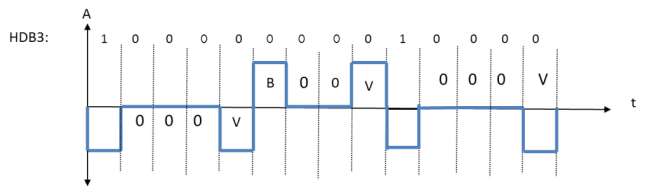
**HDB3 SCRAMBLING TECHNIQUE FOR LINE CODING WITH MATLAB CODE FOR ENCODING AND DECODING**

**Basic**

High Density Bipoler 3 Zero (HDB3) works in a similar way to AMI by changing poles for each binary 1. Moreover when it found 4 consecutive 0’s it replace them by ‘000V’ or ‘B00V’ depending on number of non-zero pulses (odd or even) after last substitution If number of non-zero pulses after last substitution is odd then ‘0000’ will be replaced by ‘000V’ if even ‘0000’ will be replaced by ‘B00V’. Here, V means violation (same to previous non-zero bit) and B means Balancing (opposite of previous non-zero bit).

Here is an example of HDB3 line coding for bits 10000000010000.



Conversion from bitstream to signal is called encoding.

Conversion from signal to bitstream is called decoding.

**Coding With MatLab**

For encoding we need the bitstream first. In MatLAB that can be defined as array or we can take input from user. Both ways are given below:

|  |  |
| --- | --- |
| 1 | bits = [1 0 1 1 1 0 0 1]; |

or,

|  |  |
| --- | --- |
| 1 | bits = input('prompt'); |

With bitstream we need bit rate that means how much bit(s) will appear in each second. This can be also defined as a variable or can be taken as input. Normally bit rate is considered as 1 bit per second. So we will define it to 1.

|  |  |
| --- | --- |
| 2 | bitrate = 1; |

We need a sampling for each bit. So that MatLAB needs to know how much sample point will be appeared for each bit. Low sampling can loss data or cannot define bits accurately. Very high sampling can cause slow your program. Here we will define it as 1000.

|  |  |
| --- | --- |
| 3 | n = 1000; |

MatLAB needs to know the total time needed for the bitstream. Sometimes total time T may be given or said to take as input. If not then we need to find it by dividing length of bitstream with bitrate.

|  |  |
| --- | --- |
| 4 | T = length(bits)/bitrate; |

We have a sampling for each bit we have declared previously. Now we need to find total sample point needed for whole bitstream. We can find it by multiplying sample point of each bit with the number of bits as below.

|  |  |
| --- | --- |
| 5 | N = n\*length(bits); |

Now, we need to found time for each sample point. For this we divide the whole time (T) with number of total sample points (N).

|  |  |
| --- | --- |
| 6 | dt = T/N; |

We are near to the main process. Now we will define the time domain. That is the all values of X-axis. For this we need to declare an array from time 0 to time T by partitioning with dt. That is an array like this.

|  |  |
| --- | --- |
| 7 | t = 0:dt:T; |

For better performance we need to set all values of the result (amplitue domain) to 0. We can do this task with a build in function zeros().

|  |  |
| --- | --- |
| 8 | x = zeros(1,length(t)); |

Here we need an additional variable to save the previous bit. The previous bit at the first point is given in the problem statement. If not by default we will declare it to 1. Here I am declaring it 1 that is the previous bit had positive voltage.

|  |  |
| --- | --- |
| 9 | lastbit = 1; |

Here we need another variable for counting number of consecutive 0’s. That is the counter variable. As there is no zero counted at start it is declared as 0.

|  |  |
| --- | --- |
| 10 | counter = 0; |

In HDB3 we need to count the number of non zero pulse after last substitution. So we will declare a variable named pulse and the initial value of pulse will be 0. That is there is no pulse at starting.

|  |  |
| --- | --- |
| 11 | pulse = 0; |

Here each bit is checked. If the bit is 0 then counter is increased by 1. In this case there is no need to make any change in amplitude domain because that (x(i)) is by default 0.

