Name: **Jonathan Lemarroy**

ECE 3544: Digital Design 1
Homework Assignment 1 (60 points)

You may handwrite your homework. Handwritten solutions should be neat and easy to follow.

When writing a Verilog module or naming a file, replace YOURPID with your Virginia Tech PID. Name the file according to the module name that you use.

For each design that you represent in a Verilog module, copy your source code into the document. For each design that you simulate in ModelSim, include waveforms displaying the correct operation of each module

In addition, submit the .v file containing each *design* **and** each *test bench* that you write. Unless indicated in the problem, *each Verilog design file must have an accompanying test bench*. All your files should contain header information as described in Project 1A and be neatly formatted and commented. Submit your files separately on Canvas. Do not put them into an archive. *Make sure that you upload all your files.*

Helpful Hints for Analyzing and Implementing CMOS Logic Gates

Remember that our approach to implementing CMOS logic gates relies on several principles:

- In a CMOS logic gate, n-type transistors provide switched pathways from GND (as the source) to the output (which is the drain). But these pathways are enabled (switched ON) only when the gate voltage is at logic-1.

- Similarly, p-type transistors provide switched pathways from V_{DD} (as the source) to the output (which is the drain). But these pathways are enabled (switched ON) only when the gate voltage is at logic-0.

- Transistors of the same type in series represent AND connections. For the source value of one transistor to be visible on the drain of the last transistor, **all** transistors must have the appropriate gate voltage for that transistor type. For example, given two n-type transistors in series, the GND value on the source of one transistor will only be visible on the drain of the second transistor if **both** transistors have gate voltages of logic-1. Similarly, given two p-type transistors in series, the V_{DD} value on the source of one transistor will only be visible on the drain of the second transistor if **both** transistors have gate voltages of logic-0.

- Transistors of the same type in parallel represent OR connections. For the common source value of both transistors to be visible on the common drain, **any** transistor can have the appropriate gate voltage for that transistor type. For example, given two n-type transistors in parallel, the GND value on the common source will be visible on the common drain if **either** transistor has a gate voltage of logic-1. Similarly, given two p-type transistors in parallel, the V_{DD} value on the common source will be visible on the common drain if **either** transistor has a gate voltage of logic-0.

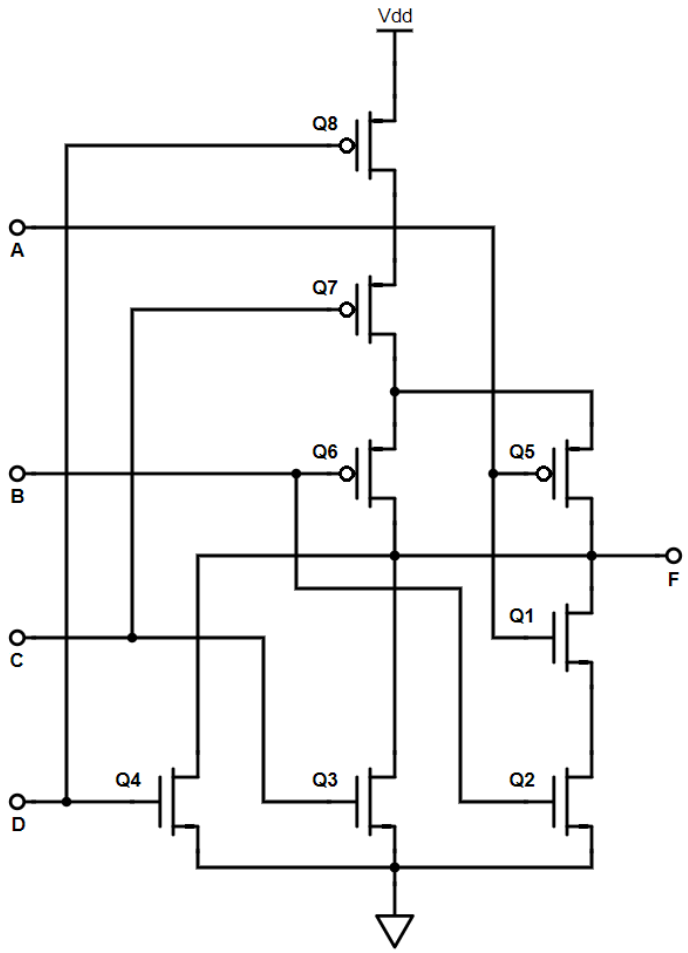
- The two branches of a CMOS gate – the switched pathway from GND to output and the one from V_{DD} to output must be duals of each other. That is, they must be complementary. This ensures that there is never a static pathway from power to ground in the gate. In general, if there are transistors or pathways that are in series in one branch, they will be in parallel in the other branch, and vice versa. Each branch still needs to have the correct transistor type: n-type transistors in the branch that provides the pathway to GND, and p-type transistors in the branch that provides the pathway to V_{DD}.

