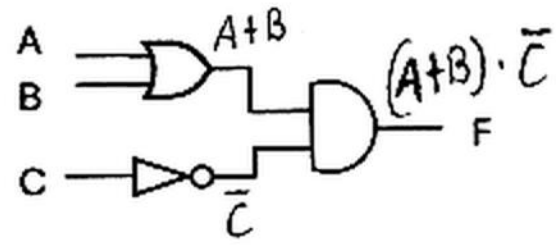
CSE369 Practice Problems (lec 3)

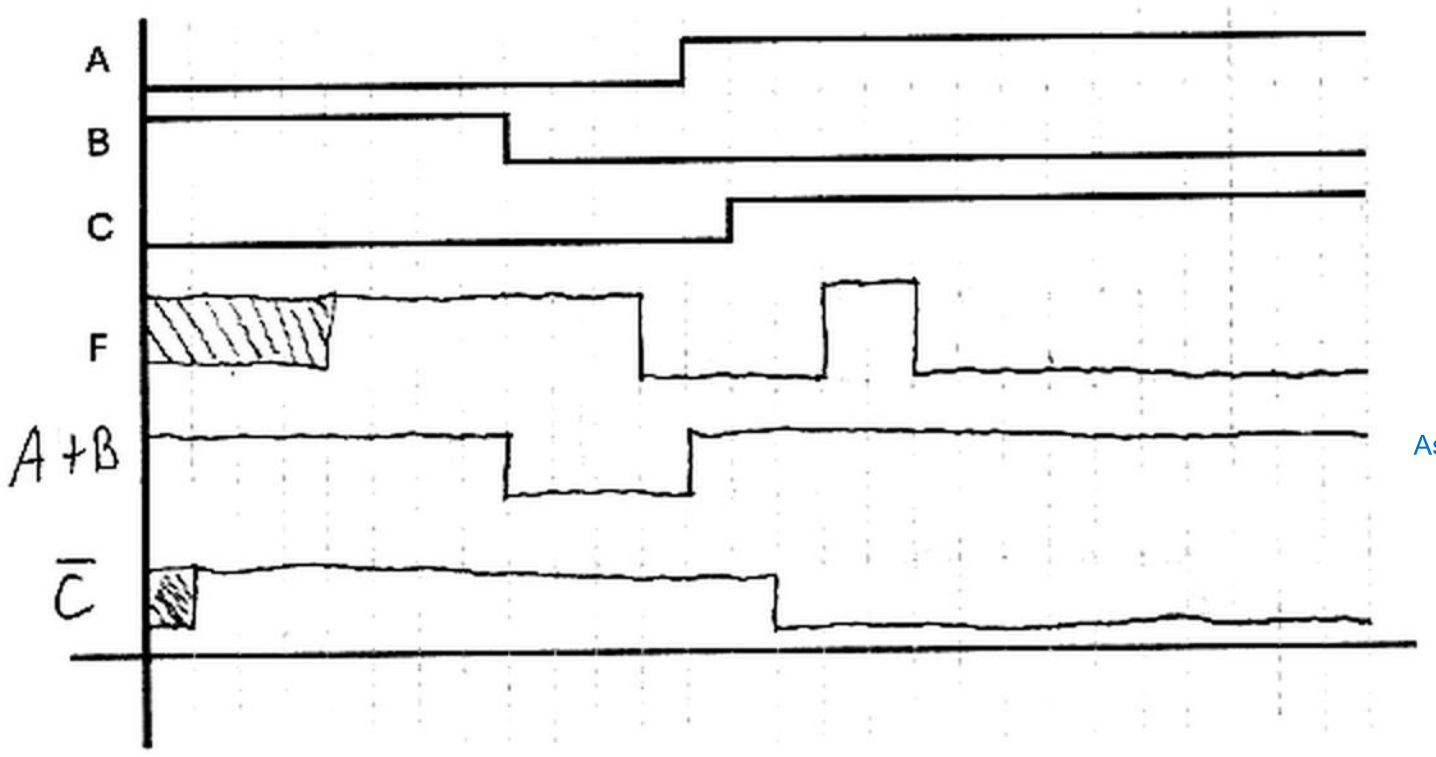
1. Waves

Consider the following circuit:



1.1] Label the intermediate nodes between gates with their algebraic representation

1.2] Complete the following waveform diagram. Include the intermediate signals you labelled above (start with those). Assume AND takes 3 time units and NOT takes 1 time unit



Assuming OR takes 0 delay

1.3] How long after power-up does it take for F to output a known value?

