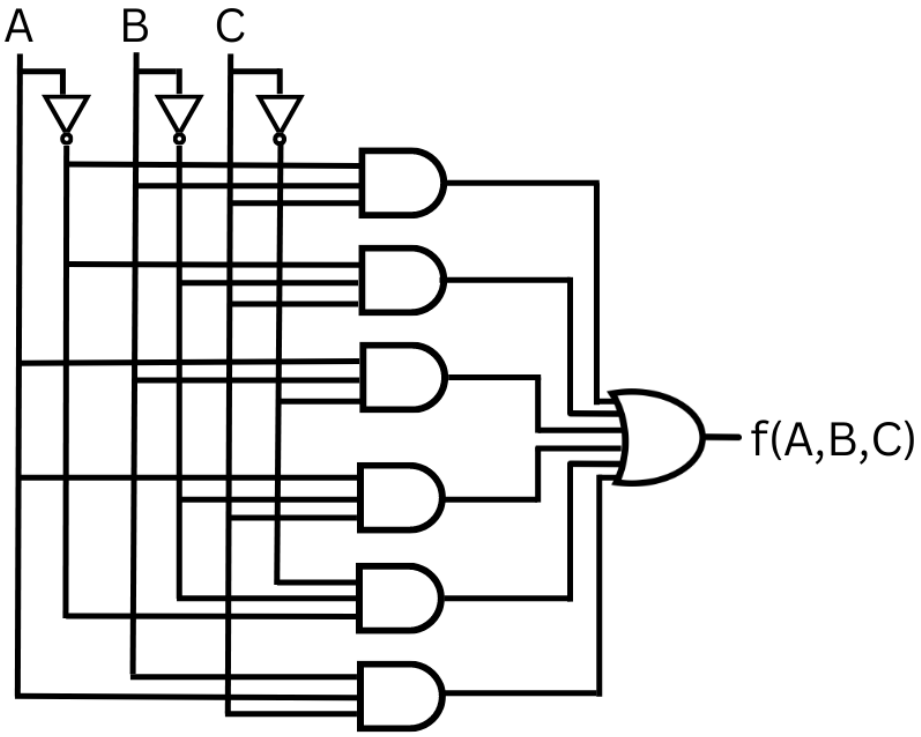
CS4341.504 HW#2 - solution
FALL SEMESTER 2023

INSTRUCTOR: DR. ALICE WANG
alice.wang@utdallas.edu

Part 1 - Boolean Theorems (20pts)

Given the following circuit:



- Convert the circuit into an equation

/ABC + /A/BC + A/BC + A/BC + /A/B/C + ABC

- Convert the circuit/equation into a truth table.

#	A	B	C	f(A, B, C)
0	0	0	0	1
1	0	0	1	1

2	**0**	**1**	**0**	**0**
3	**0**	**1**	**1**	**1**
4	**1**	**0**	**0**	**0**
5	**1**	**0**	**1**	**1**
6	**1**	**1**	**0**	**1**
7	**1**	**1**	**1**	**1**

- Convert the truth table into Sum-of-Products.

SOP = /ABC + /A/BC + A/BC + A/BC + /A/B/C + ABC

- Convert the truth table into Product-of-Sums

POS = (A + /B + C) * (/A + B + C)

- Show the Sigma Notation for the truth table.

Σ(0,1,3,5,6,7)

- Show the Pi Notation for the truth table.

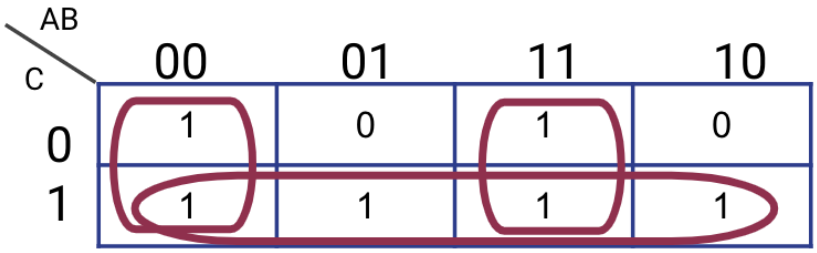
π(3,5)

