# *1. Please give the logical expression of the logical circuit in the box below (Task 1).*

Note, the detailed step-by-step procedure of derivation is needed. You will need the four special symbols, namely, ∧, ⊕, ∨ and ¬ in formulating the logical expression.

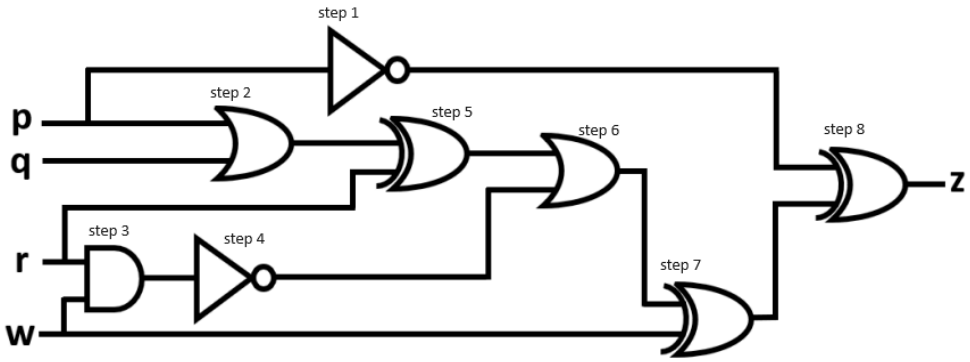


Fig. 1. The logical circuit

For the given logical circuit, which is provided by the lecturer for EEEM005 Coursework, the listed steps have been followed. For a better explanation step by step approach followed and the logic gates of the given circuit labelled as number of steps.

Step 1: In this step Not gate is used. The input is p and the output is ¬p

Step 2: In this step Or gate is used. The inputs are p and q; the output is p∨q

Step 3: In this step And gate is used. The inputs are r and w; the output is r∧w

Step 4: In this step Not gate is used. The input is r∧w and the output is ¬(r∧w)

Step 5: In this step XOR gate is used. The inputs are p∨q and r. The output is (p∨q)⊕r

Step 6: In this step Or gate is used. The inputs are (p∨q)⊕r and ¬(r∧w). The output is ((p∨q)⊕r)∨(¬(r∧w))

Step 7: In this step XOR gate is used. The inputs are ((p∨q)⊕r)∨(¬(r∧w)) and w. The output is (((p∨q)⊕r)∨(¬(r∧w)))⊕w

Step 8: In this step XOR gate is used. The inputs are ¬p and (((p∨q)⊕r)∨(¬(r∧w)))⊕w. The output is ¬p⊕((((p∨q)⊕r)∨(¬(r∧w)))⊕w)

With the step 8 it is seen that the deerivation of the logic, which equals to z is ¬p⊕((((p∨q)⊕r)∨(¬(r∧w)))⊕w).

In this logical circuit there are 8 logical gates which are 2 Not gates, 2 Or gates, 1 And gate and 3 XOR gate. It is also observed that some of the inputs are actually the outputs of the steps before them. Step 8 gets the outputs as input from step 7 and step 1. Step 7 gets one of the input as the output of step 6. Step 6 uses the outputs of the step 5 and step 4 as input. Step 4 uses the output from step 3 and step 5 uses step 2.

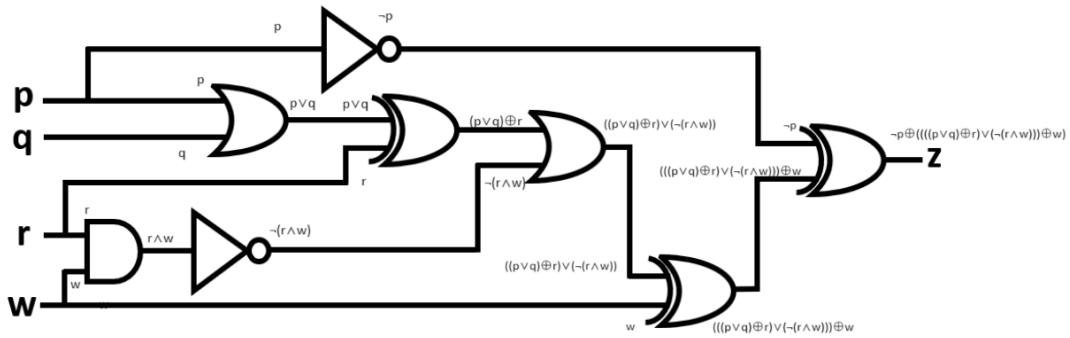


Fig. 1. The logical circuit

# *2. Please complete the truth table of the logical circuit given by the text box below (Task 1).*

Please provide the missing values in the z column to complete the truth table.

**Truth table of the logical circuit**

| p | q | r | w | z |
|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 1 |
| 1 | 1 | 1 | 0 | 1 |
| 1 | 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 | 1 |
| 1 | 0 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 0 | 0 | 1 |
| 0 | 1 | 1 | 1 | 0 |
| 0 | 1 | 1 | 0 | 0 |
| 0 | 1 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 1 |
| 0 | 0 | 1 | 0 | 0 |
| 0 | 0 | 0 | 1 | 1 |
| 0 | 0 | 0 | 0 | 0 |

With using the given steps according to the z= ¬p⊕((((p∨q)⊕r)∨(¬(r∧w)))⊕w) step by step the values are found. In order to do that the equation partially calculated from inside to outside and the values are put into the steps in order to find z.

Step 1: (p ∨ q) - it is an OR gate it will be 1 when either one of p or q is 1; else 0.