4 time units

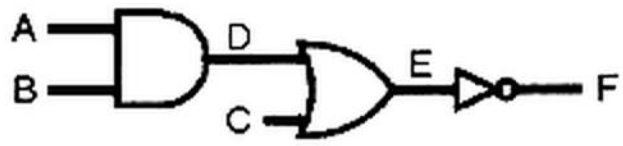
1.4] Does the waveform above display any hazards? If so, which kind?

Yes, there is a momentary wiggle in the output of F between 15 and 17 time units. It is a static zero.

A static zero is a type of hazard where the output jumps to 1 when it is supposed to stay at zero

2. More Waves

Consider the following circuit. Assume all gates have a delay of 3 time ticks.



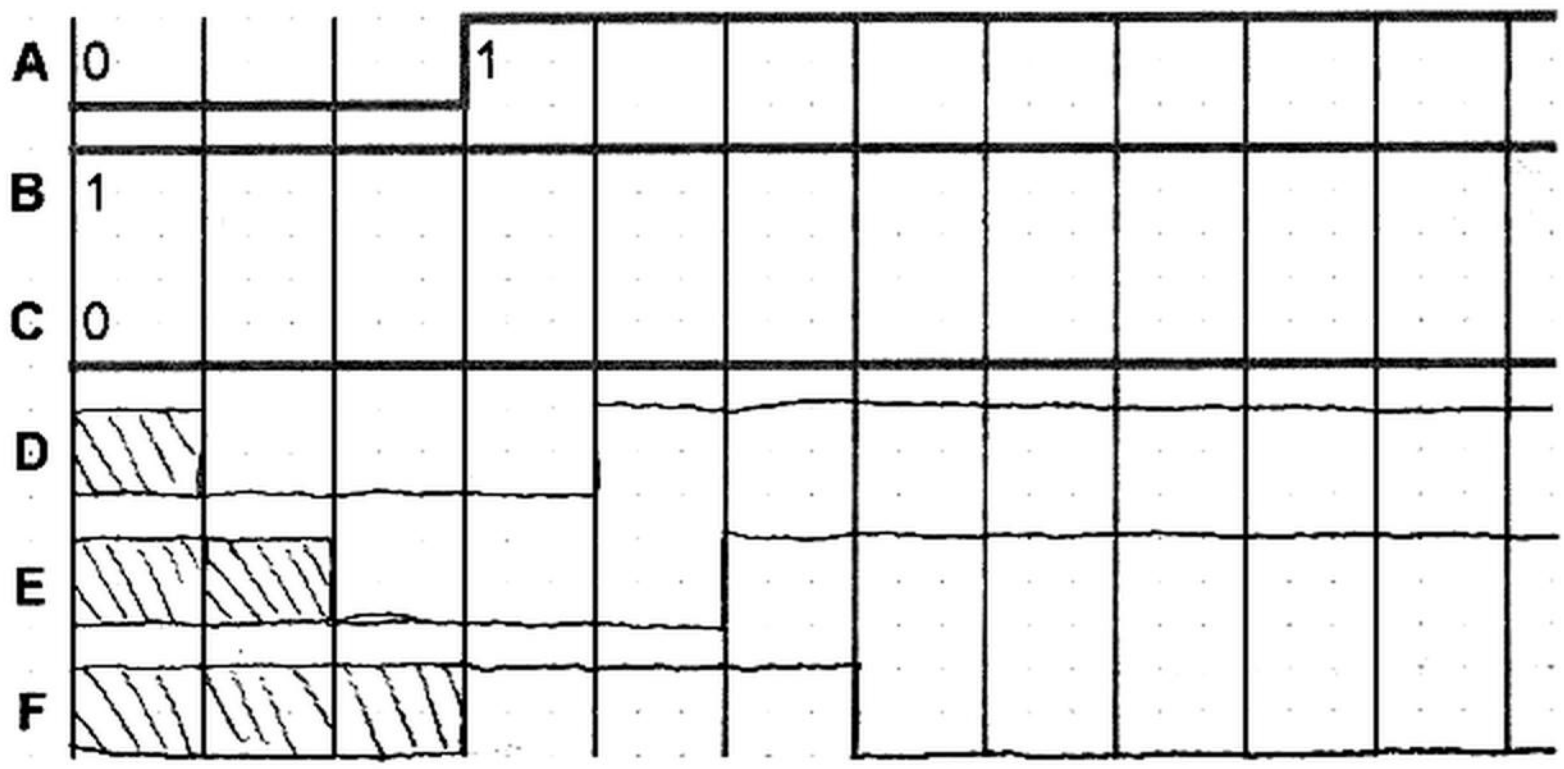
2.1] Label the intermediate nodes between gates with their algebraic representation

$D = AB$

$E = AB + C$

$F = \overline{(AB + C)}$

2.2] Complete the waveform diagram below assuming the following inputs:



2.3] How long after power-up does it take for F to output a known value?

