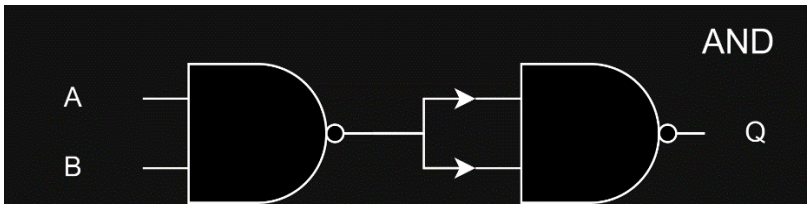
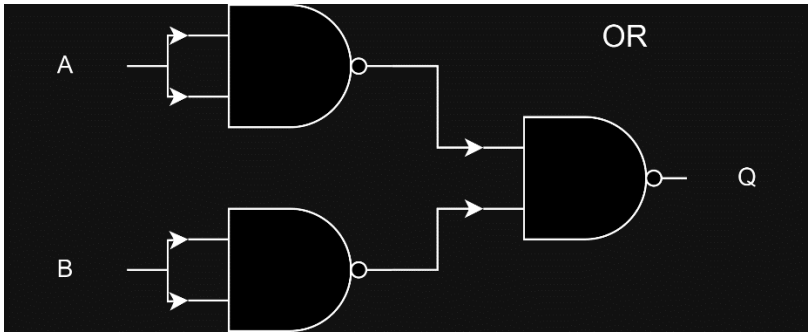
硬體設計與實驗 Lab2 Report 110062271 林奕為 資工 25

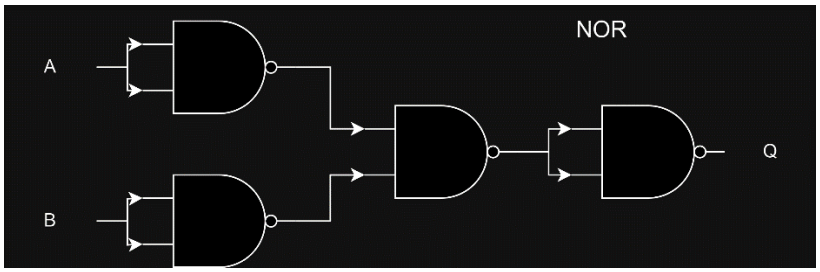
What I have learned:
In this lab, I have learned how to implement various basic logic gates using NAND universal gate. Also, I implemented the Full adder with Majority modules, and ripple-carry-adder, carry-lookahead-adder and multiplier also.
Last, I learn how to program fpga board with representing the output as a single hexadecimal number.

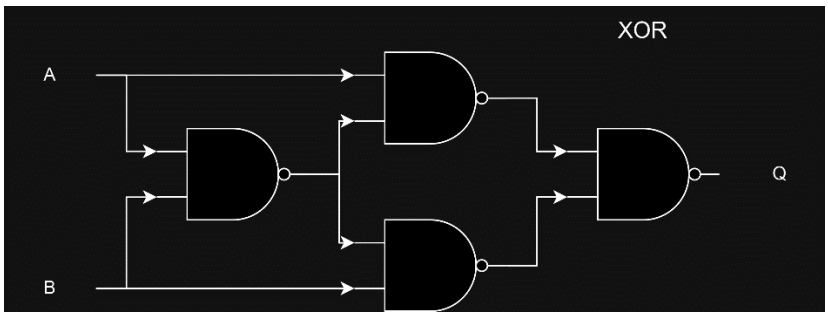
Basic :

