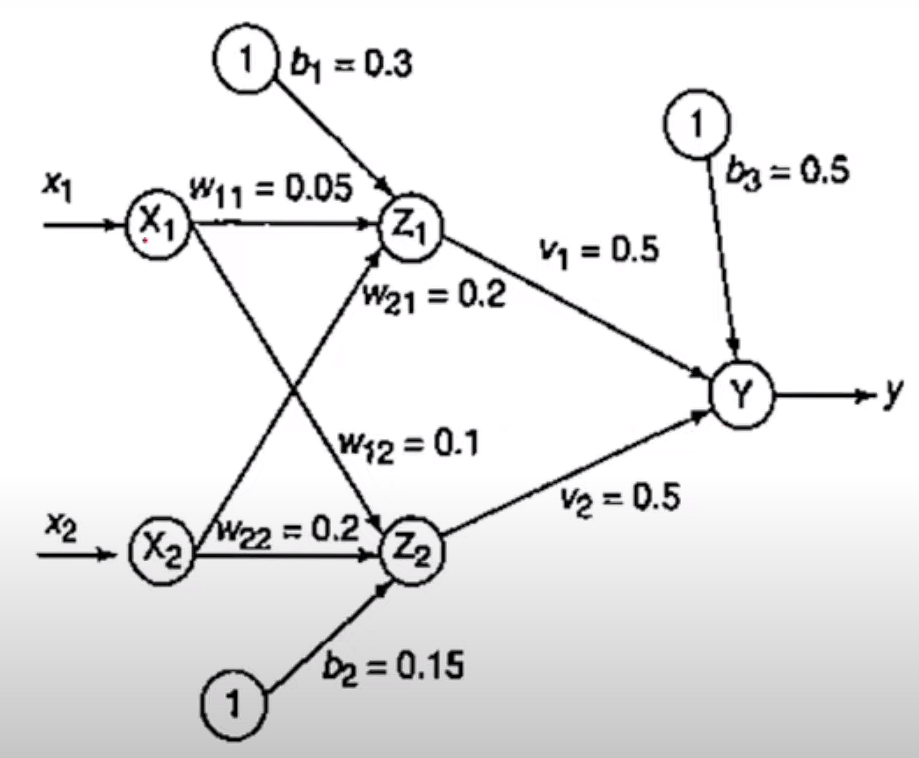
**Experiment No.: 07**

**Aim:** Write a program to implement logical XOR using the Madaline network



**Steps:**

1. **Initialize Weights and Biases:**
   * Set initial values for weights (**w11**, **w12**, **w21**, **w22**, **v1**, **v2**) and biases (**b1**, **b2**, **b3**).
2. **Define Inputs:**
   * Define the input combinations (xi) for XOR: (1, 1), (1, -1), (-1, 1), (-1, -1).
3. **Iterative Training:**
   * Use a while loop to iterate until convergence or a maximum number of iterations (**count <= 4**).
4. **Compute Zin1 and Zin2:**
   * Calculate **zin1** and **zin2** using the input values, weights, and biases.
5. **Activation Function (Step Function):**
   * Define an activation function that returns 1 if the input is greater than or equal to 0, else -1.
6. **Apply Activation to Zin1 and Zin2:**
   * Apply the activation function to **zin1** and **zin2** to get **z1** and **z2**.
7. **Compute Yin:**
   * Calculate **yin** using the outputs **z1**, **z2**, weights **v1**, **v2**, and bias **b3**.
8. **Apply Activation to Yin:**
   * Apply the activation function to **yin** to get the final output **y**.
9. **Update Weights and Biases:**
   * If the output (**y**) does not match the target, update the weights and biases based on the error.
     + If the target is -1, update both sets of weights and biases.
     + If the target is +1, update the weights and biases of the input that is closer to 0.
10. **Display Results:**
    * Display the results in a DataFrame showing the inputs, targets, intermediate values, and updated weights and biases.
11. **Check for Convergence:**
    * Check if the target and output columns match. If they do, break out of the loop.
12. **Repeat or Conclude:**
    * Repeat the training process until convergence or the maximum number of iterations is reached.

**Code:**

import numpy as np

import pandas as pd

def Xor(w11, w12, w21, w22, v1, v2, b1, b2, b3, count=0, learning\_rate = 0.5):

while(count<=4):

print("\n")

xi = [(1, 1), (1, -1), (-1, 1), (-1, -1)]

outputs = []

for x in xi:

target = -1 if (x[0] ^ x[1]) == 0 else 1

# Calculate zin1 and zin2

zin1 = x[0] \* w11 + x[1] \* w21 + b1

zin2 = x[0] \* w12 + x[1] \* w22 + b2

# Activation function: z=1 if >=0 else 1

activation = lambda z: 1 if z >= 0 else -1

# Apply activation function to zin1 and zin2

z1 = activation(zin1)

z2 = activation(zin2)

# Calculate yin

yin = (z1 \* v1) + (z2 \* v2) + b3

y = activation(yin)

#if target if -ve update both inputs

if y != target and target==-1:

w11 = w11 + learning\_rate \* (target - zin1) \* x[0]

w21 = w21 + learning\_rate \* (target - zin1) \* x[1]

b1 = b1 + learning\_rate \* (target - zin1)

w12 = w12 + learning\_rate \* (target - zin2) \* x[0]

w22 = w22 + learning\_rate \* (target - zin2) \* x[1]

b2 = b2 + learning\_rate \* (target - zin2)

#if target if +ve update input close to zero

else:

zin1\_diff = 0-zin1

zin2\_diff = 0-zin2

if(zin1\_diff < zin2\_diff):

w11 = w11 + learning\_rate \* (target - zin1) \* x[0]

w21 = w21 + learning\_rate \* (target - zin1) \* x[1]

b1 = b1 + learning\_rate \* (target - zin1)

else:

w12 = w12 + learning\_rate \* (target - zin2) \* x[0]

w22 = w22 + learning\_rate \* (target - zin2) \* x[1]

b2 = b2 + learning\_rate \* (target - zin2)

outputs.append([x[0], x[1],target, zin1, zin2, z1, z2, yin, y, w11, w21, b1, w12, w22, b2])

# Display results

output\_frame = pd.DataFrame(outputs, columns=['Input1', 'Input2', 'Target', 'zin1', 'zin2','Z1', 'Z2', 'Yin', 'Output', 'w11', 'w21', 'b1', 'w12', 'w22', 'b2'])

print(output\_frame.to\_string(index=False))

if output\_frame['Target'].equals(output\_frame['Output']):

print("Target and Output columns matched.")

break

count=count+1

w11 = 0.05

w12 = 0.1

w21 = 0.2

w22 = 0.2

v1 = 0.5

v2 = 0.5

b1 = 0.3

b2 = 0.15

b3 = 0.5

# Run the XOR function

Xor(w11, w12, w21, w22, v1, v2, b1, b2, b3,0)

**Output:**

