University of Pennsylvania
Department of Electrical and System Engineering
Circuit-Level Modeling, Design, and Optimization for Digital Systems

ESE3700, Spring 2024 — Proj1: Logic Optimization — Wednesday, March 13

Report Due: Friday, March 29, 11:59PM

Read the entire handout before starting.

Design Problem: Design the circuit-level implementation of a full-adder bit-slice (for a ripple-carry adder).

- You must work **individually**. You can discuss strategies with classmates, but all work must be your own.
- Target technology is the High Performance 22nm process (`/home1/e/ese3700/ptm/22nm_HP.pm`)
- Your design must be cascadable to build adders of arbitrary bit width and usable in adder trees (*e.g.* as used in multipliers).
- You must stay with a cascadable, bit-slice, ripple-carry design; we are not exploring organization tradeoffs in this assignment.
- You may explore polarity[1] optimizations on the carry chain and use multiple types of single bit-slice cells. Nonetheless, the number of cell types needed should be a small constant number independent of the size of the adder.

Designs: You will create two designs of your adder, one baseline CMOS design and another delay-optimized design.

- Baseline CMOS design: Use CMOS logic discipline, nominal 0.8V$= V_{dd}$, and minimum size transistors. This is the starting point.
- Delay-optimized design:
  - You select gate structure, gate types, transistor sizes, and logic discipline.
  - You may use any voltage $V_{dd} \leq 1.0$V.
  - Make an initial list of issues/design-options to explore before you start detailed SPICE simulation to test your hypotheses and refine your designs (maybe even before you start detailed manual analysis).

  Hint: Verify design in components. Test sub-circuit (logic gates, bit-slice, etc.) functionality before full adder. Create different test schematics for each design metric of your adder.

[1] That is, whether a high input or a low input represents that a particular condition is true. Alternating carry polarities can result in about twice the speed.

Design Metrics:

Both 8b adder designs you implement must be measured with the following 5 design metrics:

- *delay*: Measure the delay of a 8b addition. This is the worst-case delay from two 8b inputs to all 9b of output (including final carry out). Since the input is only the two 8b inputs, the carry-in to the first bit-slice will be tied to a logic low ('0'). Load the outputs by the equivalent load as the input to your adder. Drive the inputs by the equivalent drive of one of your adders. Both input drive and output load are consistent with this being used in an adder tree. You must consider what input case would give the worst case delay.
- *active energy*: Measure the energy for the 8b addition in two cases:
  1. Maximum switching energy case (All inputs switch from $0 \rightarrow 1$)
  2. Average switching energy case (Half the inputs switch from $0 \rightarrow 1$)
- *leakage energy*: measure the leakage energy for one adder delay period when the inputs do not change in two cases:
  1. Maximum leakage energy case
  2. Minimum leakage energy case

  For leakage energy, the maximum and minimum case could vary depending on your implementation. You should identify and justify your choice of inputs for each of the cases. (**Hint**: Do the inputs switch for leakage energy? How many cases are there for each bit slice? How does the overall delay, leakage energy, or active energy relate to the results you get for a single bit slice?)
- *area*: Sum the total transistor width for the design. This is a crude metric, but it is simple to calculate since we are not performing layout.

Milestone (March 22): Only for you to check your progress (no need to turn in anything).

- Baseline Design Schematics
- Description of how you validated logical correctness of the baseline design. What test cases did you use? How did you select them? Why do these test cases demonstrate the correctness of your design for all input cases?
- Explanation of the delay in the design in τ units.
- Summary table of the design metrics for baseline design to two significant figures. Include supporting evidence in the form of equations and simulation results.
- Identification and justification of test cases used for evaluation metrics.
- Initial list of issues/design-options to explore containing at least 6 ideas.

This won't need to be as extensive as the report. However, it is a good chance to get feedback and make corrections before you prepare the report. We will also make an effort to give you timely feedback on the list of issues/design-options you plan to explore.

Final Report: Your report should be a single, stand-alone PDF document turned in and should include:

- Final report should be a typed document and all diagrams, graphs, equations, and results should be computer generated. Hand written labels of axes and annotations in simulation results will not be accepted for the final report.
- For each of the **baseline and final** design, provide:
  1. All schematic(s) for the design – make sure transistor sizing annotations are easily readable in the diagram you include in the report.
  2. Text description of the logic and operation. Use equations as necessary. This is just a few sentences for the baseline design. The optimized description might be longer if your design is tricky. Your description should make it easy for us to understand how and why your design works. The description should explain what features of the design make it fast.
  3. Explain the delay of the design using τ units and (where appropriate) Elmore delay model. If SPICE simulations points you in a different direction from these simpler models, call that out in your description.
  4. Description of how you validated logical correctness of the design. What test cases did you use? How did you select them? Why do these test cases demonstrate the correctness of your design for all input cases?
  5. Identification and justification of test cases used for evaluation metrics. Include your test schematics with the test cases.
  6. Summary table of the design metrics to two significant figures. Include supporting evidence in the form of equations and simulation results. Annotate all simulation results for relevant information and make sure to label all graphs with axis labels and titles.
- A brief description of alternate designs or variations that you tried and why they were inferior. This will include both the items on your initial list from the milestone and additional options that became clearer to you as you explored the options and refined your design. If you can clearly and succinctly describe how an alternate differed with words and numbers that is sufficient. You may include schematics when words alone would be inefficient.
- A brief description of how you used the alternates and variations to arrive at the final design, highlighting what you learned from the designs. Include graphs and tables as appropriate to show how the alternatives compared and support your final design selection.
- A brief description of how the area and delay of your adder would vary when reducing the number of bits? (e.g., 8b adder, 4b adder, 2b adder and 1b adder). Include graphs and tables as appropriate to show how these reduced-precision adders compared and support your conclusion. (Note, in modern machine learning and AI chip design, low precision arithmetic unit is much more popular. Think about why.)

- Please include a statement on your final submission:

  I, *your-name-here*, certify that I have complied with the University of Pennsylvania's Code of Academic Integrity in completing this project.

  You can review the Code of Academic Integrity here: `http://www.upenn.edu/academicintegrity/ai_codeofacademicintegrity.html` If any cheating is suspected, all parties will be penalized and reported to Office of Student Conduct.

Where do I start? First start at the function level. Given three inputs A, B and CarryIn, what are the outputs Sum and CarryOut? Write a boolean function describing this. You now know how to design this with multiple fundamental CMOS logic gates (NAND, NOR, INV, XOR, etc.) or a single CMOS logic gate. Your ultimate objective is to decide on a topology and size it to minimize delay.