Solutions PDF Generated from: solutions-openai-generated/midterm-exam/midterm-exam-solutions-set-03.json

Question A

A diagram of a dual in-line package (DIP) module containing 4 NOR gates is shown in the figure, it is similar to modules available in lab 1 project. Describe how would you connect the pins of the module to realize the Boolean expression Y = AC + BC. Do this by drawing the module and the connections (wires) of your solution. Since the module only has NOR gates, you need to convert the expression for Y into equivalent one using only NOR gates. Then, for each pin (1 to 14) of the module, you should show if its connected to A, B, C, Y, Ground, VDD, or if the pin is connected to another pin of the module, or if the pin not connected. You can use any number of NOR gates, not all NOR gates of the module need to be used, if some NOR gates are unused label the pins as unconnected.

```
Answer:

Below is a diagram of the DIP module containing 4 NOR gates and the connections of the pins to
    realize the Boolean expression Y = AC + BC:

Pin Number |  1  |  2  |  3  |  4  |  5  |  6  |  7  |  8  |  9  | 10  | 11  | 12  | 13  | 14
 Connection | Y | B |A    |B'   |C'   |C    |Ground| VDD | - | -   | -   | -   | -   | -


The expression Y = AC +    BC    has first been converted to its equivalent in NOR gates: Y = (A +
    B)(C + C). As shown in the diagram, pins 1 and 2 are connected to B and B respectively, pins 3
    and 4 are connected to A and A respectively, and pins 5 and 6 are connected to C and C
    respectively. Pins 7 and 8 are connected to Ground and VDD respectively. Finally, pins 1 and 3
    are connected to each other (i.e., Y is connected to A + B) and pins 5 and 4 are connected to
    each other (i.e., C is connected to C). All other pins are not connected.
```

Question B

Draw a transistor level diagram of CMOS AND gate. Assume you only have two 1-bit inputs A and B; draw transistor diagrams of inverters if you need to generate A or B.

```
![Diagram of CMOS AND Gate](https://drive.google.com/uc?export=view&id=1LCnhLi3tJ_kd2a22MUhgYzNzbA
    -30oKZ)
```

Question C, Part 1

Draw a truth table for the expression Y = ACD + ABD + ABC + BCD + ACD.

```
Truth Table for Y =   ACD   +  ABD   +  ABC   + BCD + ACD

A | B | C | D | Y
```

```
--|---|---|---|--
0 | 0 | 0 | 0 | 0
0 | 0 | 0 | 1 | 0
0 | 0 | 1 | 0 | 0
0 | 0 | 1 | 1 | 1
0 | 1 | 0 | 0 | 0
0 | 1 | 0 | 1 | 0
0 | 1 | 1 | 0 | 1
0 | 1 | 1 | 1 | 1
1 | 0 | 0 | 0 | 0
1 | 0 | 0 | 1 | 1
1 | 0 | 1 | 0 | 1
1 | 0 | 1 | 1 | 1
1 | 1 | 0 | 0 | 1
1 | 1 | 0 | 1 | 1
1 | 1 | 1 | 0 | 1
1 | 1 | 1 | 1 | 1
```

Question C, Part 2

Draw a K-Map and find minimal sum-of-products expression for Y = ACD + ABD + ABC + BCD + ACD.

```
Y = (A + B'D + C) * (A'C' + B' + B'C + C'D + CD)

 K -Map:

Y =   AC   +  BD  +  BC  + CD
```

