CHAPTER 5

PAPR REDUCTION USING HUFFMAN AND ADAPTIVE HUFFMAN CODES

5.1 INTRODUCTION

In this work the peak powers of the OFDM signal is reduced by applying Adaptive Huffman Codes (AHC). First the encoding procedure is carried out for Huffman codes (Ashraf Eltholth et al 2008). Then the algorithm for adaptive Huffman codes was constructed. PAPR technique applied in this study is coding scheme. The PAPR values is calculated and compared for the applied two codes. Simulation result shows that the encoding of adaptive huffman codes shows a good performance of PAPR reduction when compared to huffman codes in OFDM systems.

5.2 HUFFMAN ENCODING

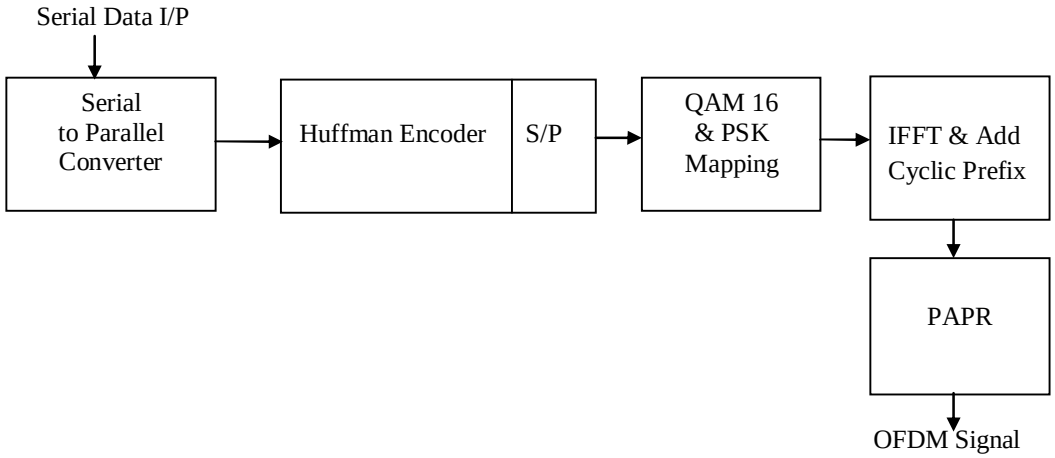


Figure 5.1 Transmitter block diagram of the OFDM using Huffman Encoder S/P : Serial to parallel converter

Figure 5.1 shows the transmitter block diagram of the OFDM system using Huffman Encoder. In an OFDM system, each channel is broken into various sub-carriers. The use of sub-carriers makes optimal use of the frequency spectrum but also requires additional processing by the transmitter and receiver. This additional processing is necessary to convert a serial data stream into several parallel data streams which is to be divided among the individual carriers. Randomly generated inputs from S/P converter may include characters (both upper case A-Z and lower case a-z), special characters and numbers. The serial data streams placed in individual parallel sub carrier forms the output of S/P converter.

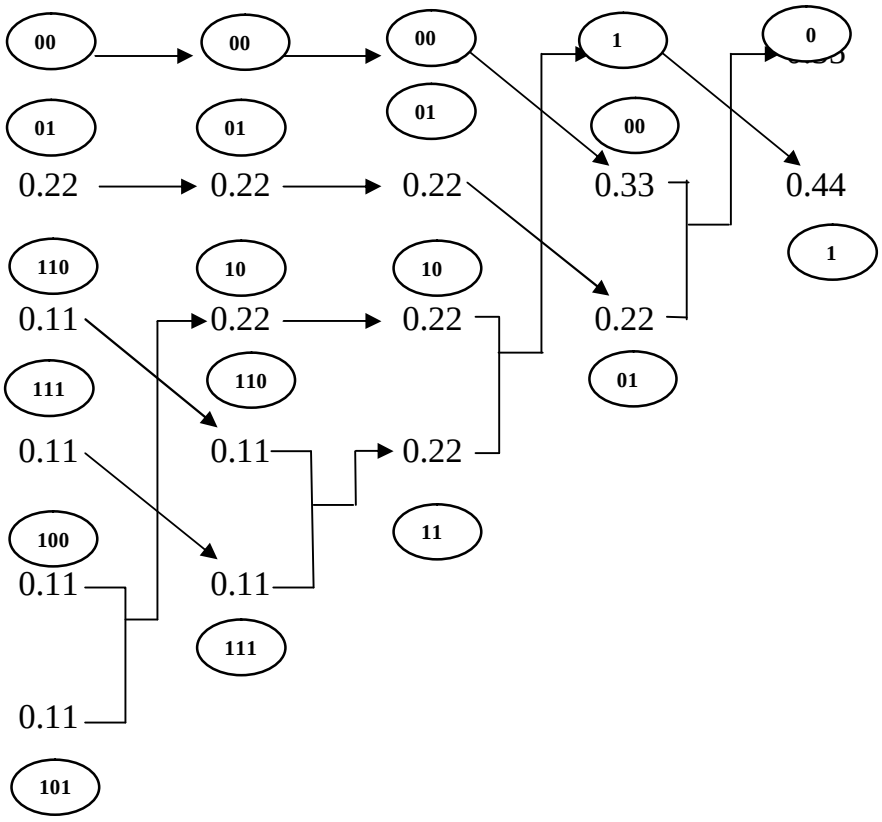
The output from the serial to parallel converter is encoded using Huffman codes. Huffman coding will reduce the PAPR because Huffman coding assigns fewer bits for frequently occurred symbols and more bits for seldom occurred. The features of Huffman encoding are to assign the variable length codes to each character based on their probability values. To implement Huffman codes, the compression and uncompression algorithms must agree on the binary codes which are used to represent each character (Xrysovalantis et al 2007).. This can be handled in one of two ways. The simplest is to use a predefined the encoding table that is always the same, regardless of the information being compressed. More complex schemes use encoding which is optimized for the particular data being used. This requires that the encoding table can be included in the compressed data to be used by the uncompressed program, but both methods are common. In this work, we have chosen the second method, this implies that the encoding table was considered but this will not affect the bandwidth efficiency because the

compression gained by the Huffman coding will counteract the increase in data rate due to the side information to be transmitted.

The probabilities for each character is calculated and it is coded based on their probability values. The characters are coded accordingly such that the most frequently occurring character can be coded with single bit. The encoding mechanism results in long serial bit streams of ones and zeros. These long serial strings are again converted to parallel bit streams before mapping. The encoding of Huffman codes takes place in three steps. The first step is to arranging the inputs based on their probability, the Second step is to create code book for each character and the third step is to assign the codes for each symbol (Reza Hashemian, 2004). The Table 5.1 defines that, how each input is encoded based on their probabilities using variable length codes and the Huffman encoding procedure is shown in Fig 5.2. Let us consider the inputs 1, 6, 5, 1, 1, 4, 2, 3 and 4.

Table 5.1 Huffman encoding table

Inputs	Probability
1	0.33
2	0.11
3	0.11
4	0.22
5	0.11
6	0.11



Encoded output data : 001011000000011101110

Figure 5.2 Huffman encoding procedure

The encoded data is divided into several parallel data streams or channels, one for each sub-carrier. Each sub-carrier is modulated with a conventional modulation scheme such as QAM and PSK at a low symbol rate maintaining total data rates similar to conventional single carrier modulation schemes in the same bandwidth. The frequency domain data from the mapping is converted into time domain using IFFT. After IFFT cyclic prefix was added. Finally PAPR is calculated from the generated OFDM signal.

5.3 ALGORITHM FOR HUFFMAN CODING

Step1: Get the input characters.

Step2: Convert the serial input sequence to parallel sequence.

Step3: Calculate the probability for each character.

Step4: Assign bits according to probability.

Step5: Before Mapping check for zero padding.

Step6: Map the zero-padded bits using the modulation techniques QAM and PSK

Step7: Compute IFFT for the mapped sequence and add cyclic prefix.

Step8: Calculate PAPR using the formula.

Step9: Plot the CCDF graph between threshold and CCDF values.

5.4 FLOW CHART FOR HUFFMAN CODING

The flow chart of Huffman coding procedure is shown in Figure 5.3.

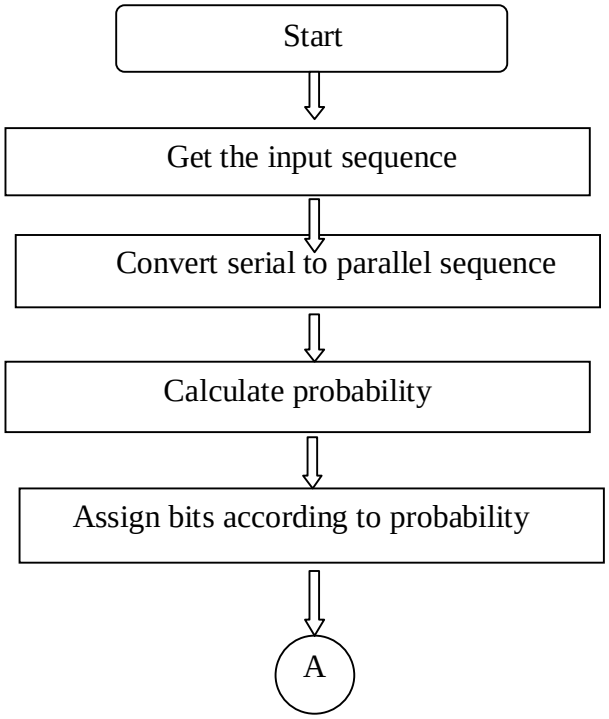


Figure 5.3 (Continued)