  For example, the two-input NAND gate F = (AB)′ is one where F = 0 if A = 1 AND B = 1. This implies two n-type transistors in series connecting GND to the gate output. But since F = A′ + B′, the same gate also has two p-type transistors in parallel connecting V_{DD} to the gate output. Review Lecture Slides 2.14 and 2.15 to see the structure of the 2-input NAND gate.

Problem 1 (6 points)

The circuit shown in the diagram below is a type of CMOS AND-OR-INVERT gate. Determine the logic function that it implements. Your solution should justify your answer.

- You could include a function table where you show the status of each transistor (ON or OFF) for all possible combinations of the gate inputs and use that information to determine the logic level of the output.
- Instead of showing a function table, you could provide an explanation of the logic that you used to derive the logic expression from the CMOS gate schematic. This explanation should follow from the CMOS gate principles that you learned in class and that are summarized on the cover page for this assignment.



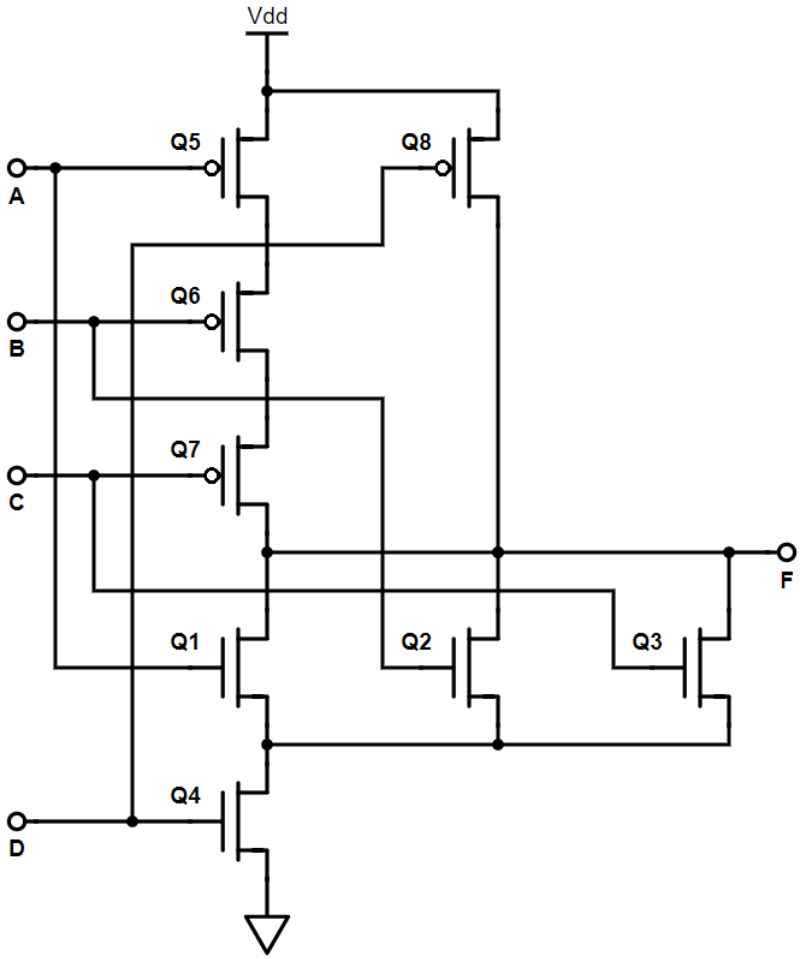
A	B	C	D	F
0	0	0	0	1
0	0	0	1	0
0	0	1	0	0
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	1	1	0	0
0	1	1	1	0
1	0	0	0	1
1	0	0	1	0
1	0	1	0	0
1	0	1	1	0
1	1	0	0	0
1	1	0	1	0
1	1	1	0	0
1	1	1	1	0

