University of Washington – Computer Science & Engineering
Winter 2026 Instructor: Naomi Alterman 2026-03-10

CSE 369 QUIZ 3

Name: Salvatore Solutionetti

Student ID Number: 0xFFF

Please do not turn the page until 11:50.

Instructions

- This quiz contains **8** pages, including this cover page. You may use the backs of the pages for scratch work.
- Show scratch work for partial credit, but put your final answers in the boxes and blanks provided.
- The quiz is closed book and closed notes.
- Please silence and put away all cell phones and other mobile or noise-making devices.
- Remove all hats, headphones, smart glasses, watches, and other digital wearables.
- You have 60 (+10) minutes to complete this quiz.

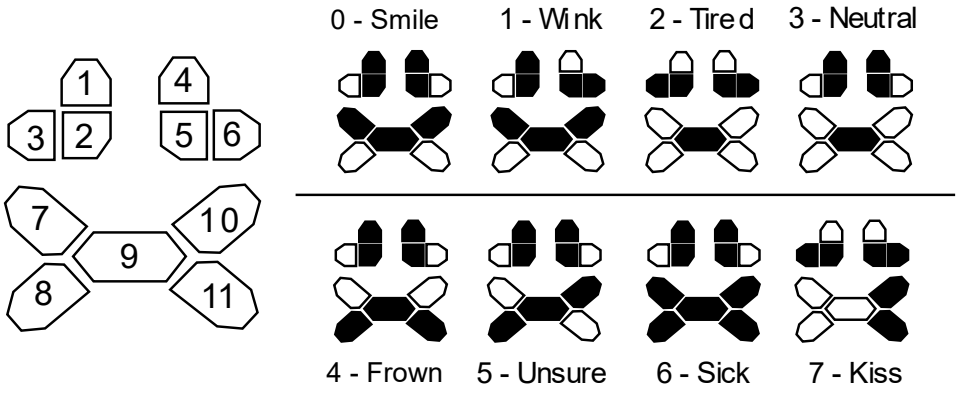
Advice

- Read questions carefully before starting. Read *all* questions first and start where you feel the most confident to maximize the use of your time.
- There may be partial credit for incomplete answers; please show your work.
- Relax. If you've been practicing, you got this. If you haven't, you'll learn something now.

Question	Points	Score
(1) Decoders	10	
(2) Counters	24	
(3) Shift Registers	14	
Total:	**48**	

Question 1: Decoders [10 pt]

We're building an electronic emoji keychain using **decoders** and a specially-built **11-segment display**:



The input to the circuit is the 3-bit binary ID number for the emoji to display (listed above next to each emoji name). The output is an 11-bit bus called "S." Each bit of S drives one of the 11 display segments (numbered above-left). '1' means the segment should be filled in/black, and '0' means the segment should be empty/white.

(A) Complete the truth table to the right. [4 pt]

(B) In the space below, solve for the minimal logical expression for S_{10}. Show your work. [5 pt]

ID_2	ID_1	ID_0	S_3	S_{10}
0	0	0	**0**	1
0	0	1	**0**	1
0	1	0	**1**	0
0	1	1	**0**	0
1	0	0	**0**	0
1	0	1	**0**	1
1	1	0	**0**	1
1	1	1	**1**	1