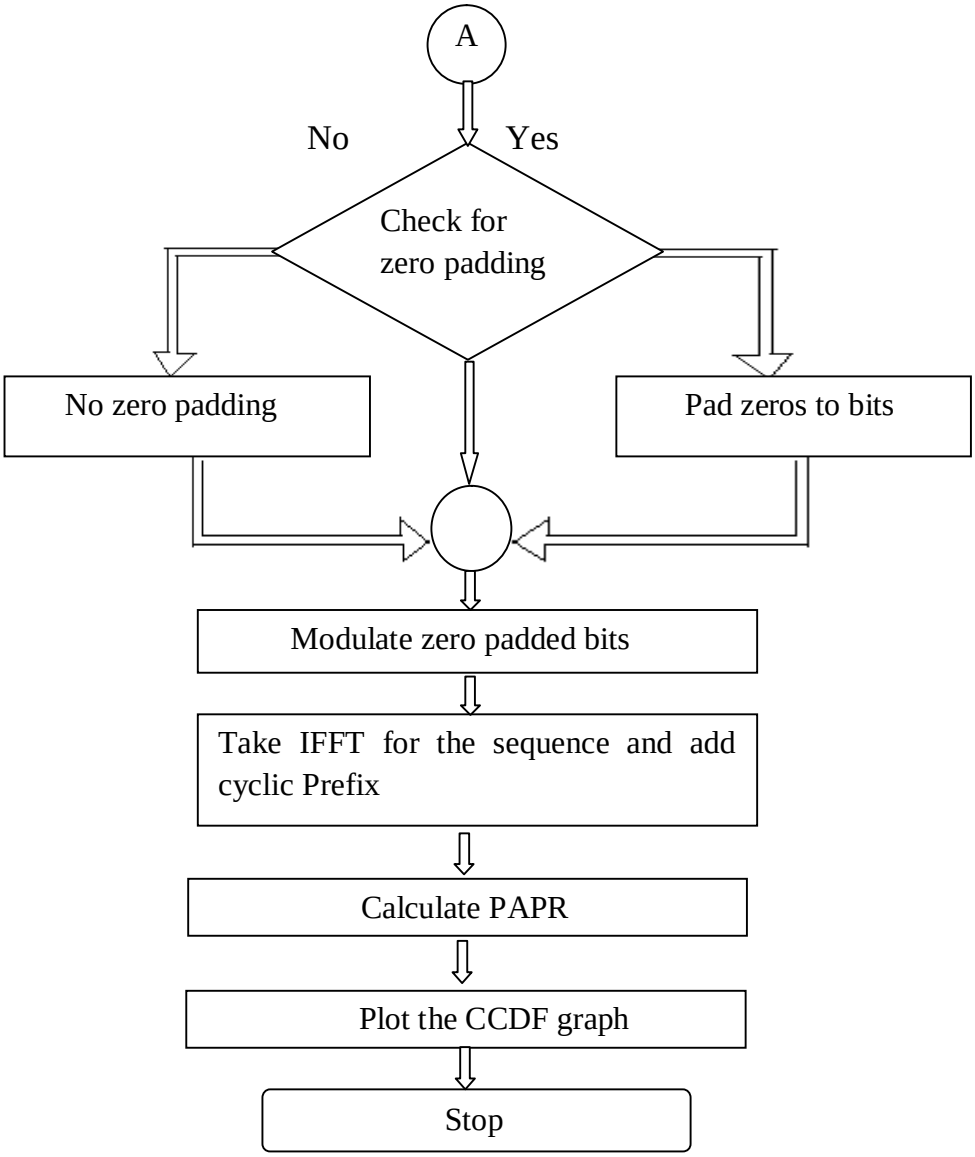


Figure 5.3 Flow chart of Huffman coding

5.5 ADAPTIVE HUFFMAN ENCODING

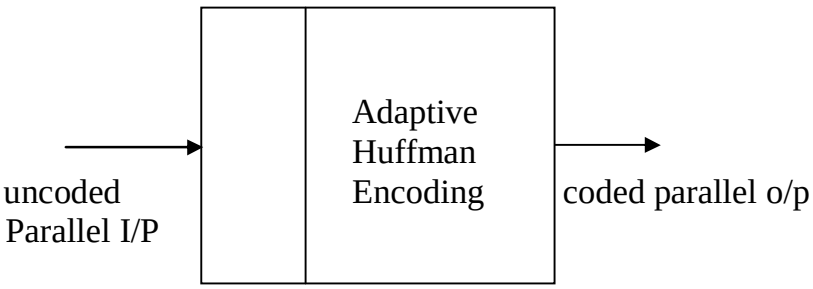


Figure 5.4 Block diagram of Adaptive Huffman Encoder

Figure 5.4 represents the block diagram of Adaptive Huffman Encoding. Adaptive Huffman coding or Dynamic Huffman is a coding technique for lossless compression based on Huffman coding. It permits building the code as the symbols is being transmitted, having no initial knowledge of the source and allows generating the code from a single scanning of the source (Pushpa Suri and Madhu Goel,2009). Also another property of this method of coding is that same characters situated on distinct positions are differently encoded. Adaptive Huffman Coding has the advantage over static coding that the entire dataset does not need to be known in advance and the dictionary does not need to be transmitted separately from the data (Rashmi et al 2010). The Parallel data from the S/P converter block enter as input to the Adaptive Huffman encoding block. Vitter algorithm is used for encoding the input streams (Rajeshree Nair and Ben Soh 2006).

5.5.1 Vitter Algorithm

Step1: Code is represented as a tree structure in which every node has a corresponding weight and a unique number.

Step2: Numbers go down, and from right to left.

Step3: Weights must satisfy the sibling property, which states that nodes must be listed in the order of decreasing weight with each node adjacent to its sibling.

Step4: The weight is merely the count of symbols transmitted which codes are associated with children of that node.

Step5: A set of nodes with same weights make a block.

Step6: To get the code for every node, in case of binary tree we could just traverse all the path from the root to the node, writing down (for example) "1" if we go to the right and "0" if we go to the left.

Step7: We need some general method to transmit symbols that are "Not Yet Transmitted" (NYT) . We could use, for example, transmission of binary numbers for every symbol in alphabet.

Step8: Encoder and Decoder start with only the root node, which has the maximum number. In the beginning it is our initial NYT node.

Step9: When we transmit an NYT symbol, we have to transmit code for the NYT node, then for its generic code.

Step10: For every symbol that is already in the tree, we only have to transmit code for its leaf node.

Step11: For every symbol transmitted both the transmitter and receiver execute the update procedure:

Step11.1: If current symbol is NYT, add two child nodes to NYT node. One will be a new NYT node the other is a leaf node for our symbol. Increase weight for the new leaf node and the old NYT then go to step 11.4. If not, go to symbol's leaf node.

Step11.2: If this node does not have the highest number in a block, swap it with the node having the highest number, except if that node is its parent.

Step11.3: Increase weight for current node.

Step11.4: If this is not the root node go to parent node then go to step 11.2.

5.5.2 Sibling Property

Sibling property is an important parameter in vitter algorithm. The condition for sibling property is the weight of the nodes from left to right until to the top must be in descending order. (i.e $w_2 \leq w_3 \leq w_1$, for the tree.). Fig 5.5 shows the nodes representing the sibling property.

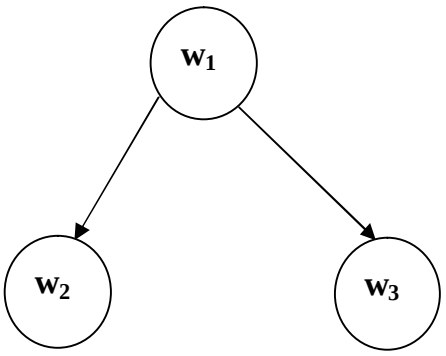
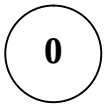


Figure 5.5 Nodes representing the sibling property

After entering each symbol the condition for sibling property need to be checked only if the condition is satisfied the next symbol that will be added to the tree, if the condition is not satisfied the nodes need to be interchanged to satisfy the condition, this step will continue until satisfying the condition.

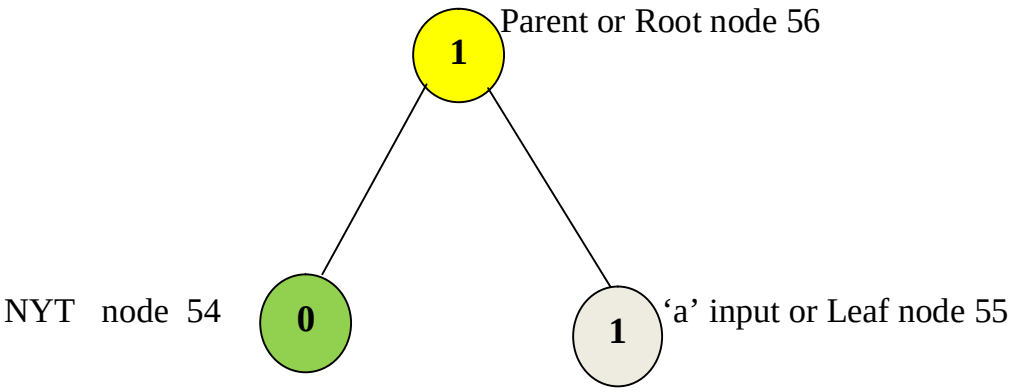
Let us consider the input for the variables a , a ,r, d and v

The starting node will be not yet transmitted node (NYT). Weight of the node is 0. The selected node number to start the procedure is ‘56’.

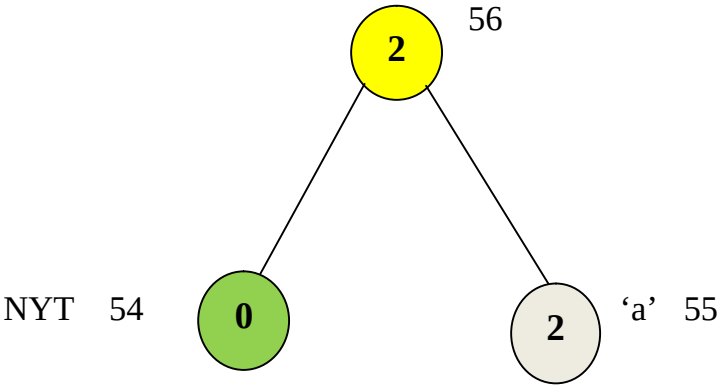


NYT 56

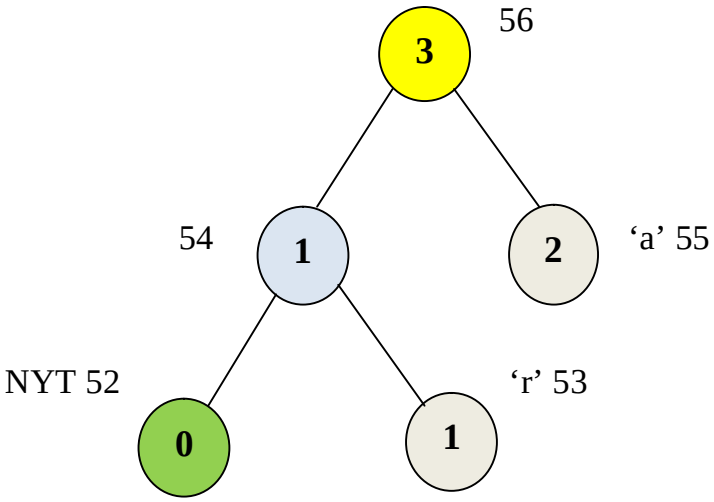
The first symbol is a. To add this symbol with tree we create a leaf node along with one NYT node. Now add the two node with the root node. The weight of the node in the left side is 0 and the weight of the node in the right side is 1. The node number will get decreased from right to left. Check for sibling property condition . If it is satisfied go for the next symbol



Next input symbol is 'a' where was already represented. So just increase the weight of the input node by one time and check for sibling property.

