Assignment 1

Bit Stuffing/Unstuffing

Name: Shirish Manoj Bobde

Reg. No.: 812

Roll No.: ECE/21152

Problem Statement:

Consider an arbitrary framing of 64 bits having start and end flag pattern is 1110111. Design the best bit stuffing/unstuffing mechanism interns of bandwidth utilisation.

Solution:

To design an efficient bit stuffing/unstuffing mechanism with the given framing pattern, we need to ensure that the transmitted data stream doesn't contain the same pattern as the start and end flags. This is crucial to avoid any ambiguity in detecting the start and end of frames.

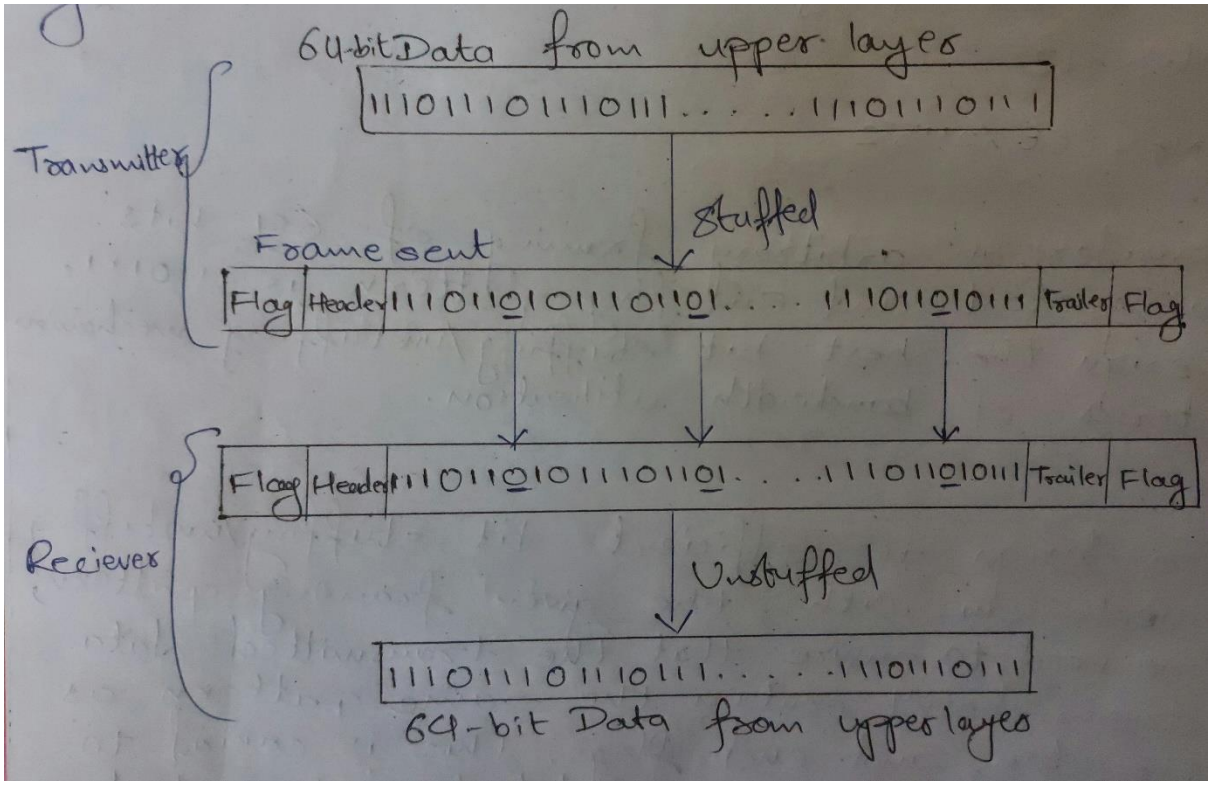
Bit Stuffing Mechanism:

1. Whenever the transmitter encounters pattern ‘111011’ in the data to be transmitted (excluding the flag pattern), it inserts a '0' bit after the sixth '1'.
2. This ensures that the flag pattern '1110111' doesn't appear within the transmitted data stream.
3. The receiver, upon receiving the data, removes the stuffed bits (the '0' bits) whenever it encounters pattern 111011.

Unstuffing Mechanism:

1. The receiver keeps track of the bit pattern.
2. Whenever it encounters bit pattern 111011 followed by a '0', it removes the '0' bit and continues decoding the data.
3. This ensures that the original data is reconstructed correctly.

Here worst-case scenario is assumed to establish most bandwidth utilised mechanism.



Why the bit was stuffed after 6th position?

1. As the demand of creating bandwidth efficient mechanism, adding bit at last second position has **50% probability** that the flag pattern is going to be encountered at receiver end.
2. By moving towards left from last bit, probability of flag encounter changes as **50%, 25%, 12.5%, 6.25%** and so on. So, to be a bandwidth efficient the stuffing should be as less as possible.
3. Another method is to identify flag pattern, if last bit is 0: then stuff ‘1’ at last second position else stuff '0' at last second position. This will be method the most bandwidth efficient method.

Bandwidth Efficiency:

Frame size = **64 bits**

Assuming worst case scenario we have to stuff bit after 6th position for best bandwidth efficiency.

i.e. 11101101011101101O…

Stuffed bits count = Frame size/Repetitive length

= 64/8(before stuffing)

Stuffed bits count = **8**

So, 8 bits should be stuffed in worst case scenario.

Stuffed Frame size = 64+8 = **72 bits**

Bandwidth Efficiency (%) = (Original Frame Size) / (Stuffed Frame size) *100%

= (64/72) *100%

Bandwidth Efficiency (%) = 88.89%

Code:

```
# Input from the user
binary_input = input("Enter a binary array (e.g., '111011101110'): ")
binary_array = [int(bit) for bit in binary_input]

#bit stuffing
result = []
pattern = [1, 1, 1, 0, 1, 1]
i = 0

#zero adding process
while i < len(binary_array):
    # Check if the current position matches the pattern
    if binary_array[i:i+len(pattern)] == pattern:
        # Add the pattern to the result
        result.extend(pattern)
```

```
        # Add a '0' after the pattern
        result.append(0)
        # Move the index to skip the pattern
        i += len(pattern)
    else:
        # If the pattern is not found, add the current bit to the result
        result.append(binary_array[i])
        i += 1

# Print the original binary array and the result
print("Original binary array                    :", binary_array)
print("Result after adding '0' after every '111011' pattern   :", result)



#unstuffing
result1 = []
i = 0
while i < len(result):
    # Check if the current position matches the pattern
    if result[i:i+len(pattern)] == pattern:
        # Add the pattern to the result
        result1.extend(pattern)
        # Skip the '0' added after the pattern
        i += len(pattern) + 1
    else:
        # If the pattern is not found, add the current bit to the result
        result1.append(result[i])
        i += 1

# Print the original binary array and the result
print("Result after removing '0' after every '111011' pattern :", result1)
```