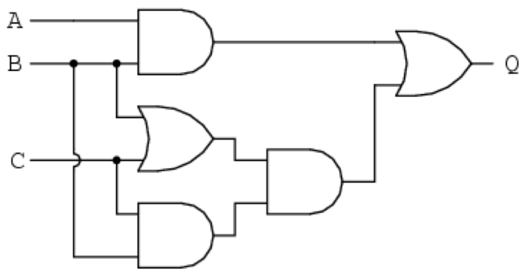
Exercise 1 – Gates Galore

a) Write out the Boolean Algebra expression for Q for the following circuit.

b) Simplify the expression for Q.



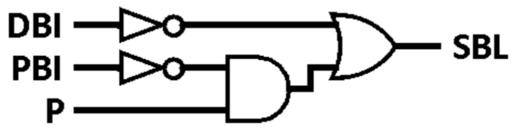
Exercise 2 – Universal Gates

Implement the Boolean expression B(A + C) with the fewest number of a single universal gate. What does your solution look like?

Exercise 3 – Seat Belt Light

Write a SystemVerilog module that implements the Seat Belt Light circuit from Lecture 1:

- SeatBeltLight (DriverBeltIn, PassengerBeltIn, Passenger)
- Don't mix-and-match – use either all built-in operators or all built-in gates



```
module seatbelt_light(input  logic _______, _______, _______,
                      output logic _______);




endmodule  // seatbelt_light
```

Exercise 4 – Comparator

Design a circuit that compares two numbers with the following specifications:

Inputs:

- `A`: first number
- `B`: second number
- Inputs assumed signed

Outputs:

- `is_gt` (>): asserted when $A > B$
- `is_eq` (=): asserted when $A = B$
- `is_lt` (<): asserted when $A < B$

For simplicity, the design shall take advantage of the subtraction operator in Verilog by computing $A - B$.

- `is_lt`: (Most significant bit of $A - B$) == 1 (negative)
- `is_eq`: NOR all bits of $A - B$
- `is_gt`: MSB of $A - B$ is 0 *and* ¬`is_eq`

Note: These fail some edge cases but we will ignore those for now.

```
module comparator(input logic [2:0] _______, _______,
                        output logic _______, _______, _______);
```

```
endmodule  // comparator
```

Exercise 5 – Guessing Game

Create a magic number guessing game using the comparator module with the following specifications:

- The system shall contain a *secret*, hard-coded number of your choosing. Recall that a constant in SystemVerilog is written in the form `3'b001`.
- `SW[2:0]` represents the user's guess.
- `KEY[0]` is used to check the user's guess. All `KEY` inputs are *active-low*, meaning a value of `0` indicates the button is pressed.

When `KEY[0]` is asserted, the LEDs shall indicate the result of the comparison as follows:

- `LEDR[0]` shall be illuminated if the user's guess is greater than the secret number (signed comparison).
- `LEDR[1]` shall be illuminated if the user's guess is equal to the secret number.
- `LEDR[2]` shall be illuminated if the user's guess is less than the secret number (signed comparison).

a) Draw a block diagram of your proposed system:

KEY[0]
SW[2]
SW[1]
SW[0]
LEDR[2]
LEDR[1]
LEDR[0]

b) Implement the system in SystemVerilog:

```
module guessing_game(output logic [9:0] LEDR,
                     input  logic [3:0] KEY,
                     input  logic [9:0] SW);



endmodule  // guessing_game
```