import numpy as np

# Function to generate Walsh matrix

def generate\_walsh\_matrix(n):

    if n == 1:

        return np.array([[1, 1], [1, -1]])

    else:

        prev\_matrix = generate\_walsh\_matrix(n-1)

        upper = np.concatenate((prev\_matrix, prev\_matrix), axis=1)

        lower = np.concatenate((prev\_matrix, -prev\_matrix), axis=1)

        return np.concatenate((upper, lower), axis=0)

# Function to encode data using CDMA technique

def cdma\_encode(data\_bits, walsh\_matrix):

    encoded = np.zeros\_like(walsh\_matrix[0])

    for i, bit in enumerate(data\_bits):

        encoded += bit \* walsh\_matrix[i]

        return encoded

# Function to decode a specific channel in CDMA

def cdma\_decode(encoded\_data, walsh\_matrix, channel):

    decoded = np.dot(encoded\_data, walsh\_matrix[channel]) / len(walsh\_matrix[channel])

    return decoded

# Get number of data bits from user

num\_bits = int(input("Enter the number of data bits: "))

# Get data bits from user

cdata\_bits = []

print("Enter the data bits:")

for i in range(num\_bits):

    bit = int(input(f"Enter bit {i + 1}: "))

    data\_bits.append(bit)

#Defining

x=1

while (2\*\*x<num\_bits):

    x=x+1

# Generate Walsh matrix

walsh = generate\_walsh\_matrix(x)

np.set\_printoptions(threshold=np.inf)

# Set printing options

print("\nGenerated Walsh Matrix:")

print(walsh)

# Encode data using CDMA technique

encoded\_data = cdma\_encode(data\_bits, walsh)

# Get channel to decode from user

decode\_channel = int(input(f"\nEnter the channel to decode (0 - {num\_bits - 1}): "))

# Decode the selected channel

decoded\_channel = cdma\_decode(encoded\_data, walsh, decode\_channel)

print(f"\nEncoded data: {encoded\_data}")

print(f"Decoded data from channel {decode\_channel}: {decoded\_channel}")

print(f"Input data for comparison: {data\_bits}")