**The University of Michigan**
**Department of Electrical Engineering and Computer Science**

## EECS 270 Fall 2003

# Practice Final Exam Solutions

---

Name:_____ UM ID:_____

---

*For all questions, show all work that leads to your answer.*

| Problem # | Possible Points | Points Earned |
|---|---|---|
| 1 | 13 | |
| 2 | 19 | |
| 3 | 14 | |
| 4 | 17 | |
| 5 | 15 | |
| 6 | 12 | |
| 7 | 10 | |
| Total | 100 | |

*I have neither given nor received any unauthorized aid on this exam.*

*Signed:* _____

1. *Number Conversion: 13 Points Total*

(a: 3 pts) $-20_{10} = ?_2$ using 6-bit two's-complement representation

$20_{10} = 10100$; Sign extend = 010100
Flip all bits: 101011; add 1 = 101100
$-20_{10} = 101100_2$ (two's-complement representation)

(b: 3 pts) $AB.9_{16} = ?_8$

$A \quad B \; . \; 9_{16}$
$= 1010\ 1011.1001_2 = 010\ 101\ 011.\ 100\ 100_2$
$= 2 \quad 5 \quad 3 \quad 4 \quad 4_8$
$= 253.44_8$

(c: 4 pts) Represent the decimal number 26.6 with a binary number containing no more than eight bits. What is the numerical error in this representation of the number 26.6?

$26_{10} = 11010_2$

With 8 bits, $26.6_{10}$ is represented by $11010.100_2$

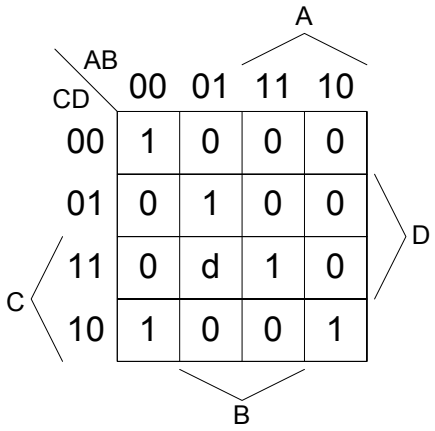
.6 * 2
1 | .2 * 2
0 | .4 * 2
0 | .8 * 2

This is equal to 26.5, so the numerical error in this 8-bit representation of 26.6 is 26.6 – 26.5 = 0.1

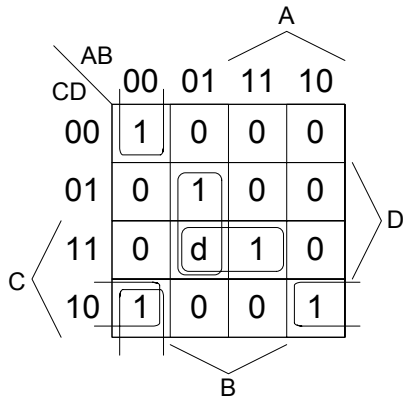
2. *K-Maps: 19 Total Points*

Consider the following function and its K-map:

$$F = \Sigma_{ABCD}(0, 2, 5, 10, 15) + d(7)$$



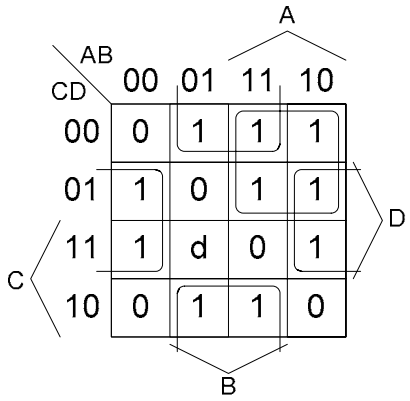

(a: 3 pts) Construct the minimal S.O.P. expression for F.




$F = A' \cdot B' \cdot D' + A' \cdot B \cdot D + B \cdot C \cdot D + B' \cdot C \cdot D'$

(b: 4 pts) Construct the minimal P.O.S. expression for F.