Step 2: ((p ∨ q) ⊕ r) - it is an XOR gate if the output of (p ∨ q) is 1 and r is 1 then the output is 1; else 0.

Step 3: (¬(r ∧ w)) - it is and AND gate therefore it is 1 if both of them is 1; else 0. Since the equation has also a NOT gate, the output will be 0, if both of them is 1 and output will be 1 otherwise.

Step 4: (((p ∨ q) ⊕ r) ∨ (¬(r ∧ w))) - since there is an OR gate between the sides if any of the sides is 1, the output will be 1, otherwise 0.

Step 5: ¬p ⊕ ((((p ∨ q) ⊕ r) ∨ (¬(r ∧ w))) ⊕ w) - the sides are relationed with XOR gate and on the left side there is a NOT gate. Without the NOT gate if the left side is 1 right should be one as well for output of 1 but with the NOT gate, if two side is different the output will be 1, otherwise it will be 0.

## *3. Please complete the code template of the logical circuit given by the code box below (Tasks 2 & 3).*

In [33]:

```
def Logical_Circuit(p,q,r,w):
    #AND Gate - Conjunction Operation
    def AND(a, b):
        a = int(a)
        b = int(b)
        if a == 1 and b == 1:
            return 1
        else:
            return 0

    #OR Gate - Disjunction Operation
    def OR(a, b):
        a = int(a)
        b = int(b)
        if a == 1:
            return 1
        elif b == 1:
            return 1
        else:
            return 0

    #NOT Gate - Negation Operation
    def NOT(a):
        a = int(a)
        if a == 1:
            return 0
        else:
            return 1


    #XOR Gate - Exclusive Disjunction Operation
    def XOR(a, b):
        a = int(a)
        b = int(b)
        if a != b:
            return 1
        else:
            return 0

    p = None
    while p != 0 and p != 1:
        p = int(input("p: "))
        if p != 0 and p != 1:
            print("Invalid input. Should be either 0 or 1")

    q = None
    while q != 0 and q != 1:
        q = int(input("q: "))
        if q != 0 and q != 1:
            print("Invalid input. Should be either 0 or 1")

    r = None
    while r != 0 and r != 1:
        r = int(input("r: "))
        if r != 0 and r != 1:
            print("Invalid input. Should be either 0 or 1")

    w = None
    while w != 0 and w != 1:
        w = int(input("w: "))
        if w != 0 and w != 1:
```

```
            print("Invalid input. Should be either 0 or 1")


    #calculation of the equation starts from the left and the importance of
    p_or_q = OR(p, q) #the centered logic (p∨q)
    p_or_q_xor_r = XOR(p_or_q, r) #the second logic gate, XOR with the step
    r_and_w = AND(r, w) #the centered logic on the right hand side after the
    not_r = NOT(r) #the negotiation of r (¬r)
    not_r_and_w = NOT(r_and_w)#the negotiation of the centered logic ¬(r∧w)
    p_or_q_xor_r_or_not_r_and_w = OR(p_or_q_xor_r, not_r_and_w)
    # named as step7(as its label in the graph above) since the variable nam
    step7 = XOR(p_or_q_xor_r_or_not_r_and_w, w)  # both side of the OR gate(
    not_p = NOT(p) #the negotiation of p  (¬p)
    z = XOR(not_p, step7) # both side of the XOR gate on the very left. Sinc
    return z  #result of the logic gate



z = Logical_Circuit(p,q,r,w)
#print(type(z))
print("z:", z)
```

```
p: 0
q: 1
r: 0
w: 0
z: 0
```

The function which is given, Logical_Circuit() calculates the logical expression according to the given figure 1, logical circuit. Takes 4 arguments p,q,r,w as integers and returns integer z, as the output which is also the result of the truth table given the inputs.

The function takes the inputs from the user one by one and if they are valid calculates the z value. If the input is not 0 or 1 it ensures that it is either 0 or 1 untill it is inputted.

As given in the task, in order to make sure that the function works as intented, few approaches implemented as for controls. The input for the arguments turned into integers while being inputted and as the second level control the 0 or 1 check implemented. In this way, incorrect type of inputs handled instead of creating different subfunctions for data type check or out of correct range errors with while loops.

4 logical gates implemented in order to implement the given expression ¬p⊕((((p∨q)⊕r)∨(¬(r∧w)))⊕w). This sub-functions named according to the operation they do. The gates are implemented according to their logic gates algorithms. All of the gates also takes integers as the data type. Also the control of the correct type of input is done while inputting the parameters at the beginning from user.

The implementation of the expression ¬p⊕((((p∨q)⊕r)∨(¬(r∧w)))⊕w) started from the paranthesis. ¬p section is done at the very last step. Function started the calculate expression right hand side of the XOR gate ((((p∨q)⊕r)∨(¬(r∧w)))⊕w). As the first step, due to the importance of the paranthesis, the left side of the OR gate, centered equation,p∨q, is calculated. Later the operation with XOR gatw with r calculated. As the third step according to the paranthesis r∧w, not r and ¬(r∧w) is calculated. Since with the paranthesis the left hand side and the right hand side of the OR gate,(((p∨q)⊕r)∨(¬(r∧w))), is calculated. The last step before the whole equation is calculated(((p∨q)⊕r)∨(¬(r∧w)))⊕w) and as the very last step which is the XOR gate with ¬p, which also seperated from the implementation at the beginning is calculated.

As a summary Logical_Circuit function implements the given logical circuit expression using logic gates AND, OR, NOT, XOR. The function takes four integer input parameters p, q, r, w, which must be either 0 or 1, and outputs the result of the implemented specific logic expression, integer z.