Combinatorial Circuits & Storage Elements

CS 350: Computer Organization & Assembler Language Programming

Due Fri Apr 5 (2400 hrs)

[4/8 Program solution on alpha]

A. Why?

- Combinatorial logic circuits correspond to pure (state-free) calculations on booleans.
- Storage elements are the basic circuits that store data, which is used in logic circuits that use memory.

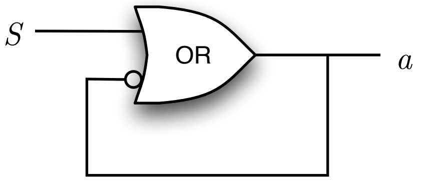
B. Outcomes

After this lecture lab, you should

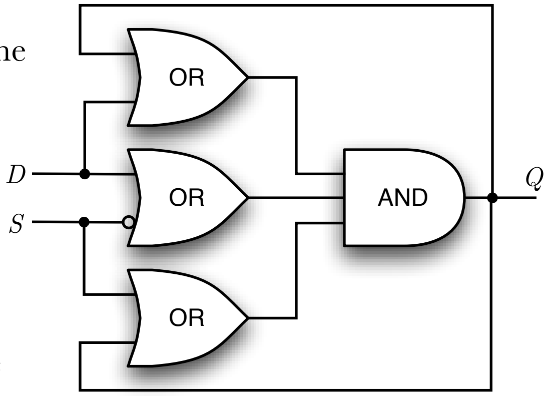
- Be able to analyze/draw the circuitry for simple arithmetic/logical calculations.
- Be able to determine whether a logic circuit has a logically stable state.

C. Problems [50 points total]

1. [18 = 3*6 pts] Consider the circuit to the right. (a) Describe the new value of a as a function of S and the current value of a. (b) When does this circuit have a logically stable values for a? A logically unstable value for a? (c) Can this circuit be used to remember a bit?



2. [18 = 3*6 pts] Consider the logic circuit to the right. (a) Describe the new value of Q as a function of D, S, and the current value of Q. (b) When does this circuit have logically stable or unstable values for Q? (c) Can this circuit be used to store a bit? (I.e., does it remember a bit? Can we set the bit when we wish?)



3. [14 = 2*7 pts] Exercise 3.24: (a) [The figure below] shows a block-level logic circuit that appears in many of today's processors. Each of the boxes labelled "+" is a full-adder circuit. What does the value on the wire X do? That is, what is the difference in the output of this circuit if $X = 0$ versus if $X = 1$? (b) Modify the logic diagram below so that implements an adder/subtracter. That is, the logic circuit will compute $A + B$ or $A - B$ depending on the value of X. [Hint: Replace each C_i with a circuit that uses B_i and possibly other inputs.]

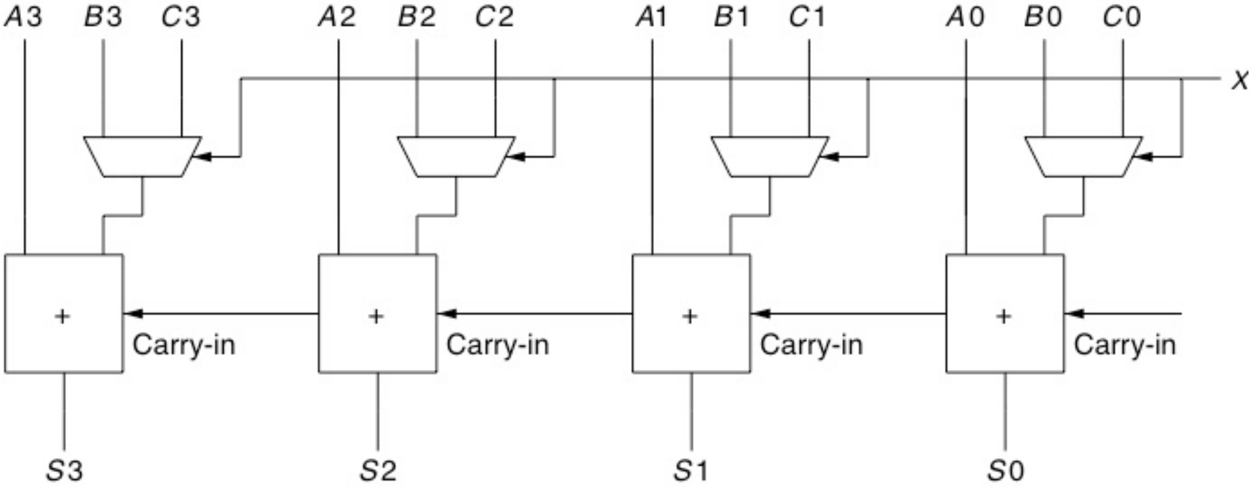


Figure for Exercise 3.24

D. Programming Problem [50 points total]

- For Labs 7 and 8, you'll be implementing a version of the Simple Decimal Computer (SDC) from lecture, in C. You'll write a line-oriented program that reads in the initial memory values and then lets the user execute the program one instruction at a time.
- The SDC is a decimal computer with memory addresses `0000` – `9999`, ten general-purpose registers numbered `0` – `9`, and ten instructions. The SDC uses word addressability, with a word being 4 decimal digits (plus a sign-magnitude sign).
- For Lab 7, we'll work on reading in memory and reading commands (execute a instruction, print some help, or quit). Lab 8 will address the actual execution of instructions.

What Should Your Program Do?

1. Prompt for and read in the values for memory locations `00`, `01`, etc. Read until you see a number > `9999` or < `-9999`, and initialize the rest of memory to all zeros. (Each memory location is supposed to contain a value ≤ `9999` and ≥ `-9999`, so we can use values outside that range as sentinels.)
2. Print out the memory values and initialize the control unit. (For Lab 8, set the registers to zero and set the `done` flag to false.)
3. While not done
4. Prompt for and read a command (read the rest of the line including the carriage return).
5. For command `q`, set `done` to true.
6. For command `h` or `?`, print out a help message. (For Lab 8, this is just a placeholder message.)
7. For the null command (just a carriage return), call the `instruction_cycle` function to execute an instruction. (For Lab 8, this function just prints out a message saying how many times we've called it. On the tenth time, it also sets `done` to true.)
8. When you're done, print out the memory values and the control unit. (For Lab 8, just print the registers [which are still all zero].)

E. Sample Solution and Skeleton

- To get you started with the program, there's a partial non-working skeleton at http://www.cs.iit.edu/~cs350/Class350/Lab07_skeleton.c . Add, change, or delete lines in the skeleton as necessary; the STUB comments can be replaced with code. You don't have to use the skeleton if you don't want to.
- There's a sample output at at http://www.cs.iit.edu/~cs350/Class350/Lab07_soln_out.txt .
- I've posted ~~I'll post a~~ sample executable solution on alpha. You can execute `~sasaki/Lab07_soln` .