Then we will seek for 4 consecutive 0. If counter is equal to 4 that means 4 consecutive zero. Otherwise it will enter in else statement. There it will set the counter value to 0. Because the consecution of 0 is broken. The the value of amplitude domain (x) will be redefined by opposite value of lastbit for that bit. And lastbit will be changed (negated). And the pulse value will be increased by 1 as one pulse occurred here.

If there is found 4 consecutive 0,s then it will enter the if condition (see in code below).

There is another if condition is the pulse after last substitution is even? If pulse is even (pulse%2==0) then we will replace 4 consecutive 0’s with ‘B00V’. And for this case we will start re filling from 3 bit earlier. First bit of them is filled by opposite value of the lastbit and lastbit will be changed (negated). Second and Third bit are filled with 0. Fourth bit (current bit that is passed through condition immediately) will be filled by same value of lastbit. After replacing the 4 bits the value of counter and pulse will be reset(set 0 again).

Is the pulse after last substitution is not even (else statement will be true)? If pulse is not even (odd) then we will replace 4 consecutive 0’s with ‘000V’. In this case we will start filling from three bits earlier also (as if statement). First, second and third bit will be filled by 0. Fourth bit will be filled by the same value of lastbit. After replacing the 4 bits the value of counter and pulse will be reset(set 0 again).

|  |  |
| --- | --- |
| 12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39 | for i=1:length(bits)    if bits(i)==0      counter = counter + 1;      if counter==4        if(mod(pulse, 2)==0)          x((i-1-3)\*n+1:(i-3)\*n) = -lastbit;          lastbit = -lastbit;          x((i-1-2)\*n+1:(i-2)\*n) = 0;          x((i-1-1)\*n+1:(i-1)\*n) = 0;          x((i-1)\*n+1:i\*n) = lastbit;          counter = 0;          pulse = 0;        else          x((i-1-3)\*n+1:(i-3)\*n) = 0;          x((i-1-2)\*n+1:(i-2)\*n) = 0;          x((i-1-1)\*n+1:(i-1)\*n) = 0;          x((i-1)\*n+1:i\*n) = lastbit;          counter = 0;          pulse = 0;        end      end    else      counter = 0;      x((i-1)\*n+1:i\*n) = -lastbit;      lastbit = -lastbit;      pulse = pulse + 1;    end  end |

Everything required for encoding is done. We will now plot the result (amplitude domain) on respect to time domain. Here is the plot. Here ‘Linewidth’, 3 means the signal line will be bold three times than original signal.

|  |  |
| --- | --- |
| 40 | plot(t, x, 'Linewidth', 3); |

**Code For Encoding**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40 | bits = input('prompt');  bitrate = 1;  n = 1000;  T = length(bits)/bitrate;  N = n\*length(bits);  dt = T/N;  t = 0:dt:T;  x = zeros(1,length(t));  counter = 0;  lastbit = 1;  pulse = 0;  for i=1:length(bits)    if bits(i)==0      counter = counter + 1;      if counter==4        if(mod(pulse, 2)==0)          x((i-1-3)\*n+1:(i-3)\*n) = -lastbit;          lastbit = -lastbit;          x((i-1-2)\*n+1:(i-2)\*n) = 0;          x((i-1-1)\*n+1:(i-1)\*n) = 0;          x((i-1)\*n+1:i\*n) = lastbit;          counter = 0;          pulse = 0;        else          x((i-1-3)\*n+1:(i-3)\*n) = 0;          x((i-1-2)\*n+1:(i-2)\*n) = 0;          x((i-1-1)\*n+1:(i-1)\*n) = 0;          x((i-1)\*n+1:i\*n) = lastbit;          counter = 0;          pulse = 0;        end      end    else      counter = 0;      x((i-1)\*n+1:i\*n) = -lastbit;      lastbit = -lastbit;      pulse = pulse + 1;    end  end  plot(t, x, 'Linewidth', 3); |

For decoding we need a counter that will count how much bit has been decoded. We will define the initial value of the counter to 0.

|  |  |
| --- | --- |
| 41 | counter = 0; |

Here we also need a variable to store lastbit. We will use the same variable we used in encoding. But as the value of that has been changed so we will define it’s value again.

|  |  |
| --- | --- |
| 42 | lastbit = 1; |

Now we need a loop again to decode the signal. The loop will run from 1 to length of t. That is same times of total sample point.