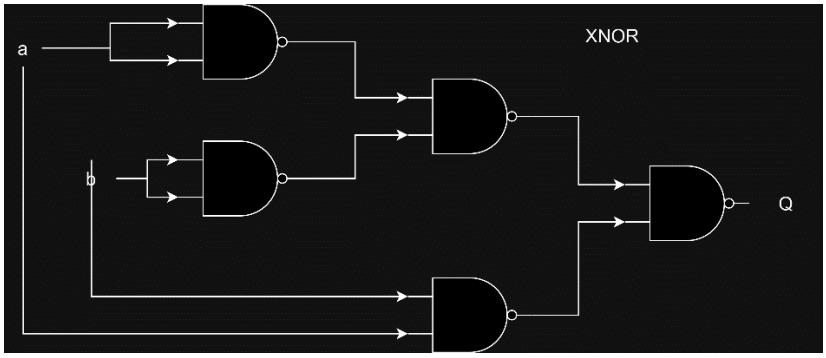
Question 1:

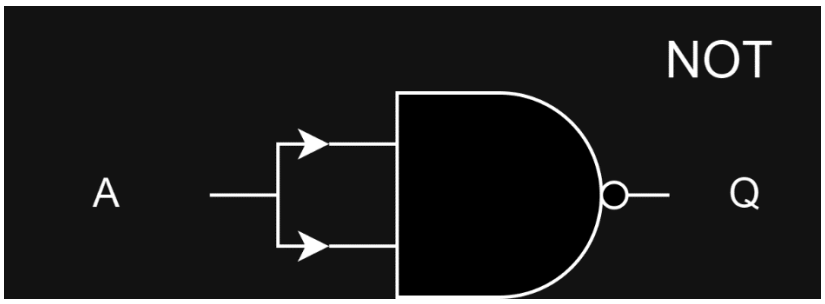


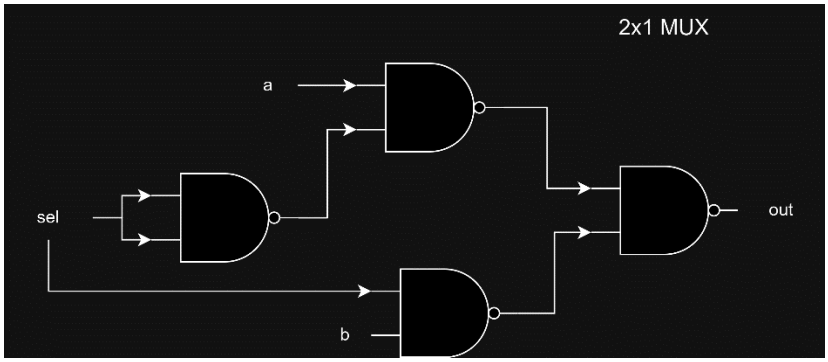


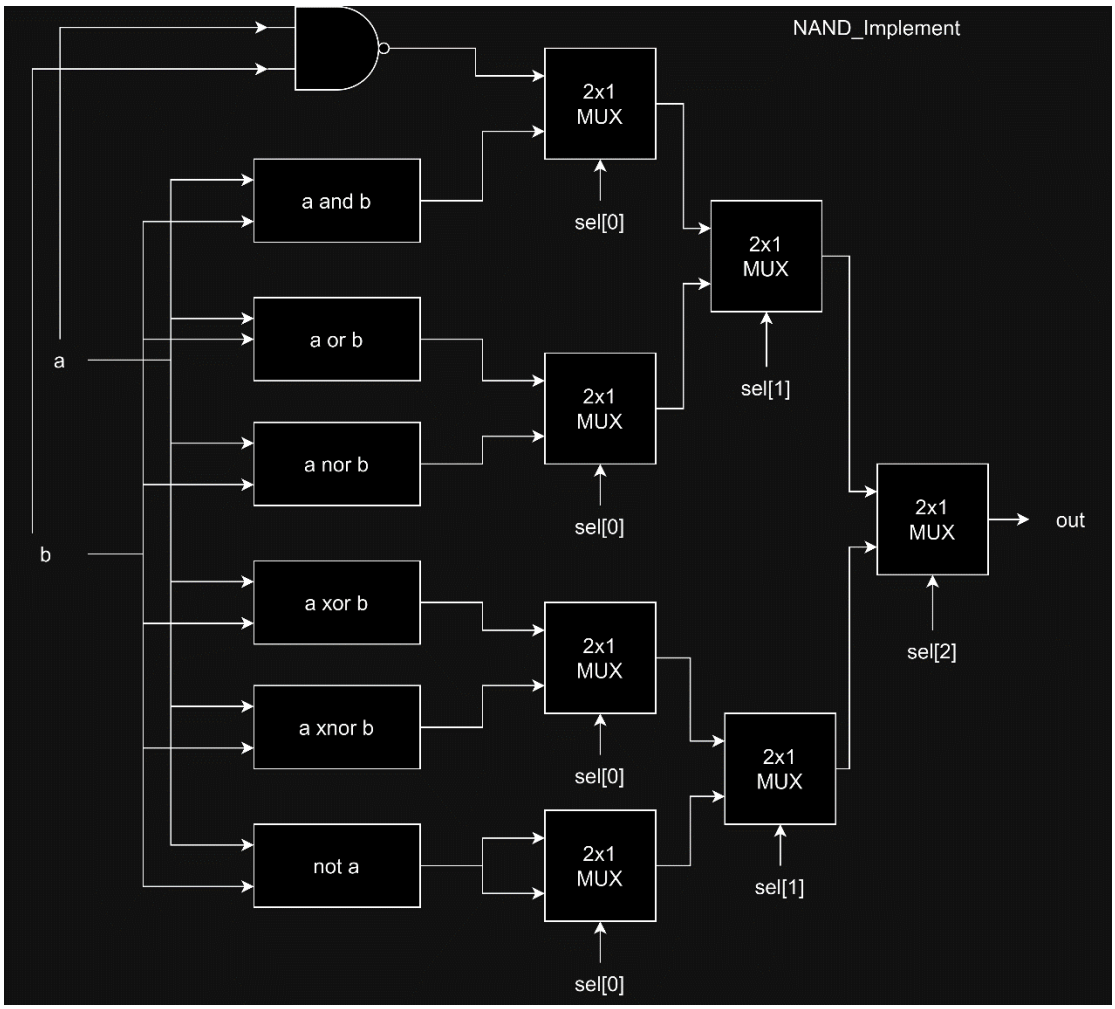





XNOR
a
b
Q


NOT
A
Q


2x1 MUX
a
sel
b
out



Question 3:

The main difference between a half adder and a full adder is that a full adder can take an additional input cin and calculate