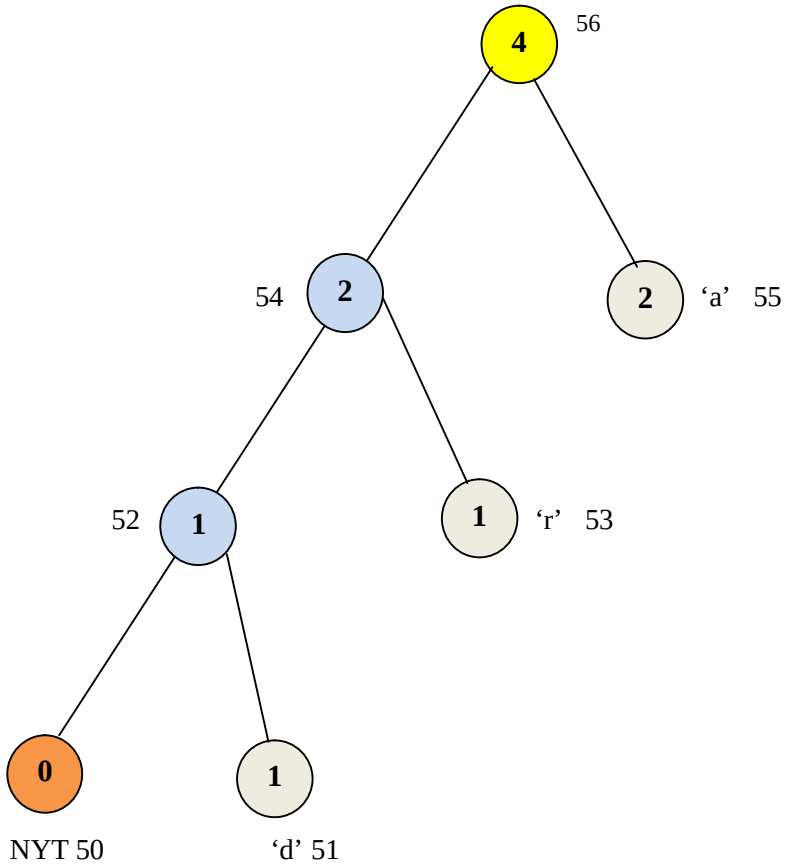


The next input symbol is 'r' which is new to the tree. Hence create a new node along with NYT node and increase the weight. Update the tree and Check for Sibling Property (CSP). Here the condition is satisfied and hence go for next input symbol.



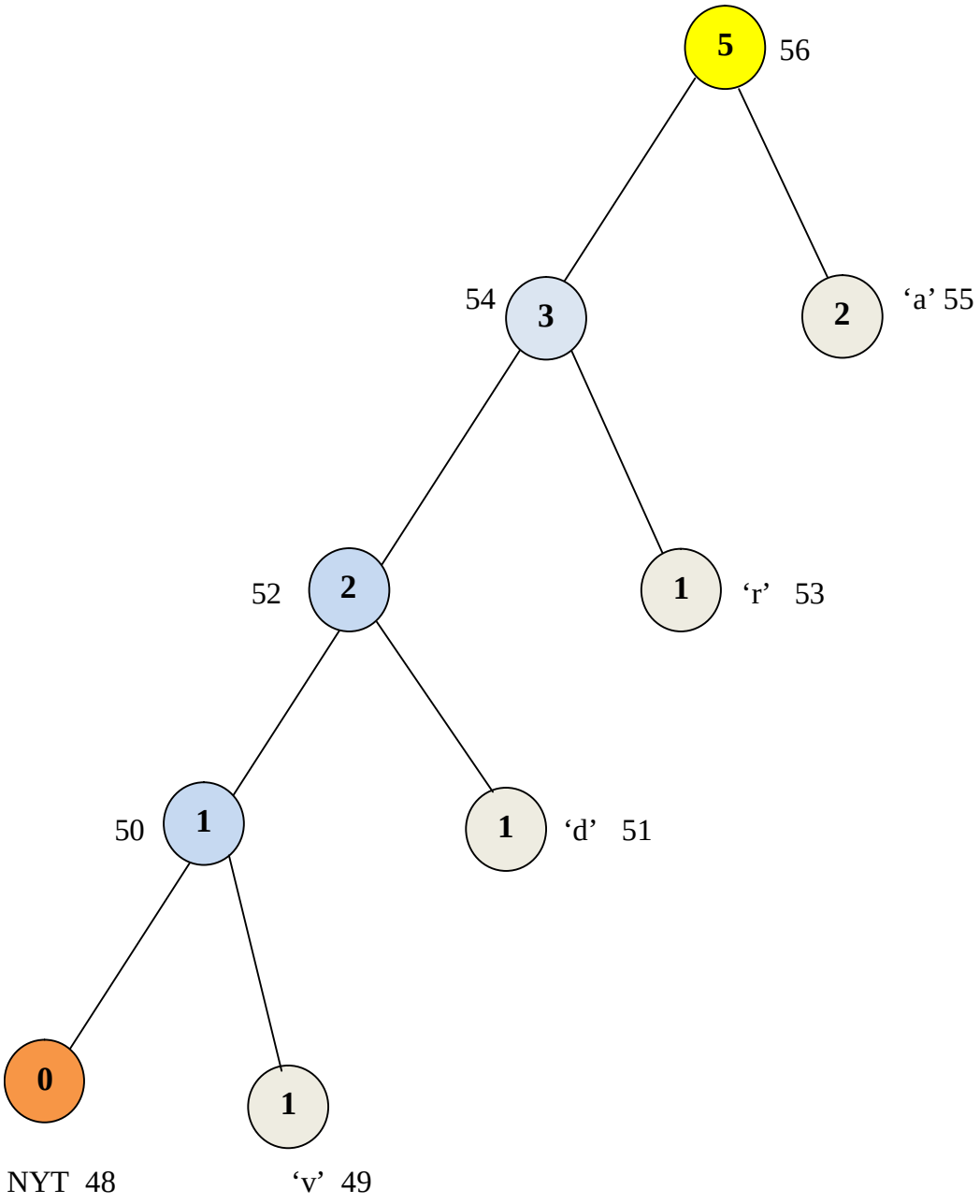
CSP : 0≤ 1 ≤ 1 ≤ 2 ≤ 3

The next input symbol is ‘d’ which is new to the tree. Hence repeat the steps as we proceeded for the input symbol ‘r’. The sibling property is satisfied here. Assign numbers to all the nodes.



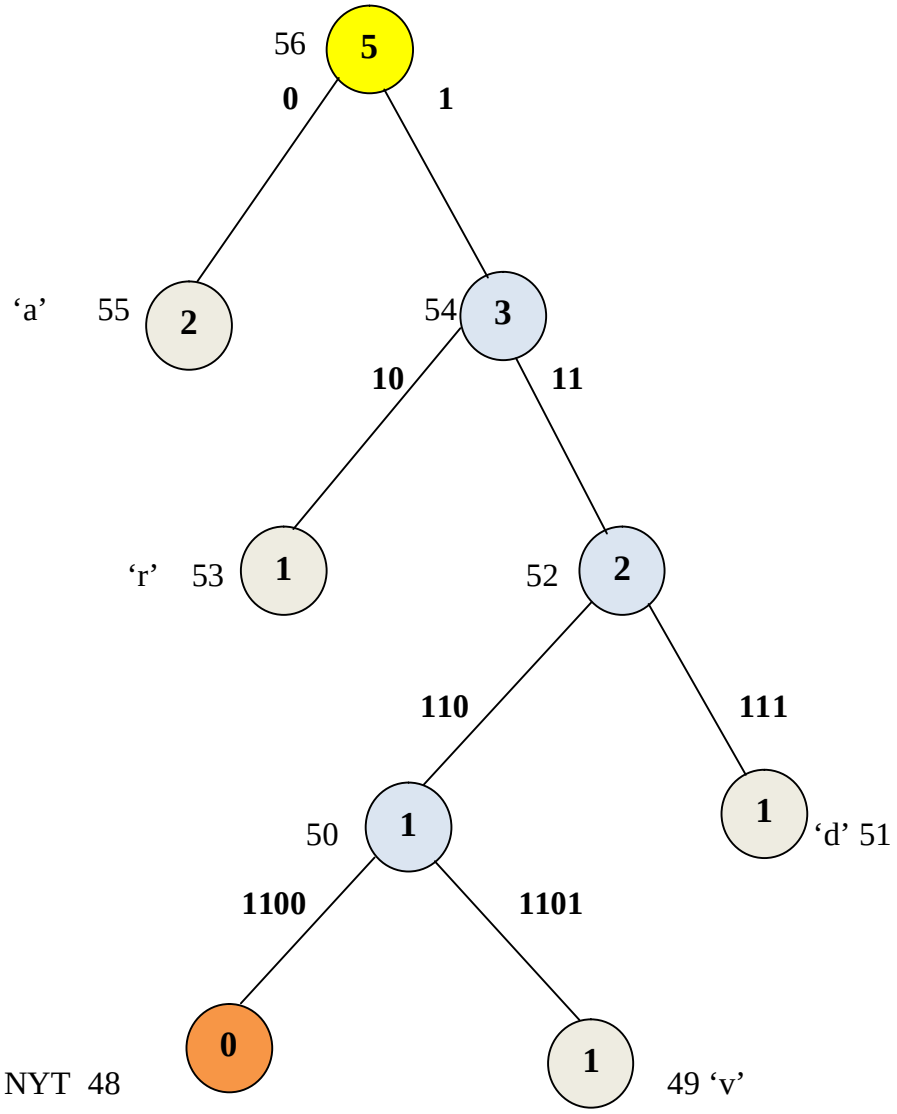
CSP : 0 ≤ 1 ≤ 1 ≤ 1 ≤ 2 ≤ 2 ≤4

The next input symbol is v. It is new to the tree, create a new node and a NYT node. Assign weights and node number. Use update procedure to update the tree and check for sibling property.