In the loop we will test whether t(i) is greater than the counter. If it is true then we will increase the value of counter. Here we will check each bit whether it is equal to lastbit value (Note: We will never assign the value of lastbit variable to 0. So lastbit is either 1 or -1).

If the condition is matched then we found a set of 4 consecutive 0 ‘s. If the condition is false program will turn to else statement. There if the bit is 0 we will assign 0 to result(counter). If not (the bit is 1 or -1) we will assign 1 to result(counter) and change (negate) the value of lastbit (because 1 caused a transition while encoding).

If the condition is true that is found any set of 4 consecutive 0’s then we will enter the if statement. Carefully take it where we assigne ‘000V’ in encoding there the current bit and last non zero bit is differenced by three bits between them. If we replaced with ‘B00V’ then there is two bits among the current bit and last non-zero bit. So, we will fill last three bit and current bit with 0. That is ‘B00V’ or ‘000V’ will be replaced by ‘0000’. After this the loop will proceed on.

Look the loop below.

|  |  |
| --- | --- |
| 43  44  45  46  47  48  49  50  51  52  53  54  55  56  57 | for i = 1:length(t)    if t(i)>counter      counter = counter + 1;      if x(i)==lastbit        result(counter-3:counter) = 0;      else        if(x(i)==0)          result(counter) = 0;        else          result(counter) = 1;          lastbit = -lastbit;        end      end    end  end |

After this decoding is also done. We need to display it. We can display that by just writing the variable without semi-colon. Or this can shown with disp() function.

|  |  |
| --- | --- |
| 58  59 | disp('HDB3 Decoding:');  disp(result); |

**Full Code (Encoding and decoding)**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59 | bits = input('prompt');  bitrate = 1;  n = 1000;  T = length(bits)/bitrate;  N = n\*length(bits);  dt = T/N;  t = 0:dt:T;  x = zeros(1,length(t));  counter = 0;  lastbit = 1;  pulse = 0;  for i=1:length(bits)    if bits(i)==0      counter = counter + 1;      if counter==4        if(mod(pulse, 2)==0)          x((i-1-3)\*n+1:(i-3)\*n) = -lastbit;          lastbit = -lastbit;          x((i-1-2)\*n+1:(i-2)\*n) = 0;          x((i-1-1)\*n+1:(i-1)\*n) = 0;          x((i-1)\*n+1:i\*n) = lastbit;          counter = 0;          pulse = 0;        else          x((i-1-3)\*n+1:(i-3)\*n) = 0;          x((i-1-2)\*n+1:(i-2)\*n) = 0;          x((i-1-1)\*n+1:(i-1)\*n) = 0;          x((i-1)\*n+1:i\*n) = lastbit;          counter = 0;          pulse = 0;        end      end    else      counter = 0;      x((i-1)\*n+1:i\*n) = -lastbit;      lastbit = -lastbit;      pulse = pulse + 1;    end  end  plot(t, x, 'Linewidth', 3);  counter = 0;  lastbit = 1;  for i = 1:length(t)    if t(i)>counter      counter = counter + 1;      if x(i)==lastbit        result(counter-3:counter) = 0;      else        if(x(i)==0)          result(counter) = 0;        else          result(counter) = 1;          lastbit = -lastbit;        end      end    end  end  disp('HDB3 Decoding:');  disp(result); |

I think, your experience with this tutorial is quite good. You can notify me anything related to this tutorial via comment or contact form.