ECE/COMPSCI 350 – Digital Systems

Section 001 – Fall 2023 – Midterm Exam #1

Name: Edmond Niu

READ THIS:

- This is a closed-note, closed-book, closed-internet, closed-peer, calculator-free exam, with the exception of one *single-sided* 8.5x11 or A4 reference sheet with the content of your choice. **If you are caught violating these rules by the teaching staff, you will receive a -100 on the exam and will have an academic integrity violation filed against you.**
- We will likely be scanning this exam for grading, so to keep things together, please **put your NetID on the upper right of each page before you begin!** There are **9** pages total.
- Time management is part of the exam. You have 75 minutes for 7 questions, so aim to average ~10 minutes per question. Make sure you get to all of them!

Please sign the **honor pledge** below to affirm that you understand the rules of this exam period. Your exam will not be graded if you do not sign the honor pledge.

"I have neither given nor received unauthorized aid on this test or assignment"

Signature: Date:

Question 1 [32 points]: You are given the function $F(A,B,C) = \sum m\,(3,4,5,7)$.

(a) [8 pts] Give the canonical sum-of-products representation of F.
(The logical expression in full written-out form)

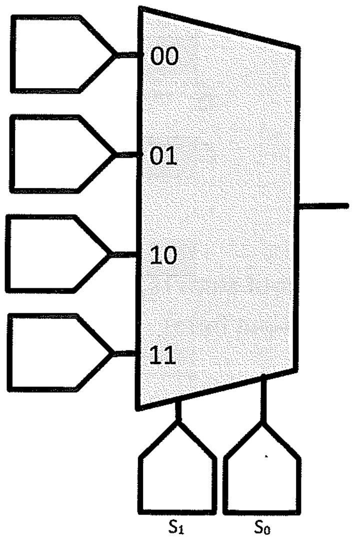
(b) [12 pts] Give the logic expression for the minimal sum-of-products form of F. Space if given below for you to work out the simplification, but put your final expression in the box provided. Hint: the minimal form has two terms of two inputs each.

(c) [4 pts] Draw a circuit representing the expression from part (b) as a 2-level circuit ***using only NAND gates***. The inputs and their complements are freely available.

(d) [8 pts] Give the canonical product-of-sums representation of F.
(The logical expression in full written-out form)

Question 2 [12 pts]: You have just a single 4:1 mux. You have *no* other gates at all. Further, you *don't* have access to the complemented form of the inputs. Show how you can implement the following logic expression under these constraints. Do so by writing one of {0, 1, ***a***, ***b***, ***c***, ***d***} into each mux input box.

Expression: $\bar{d}(a\bar{c}+bc)+\bar{c}d$



Question 3: [10 pts] Below is Verilog for a 6-bit sign detector that's kind of silly.

```
// give is_neg=1 for negative input, else 0
// give is_pos=1 for positive input, else 0
// zero is neither positive nor negative
module sign_finder6(v, is_neg, is_pos);
    input [5:0] v;
    output is_neg, is_pos;
    wire [5:0] neg_v;

    assign is_neg = v[5];      // sign bit of v
    assign neg_v  = -v;        // 2's complement negation, so NOT(v)+1
    assign is_pos = neg_v[5];  // sign bit of -v
endmodule
```

(a) [6 pts] One specific input value will give incorrect results. What is it? Why?

(b) [4 pts] What simple change could you make to fix it? You may use behavioral or structural Verilog.