the sum as (a + b + cin) while a half adder calculates the sum as (a + b).

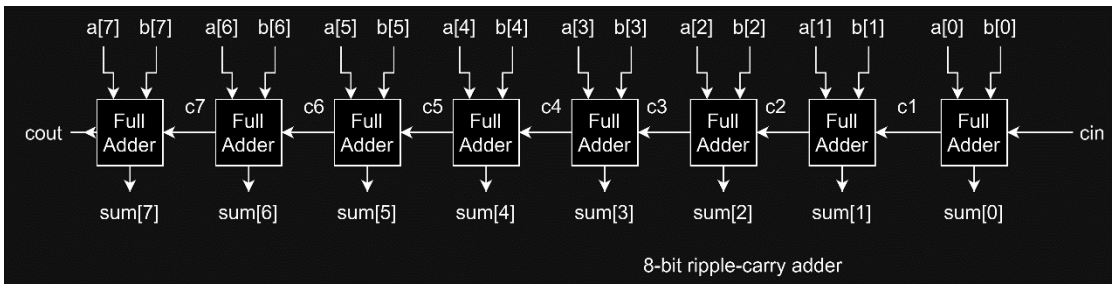
As a result, a full adder can be used to implement a ripple carry adder since it can take the cout of the previous bit-calculation as the cin to the current bit-calculation.

Additionally, a Full adder can also be implemented by 2 half adders.