$ID_2 \backslash ID_{1,0}$	00	01	11	10
0	1	1	0	0
1	0	1	1	1

$$S_{10} = \overline{ID_2}\,\overline{ID_1} + \overline{ID_1}ID_0 + ID_2ID_1$$

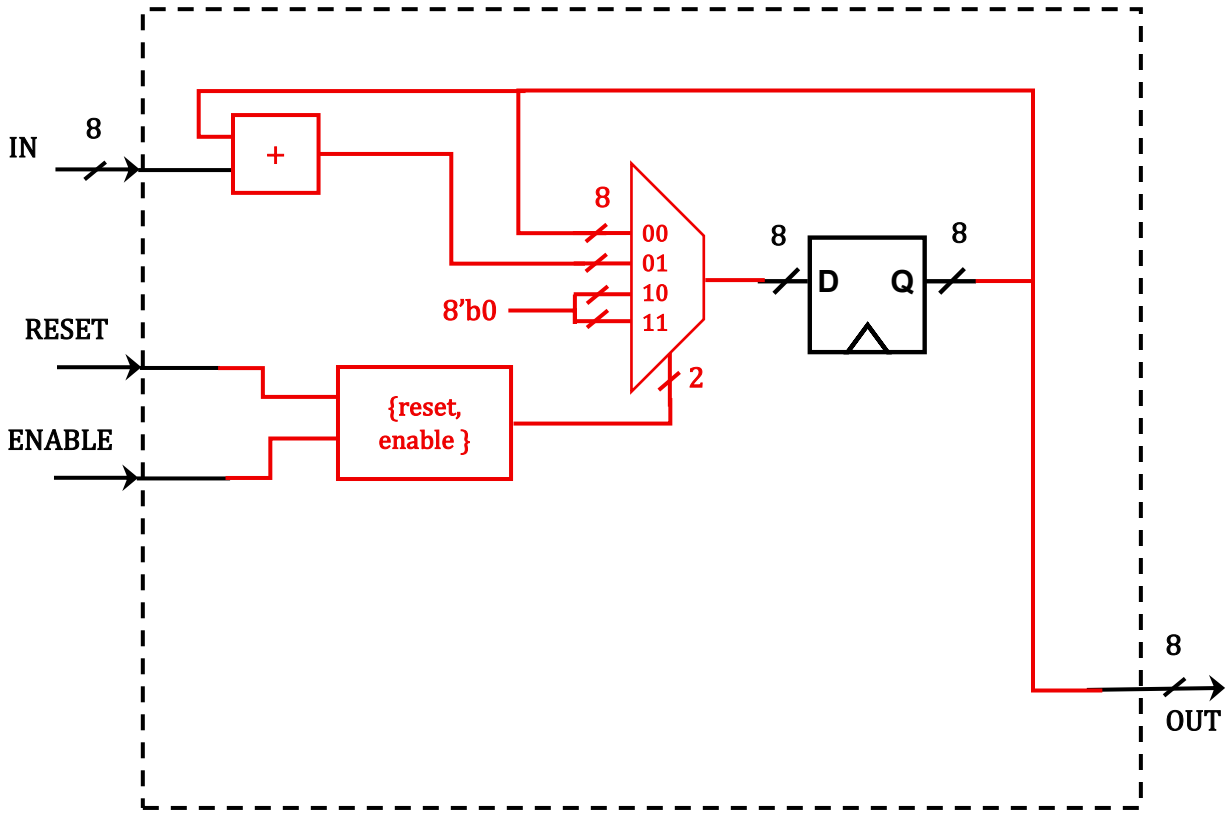
$$S_{10} = \overline{ID_1}(\overline{ID_2} + ID_0) + ID_2ID_1$$

(C) Is the "S" bus in our circuit a one-hot signal? Why or why not? [1pt]