**F':**




$$F' = B \cdot D' + A \cdot C' + B' \cdot D$$
$$F = (F')' = (B' + D) \cdot (A' + C) \cdot (B + D')$$

(c: 3 pts) What is the cost of each expression (S.O.P and P.O.S.) in terms of number of literals?

S.O.P. cost is 12 literals and P.O.S. cost is 6 literals.

(d: 4 pts) Are the S.O.P. and P.O.S. identical functions?

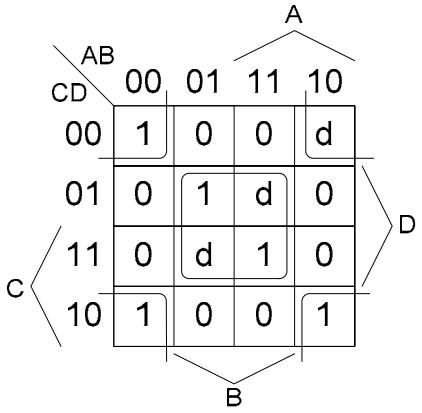
Yes: In the minimal S.O.P., the don't care cell is circled by a prime implicant and therefore assigned to be 1. In the minimal P.O.S., the don't care cell is not circled and therefore also assigned to be 1. Therefore, the functions are equal.

(e: 5 pts) Change two cells in the K-map above to don't cares, and derive the new minimal S.O.P and P.O.S. expressions. Choose your don't cares such that:

1) The new minimal S.O.P. and P.O.S. are equal functions
2) The sum of the number of literals from the S.O.P. and P.O.S. is minimal.

Change cells 8 and 13 to don't cares…

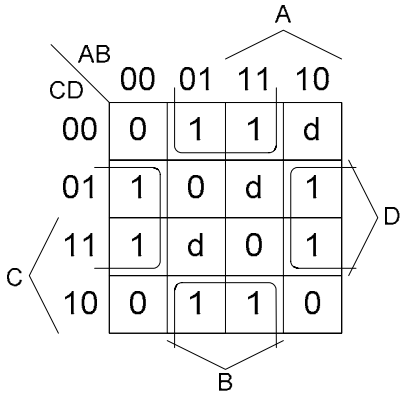
New S.O.P. expression:




$$F = B' \cdot D' + B \cdot D$$

New P.O.S. expression:

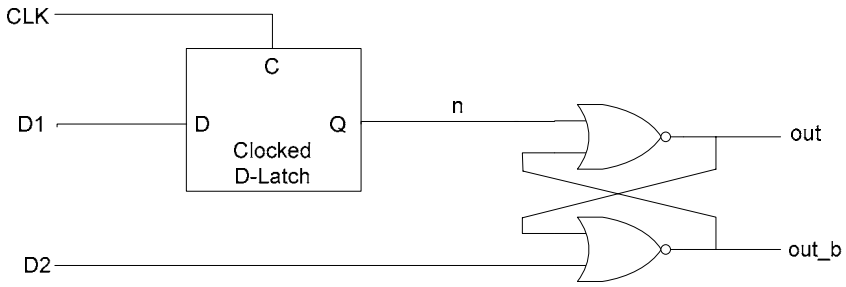
**F':**




$$F' = B \cdot D' + B' \cdot D$$
$$F = (F')' = (B' + D) \cdot (B + D')$$

All don't cares are assigned to 1 in both the S.O.P. and the P.O.S. so the functions are equal. The sum of the number of literals from the S.O.P. and the P.O.S is 8, which is minimal.