Advanced:

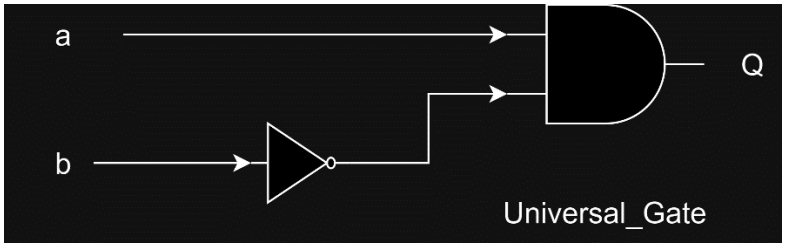
Question 1:

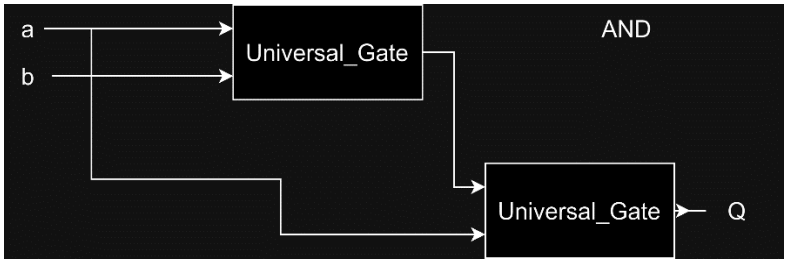
Diagram :

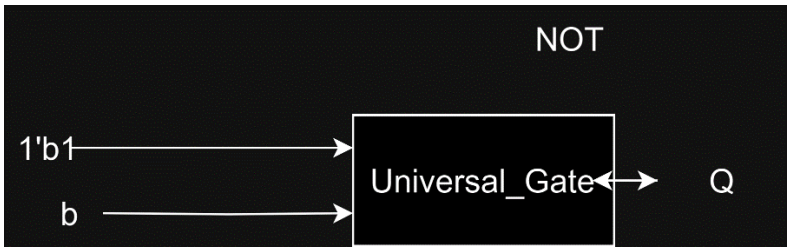


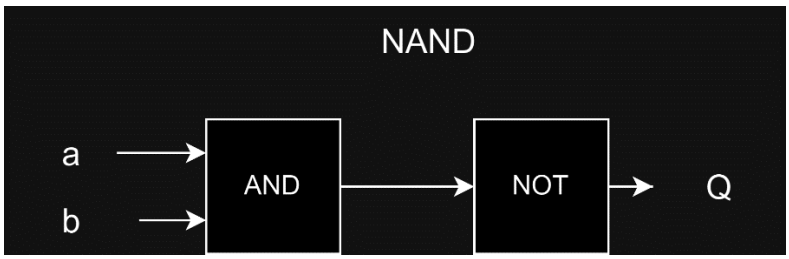
Question 2:

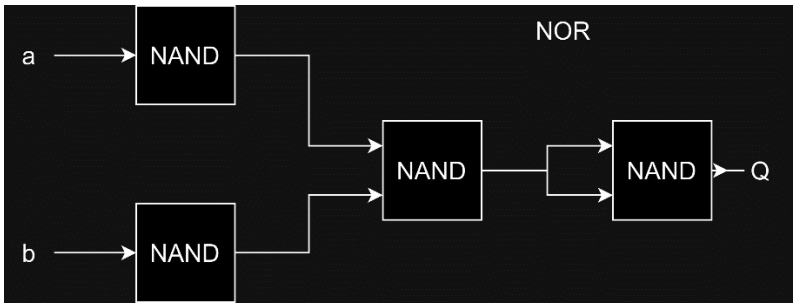
Diagram :