9 time units (one delay per gate)

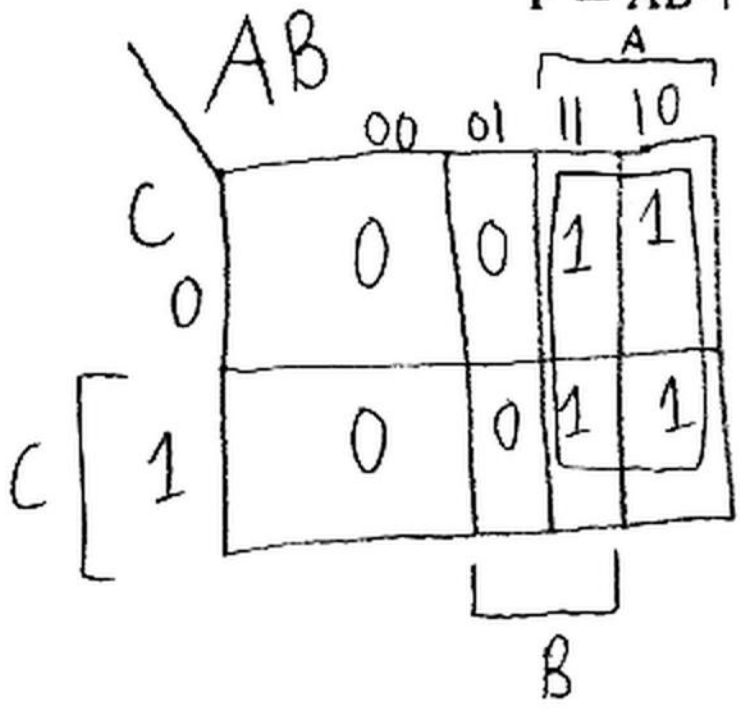
2.4] Does the waveform above display any hazards? If so, which kind?

NO, there are no glitches in the output.

3. "Pure" K-Maps

3.1] Find a simplified equation for **F** using a k-map:

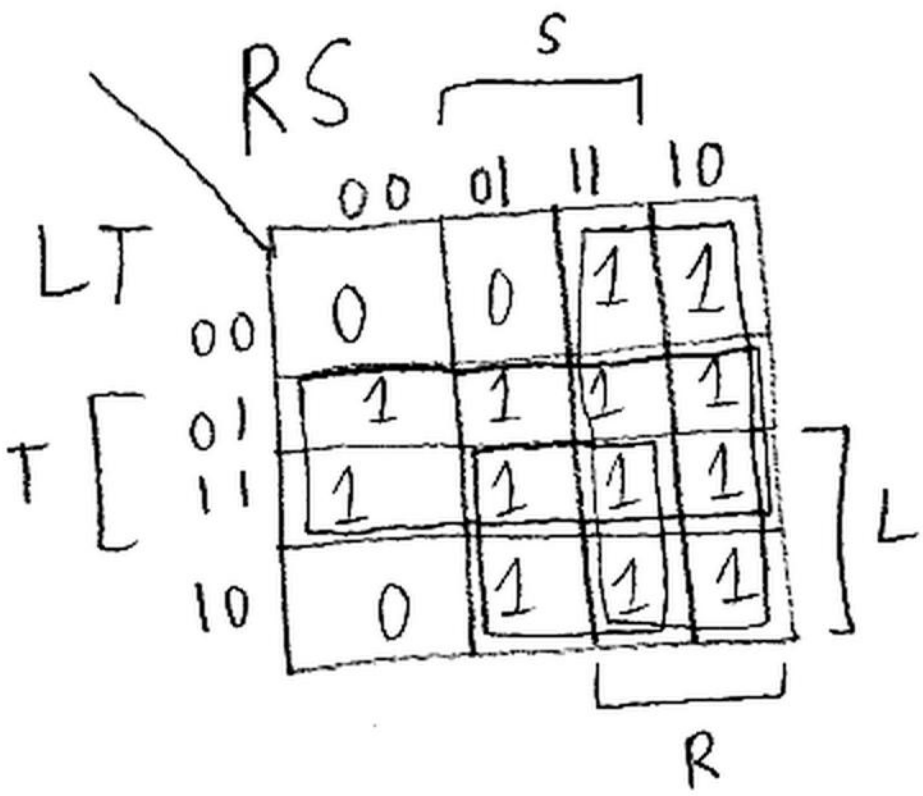
$$F = AB + A\overline{BC} + A\overline{B}C + AC$$