- Using Boolean Theorems step-by-step simplify the equation to the normalized form. List each theorem used. Underline the sections you will apply the theorem

Use the POS to simplify:

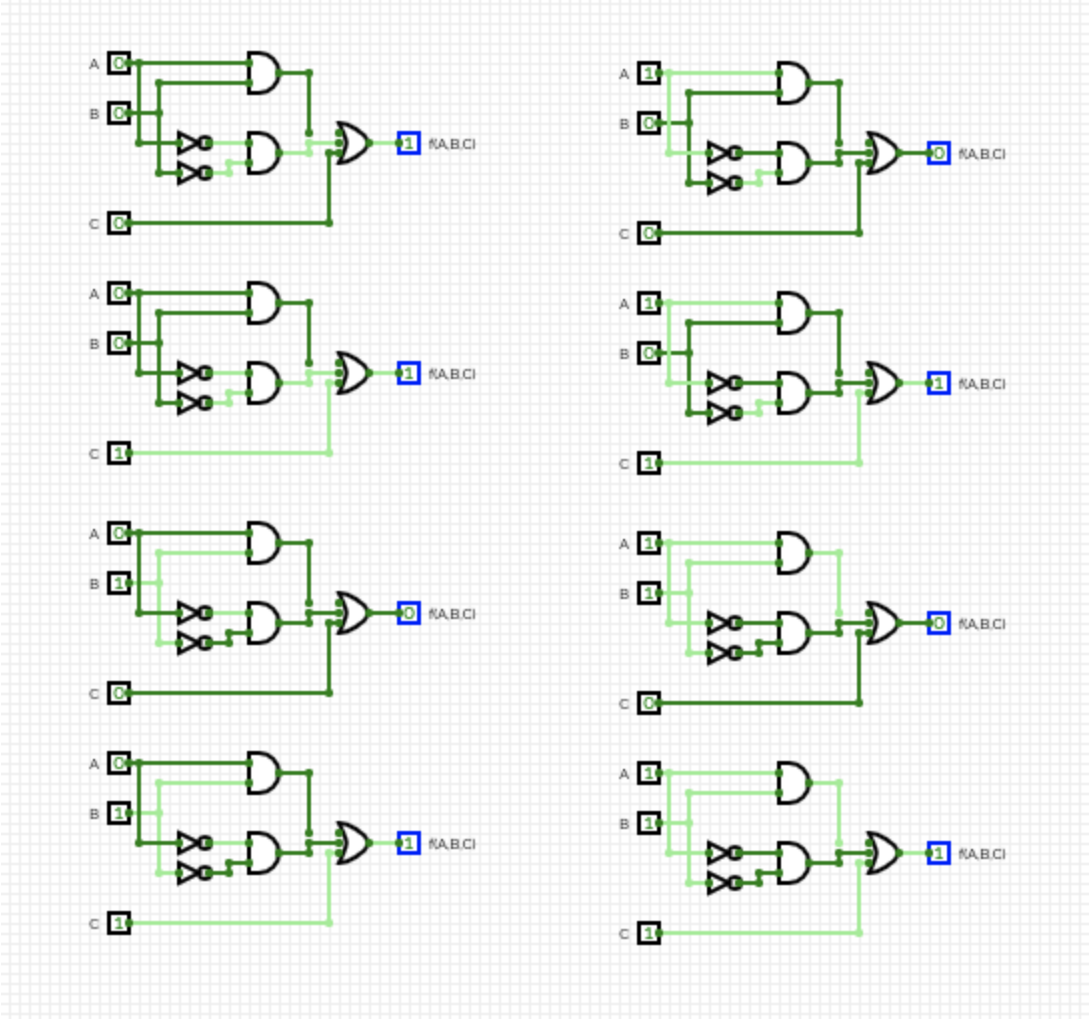
1. (A + /B + C) * (/A + B + C) — T8 Distributivity
2. A/A + AB + AC + /A/B + /BB + /BC + /AC + BC + CC — T3 Idempotency
3. A/A + AB + AC + /A/B + /BB + /BC + /AC + BC + C — T5 Complements
4. 0 + AB + AC + /A/B + 0 + /BC + /AC + BC + C — T1 Identity
5. AB + AC + /A/B + /BC + /AC + BC + C — T6 Commutativity
6. AB + /A/B + AC + /AC + /BC + BC + C — T7 Associativity
7. AB + /A/B + (AC + /AC) + (/BC + BC) + C — T10 Combining
8. AB + /A/B + C + C + C — T3 Idempotency
9. AB + /A/B + C — final