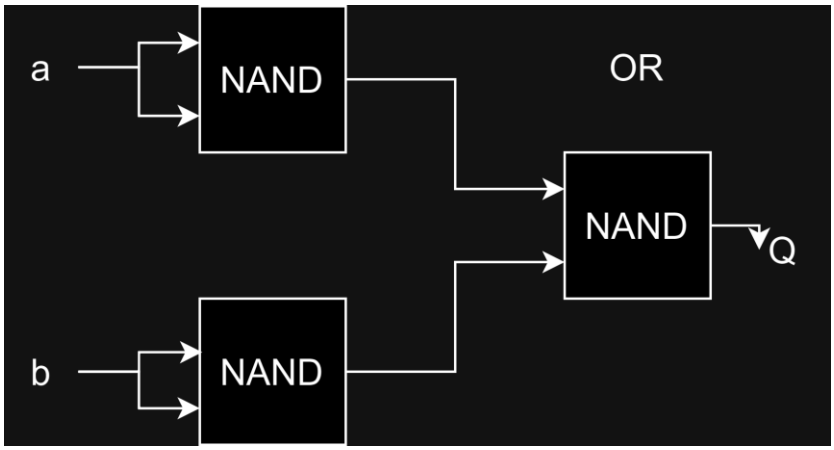


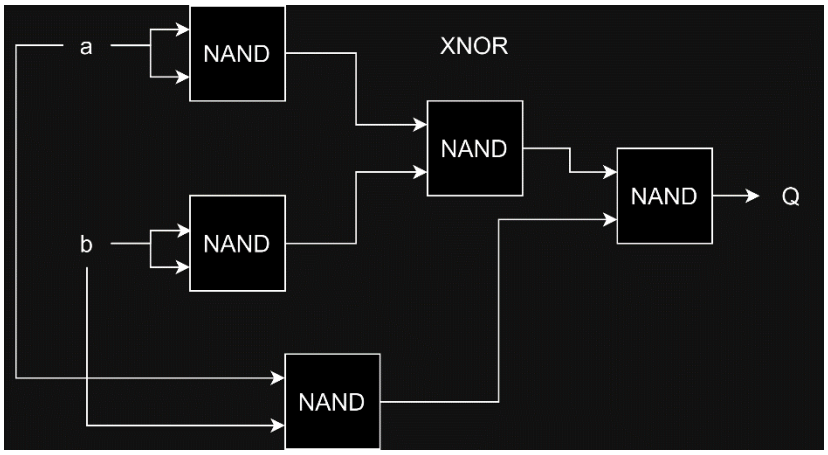


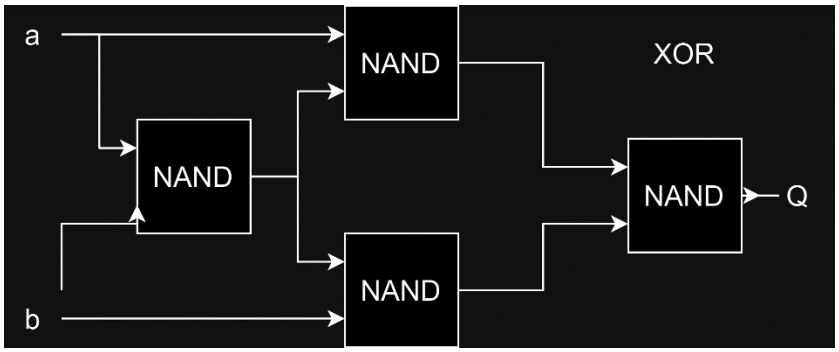





NOR
a
b
NAND
NAND
NAND
NAND
Q


OR
a
b
NAND
NAND
NAND
Q


XNOR
a
b
NAND
NAND
NAND
NAND
NAND
Q


XOR
a
b
NAND
NAND
NAND
NAND
Q

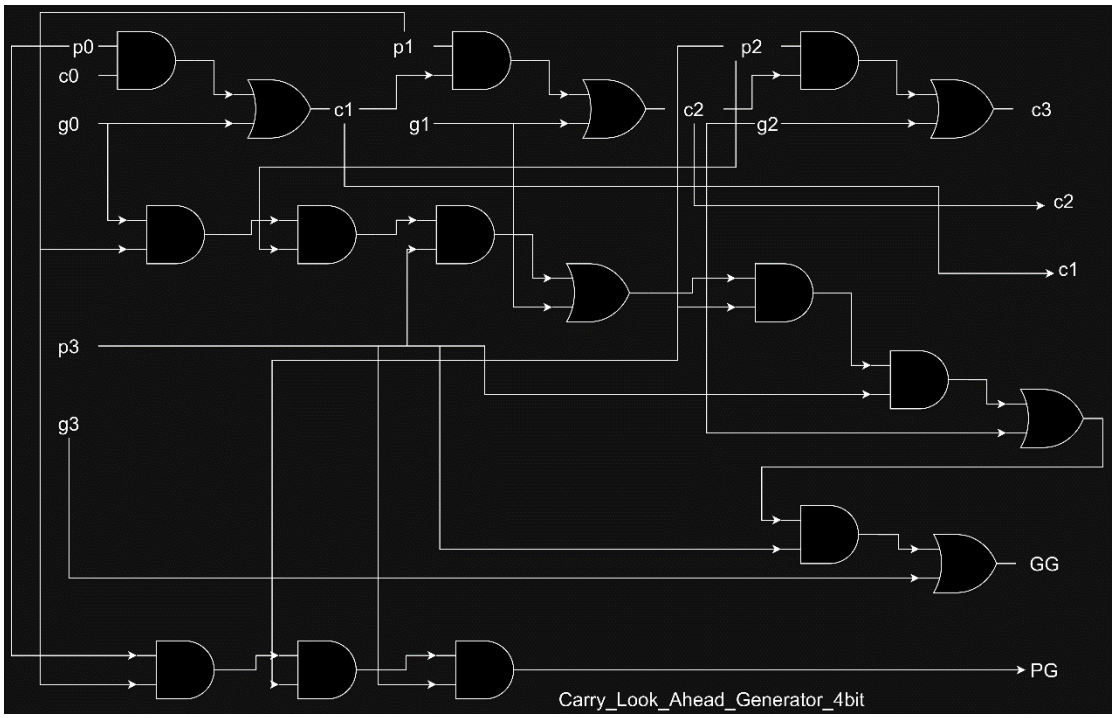
Question 3:

By using the formula below, we can get the cin to each Full adder in parallel manner without the waiting time to calculate the cin bit each time before the addition of the next Full adder.