Because C and D are connected in series to PMOS transistors they form the expression C'D'. That expression is then connected two more PMOS transistors connected in parallel controlled by A and B forming the expression A' + B'. Therefore, the solution is

F = (A' + B')C'D'.

Problem 2 (6 points)

The circuit shown in the diagram below is a type of CMOS OR-AND-INVERT gate. Determine the logic function that it implements. Your solution should justify your answer. Follow the guidance given in Problem 1.



A	B	C	D	F
0	0	0	0	1
0	0	0	1	1
0	0	1	0	1
0	0	1	1	0
0	1	0	0	1
0	1	0	1	0
0	1	1	0	1
0	1	1	1	0
1	0	0	0	1
1	0	0	1	0
1	0	1	0	1
1	0	1	1	0
1	1	0	0	1
1	1	0	1	0
1	1	1	0	1
1	1	1	1	0

Because D is connected to a PMOS transistor that is in parallel with 3 PMOS transistors in series controlled by A, B, and C. Therefore, the solution is

F = A'B'C' + D'

Problem 3 (8 points)
The Verilog HDL includes primitives that describe the NMOS and PMOS transistors:

```
nmos instance_name(drain, source, gate);
pmos instance_name(drain, source, gate);
```

To investigate the operation of these transistors, reproduce the following test bench in ModelSim.

- The test bench instantiates primitives, so there is no other module that is the design under test.
- DO NOT COPY AND PASTE THE MODULE FROM WORD TO MODELSIM. Word processor applications add characters to text that a text-editor cannot parse.

```
`timescale 1ns/1ns

module tb_cmos();
   reg  source, gate;
   wire nmos_drain, pmos_drain;

   nmos u1(nmos_drain, source, gate);
   pmos u2(pmos_drain, source, gate);

   initial begin
      gate = 1'b0;
      source = 1'b0;
      #100;
      source = 1'b1;
      #100;
      gate = 1'b1;
      source = 1'b0;
      #100;
      source = 1'b1;
      #100;
   end

endmodule
```

Problem 3 (continued)

a. Simulate the test bench and use the waveform to complete these truth tables

Gate	Source	NMOS Drain	PMOS Drain
0	0	z	0
0	1	z	1
1	0	0	z
1	1	1	z

b. Connect the information in the table with what we have learned in class about how NMOS and PMOS transistors operate in CMOS logic by completing the following statements.

To drive the drain of an **NMOS** transistor, the *gate value* **mus**t be logic-____1_____.

However, for that gate value, the transistor is in *saturation* when the *source value* is logic- ____1_____.

To drive the drain of a **PMOS** transistor, the *gate value* **must** be logic-____0_____.