- Use Karnaugh-map to simplify the equation to the normalized form
    - Correctly draw the K-Map and mark the implicants.
    - List out each of the implicants, and the terms generated.



AB + /A/B + C

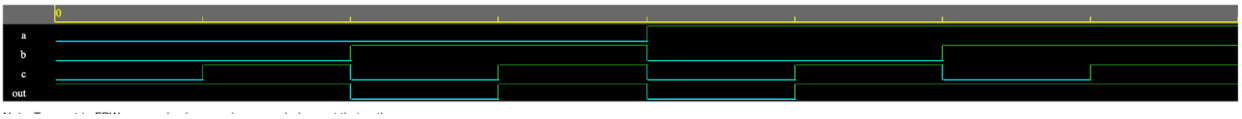
- Draw the schematic of the circuit for the normalized form and verify its function using the circuitverse website. For verification capture screenshots showing the outputs for each input combination (hint: toggle the inputs and verify that the output is correct.)



This matches the Truth Table.

- Write the Verilog code for the circuit and verify its function using edaplayground by showing the waveform of the output with all combinations of inputs.

```
# KERNEL: Start
# KERNEL: a:0,b:0,c:0,out:1
# KERNEL: a:0,b:0,c:1,out:1
# KERNEL: a:0,b:1,c:0,out:0
# KERNEL: a:0,b:1,c:1,out:1
# KERNEL: a:1,b:0,c:0,out:0
# KERNEL: a:1,b:0,c:1,out:1
# KERNEL: a:1,b:1,c:0,out:1
# KERNEL: a:1,b:1,c:1,out:1
```



- Submit your work (PDF format). For the schematic from circuitverse and the waveform window from edaplayground use a screen capture utility and cut-n-paste to your report.

Part 2 - Evaluate Boolean Expressions (10pts)

Evaluate the Boolean equation

F = a AND (b OR (c AND d))

for the given values of variables a, b, c, and d:

1. a=1, b=1, c=0, d=1
2. a=0, b=0, c=0, d=1
3. a=1, b=0, c=0, d=0
4. a=1, b=0, c=1, d=1
5. a=1, b=0, c=0 d=1

1) F = 1 AND (1 OR (0 AND 1)) = 1 AND (1 OR 0) = 1 AND 1 = 1

2) F = 0 AND (0 OR (0 AND 1)) = 0 AND (0 OR 0) = 0 AND 0 = 0

3) F = 1 AND (0 OR (0 AND 0)) = 1 AND (0 OR 0) = 1 AND 0 = 0

4) F = 1 AND (0 OR (1 AND 1)) = 1 AND (0 OR 1) = 1 AND 1 = 1

5) F = 1 AND (0 OR (0 AND 1)) = 1 AND (0 OR 0) = 1 AND 0 = 0

Part 3 - Multiplexors & Adders (50pts)

In part we will create combinational logic subcircuits which are building blocks for a small processor in future assignments.