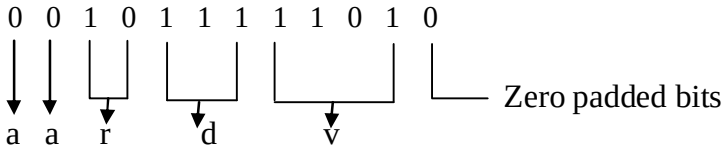
CSP : 0 ≤ 1 ≤ 1 ≤ 1 ≤ 2 ≤ 1 ≤ 3 ≤ 2 ≤ 5

In the above procedure, the sibling property is unsatisfied for the node 54 and 55. So swap the nodes 54 and 55, again check for sibling property. Again the sibling property is unsatisfied for the node 52 and 53. Then swap the nodes 52 and 53, check for the sibling property.



CSP : 0 ≤ 1 ≤ 1 ≤ 1 ≤ 1 ≤ 2 ≤ 2 ≤ 3 ≤ 5

Now the condition is satisfied and v is the last input symbol in the sequence. Hence stop updating the tree. Now for each symbol write the transmitted code. Assign bit ‘0’ bit for left side and ‘1’ bit for right side. Then the obtained bits is a- 0 , r - 10 , d- 111, v-1101. The encoded bits for the given inputs a ,a, r, d and v is



The encoding mechanism results in long serial bit streams of ones and zeros. This long serial strings are again converted to parallel bit streams before mapping. Except the encoding mechanism all the transmitter block concepts are same as in the Huffman codes.

5.6 ALGORITHM FOR ADAPTIVE HUFFMAN CODING

Step1: Get the input characters.

Step2: Convert the serial input sequence to parallel sequence.

Step3: Encode the bits using Adaptive Huffman-Vitter Algorithm.

Step4: Before mapping check for zero padding.

Step5: Map the zero-padded bits using modulation techniques (QAM16 and PSK16).

Step6: Compute IFFT for the mapped sequence and add cyclic prefix.

Step7: Calculate PAPR

Step8: Plot the graph between threshold and CCDF values.

5.7 FLOW CHART FOR ADAPTIVE HUFFMAN CODING

Figure 5.6 and 5.7 represents the flow chart of AHF coding and Vitter algorithm.

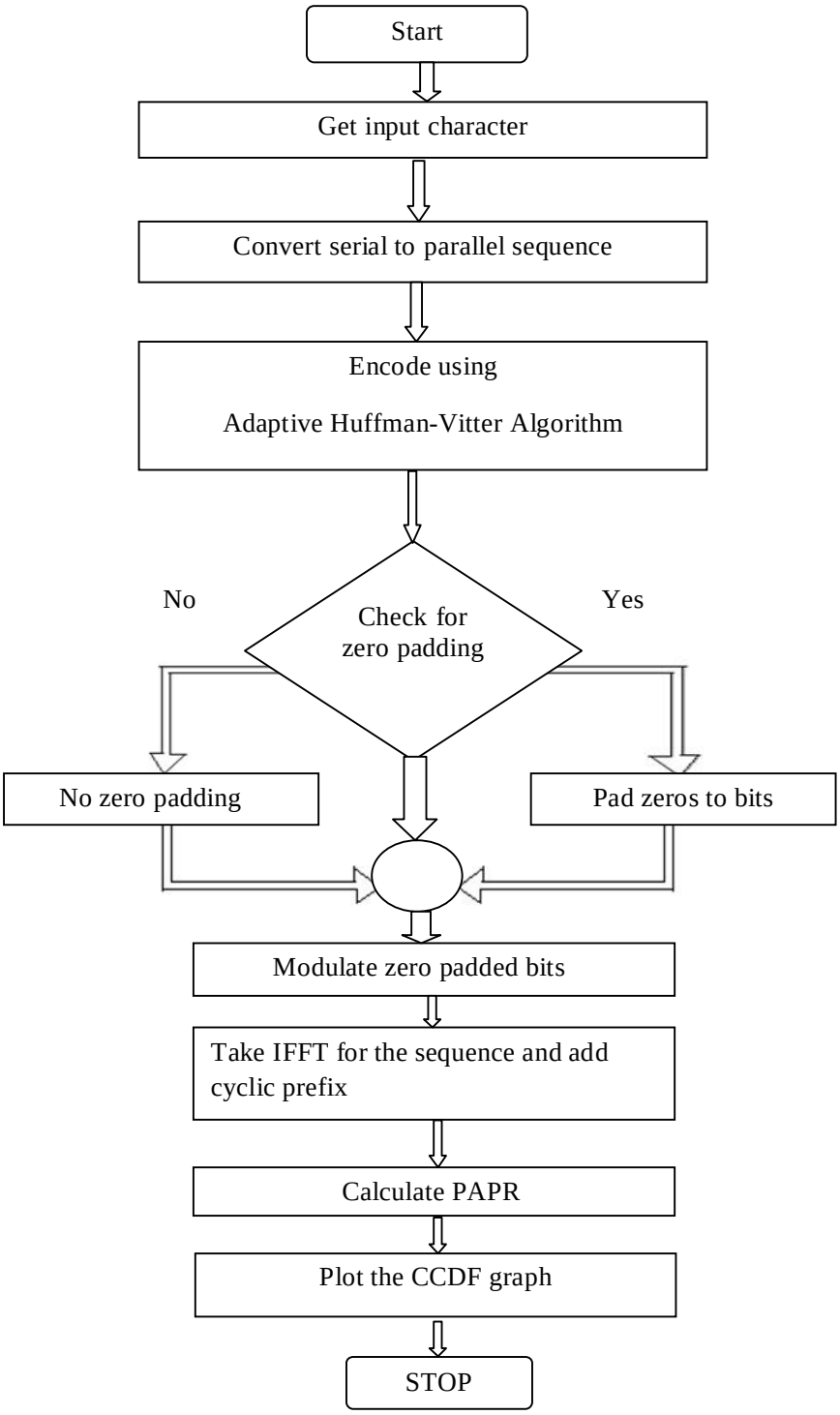


Figure 5.6 Flow chart of adaptive Huffman encoding procedure

5.7.1 Flow Chart of Vitter Algorithm

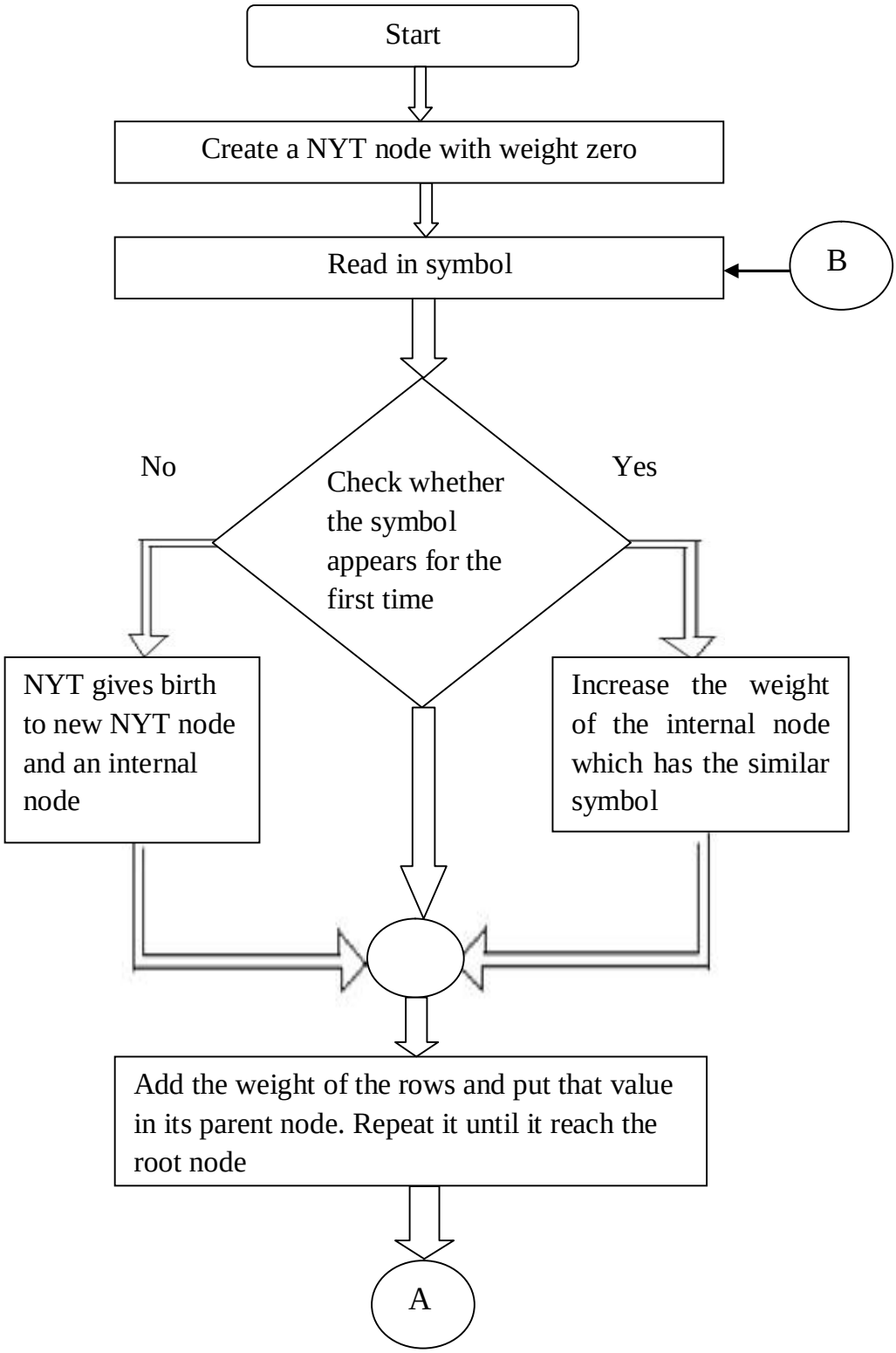


Figure 5.7 (Continued)