$F = A$

3.2] Find a simplified equation for **G** using a k-map:

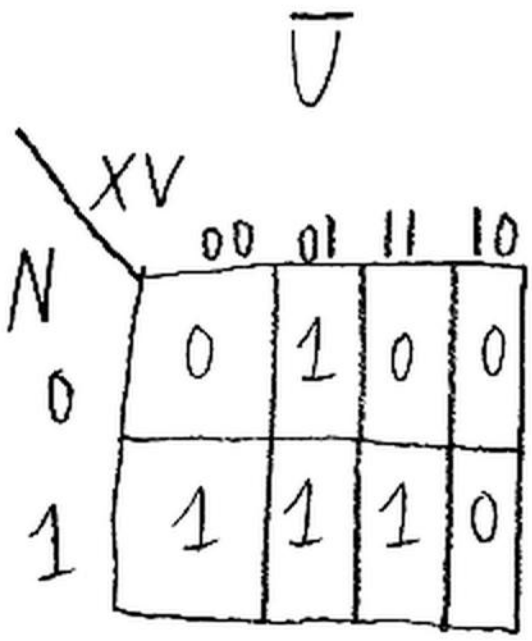
$$G = RS + LS + \overline{R}LS + R\overline{S} + T$$

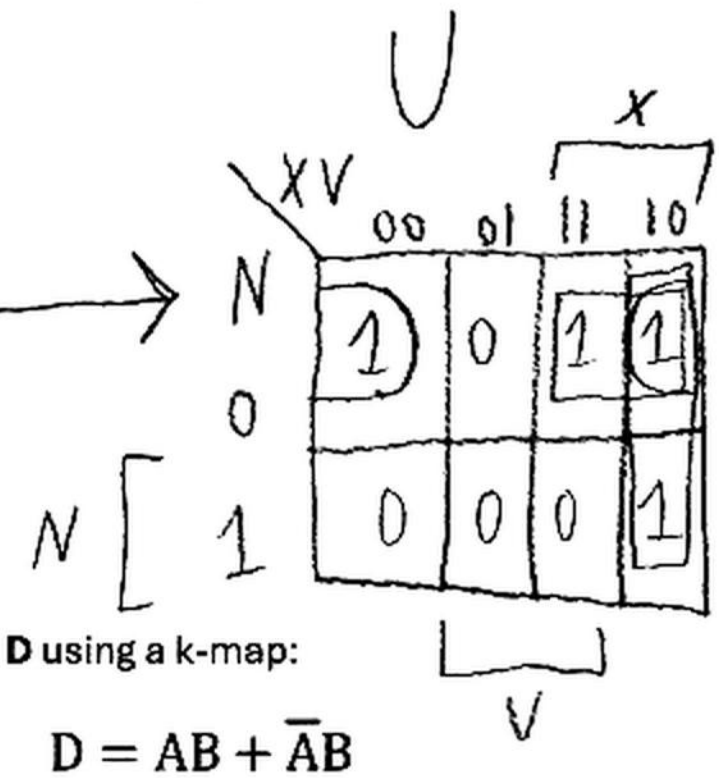


$G = R + T + LS$

3.3] Find a simplified equation for **U** using a k-map:

$$\overline{U} = (V + \overline{X})(X + V + N)(\overline{X} + \overline{V} + N)$$