Pi = Ai OR Bi, Gi = Ai XOR Bi

C i+1 = Gi OR (Pi AND Ci)

Diagram : (The basic logic gates in the diagram is implemented by NAND gates, with pictures in the NAND_Implement)



How does it work?

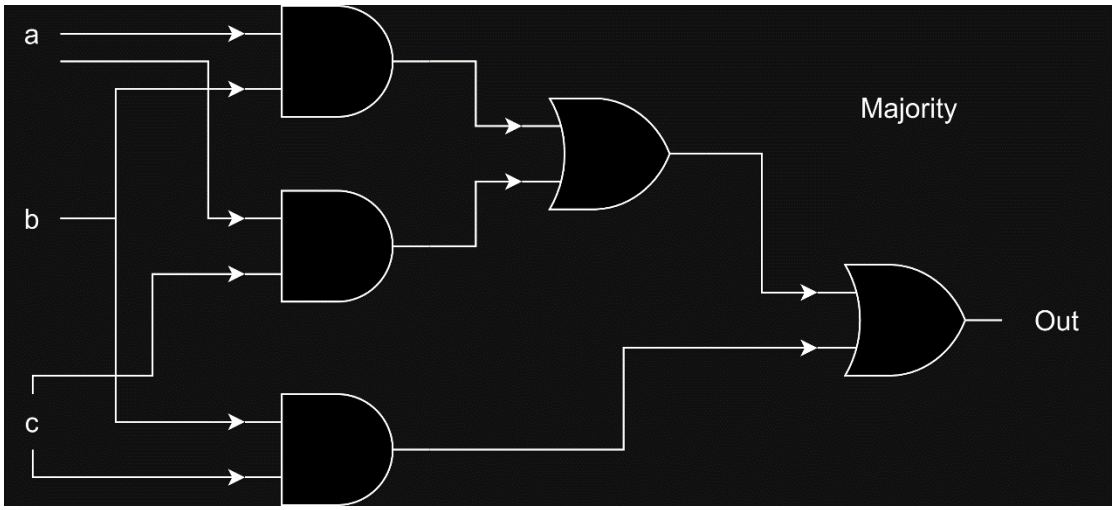
I followed the diagram provided and implement it with two 4-bit CLA generator and one 2-bit CLA generator and 8 Full-adder(with additional output p and g).

