In [3]:
```
import cv2
import numpy as np

# Загружаем изображение в градациях серого
image = cv2.imread('sar_1_gray.jpg', cv2.IMREAD_GRAYSCALE)
if image is None:
    raise FileNotFoundError("Ошибка: изображение не загружено. Проверьте

# Сохраняем оригинальное изображение в текстовый файл
np.savetxt('original_image.txt', image, fmt='%d')
# Выполняем преобразование Хаара
rows, cols = image.shape
LL = (image[0::2, 0::2] + image[0::2, 1::2] + image[1::2, 0::2] + image[1
LH = (image[0::2, 0::2] + image[0::2, 1::2] - image[1::2, 0::2] - image[1
HL = (image[0::2, 0::2] - image[0::2, 1::2] + image[1::2, 0::2] - image[1
HH = (image[0::2, 0::2] - image[0::2, 1::2] - image[1::2, 0::2] + image[1
```

In [ ]:
```
# Квантуем коэффициенты
levels = 4 #(как в примере, =4)
min_val = np.min(LH)
max_val = np.max(LH)
step = (max_val - min_val) / levels
LH_quantized = np.floor((LH - min_val) / step) * step + min_val

min_val = np.min(HL)
max_val = np.max(HL)
step = (max_val - min_val) / levels
HL_quantized = np.floor((HL - min_val) / step) * step + min_val

min_val = np.min(HH)
max_val = np.max(HH)
step = (max_val - min_val) / levels
HH_quantized = np.floor((HH - min_val) / step) * step + min_val
```

In [ ]:
```
# Кодируем с помощью кодирования с длиной пробега
def run_length_encode(data):
    flattened = data.flatten()
    encoded = []
    prev = flattened[0]
    count = 1
    for value in flattened[1:]:
        if value == prev:
            count += 1
        else:
            encoded.append((prev, count))
            prev = value
            count = 1
    encoded.append((prev, count))  # Добавляем последний элемент
    return encoded
LH_encoded = run_length_encode(LH_quantized)
HL_encoded = run_length_encode(HL_quantized)
HH_encoded = run_length_encode(HH_quantized)
```

In [ ]:
```
# Сохраняем результаты в файл
with open('haar_output.txt', 'w') as f:
    np.savetxt(f, LL, fmt='%f')
```

```
    for label, encoded in zip(["LH", "HL", "HH"], [LH_encoded, HL_encoded
        f.write(f"{label}:\n")
        for value, count in encoded:
            f.write(f"{value} {count}\n")
```

In [2]:
```
# Вычисляем размеры
original_size = image.nbytes
LL_size = LL.nbytes
LH_size = sum(len(str(value)) + len(str(count)) + 2 for value, count in L
HL_size = sum(len(str(value)) + len(str(count)) + 2 for value, count in H
HH_size = sum(len(str(value)) + len(str(count)) + 2 for value, count in H
compressed_size = LL_size + LH_size + HL_size + HH_size
# Выводим информацию о размерах
print(f"Исходный размер: {original_size} байт")
print(f"Размер после сжатия: {compressed_size} байт")
print(f"Коэффициент сжатия: {original_size / compressed_size:.2f}")
```

```
Исходный размер: 240000 байт
Размер после сжатия: 1341486 байт
Коэффициент сжатия: 0.18
```

In [ ]:
```
```