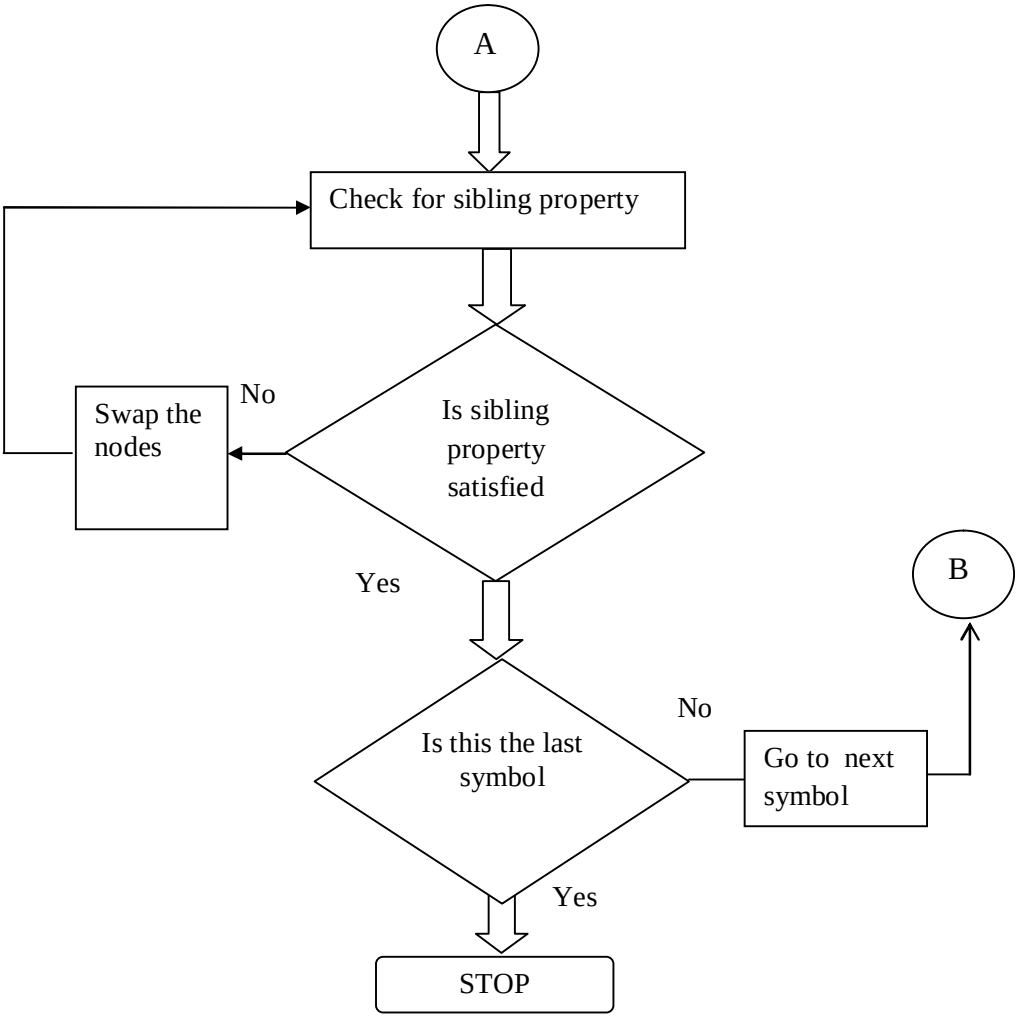


Figure 5.7 Flow chart of Vitter algorithm

5.8 SIMULATION AND RESULTS

In this work the simulation was carried out by using MATLAB software. Randomly generated input is uniformly distributed. The work is constructed for any number of sub carriers and the modulation technique considered in this work is QAM 16 and PSK. For simulation the selected sub carriers are N=32, 64, 128 and 256. The Figure 5.8 to Figure 5.11 represents the proposed transmitter block diagram outputs for Adaptive Huffman codes with N=128.