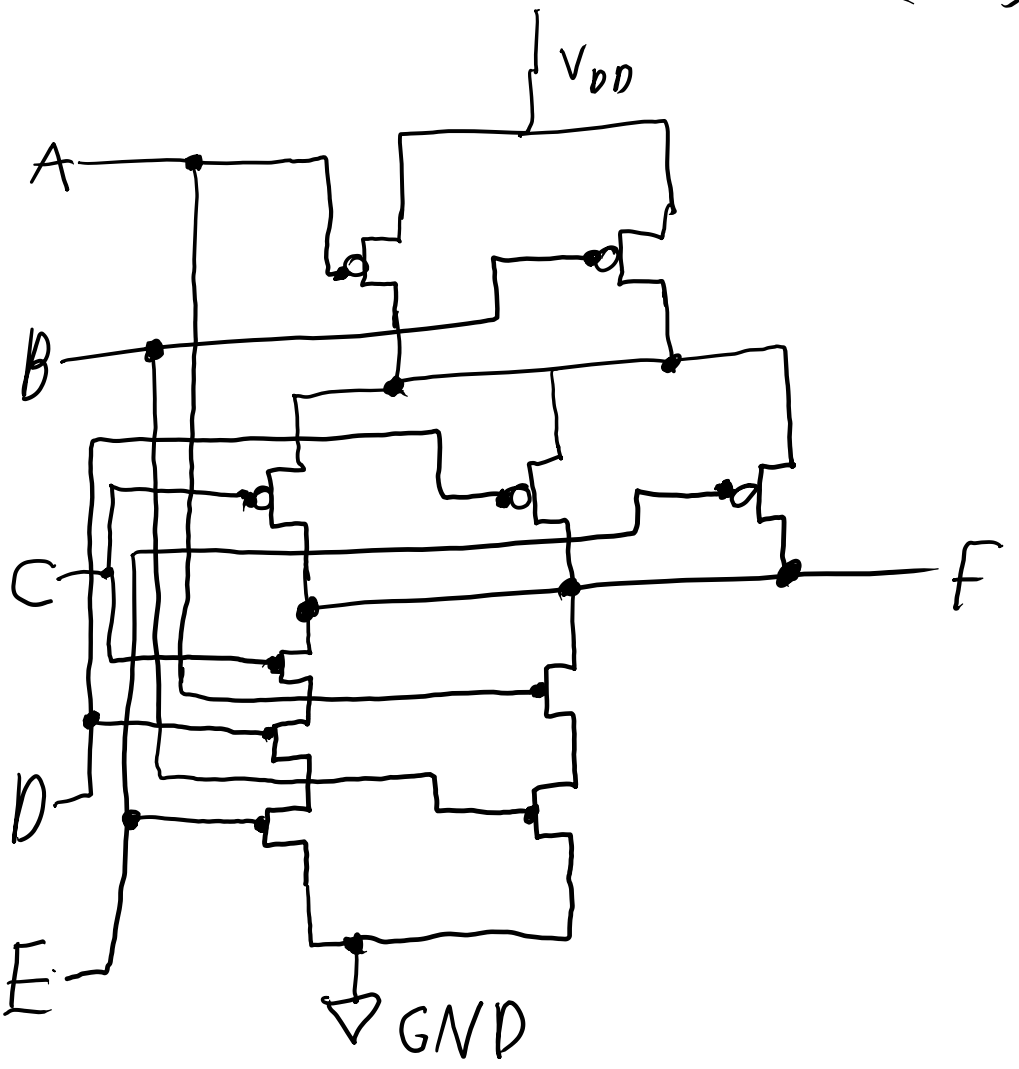
However, for that gate value, the transistor is in *saturation* when the *source value* is logic- ___1_____.

Problem 4 (16 points)

a. (6 points) Draw the CMOS transistor schematic for a "gate" that implements the Boolean function. Your implementation must be transistor-efficient.