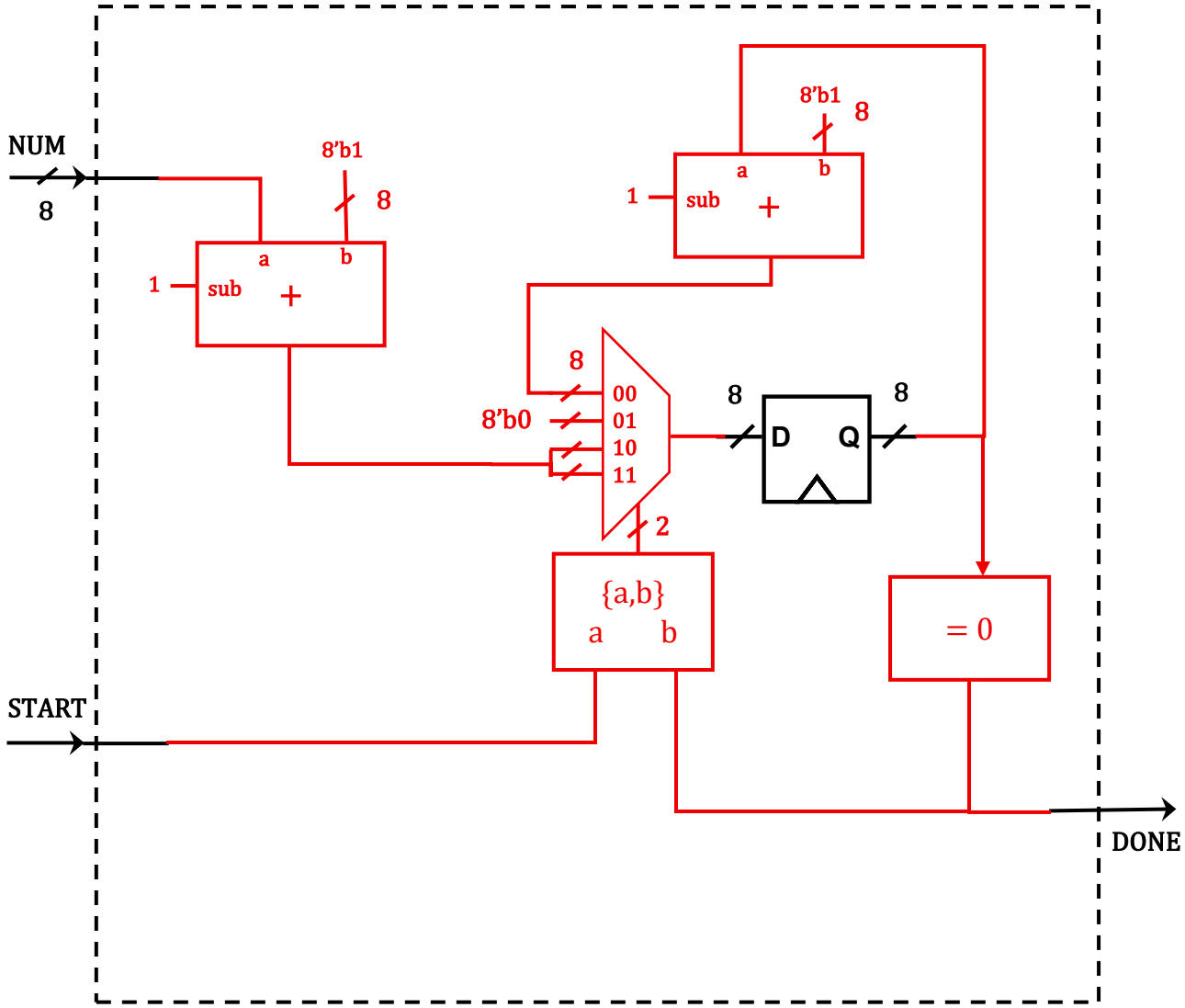
No – one-hot signals guarantee that exactly one bit of the bus will be "1" at any given time, which is not true in this case.

Question 2: Counters [24 pt]

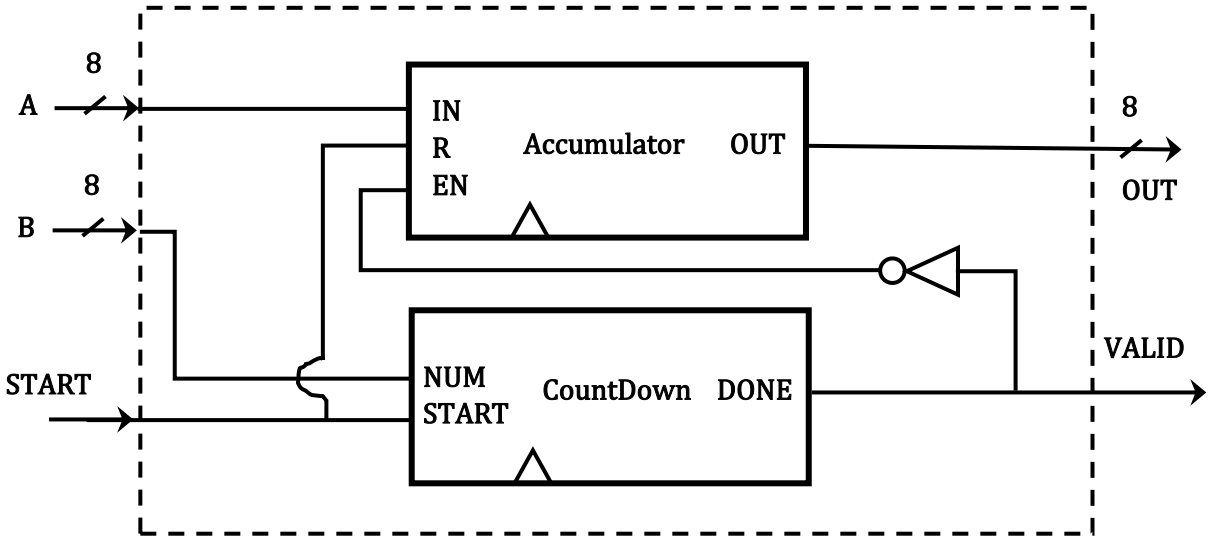
A) Complete the block diagram for a module called "Accumulator", which sums up the numbers provided to its input bus. Its 8-bit output "OUT" should start at 0 the cycle after RESET is high. Then every cycle after that, if ENABLE is high, it should increase by the number from 8-bit input "IN". If RESET and ENABLE are both high, RESET should take precedence. You may use any logic gates and blocks from the "design library" stapled to the back of the exam. Assume all clocks are properly connected for you. [5 pt]