Question D

The figure shows the truth table for the 7-segment decoder. Write Verilog code to realize the 7-segment decoder. The inputs are D, C, B, A, and outputs are a, b, c, d, e, f, g. For any inputs not shown in the table, the output should be 0 for all output signals. Remember Verilog is case sensitive. You are encouraged to use the case statement. Also, you may want to use to concatenate signals, as well as you may want to define new internal wires used for the concatenated inputs and outputs.

```
module SevenSegment ( input A,B,C,D, output a,b,c,d,e,f,g);
       wire [3:0] ABCD;
        >
       assign {ABCD[3],ABCD[2],ABCD[1],ABCD[0]} = {A,B,C,D};

       always @ (ABCD)
       begin
              case (ABCD)

                     4'b0000 : {a,b,c,d,e,f,g} = 7'b1000000;
                     4'b0001 : {a,b,c,d,e,f,g} = 7'b1111001;
                     4'b0010 : {a,b,c,d,e,f,g} = 7'b0100100;
                     4'b0011 : {a,b,c,d,e,f,g} = 7'b0110000;
                     4'b0100 : {a,b,c,d,e,f,g} = 7'b0011001;
                     4'b0101 : {a,b,c,d,e,f,g} = 7'b0010010;
                     4'b0110 : {a,b,c,d,e,f,g} = 7'b0000010;
                     4'b0111 : {a,b,c,d,e,f,g} = 7'b1111000;
                     4'b1000 : {a,b,c,d,e,f,g} = 7'b0000000;
                     4'b1001 : {a,b,c,d,e,f,g} = 7'b0010000;
                     4'b1010 : {a,b,c,d,e,f,g} = 7'b0001000;
```

```
                    4'b1011 : {a,b,c,d,e,f,g} = 7'b0000011;
                    4'b1100 : {a,b,c,d,e,f,g} = 7'b1000110;
                    4'b1101 : {a,b,c,d,e,f,g} = 7'b0100001;
                    4'b1110 : {a,b,c,d,e,f,g} = 7'b0000110;
                    4'b1111 : {a,b,c,d,e,f,g} = 7'b0001110;
                    default: {a,b,c,d,e,f,g} = 7'b0000000;

             endcase
      end

endmodule
```

Question E, Part 1

Recall that a Fibonacci Sequence is: 1, 1, 2, 3, 5, ... I.e. two prior numbers are added to get the next number. Draw a state diagram for a Moore finite state machine which outputs only the first 3 Fibonacci numbers. The finite state machine should start in an idle state where it outputs 0. Assume that there is a start signal named S that is 1 when the output sequence should be started. When S is 1, finite state machine should exit the idle state and output one Fibonacci number per cycle (i.e. output 1, then next 1, then 2). When last number is output it should go back to the idle state. S is a dont care value in any state other than the idle state. The state machine should output 0 in the initial idle state, and any time it goes back to the idle state. If machine enters idle state, it should stay there until S becomes 1 again.

```
State Diagram:

State: Idle
Input: S
Output: 0

State: 0
Input: S
Output: 1

State: 1
Input: S
Output: 1

State: 2
Input: S
Output: 2

State: Finish
Input: S
Output: 0
```

Question E, Part 2

Select binary encoding for the states of a Moore finite state machine which outputs only the first 3 Fibonacci numbers. Draw the next-state and output tables based on your encoding; note the output is the Fibonacci number, so the output needs to be more than one bit.

```
Let the binary encoding of the states be:

00 - S (start)
01 - A
10 - B
```

```
11 - C

Next-State Table:

S       A       B       C
00      01      00      00
01      10      01      01
10      11      10      10
11      00      11      11

Output Table:

A       B       C
0       1       1
1       1       2
2       3       5
```

Question E, Part 3

Use K-Maps to find the minimal sum-of-products form equations for the next-state logic and the output logic of a Moore finite state machine which outputs only the first 3 Fibonacci numbers.

```
Next-State Logic:
Q2'= D'Q0 + DQ1
Q1'= D'Q0 + DQ2
Q0'= DQ2

Output Logic:
S1= Q2
S2= Q1
S3= Q0
```

Question E, Part 4

Determine the propagation delays and contamination delays for the next state logic and output logic of a Moore finite state machine which outputs only the first 3 Fibonacci numbers. Use the gate delays table shown in the figure.

```
Propagation Delay

Next State Logic: NAND gate (45ns) + NAND gate (45ns) + Long Inverter (50ns) = 140 ns

Output Logic: NAND gate (45ns) + NAND gate (45ns) + Long Inverter (50ns) = 140 ns

Contamination Delay

Next State Logic: NAND gate (2ns) + NAND gate (2ns) + Long Inverter (0ns) = 4 ns

Output Logic: NAND gate (2ns) + NAND gate (2ns) + Long Inverter (0ns) = 4 ns
```

Question E, Part 5

Assume each flip-flop has a setup time of 30 ps, a hold time of 10 ps, a clock-to-Q maximum delay of 35 ps, and a clock-to-Q minimum delay of 25 ps. What is the maximum operating clock frequency of the Moore finite state machine which outputs only the first 3 Fibonacci numbers equivalently what is the minimum clock period?

```
The minimum clock period is 85 ps. The maximum operating clock frequency is 11.76 MHz.
```

Execution Time

```
0:01:02.373187
```

OpenAI Parameters

```
Model: text-davinci-003, Max. Tokens: 1024, Temperature: 1, N: 1
```