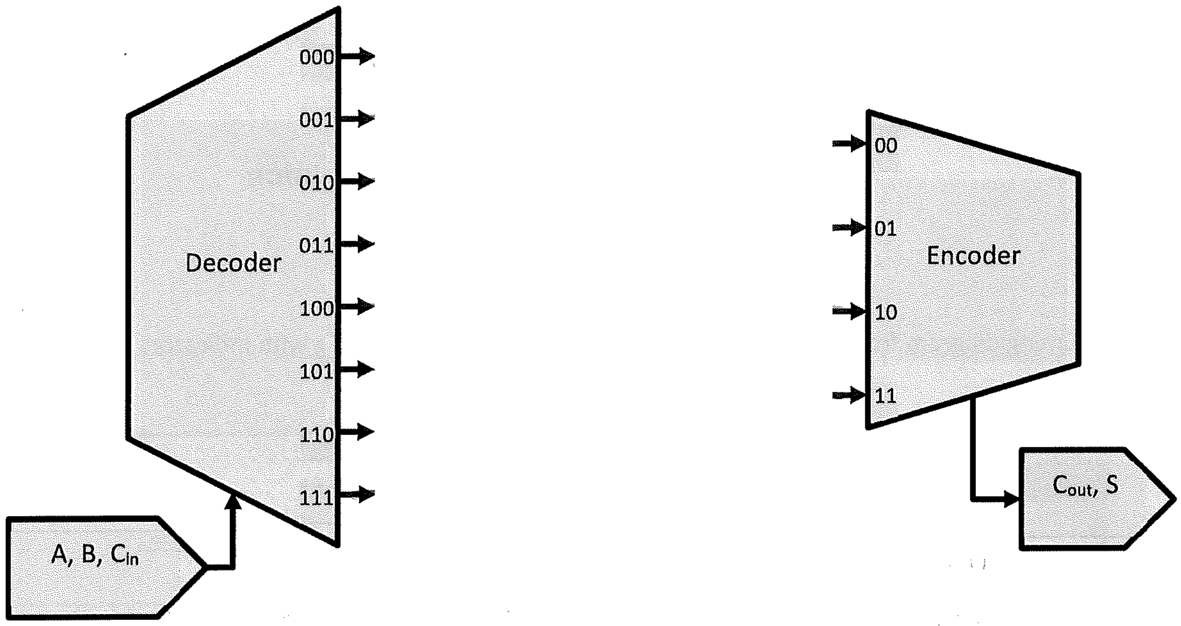
Question 4: [20 pts] Some binary arithmetic:

(a) [12 pts] Show the binary addition of the 8-bit values 01111001 and 01100100.

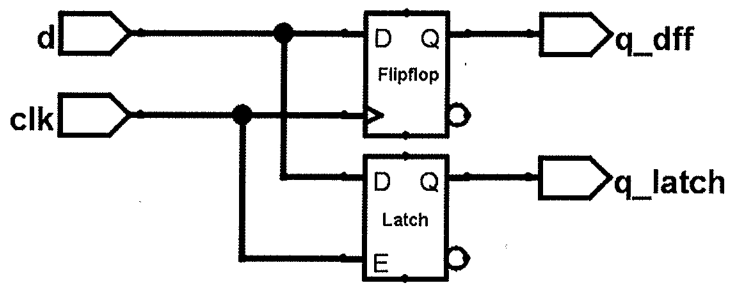
(b) [4 pts] What is the decimal interpretation of this for 2's complement signed numbers? Is there signed overflow? Why or why not?

(c) [4 pts] What is the decimal interpretation of this for unsigned numbers? Is there unsigned overflow? Why or why not?

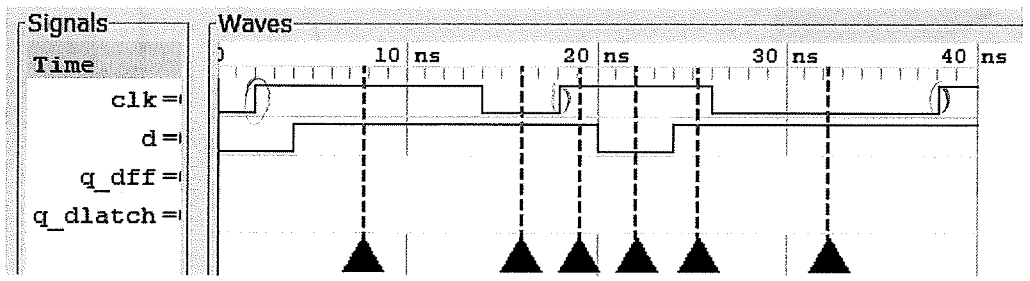
Question 5: [8 pts] Use a 3:8 decoder, a 4:2 encoder, and any number of OR gates to make a full adder that takes input bits A, B, C_{in} and produces output bits C_{out} and S. A template is provided.



Question 6: [6 pts] The circuit below has a D Flipflop and a D Latch driven from the same d and clk signals:



Below is a timing diagram from GTKwave showing changes to d and clk, along with indicators of specific points in time.



What are q_dff and q_dlatch at each of the time points indicated?

Time	q_dff	q_dlatch
8 ns		
16 ns		
19 ns		
22 ns		
25 ns		
32 ns		

Question 7: [12 pts] Use Booth's algorithm (original, not modified) to multiply -5 (the multiplicand) and 7 (the multiplier). The inputs should be treated as 4-bit values and the product will be an 8-bit value. Similar to how we saw in lecture, assume we've done the space savings hacks so the product register has both the product under construction and the multiplier being consumed. Write out each individual step (e.g. "Shift right", "Add M", or even "Add nothing") and the resulting value in the 8-bit product register.

As a hint, note that -5*7 = -35.