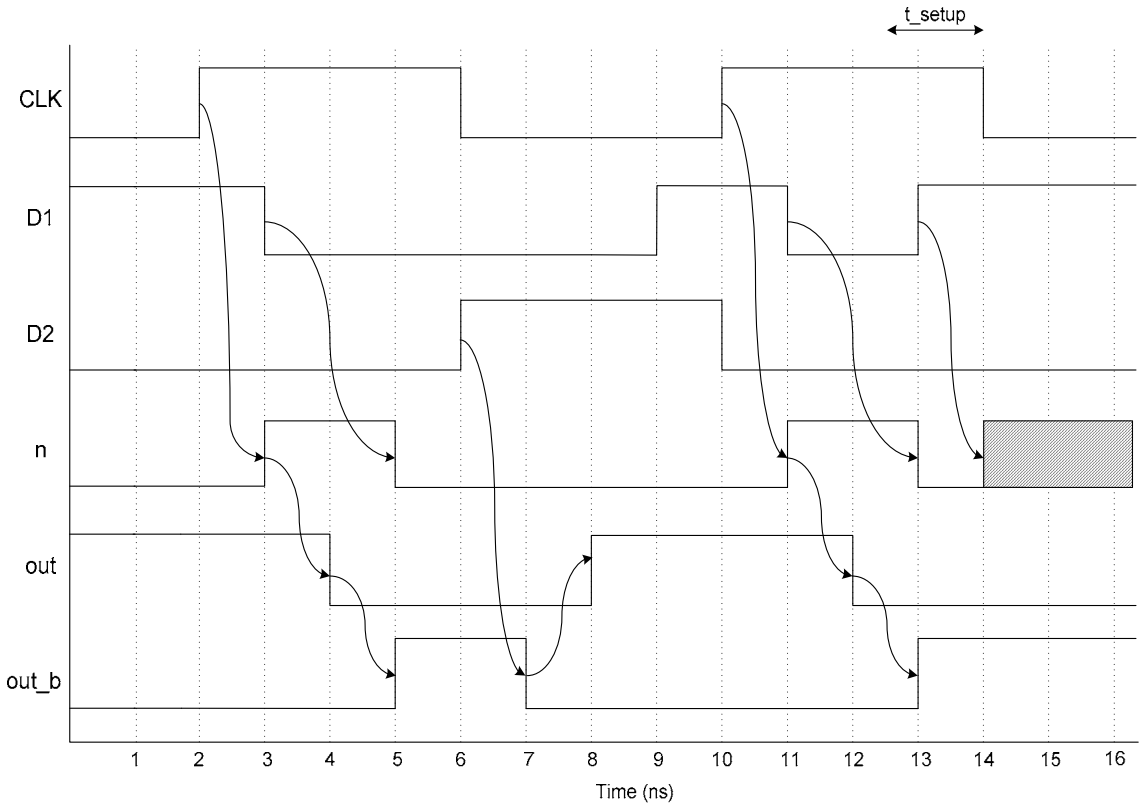
## 3. *Sequential Circuits: 14 Total Points*




Given the circuit above, complete the timing diagram for signals n, out, and out_b. Assume the following values for the D-latch:

$$t_{setup} = t_{hold} = 1.5ns$$
$$t^{CQ}_{HL} = t^{CQ}_{LH} = 1ns$$
$$t^{DQ}_{HL} = t^{DQ}_{LH} = 2ns$$

Also, assume the NOR gates have a rising and falling delay of 1ns. Mark any regions of metastability with ▨




Metastability does not occur on out and out_b because n is the Reset input to the SR-latch, and the latch is already in a reset state at the time when n goes metastable. Therefore, any uncertainty in the reset input will not affect the latch.

## 4. *State Machine Design: 17 Total Points*

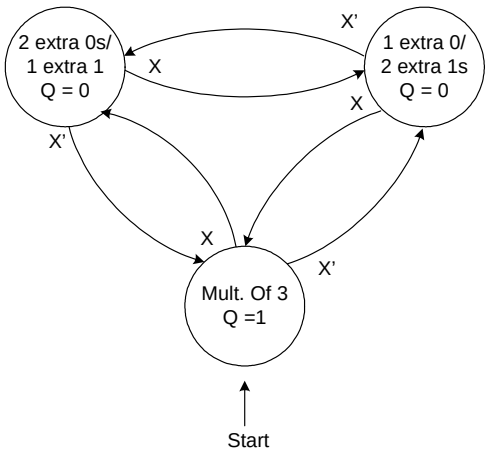
Given a state machine with one input **x** and one output **Q**. Let *num_cyc_1* denote the number of clock cycles where **x** was a one and *num_cyc_0* denote the number of clock cycles where **x** was a zero. The state machine sets **Q** = 1 if (*num_cyc_1-num_cyc_0*) is a multiple of 3, i.e (*num_cyc_1-num_cyc_0*) = …-6, -3, 0, 3, 6… When the state machine starts, num_cyc_0 and *num_cyc_1* are assumed to be zero.