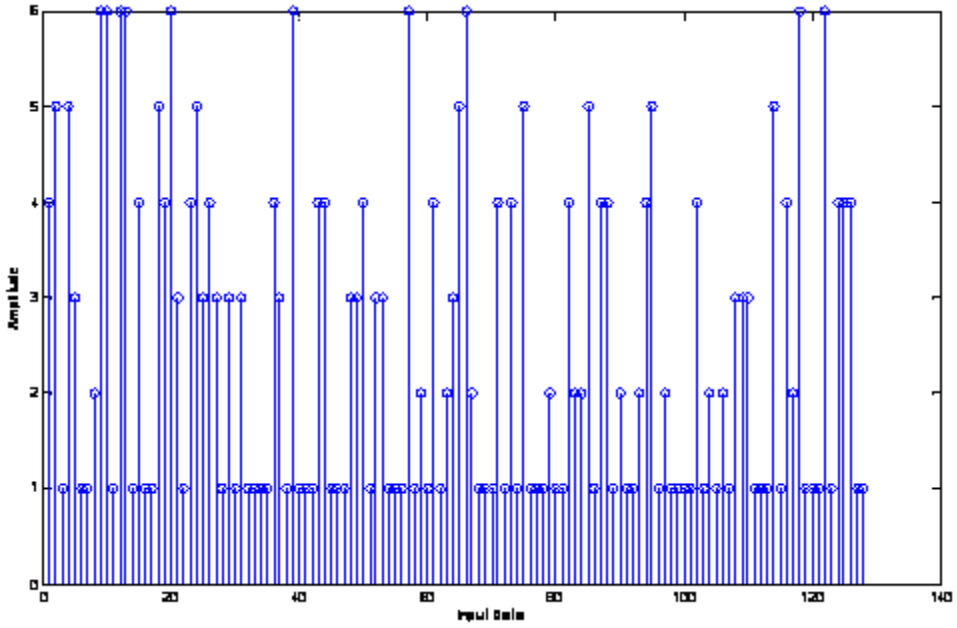


Figure 5.8 Input data streams of N=128

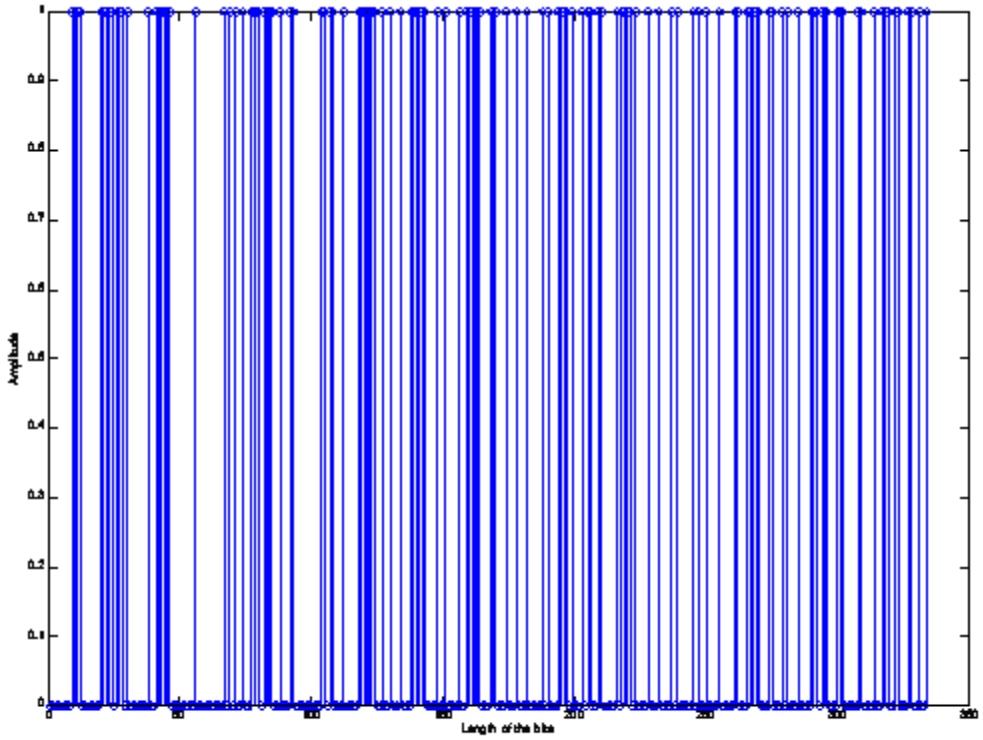


Figure 5.9 Adaptive huffman encoded output of N=128

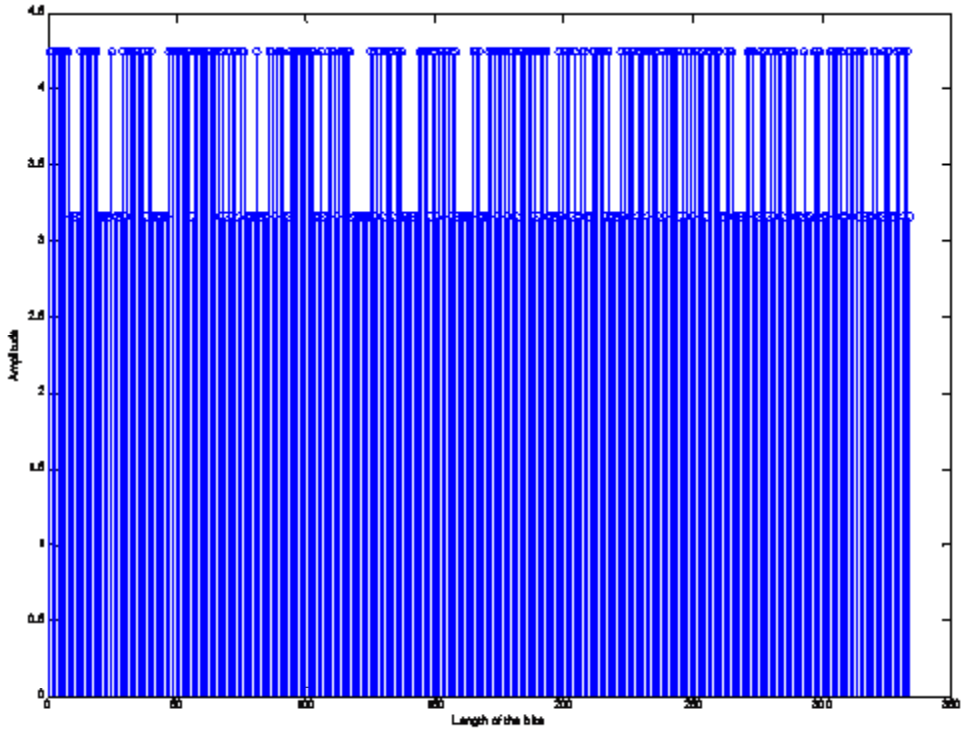


Figure 5.10 QAM Mapping block output of N=128

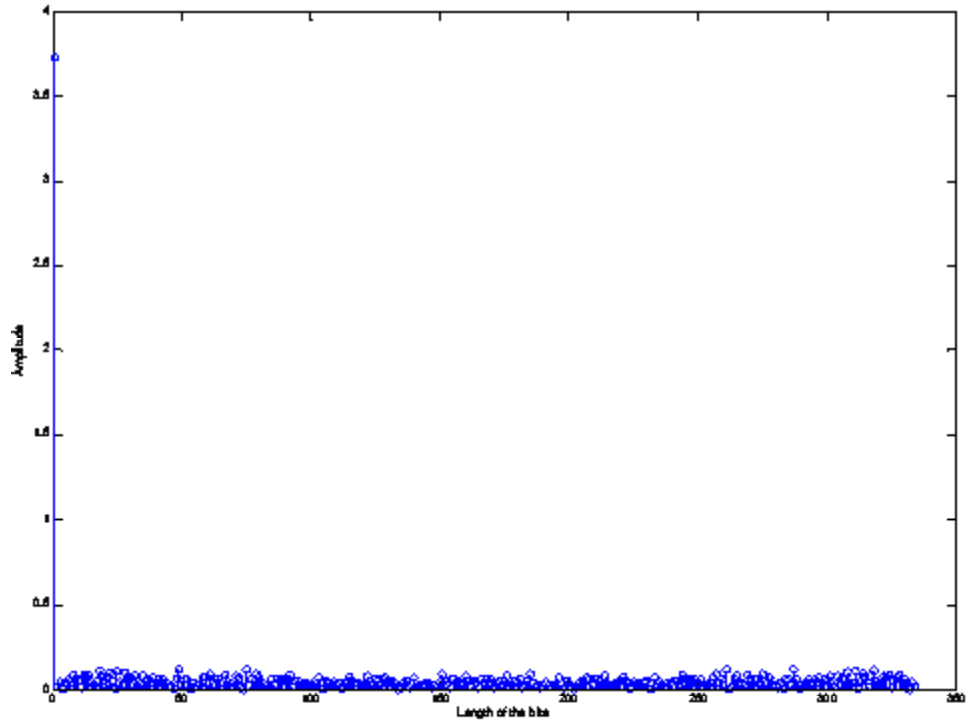


Figure 5.11 IFFT block output of N=128

Simulation result shows the CCDF plot for PAPR reduction using Huffman code, Adaptive Huffman code and PAPR with no compensation (no coding).

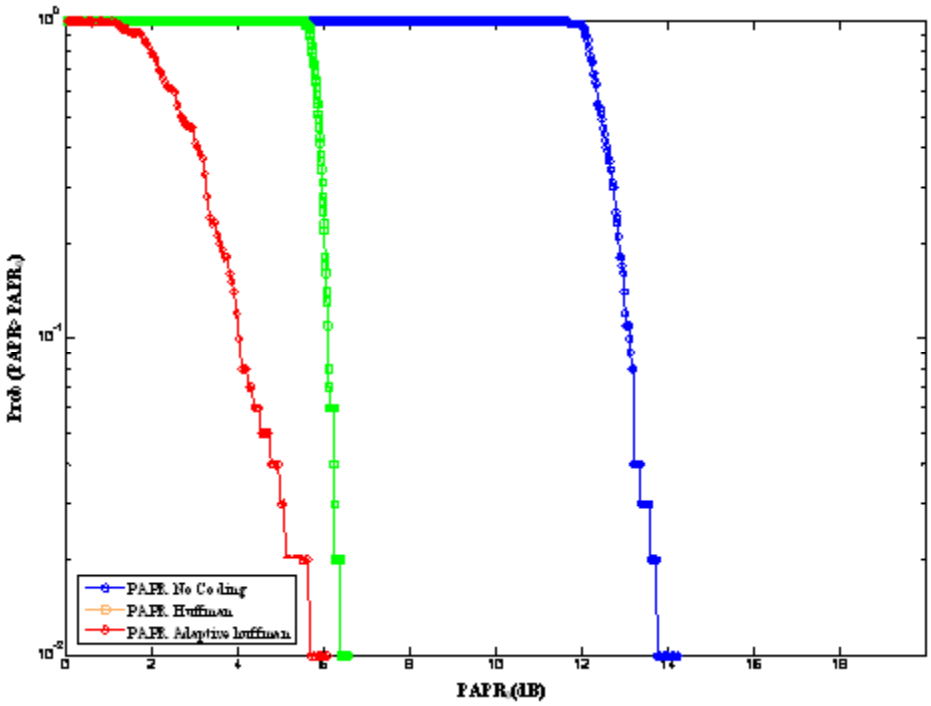


Figure 5.12 CCDF Plot representing PAPR of QAM 16 modulation for N=32

Figure 5.12 represents the CCDF plot for N=32 with QAM 16 modulation. There is a 5.8 dB PAPR reduction for Adaptive Huffman codes. The PAPR difference between Huffman and Adaptive Huffman codes (AHC) is about 0.4 dB as shown in Figure 5.11. The Figure 5.13 represents the CCDF Plot for N=64. In that there is an 8.3 dB reduction for AHC and 10.4 dB for Huffman codes. There is difference of 2.1 dB from AHC to Huffman coding.

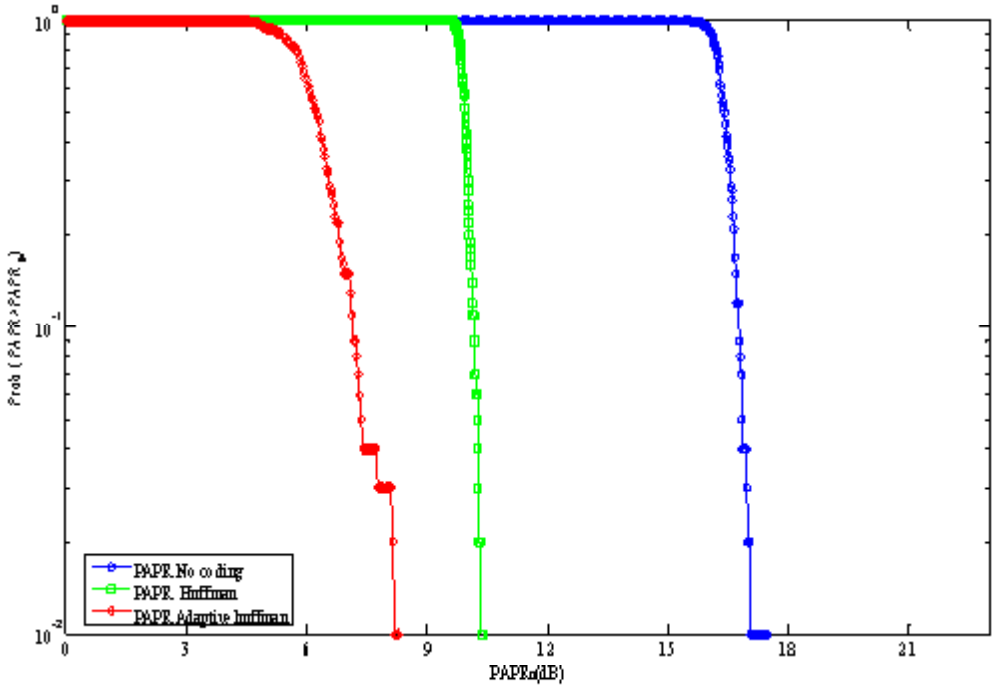


Figure 5.13 CCDF Plot representing PAPR of QAM 16 modulation for N=64

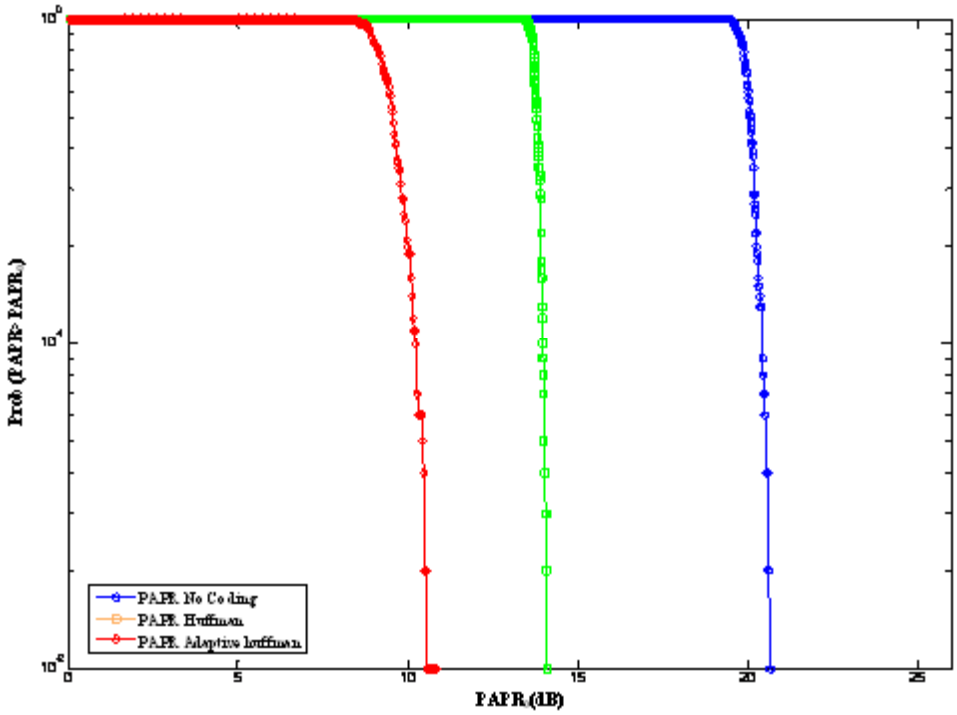


Figure 5.14 CCDF Plot representing PAPR of QAM 16 modulation for N=128

Figure 5.14 represents the CCDF plot for N=128.The performance of AHC coding is best when compared to other codes used in this study. For higher sub carriers N=256 in Figure 5.15, there is a 4.4 dB difference from AHC to huffman codes. So from Figure 5.12 to Figure 5.14 defines that if N increases the PAPR also increases.