In circuitverse.org, create

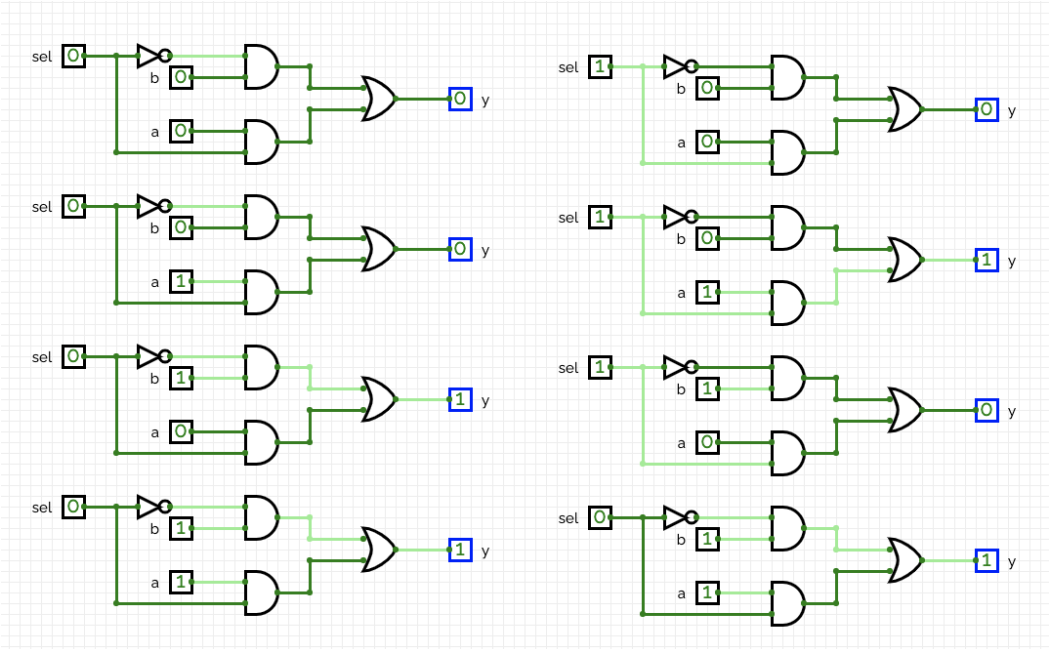
- 2:1 mux using two-level logic
- 2:1 mux using tristate buffers
- half-adder

For verification capture screenshots showing the outputs for all input combinations (hint: toggle the inputs and verify that the output match the truth table.)

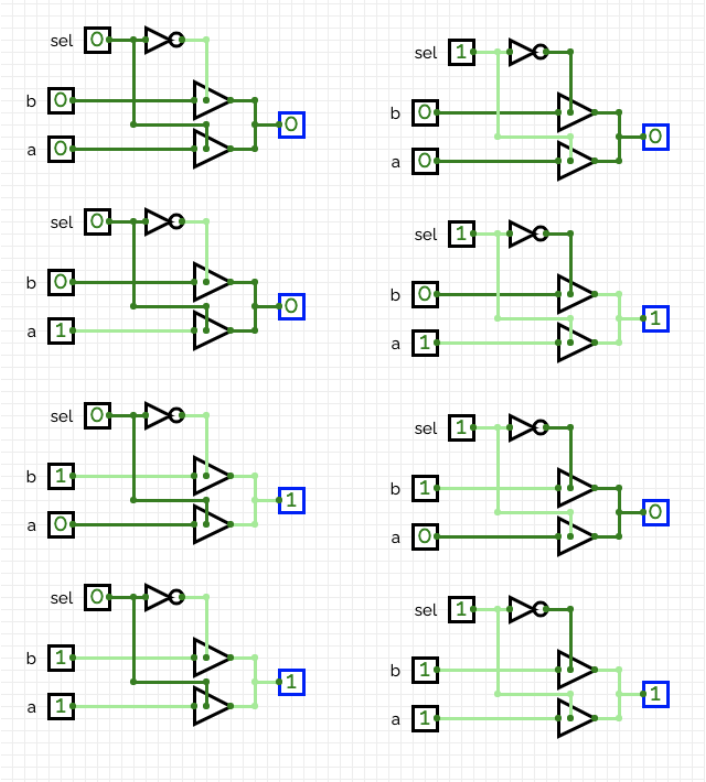
2:1 mux truth table:

sel	b	a	y
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

Two-level logic - toggling inputs and matching the truth table



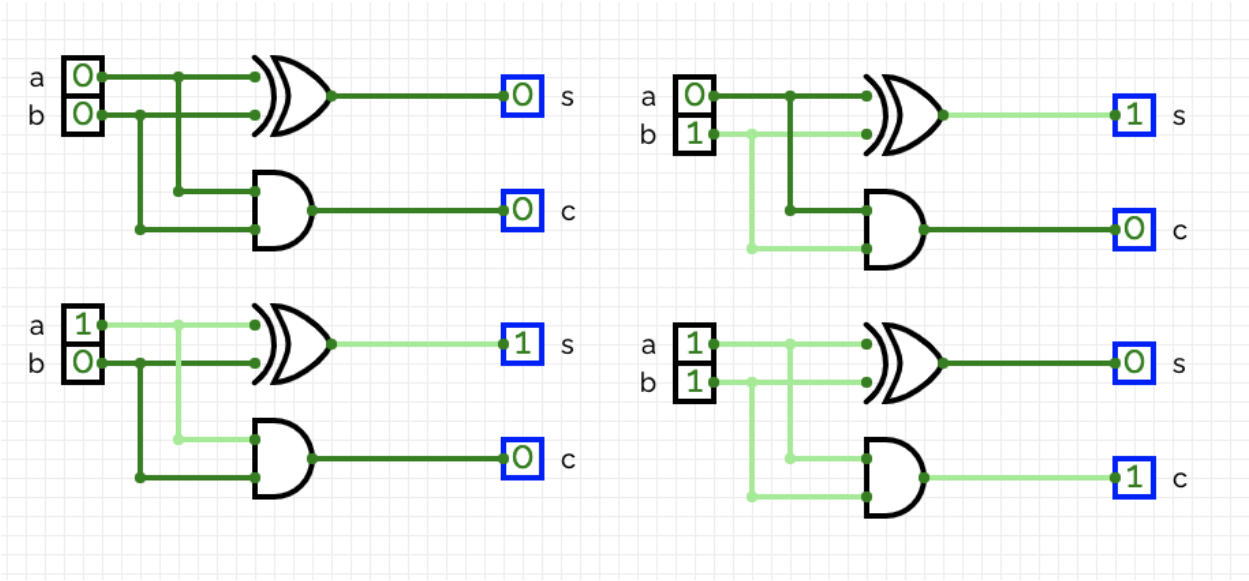
Tristate buffer - toggling inputs and matching the truth table



Half-adder truth table

b	a	s	c
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

Half-adder toggling the inputs and matching the truth table



- Write the Verilog code for a 2:1 mux and half-adder and verify their functionality using edaplayground.com by creating a testbench that tests all input combinations.

Verilog code: testbench instantiation

```
// Testbench
module test;

  reg a,b,sel,y1, y2, s, c;

  // Instantiate design under test
  mux2_two_stage_logic MYMUX0 (.a(a), .b(b), .sel(sel), .y(y1));
  mux2_tristate MYMUX1 (.a(a), .b(b), .sel(sel), .y(y2));
  half_adder HA (.a(a), .b(b), .s(s), .c(c));

  initial begin
    // Dump waves
  $dumpfile("dump.vcd");
    $dumpvars(1);

    $display("Start");
    {sel,b,a} = 3'b000;
    display;
    {sel,b,a} = 3'b001;
    display;
    {sel,b,a} = 3'b010;
    display;
    {sel,b,a} = 3'b011;
    display;
    {sel,b,a} = 3'b100;
    display;
    {sel,b,a} = 3'b101;
    display;
    {sel,b,a} = 3'b110;
    display;
    {sel,b,a} = 3'b111;
    display;
    {sel,b,a} = 3'b000;
    display2;
    {sel,b,a} = 3'b001;
    display2;
    {sel,b,a} = 3'b010;
    display2;
    {sel,b,a} = 3'b011;
    display2;
  end
```