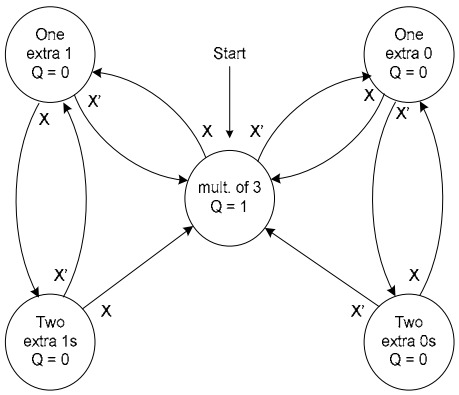
Example sequence:

```
X:            100 001 110 101 110…
Q:            101 001 001 010 100…
num_cyc_0:    001 234 444 556 666…
num_cyc_1:    011 111 234 455 678…
```

Draw the state diagram for this state machine using no more than 5 states. Identify which state your machine should start in. Draw neatly and avoid crossing lines if possible.

This problem can be solved using three states or five:

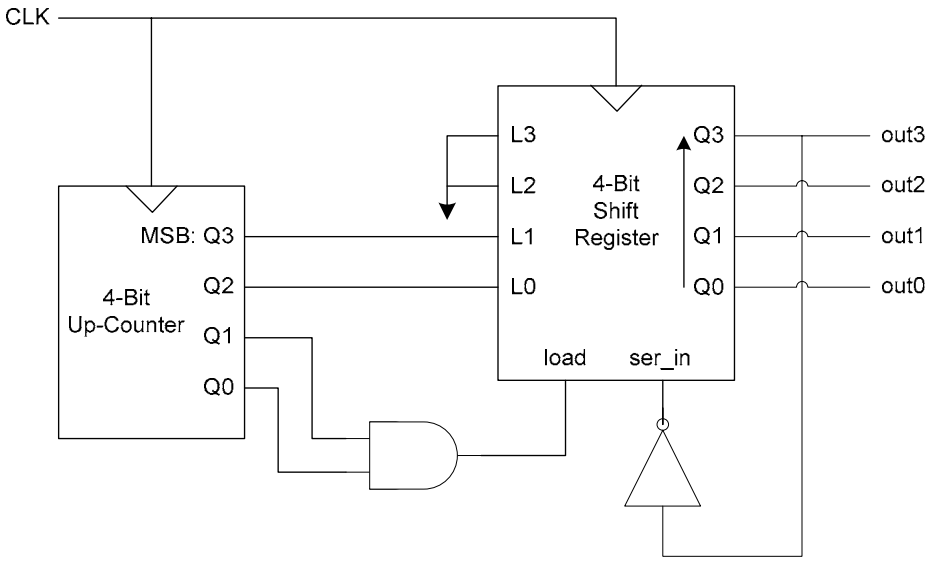


## 5. *Counter/Shifter Design: 15 Total Points*

Consider the design below, constructed with one “up” counter and one right/up shift register. Let outputs out3-out0 be the binary representation of a number *x* between 0 and 15 (out3 is the MSB). What is the repeating sequence of numbers *x* this design will produce?

