrapolating to a 0MHz clock. If you've been careful in your measurements and analysis, this number should be reasonably accurate. You can check your answer by measuring static power directly in SPICE.

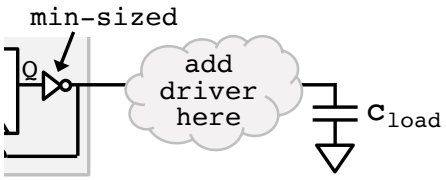
Calculate the clock frequency at which static power and dynamic power are equal.

SUBMIT: Measurements of dynamic power. Plot of average power vs frequency. Calculation of µW/MHz and explanation of consistency.
SUBMIT: Extrapolation of power at maximum frequency.
SUBMIT: Estimate of static power. Calculation of frequency at which static and dynamic power are equal.

Step 6: Layout [20pts]

Design a layout for your accumulator (use Cadence Layout XL). Your layout must pass DRC and meet the requirements that are described on page 2 of this document. Make sure your design also passes LVS. Extract a netlist from your layout with parasitic information (choose "R+C+CC" option from Calibre PEX). Repeat the simulation and analysis using the patterns from step 3 to make sure everything is working correctly. Then repeat the analysis from step 4 to find the maximum frequency at which your design can operate with the parasitics included. How does this compare to the frequency you found in step 4?

SUBMIT: An image of your layout, with the total height and width annotated. You must capture the layout with enough resolution so that it doesn't just look like a random patchwork of color. In addition to a single image showing the whole layout, you may provide several "zoomed-in" views if you feel it's necessary.
SUBMIT: Screen capture of DRC, LVS results.
SUBMIT: State the maximum clock frequency with parasitics and compare to the frequency without parasitics.
SUBMIT: As in step 4, give image of the simulator output showing correct functionality at the post-layout maximum clock frequency, and another image showing failure at 10% higher frequency. Show on the waveforms where the incorrect result is captured into state.



Step 7: Effort-based Sizing to Drive Load Capacitance [25pts]

Step 7a (ECE 558 and ECE658)**:** In this step you will design a driver for a large load capacitance that is attached to the SOUT output of your accumulator, as shown in the figure above. The driver that you design is used only for this step. The value of Cload in units of fF will be 20 + the sum of the last two digits of your student ID number (which means it will be between 20fF and 38fF). We know from lecture that driving such a large capacitance with only the existing minimum-sized inverter would result in a high delay. Therefore, you will design a 2-stage non-inverting driver with progressive sizing to reduce the delay. All delays in this step should be expressed in terms of parasitic inverter delay (τ); for example, a delay could be given as 6τ if it is calculated to be 6 times the parasitic inverter delay. You can use reasonable assumptions and approximations (for example, regarding the logical effort of inverters), but must demonstrate correct application of the principle of effort-based sizing. For simplicity, you can ignore the branching effort from the path that goes back to the input of the full/half adder. Explain your calculations.

- Estimate the delay when the load is driven only by the minimum sized inverter, with no driver added.
- Determine sizes for the first and second stage inverter in the driver that minimize the total delay between the input of inverter and the load. Size only the 2 driver stages; do not resize the inverter in the accumulator.
- Once you have sized the 2 stages, also estimate the propagation delay of each driver stage.

SUBMIT: Propagation delay when no driver is added, expressed in terms of τ.
SUBMIT: Optimized sizes for the two driver stages.
SUBMIT: The optimal total delay from the inverter to the load, expressed in terms of τ.
SUBMIT: Propagation delay of each driver stage, expressed in terms of τ.

Step 7b (ECE558 and ECE658)**:** Use SPICE to check your solution from step 7a. First run SPICE analysis of an unloaded inverter to find the actual value of τ. Then run SPICE analysis of a circuit with the inverter from the accumulator, your sized driver stages, and the load capacitance. Measure the total delay through this circuit, and the propagation delay of each stage. How well do these delays, relative to τ, agree with your analysis from step 7a?

SUBMIT: Measured value of parasitic inverter delay. Measured delay values of optimized driver circuit.
SUBMIT: Analysis of whether these values are consistent with your estimates from Step 7a.

Step 7c (ECE658 only): Now find empirically the optimal sizes for the two stages in the driver using HSPICE ".alter" statements, or any other SPICE-based technique you like. In other words, try all combinations of stage sizes, and measure the delay for each combination. Use these findings to evaluate the quality of the sizes from step 7a. Are they actually close to optimal?

SUBMIT: A plot showing how delay (reported by HSPICE) changes as a function of the stage sizes. Note that delay is a function of 1st stage size and 2nd stage size, so think about how you wish to show this. You could, for example, show a plot in which x-axis is 1st stage size, y-axis is delay, and one line is included for each value of 2nd stage size. Or you could have x-axis be 1st stage size, y-axis be 2nd stage size, and z-axis be delay.
SUBMIT: Text comparing your findings from SPICE to the analytical findings from Step 7a.

Step 7d (ECE558 and ECE658)**:** We know that, if the load capacitance gets very large, we may want to add more stages to the driver. In order to avoid changing the logic function, we would need to add two inverters to go from a two-stage driver to a four-stage driver. What is the smallest value for the load capacitor in which the optimal delay with a four-stage driver is lower than the optimal delay with a two-stage driver? What would the optimal delay of the four-stage driver be (in terms of parasitic inverter delay), and what are its sizes?

SUBMIT: Your analysis of when four-stage driver is appropriate, and your analysis of its optimized delay/sizing.