Verilog code: design instantiation

```
// Design

module mux2_two_stage_logic (a,b,sel,y);
  input a,b,sel;
  output y;

  assign y = (sel & a) | (~sel & b);
endmodule

module tristate_buffer (a,sel,y);
  input a,sel;
  output y;

  assign y = sel ? a : 1'bZ;
endmodule

module mux2_tristate (a,b,sel,y);
  input a,b,sel;
  output y;

  tristate_buffer TR1 (a,sel,y);
  tristate_buffer TR2 (b,~sel,y);
endmodule

module half_adder (a,b,s,c);
  input a,b;
  output s,c;

  assign s = a ^ b;
  assign c = a | b;
endmodule
```

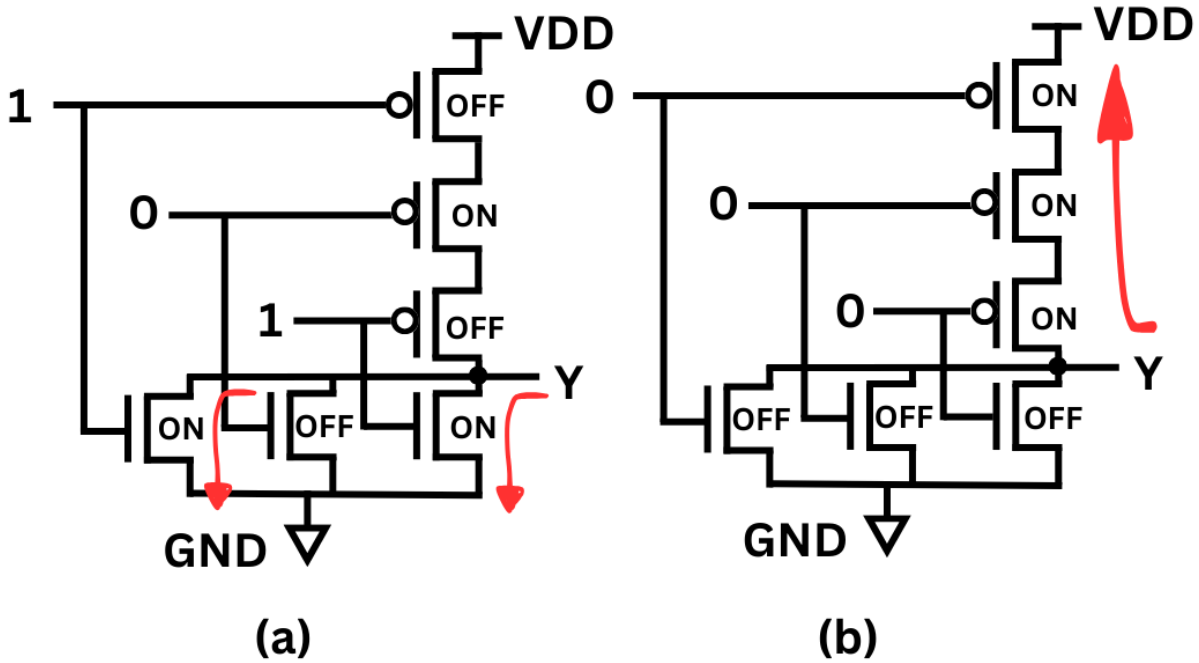
- Submit your work (PDF). For the schematic from circuitverse and the waveform window from edaplayground use a screen capture utility and cut-n-paste to your report by the due date.

Part 4 - Transistors (20pts)

Show the pull-up/pull-down path for the 3-input NOR that was shown in Lecture 4 when

(a) a=1, b=0, c = 1

(b) a=0, b = 0, c = 0


VDD
1
OFF
0
ON
1
OFF
Y
ON
OFF
ON
GND
(a)
VDD
0
ON
0
ON
0
ON
Y
OFF
OFF
OFF
GND
(b)