*Hint: Assume the counter starts in state Q3 – Q0 = 0011*




| | Counter | | | | Shift Register | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | **Q3** | **Q2** | **Q1** | **Q0** | **LD** | **out3** | **out2** | **out1** | **out0** | **ser_in** | ***out3-out0*** |
| | 0 | 0 | 1 | 1 | 1 | X | X | X | X | X | |
| | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | *0* |
| | 0 | 1 | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | *1* |
| | 0 | 1 | 1 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | *3* |
| | 0 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 1 | 1 | *7* |
| | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | *1* |
| | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | *3* |
| | 1 | 0 | 1 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | *7* |
| repeats | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | *15* |
| | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | *2* |
| | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | *5* |
| | 1 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | *11* |
| | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 | 0 | 0 | *6* |
| | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | *3* |
| | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | *7* |
| | 0 | 0 | 1 | 0 | 0 | 1 | 1 | 1 | 1 | 0 | *15* |
| | 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 0 | *14* |
| | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | *0* |

## 6. *Codes: 12 Total Points*

You have designed a system that transmits one of four 5-bit codes A, B, C, D, shown below, across a microwave link.

```
A:  0 1 1 0 0
B:  1 1 0 1 0
C:  1 0 1 0 1
D:  0 0 0 1 1
```

(a: 4 pts) What is the Hamming distance of this code?

3: The minimum number of bit changes to get from any valid code to another is three.

(b: 4 pts) Your manager explains to the customer that the microwave link is very robust and only a single bit in any 5-bit code can be flipped. If the data transmitted by the microwave link is corrupted and you receive the code X, shown below, can the received code be corrected? If yes, what was the transmitted code? If no, what is the set of possible transmitted codes?

X: 11101

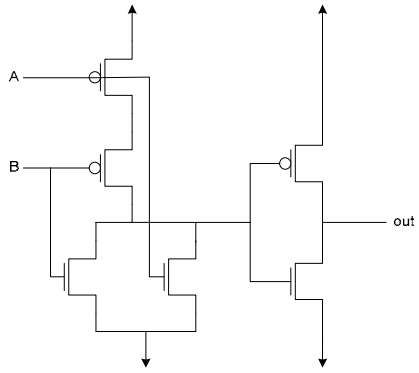
Yes, this code can be corrected if it is known that only a single bit can be corrupted. $n = 2d + c + 1$, where n is the minimum Hamming distance which is 3. Set $d = 0$, then $c = 1$, so we can correct one bit error. The only valid codeword with a distance of one from X is C (10101), so C is the code that must have been transmitted.

(c: 4 pts) After buying your system, your customer finds that the microwave link is not quite as reliable as your manager had claimed. Your manager blames you for the problem and tells you to develop a solution, once again interrupting your latch design project. Upon investigation, you find that the link is such that either no bits are flipped in a transmitted code, or exactly two bits are flipped. Given this new information, can the received code be corrected? If yes, what was the transmitted code? If no, what is the set of possible transmitted codes?

Yes, this code can be corrected if it is known that exactly two bits or no bits are flipped. The only valid codeword with a distance of 2 from X is A (01100), so A is the code that must have been transmitted.

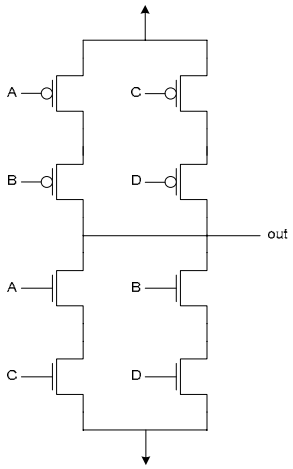
## 7. *CMOS: 10 Total Points*

(a: 5 pts) Given the circuit below, write out the Boolean equation for out as a function of inputs A and B.




out = (A'B')' = ((A + B)')' = A + B

(b: 5 pts) Is the gate shown below a valid CMOS gate? Briefly explain why or why not.




No, this is not a valid CMOS gate. Looking at the input conditions for out to be 1 and 0 shows why:

If (A'B') + (C'D'), out = 1
If AC + BD, out = 0

These two cases are *not* complements of each other. As a consequence of this, under input combinations ABCD = 0110 and 1001, *out* is not connected to Vdd or ground.