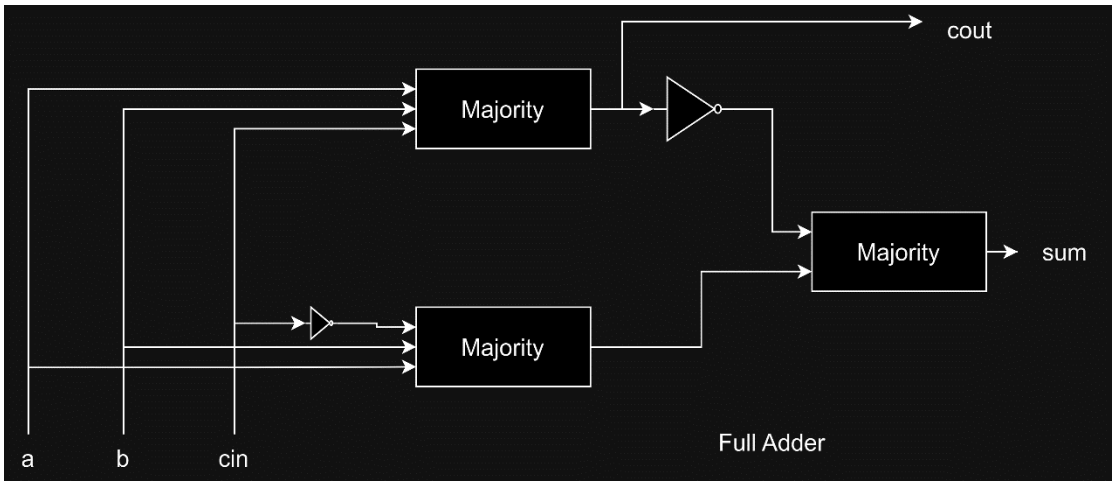
The two 4-bit CLA generator calculate c1, c2, c3, c5, c6, c7 and pass their individual PG and GG to the 2-bit CLA generator, and c0 as input, which then outputs c4 and c8.

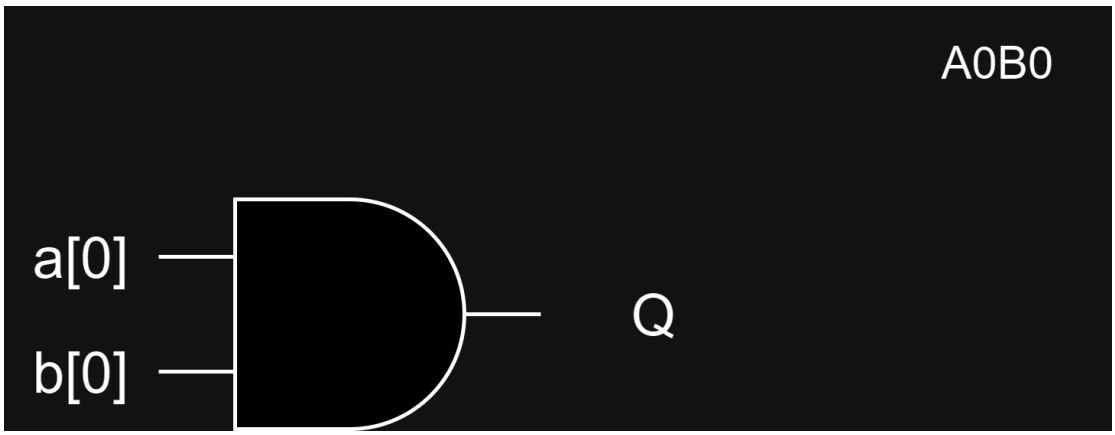
The Full adders individually outputs pi and gi to the 4-bit CLA generator first and later take c[i] as input (after CLA generator completed their tasks) with a[i] and b[i] to calculate their sum bit (sum[i]).