B) Complete the block diagram for a module called CountDown, which has an 8-bit input "NUM," a 1-bit input "START," and a 1-bit output "DONE". When START is asserted, "DONE" should become 0 for the next "NUM" clock cycles, after which it should go back up to 1 and stay that way until the next START. You can assume START won't trigger in the middle of a countdown. You may use any logic gates and blocks from the "design library" stapled to the back of the exam. Assume all clocks are properly connected for you. [10 pt]



Using the previous two counters, we built a multiplier! Since multiplication is just repeated addition, it multiplies two numbers A and B by initializing the accumulator 0 and then adding A into it every clock cycle for B clock cycles:



C) Does the above implementation work if we interpret either A or B as two's complement negative numbers? Why or why not? [3 pt]

Yes, even though a two's complement value for "b" will be interpreted as a very large positive number by our CountDown circuit, the accumulator will eventually wrap around probably to produce a result that, when interpreted negatively, will still all "just work out". Two's complement is magic.

D) Does the above implementation work if A or B are zero? Why or why not? [3 pt]

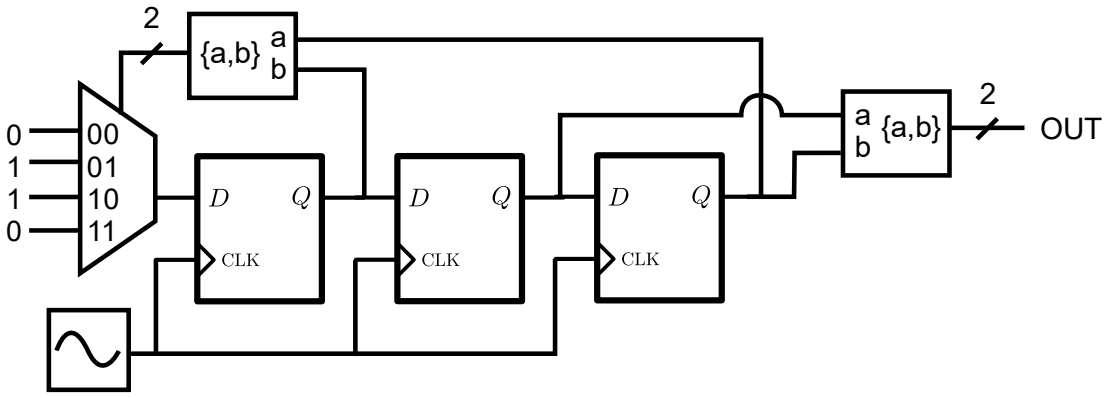
Yes, if A is 0 then the accumulator's answer will simply never change, and if B is 0 then the accumulator will never accumulate after reset.