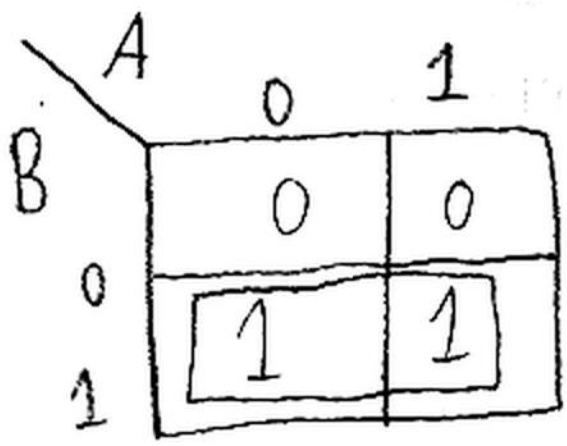




$$U = X\overline{N} + X\overline{V} + \overline{N}\overline{V}$$

3.4] Find a simplified equation for **D** using a k-map:

$$D = AB + \overline{A}B$$



$$D = B$$

4. "Design" K-Maps

Let's implement the circuit to judge a game of "Rocks Paper Scissors," where two players (player 0 and player 1) simultaneously choose one of three battle poses: rock (00), paper (01), or scissors (10). Rock beats paper, paper beats scissors, scissors beat rock. If players choose the same pose they tie rather than win or lose.

Our circuit is going to receive the two players' poses and calculate two outputs: "winner" and "tie." If the game is tied, we don't care who the winner is. If a player chooses an invalid pose, we also don't care what the circuit's output is.

Complete the truth table for our circuit, where {A,B} encodes player 1's pose and {C, D} encodes player 2's pose. Then, use the truth table to fill in a Karnaugh maps for "Win" and a Karnaugh map for "Tie". Finally, fill in the minimal two-level equations for each output at the bottom.

P1		P0			
A	B	C	D	Win	Tie
0	0	0	0	X	1
0	0	0	1	0	0
0	0	1	0	1	0
0	0	1	1	X	X
0	1	0	0	1	0
0	1	0	1	X	1
0	1	1	0	0	0
0	1	1	1	X	X
1	0	0	0	0	0
1	0	0	1	1	0
1	0	1	0	X	1
1	0	1	1	X	X
1	1	0	0	X	X
1	1	0	1	X	X
1	1	1	0	X	X
1	1	1	1	X	X

Win

CD \ AB	00	01	11	10
00	X	1	X	0
01	0	X	X	1
11	X	X	X	X
10	1	0	X	X

Tie

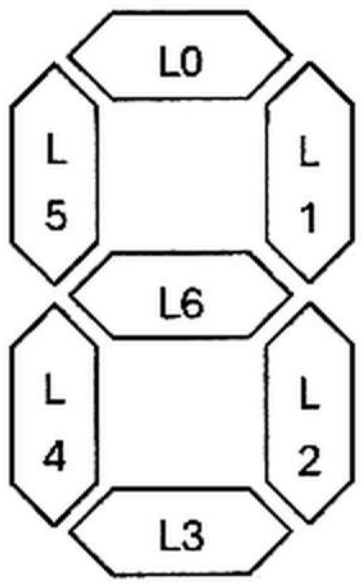
CD \ AB	00	01	11	10
00	1	0	X	0
01	0	1	X	0
11	X	X	X	X
10	0	0	X	1

Win = $B\bar{C} + AD + \bar{B}C$

Tie = $BD + AC + \bar{A}\bar{B}\bar{C}\bar{D}$

5. Verilog

5.1] Extend this Verilog implementation of a 7-segment display driver to support hexadecimal digits (0-9, followed by A-F). You'll have to choose which lights come on for each alphabetical digit. It's fine if some are uppercase and some are lowercase.




```
module seg7_hex (digit, leds);
   input  logic [3:0] digit;
   output logic [6:0] leds;
   always_comb
      case (digit)
         // 3210              6543210
         4'b0000: leds = 7'b0111111;
         4'b0001: leds = 7'b0000110;
         4'b0010: leds = 7'b1011011;
         4'b0011: leds = 7'b1001111;
         4'b0100: leds = 7'b1100110;
         4'b0101: leds = 7'b1101101;
         4'b0110: leds = 7'b1111101;
         4'b0111: leds = 7'b0000111;
         4'b1000: leds = 7'b1111111;
         4'b1001: leds = 7'b1101111;
         4'b1010: leds = 7'b1110111;
         4'b1011: leds = 7'b1111100;
         4'b1100: leds = 7'b0111001;
         4'b1101: leds = 7'b1011110;
         4'b1110: leds = 7'b1111001;
         4'b1111: leds = 7'b1110001;

         default: leds = 7'bX;
      endcase
endmodule
```

Designed for active high but the actual LEDR's on the circuit board are active low.