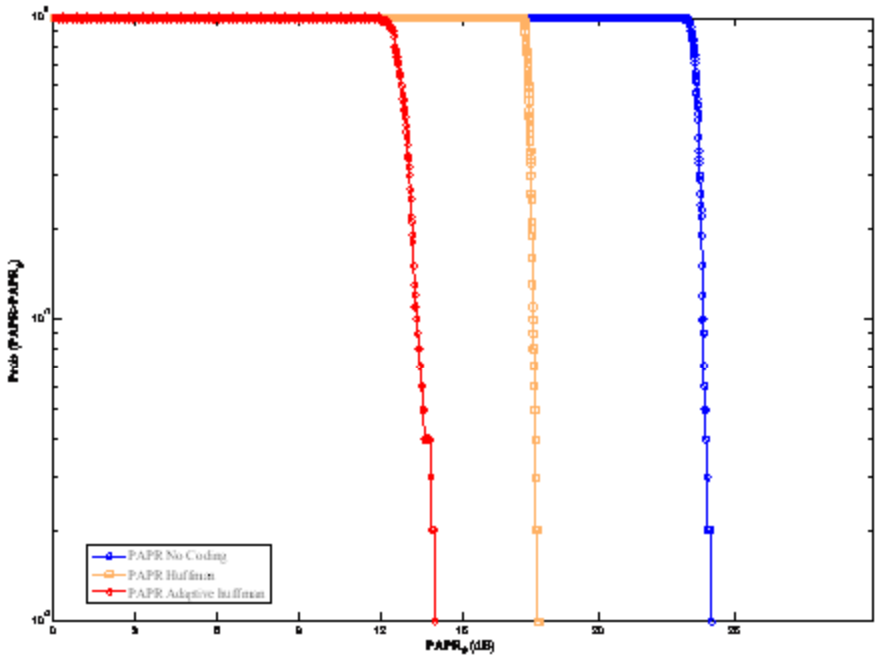


Figure 5.15 CCDF Plot representing PAPR of QAM 16 modulation for N=256

Figure 5.16 shows the CCDF plot for N=32. There is a 2.2 dB PAPR difference between PSK and QAM modulation for adaptive huffman codes and 2 dB for huffman codes. Figure 5.17 represents the CCDF plot for N=64.In that there is a 2.6 dB difference between Adaptive Huffman, 1.8 dB between Huffman codes from PSK to QAM modulation.

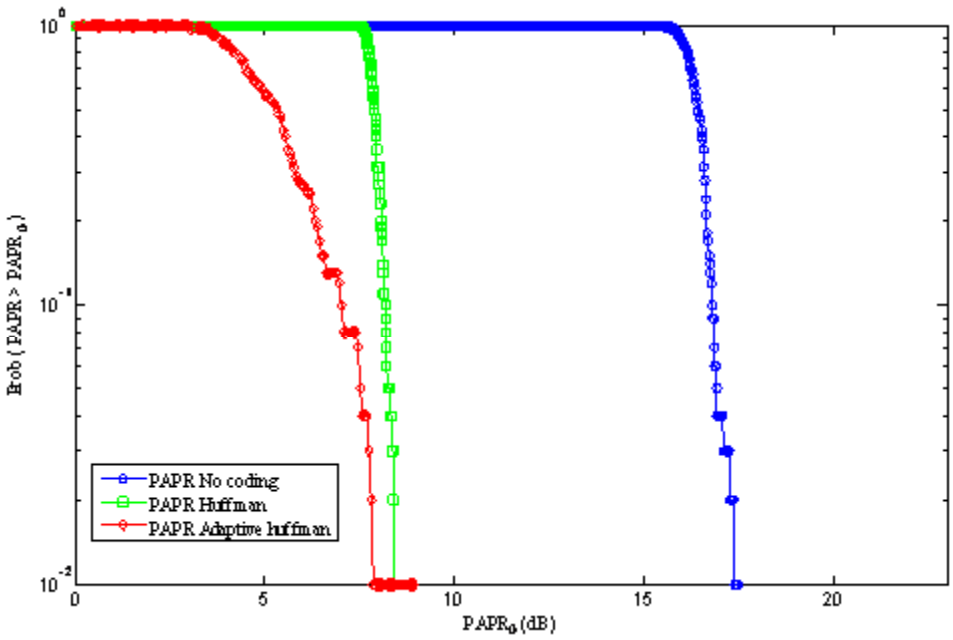


Figure 5.16 CCDF Plot representing PAPR of PSK modulation for N=32

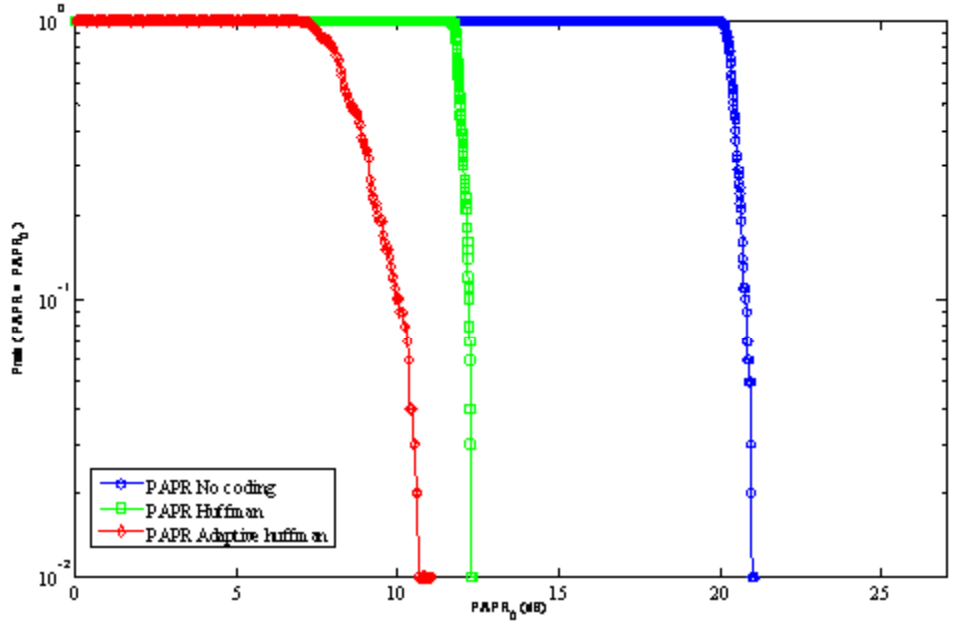


Figure 5.17 CCDF Plot representing PAPR of PSK modulation & N=64

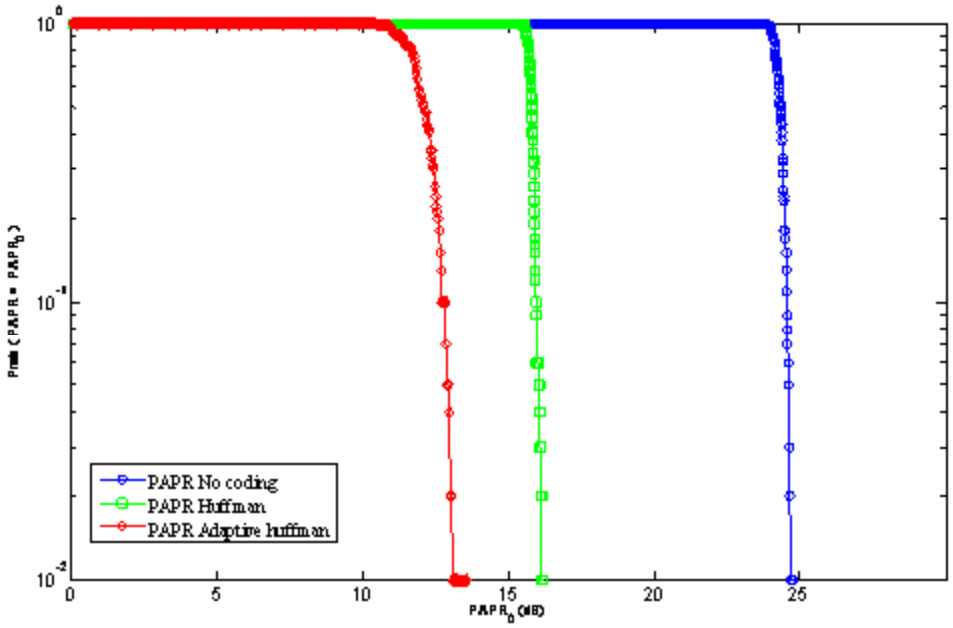


Figure 5.18 CCDF Plot representing PAPR of PSK modulation for N=128

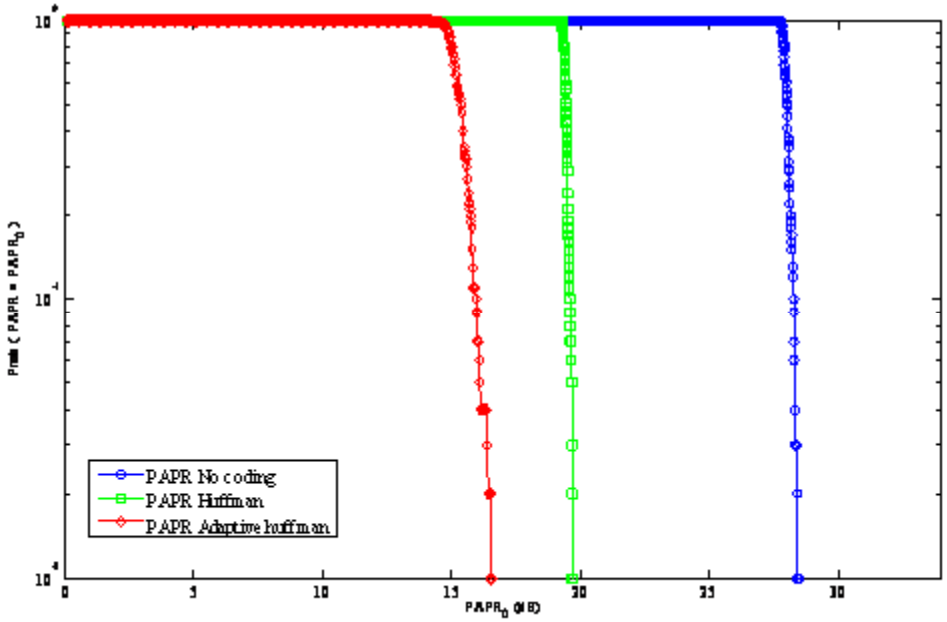


Figure 5.19 CCDF Plot representing PAPR of PSK modulation for N=256

For higher subcarriers N=128 & N=256 in Figure 5.18 and 5.19, an average of 3.1dB difference is maintained between AHF to Huffman codes using PSK modulation. The PAPR difference between QAM and PSK modulation with different 'N' values have been listed in Table 5.2.