E) What is a sequence of test inputs we could apply to the above multiplier implementation that would cause it to output a wrong answer when VALID high? Explain the cause of the failure. Assume all timing constraints are being properly met and that any A and B always stay within the range of [1, 15]. [3 pt]

Changing A in the middle of the multiplication operation will break the result, since the accumulator does not "store" its input number from one cycle to the next

Question 3: Shift registers [14 pt]

Consider the following 3-bit LFSR being used to generate 2-bit "random" numbers in a casino:



a) Assuming the state of this LFSR resets to 001 (stored in the flip-flops from left to right), write out one full cycle of the "OUT" signal. Clearly box your answer to distinguish it from scratch work. [8 pt]

Internal state:
001 -> 100 -> 110 -> 111 -> 011 -> 101 -> 010 -> 001

OUT values:
01 (1) -> 00 (0) -> 10 (2) -> 11 (3) -> 11 (3) -> 01 (1) -> 10 (2) -> 01 (1)

b) What is an initial/reset state for these flip-flops we should **absolutely avoid** if we want to make sure this LFSR functions properly? Why? [2 pt]

000, it's a sink state that just transitions back to itself (0 xor 0 = 0)

c) We've been asked to make our LFSR design more space-efficient and can no longer use the mux. What can we replace it with to maintain the same functionality? You may use any gates you like. [2 pt]

It's just implementing an XOR gate!

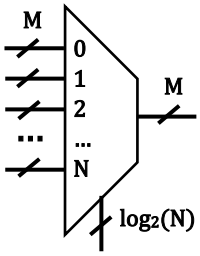
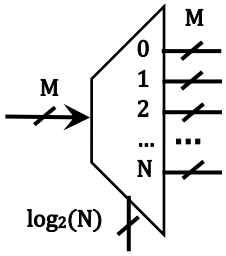
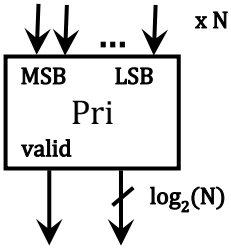
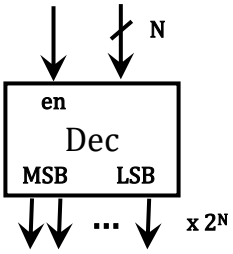
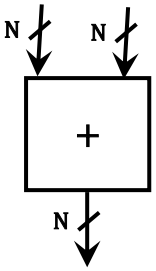
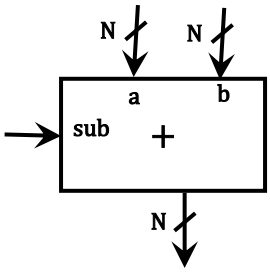
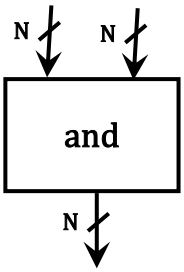
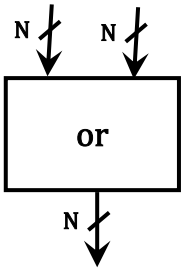
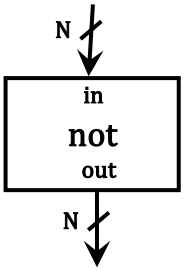
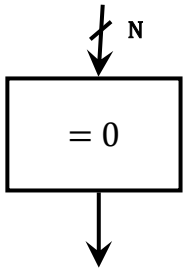
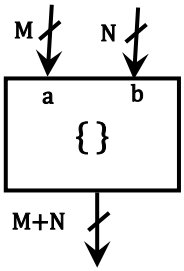
d) Based on the circuit diagram for this LFSR, do you expect it to be harder to meet setup time or hold time constraints? Why? [2 pt]

Hold time constraints.

It's not setup because there aren't any particularly long paths through the logic (the longest being from the output of a flip flop through the select signal of a mux, which is minor).

On the other hand, there are multiple flip flops that output directly into other flip flops, and we'll either need delay elements on those wires or VERY good hold time properties for our flops.

Design Library

M-bit N:1 Mux 	M-bit 1:N Demux 	N-bit Priority Encoder 
N-bit Binary to One-Hot Decoder w/ Enable 	N-bit Adder 	N-bit Adder/Subtractor (a+b / a-b) 
N-bit Bit-wise AND 	N-bit Bit-wise OR 	N-bit Bit-wise NOT 
Zero Comparator 	Concatenate ({a, b}) 	Danni