$F = \overline{AB + CDE}$

$(\bar{A} + \bar{B})(\bar{C} + \bar{D} + \bar{E})$

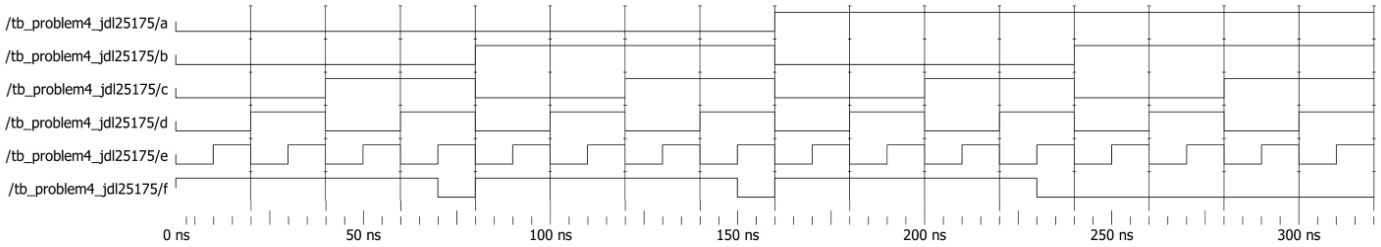


Problem 4 (continued)

b. (10 points) Using ModelSim, write a Verilog model using NMOS and PMOS transistors for the gate in part (a). Use the following module and port declaration to describe the gate. You may add wires as needed.

```
module problem4_YOURPID(a, b, c, d, e, f);
   input  a, b, c, d, e;
   output f;
```

Write a test-bench to test the gate for all input cases. Copy your source and test-bench code into this document. Include a screenshot of your waveform. In addition, submit your Verilog files along with your homework document.




```
////////////////////////////////////////////////////////////////////////////////////////////////////
// Filename:   problem4_jdl25175.v
// Author:     Jonathan Lemarroy
// Date:       07 September 2020
// Version:    1
// Description: This file implements the Boolean expression F = ~((AB) + (CDE))
////////////////////////////////////////////////////////////////////////////////////////////////////

module problem4_jdl25175(a, b, c, d, e, f);

   input  a, b, c, d, e;
   output f;

   wire ow1, aw1, aw2, aw3, aw4;

   pmos pt1(ow1, 1'b1, a), pt2(ow1, 1'b1, b), pt3(f, ow1, c), pt4(f, ow1, d), pt5(f, ow1, e);
   nmos nt1(f, aw1, c), nt2(aw1, aw2, d), nt3(aw2, 1'b0, e), nt4(f, aw4, a), nt5(aw4, 1'b0, b);

endmodule
```

