Olin College ENGR3410: Computer Architecture Fall 2022
Name:

Collaborators:

Homework 5

The git repository hws/5 directory is the basis for this homework. Space for answers intentionally left out - include these in your single pdf in the submission zip file. Also this grew beyond last year's version based on a lot of recent feedback. Let me know if this is on the right track!

Reading

- ***The verilog cheatsheet was just updated to include a ton of extra tips. Re-read it!***
- Chapter 5: 5.2 to 5.2.4, 5.4, 5.5
    - Optional: 5.3 Number Systems - Highly encouraged for anyone interested in scientific computing!
    - Next Time: 5.2, 5.6
- Context: "The Magical Number Seven, Plus or Minus Two: Some Limits on Our Capacity for Processing Information" G A Miller [pdf]
- Did you know that there is a comprehensive gtkwave manual? See Q4.

1. Context Beyond CompArch - Psychology/Information Theory

Read "The Magical Number Seven, Plus or Minus Two: Some Limits on Our Capacity for Processing Information" G A Miller. This is one of the most cited papers in the field of psychology, but it has tremendous implications to appropriate design of digital systems.

a) Describe at least one situation in which you had to use or measure more bits than were useful to a problem.

User interviews often require gathering more info. than is useful

b) Have you ever had a situation where you didn't use enough bits?

Giving a survey that didn't have enough granularity (should have used likert)

c) How many bits did you use to answer the previous question?

A lot! Depends how letters/words are encoded

d) In your own words, what is the difference between a bit and a chunk?

A bit is an individual yes/no, while a chunk is a more manageable grouping of bits that helps people retain/transmit more information

e) How are the generalizations from this paper still applicable half a century later? Alternatively, what do you feel no longer applies?

I felt that the studies would be redone (especially to see the impact of the internet on information retention

f) (optional) This is a paper I feel everyone should read[1], do you have an equivalent must-read scientific article? Meadow's "Leverage Points"

[1] I have sent this to family, friends, and colleagues. Amazingly I'm still on some of those group chats.

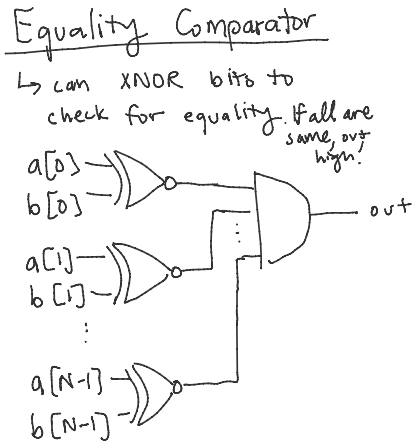
2. Combinational Review: ALU Part Two

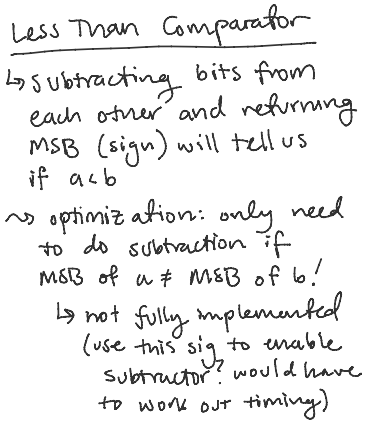
Use only structural combinational logic (no flops, no ifs/elses, etc.). That means use `always_comb` statements with only ~&|^? operators for these modules. A working adder is included in the homework folder.

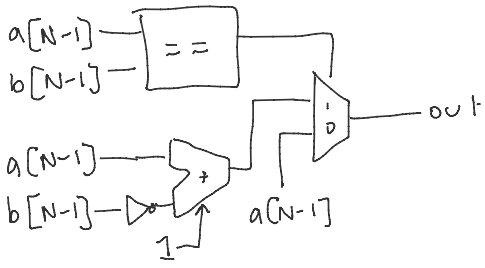
- Implement a compare_eq module that compares two n-bit numbers and outputs high if they are equal, low if they are not.
- Implement a compare_lt (less than "<" operator) that can support up to 32-bit inputs, that outputs high if a is less than b.
- Make sure the Makefile can still run the test (include any of the submodules in the target!).
- Augment the test_comparators.sv example with more test cases to give you confidence that you have implemented the two comparisons correctly.
- Descriptions **and** schematics in the top level PDF that show your approach to the two comparators.
- Bonus: Use the cheatsheet to load in test vectors from a memh/memb file. Update make test* and submission accordingly. *You'll get a guided version of this next homework.*

Confidence/Skills Check

This should feel straightforward at the block diagram level, but possibly still tricky in terms of execution in SystemVerilog. Contact an instructor ASAP if you don't know how to start, reach out to peers after ~15 min of (re)reading if you feel you can't sketch a block diagram, then follow up with more instructor time if you still feel shaky. Only start writing HDL after you have a block diagram that is at least vetted by a peer! Reach out to instructors after ~5-10 min of debugging HDL, Makefile, verilator, iverilog, or gtkwave issues.



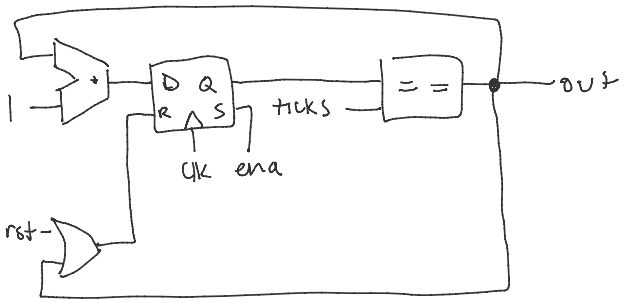




BLOCK DIAGRAMS

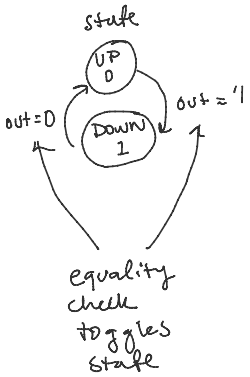
Pulse Generator

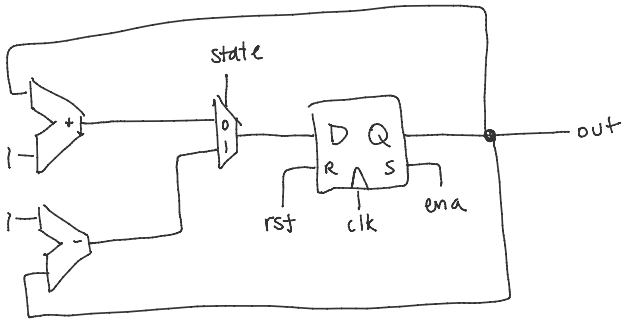
↳ need a counter to increase every clk cycle. that is compared to the input ticks. Output that comparison. If the comparison is 1, reset the counter.



Triangle Generator

2 states → count up, count down
if the output is ever 0 or high, we switch state.
↳ could do this w/ gates we made earlier, more lines of code





PWM

Have a counter, if the duty is greater than or equal to counter output high, unless duty = 0, then output low.
otherwise, output low.