Table 5.2 Comparison of PAPR reduction values by using Adaptive Huffman codes and Huffman codes with PAPR (No coding)

Mapping	No. of Sub carriers (N)	PAPR-No coding (dB)	PAPR-Huffman coding (dB)	PAPR-Adaptive Huffman coding (dB)
16 QAM	32	13.9	6.2	5.8
	64	17.8	10.4	8.3
	128	21	14	11
	256	24.1	18.8	14.4
PSK	32	17.2	8.2	7.9
	64	21	12.2	10.92
	128	24	16.2	13.1
	256	28.2	19.9	15.88

5.9 BAR CHART ANALYSIS

The PAPR difference of AHC, Huffman codes and no coding for QAM and PSK mapping has been illustrated in the bar chart analysis.

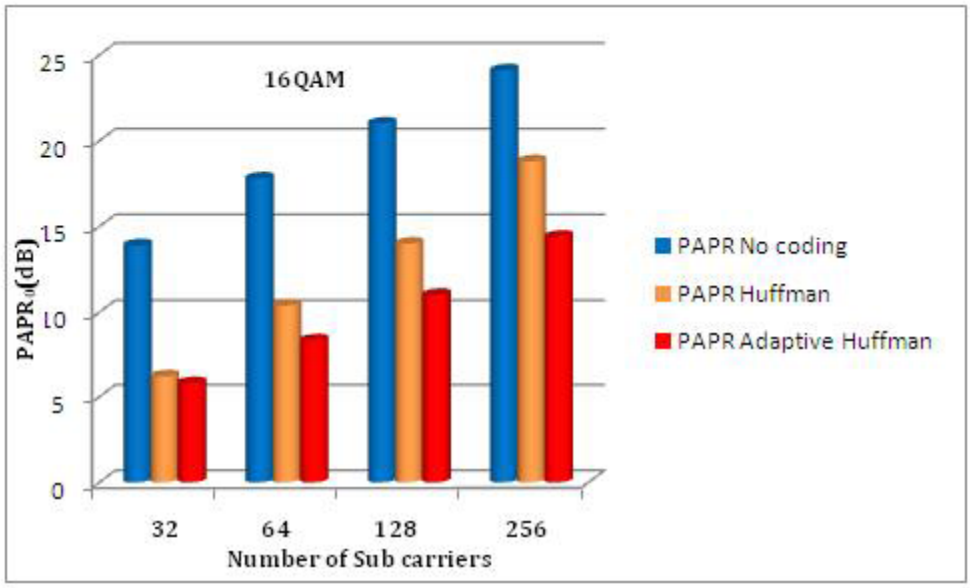


Figure 5.20 PAPR values (No coding ,Huffman and AHC)for QAM 16 mapping

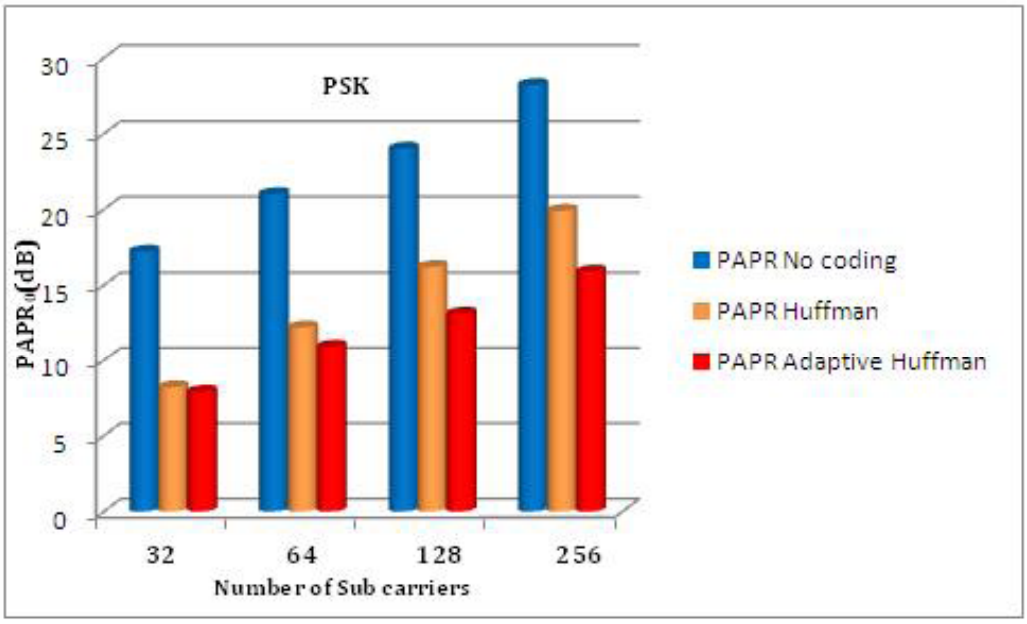


Figure 5.21 PAPR values (No coding ,Huffman and AHC)for PSK 16 mapping

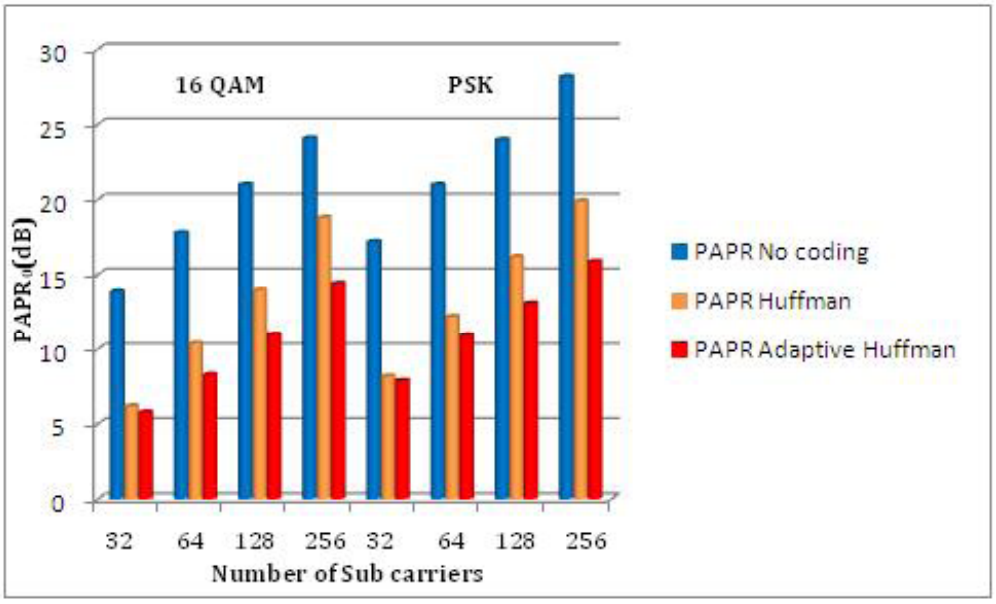


Figure 5.22 PAPR values (No coding, Huffman and AHC)for QAM 16 and PSK mapping

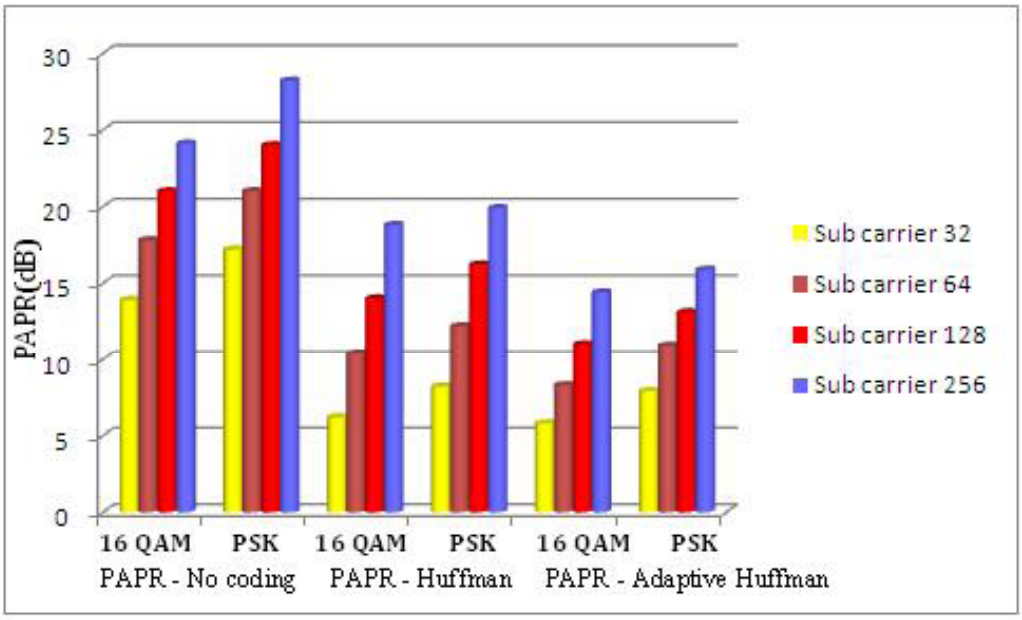


Figure 5.23 PAPR values (No coding, Huffman and AHC) for different sub carriers with QAM 16 and PSK mapping

5.10 SUMMARY

The PAPR comparison table and the Bar chart analysis states that for all ‘N’ values the Huffman codes with QAM mapping show a better PAPR reduction when compared to PAPR with no compensation. But the utilization of Adaptive Huffman codes dominates the Huffman codes for all the subcarriers usage .The AHF with QAM mapping shows a good PAPR reduction results when compared to PSK mapping.