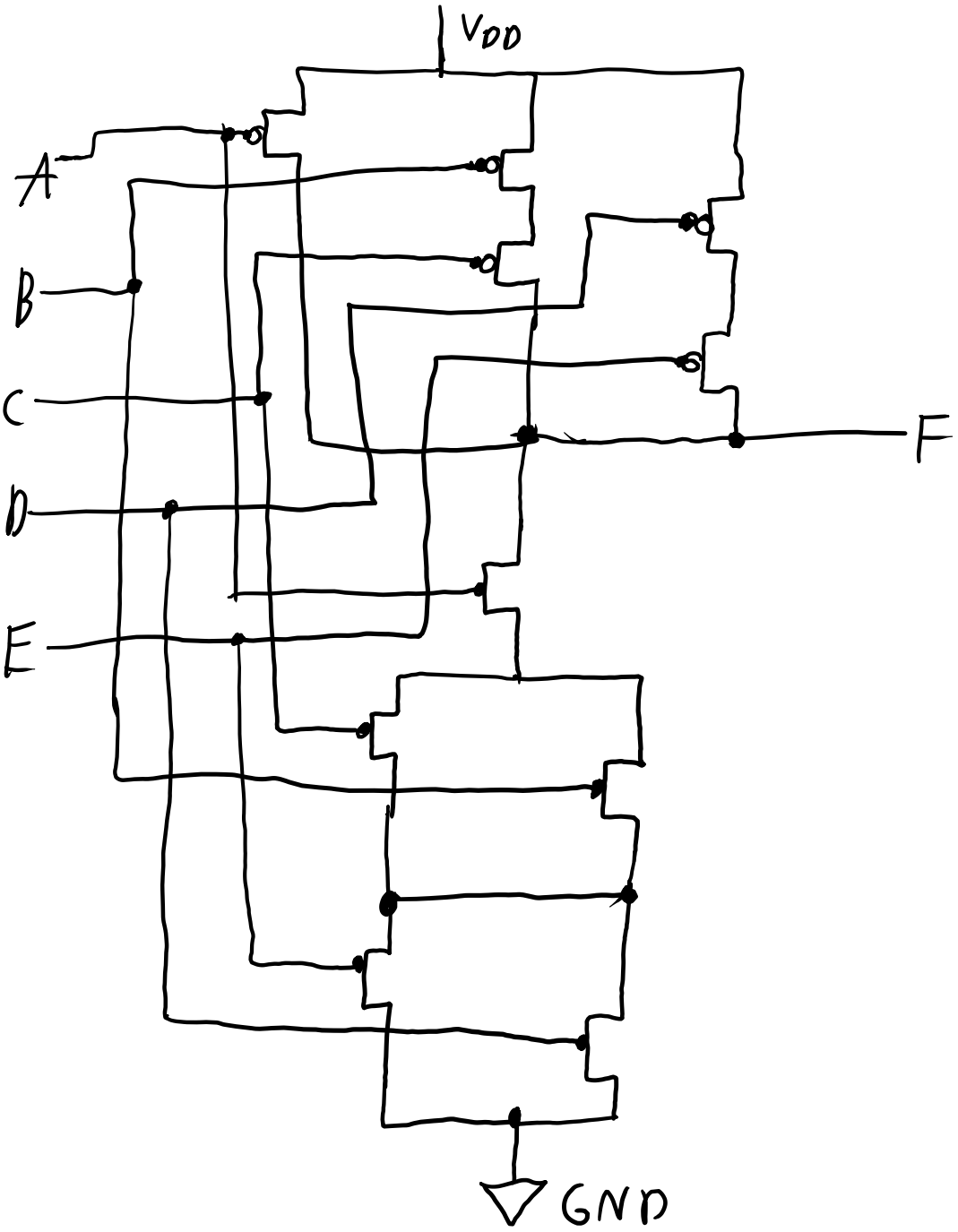
```
////////////////////////////////////////////////////////////////////////////////////////////////
// Filename:    tb_problem4_jdl25175.v
// Author:      Jonathan Lemarroy
// Date:        07 September 2020
// Version:     1
// Description: This file tests the problem4_jdl25175 module by creating the truth table for ABCDEF
////////////////////////////////////////////////////////////////////////////////////////////////

`timescale 1ns/1ns

module tb_problem4_jdl25175();
   reg a, b, c, d, e;
   wire f;
   problem4_jdl25175 mod1(a, b, c, d, e, f);

   //ABCDE
   initial begin
      a = 0'b0; //00000
      b = 0'b0;
      c = 0'b0;
      d = 0'b0;
      e = 0'b0;
      #10;
      e = 0'b1; //00001
      #10;
      d = 0'b1; //00010
      e = 0'b0;
      #10;
      e = 0'b1; //00011
      #10;
      c = 0'b1; //00100
      d = 0'b0;
      e = 0'b0;
      #10;
      e = 0'b1; //00101
      #10;
      d = 0'b1; //00110
      e = 0'b0;
      #10;
      e = 0'b1; //00111
      #10
      b = 0'b1; //01000
      c = 0'b0;
      d = 0'b0;
      e = 0'b0;
      #10;
      e = 0'b1; //01001
      #10;
      d = 0'b1; //01010
      e = 0'b0;
      #10;
      e = 0'b1; //01011
      #10;
      c = 0'b1; //01100
      d = 0'b0;
      e = 0'b0;
      #10;
      e = 0'b1; //01101
```

```
        #10;
        d = 0'b1; //01110
        e = 0'b0;
        #10;
        e = 0'b1; //01111
        #10;
        a = 0'b1; //10000
        b = 0'b0;
        c = 0'b0;
        d = 0'b0;
        e = 0'b0;
        #10;
        e = 0'b1; //10001
        #10;
        d = 0'b1; //10010
        e = 0'b0;
        #10;
        e = 0'b1; //10011
        #10;
        c = 0'b1; //10100
        d = 0'b0;
        e = 0'b0;
        #10;
        e = 0'b1; //10101
        #10;
        d = 0'b1; //10110
        e = 0'b0;
        #10;
        e = 0'b1; //10111
        #10
        b = 0'b1; //11000
        c = 0'b0;
        d = 0'b0;
        e = 0'b0;
        #10;
        e = 0'b1; //11001
        #10;
        d = 0'b1; //11010
        e = 0'b0;
        #10;
        e = 0'b1; //11011
        #10;
        c = 0'b1; //11100
        d = 0'b0;
        e = 0'b0;
        #10;
        e = 0'b1; //11101
        #10;
        d = 0'b1; //11110
        e = 0'b0;
        #10;
        e = 0'b1; //11111
        #10;
    end
endmodule
```