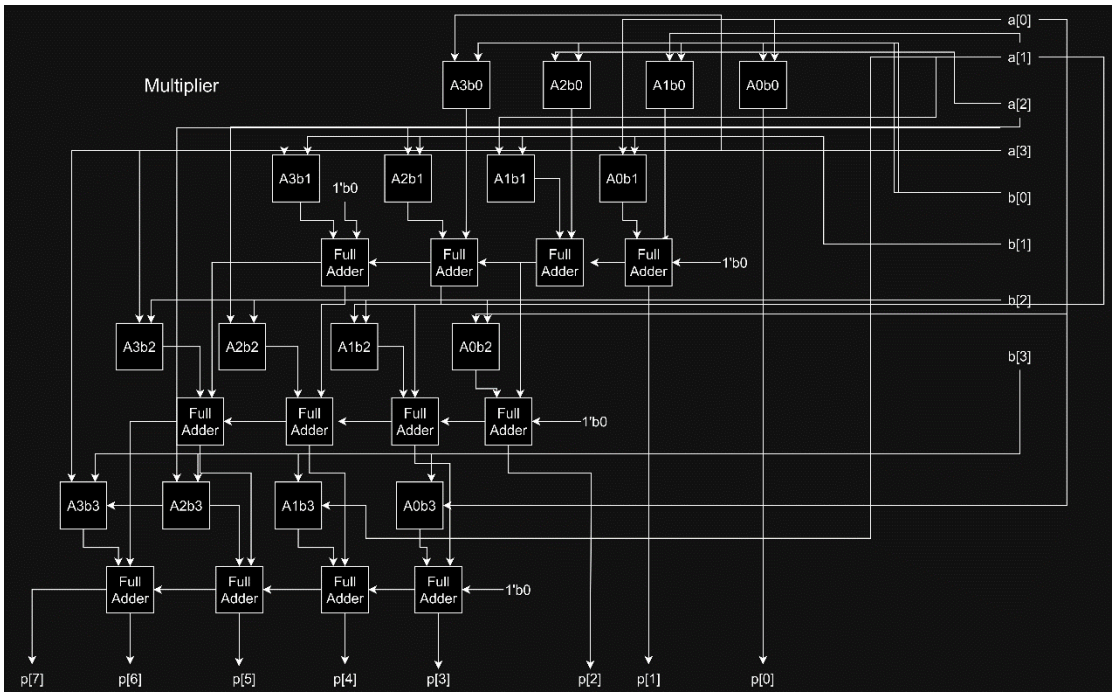
Question 4:

First, we AND 1 bit from a and 1 bit from B, which represents a[i] * b[i],
next, we sum the product vertically, while passes the cins and couts from previous to the next Full adder, which represents carry in from the addition.
According to the illustration in the PDF, we can get the result of a multiply b.
Diagram : (In which A0b0 represents a[0] AND b[0], and the basic logic gates (e.g. AND, OR) are implemented by NAND gates using the modules of Basic question about NAND_Implement)






A0B0
a[0]
b[0]
Q


Multiplier
a[0]
a[1]
a[2]
a[3]
b[0]
b[1]
b[2]
b[3]
A3b0
A2b0
A1b0
A0b0
A3b1
A2b1
A1b1
A0b1
1'b0
Full Adder
A3b2
A2b2
A1b2
A0b2
A3b3
A2b3
A1b3
A0b3
p[7]
p[6]
p[5]
p[4]
p[3]
p[2]
p[1]
p[0]