Problem 5 (16 points)

a. (6 points) Draw the CMOS transistor schematic for a "gate" that implements the Boolean function. Your implementation must be transistor-efficient.

$$F = \overline{A(B+C)(D+E)} \qquad \bar{A} + \bar{B}\bar{C} + \bar{D}\bar{E}$$

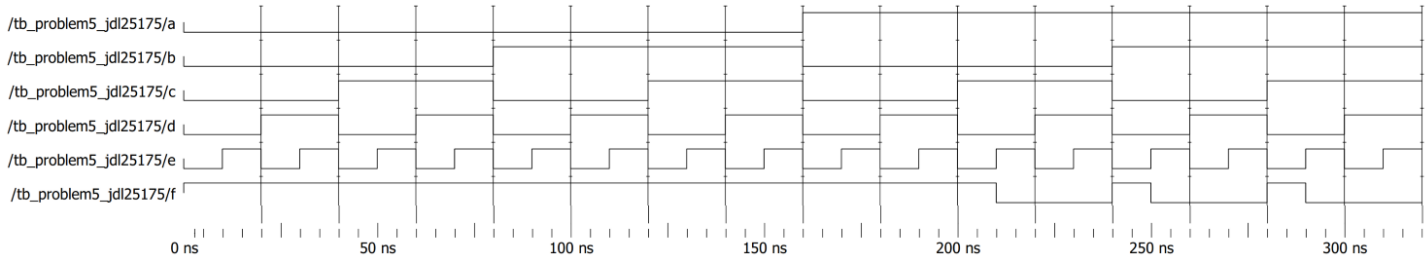


Problem 5 (continued)

b. (10 points) Using ModelSim, write a Verilog model using NMOS and PMOS transistors for the gate in Part (a). Use the following module and port declaration to describe the gate. You may add wires as needed.

```
module problem5_YOURPID(a, b, c, d, e, f);
   input  a, b, c, d, e;
   output f;
```

Write a test-bench to test the gate for all input cases. Copy your source and test-bench code into this document. Include a screenshot of your waveform. In addition, submit your Verilog files along with your homework document.




```
////////////////////////////////////////////////////////////////////////////////////////////////////
// Filename:    problem5_jdl25175.v
// Author:      Jonathan Lemarroy
// Date:        07 September 2020
// Version:     1
// Description: This file implements the Boolean expression F = ~(A(B + C)(D + E))
////////////////////////////////////////////////////////////////////////////////////////////////////

module problem5_jdl25175(a, b, c, d, e, f);

   input  a, b, c, d, e;
   output f;

   wire ow1, ow2, ow3, aw1, aw2;

   pmos pt1(f, 1'b1, a), pt2(aw1, 1'b1, b), pt3(f, aw1, c), pt4(aw2, 1'b1, d), pt5(f, aw2, e);
   nmos nt1(f, ow1, a), nt2(ow1, ow2, b), nt3(ow1, ow2, c), nt4(ow2, 1'b0, d), nt5(ow2, 1'b0, e);

endmodule
```

```
////////////////////////////////////////////////////////////////////////////////////////////////
// Filename:    tb_problem5_jdl25175.v
// Author:      Jonathan Lemarroy
// Date:        07 September 2020
// Version:     1
// Description: This file tests the problem5_jdl25175 module by creating the truth table for ABCDEF
////////////////////////////////////////////////////////////////////////////////////////////////
`timescale 1ns/1ns
module tb_problem5_jdl25175();
   reg a, b, c, d, e;
   wire f;
   problem5_jdl25175 mod1(a, b, c, d, e, f);
   //ABCDE
   initial begin
      a = 0'b0; //00000
      b = 0'b0;
      c = 0'b0;
      d = 0'b0;
      e = 0'b0;
      #10;
      e = 0'b1; //00001
      #10;
      d = 0'b1; //00010
      e = 0'b0;
      #10;
      e = 0'b1; //00011
      #10;
      c = 0'b1; //00100
      d = 0'b0;
      e = 0'b0;
      #10;
      e = 0'b1; //00101
      #10;
      d = 0'b1; //00110
      e = 0'b0;
      #10;
      e = 0'b1; //00111
      #10
      b = 0'b1; //01000
      c = 0'b0;
      d = 0'b0;
      e = 0'b0;
      #10;
      e = 0'b1; //01001
      #10;
      d = 0'b1; //01010
      e = 0'b0;
      #10;
      e = 0'b1; //01011
      #10;
      c = 0'b1; //01100
      d = 0'b0;
      e = 0'b0;
      #10;
      e = 0'b1; //01101
      #10;
      d = 0'b1; //01110
      e = 0'b0;
      #10;
      e = 0'b1; //01111
```

```
    #10;
    a = 0'b1; //10000
    b = 0'b0;
    c = 0'b0;
    d = 0'b0;
    e = 0'b0;
    #10;
    e = 0'b1; //10001
    #10;
    d = 0'b1; //10010
    e = 0'b0;
    #10;
    e = 0'b1; //10011
    #10;
    c = 0'b1; //10100
    d = 0'b0;
    e = 0'b0;
    #10;
    e = 0'b1; //10101
    #10;
    d = 0'b1; //10110
    e = 0'b0;
    #10;
    e = 0'b1; //10111
    #10
    b = 0'b1; //11000
    c = 0'b0;
    d = 0'b0;
    e = 0'b0;
    #10;
    e = 0'b1; //11001
    #10;
    d = 0'b1; //11010
    e = 0'b0;
    #10;
    e = 0'b1; //11011
    #10;
    c = 0'b1; //11100
    d = 0'b0;
    e = 0'b0;
    #10;
    e = 0'b1; //11101
    #10;
    d = 0'b1; //11110
    e = 0'b0;
    #10;
    e = 0'b1; //11111
    #10;
  end
endmodule
```

Problem 6 (8 points)

A CMOS gate operates at a switching frequency f_1, employs a load capacitance C_1, and switches the load through a voltage V_1. It therefore consumes a calculable amount of dynamic power, which we will call P.

A second CMOS gate operates at a switching frequency that is 30 percent higher than that of the first CMOS gate – that is, it has a switching frequency $f_2 = 1.3 * f_1$. However, the two CMOS gates consume the *same amount of dynamic power*.

Call the load capacitance of the second CMOS gate C_2. Call the voltage through which the second CMOS gate is switched V_2.

a. Suppose that $V_2 = V_1$. By what factor does C_2 relate to C_1?

$$C_1 V_1^2 \cdot f_1 = C_2 V_1^2 (1.3 f_1)$$

$$C_2 = C_1 / 1.3$$

b. Now suppose that $C_2 = C_1$. By what factor does V_2 relate to V_1?

$$C_1 V_1^2 f_1 = C_1 V_2^2 (1.3 f_1)$$

$$V_2 = \sqrt{\frac{1}{1.3}} \cdot V_1$$