EEE414/EEE6410: Computer Communications/Digital communications

Model Solutions for 11/12 session

1.
(a)
i[3 marks]
Bandwidth: As low as possible with low or no DC component that can be problematic in ac-coupled channels.

Synchronisation: Efficient built-in mechanism for synchronisation with low overhead, good noise immunity and possibly built-in error detection.
Complexity: Cheap and simple implementation, in particular at the receiver end

ii) [4 marks]
Polar Encoding class: which employs two levels, a positive and a negative level, in order to reduce or eliminate the DC component. Biphase encoding is a good example of this class which also employs positive to negative and negative to positive transitions to represent the 1 and 0 bits. This allows synchronisation and a degree of error-detection due to absence of a transition at the expense of increased bandwidth. Biphase encoding is typified by Manchester encoding (used in Ethernet) and Differential Manchester Encoding (used in token rings).
Bipolar Encoding class: uses 3 voltage levels, positive, negative and zero. The zero level in Bipolar encoding, unlike in RZ, represents the actual 0 bit. The 1s are represented by alternate positive and negative levels. Bipolar codes are or are made to be DC free with less transitions than biphase codes so require less bandwidth; AMI, and HDB3 are examples of bipolar encoding. HDB3 is commonly used in transmission over digital lines (E1 and E2 feg).

(b)

[7 marks]
Sequence: **1010011010**
From the timing diagram in Fig.Q.1.a, the following characteristics can be extracted for the code: [3 marks]

- A Bipolar code since it employs 3 voltage levels, positive, negative and zero.
- 1s are represented by alternate positive and negative voltage levels and the 0 with a zero voltage level.
- The code is DC free
- Has less transitions than biphase codes. Less transitions results in less bandwidth requirements but also less noise immunity.
- It keeps a long sequence of 1s synchronised, however has no mechanism to synchronise a long sequence of 0s.
- Known as AMI (Alternate Mark Inversion). Suitable for low bandwidth applications where clock synchronisation is not critical.

From the timing diagram in Fig.Q.1.b, the following characteristics can be extracted for the code: [5 marks]

- A Polar NRZ code since it employs two voltage levels (polarities) and uses positive to negative and negative to positive transitions.
- 1s are coded by full bit-period pulses alternating at each 1, and 0s are conveyed by a half-bit-period at 0 followed by a half bit-period at 1
- DC free as transitions are uniformly compensated
- Middle bit transitions for 0s only, hence less transitions overall than comparable biphase codes such as Manchester codes. Less transitions results in less bandwidth requirements .
- Binary form of AMI to synchronise long strings of 0s, simpler than HDB3
- Suitable for low bandwidth high signalling rate applications. Adopted as the line code in STS-3 for example
- This is known as Coded mark inversion (CMI) encoding (the students have not seen this in the lectures)

(c) [7 marks]

$F_{tc} = F_n(1+\delta) = 1/T_{tc}$ (transmitter clock fast)
$F_{rc} = F_n(1-\delta) = 1/T_{rc}$ (receiver clock slow)
$T_{\text{mid-start bit}} = (m/2+1)T_{rc}$ (start-bit detected 1 cycle late)
$T_{\text{character}} = (N-1) \times m \times T_{rc} + (m/2+1)T_{rc} = (mN-m/2+1)T_{rc} = N \times m \times T_{tc}$
$(mN-m/2+1)\,[1/F_n(1-\delta)] = mN\,[1/F_n(1+\delta)]$
$mN(1-\delta) = (mN-m/2+1)(1+\delta)$

mN-mNδ = mN +mNδ –m/2 –m/2δ +1 + δ
10m-0.2m = 10m +0.2m –m/2 –0.01m +1.02
0.11m = 1.02 which gives m= 9.27
The first even multiple would be 10; in practice 16x would be more suitable.

2.
i)
Need first to determine correct CRC bits using longhand division. 110 = i(x) = x^2+ x. Next we need to find the CRC bits generated by the generator polynomial gx)= $x^4+ x^3+1$. This g(x) will generate a 4-bit CRC which we can find by calculating the remainder of division of i(x)x^{n-k} =i(x) x^4 = x^6+ x^5 by g(x) using modulo-2 long hand division as follows:

$X^6+ x^5$	$x^4+ x^3+1$
$x^6+ x^5+ x^2$	x^2
x^2 (Remainder = 0100)	

Therefore the transmitted codeword is: 1100100.

ii) For this 4-bit CRC, the error-detection reliability figure is (1-1/16)x100 = 93.75%

b) (110100010) xor (011001000) = (101101010) is the erroneous received codeword (3 errors located as by the 1s in the error pattern)
To check for errors we divide the erroneous received codeword by g(x) and check the remainder.

erroneous codeword received is $x^8+ x^6+ x^5+ x^3+x$ working out the remainder as before:

$x^8+ x^6+ x^5+ x^3+x$	$x^4+ x^3+1$
$x^8+ x^7+ x^4$	$x^4+ x^3+ x$
$x^7+x^6+ x^5 + x^4+ x^3+ x$	
$x^7+x^6+ x^3$	
$x^5 + x^4+ x$	
$x^5 + x^4+ x$	

0 (Remainder = 0000)
A zero remainer for an erroneous codeword. An erroneous codeword that cannot be detected by the 4-bit CRC.This is because the error pattern is a multiple of g(x) - in this case e(x) =$x^7+x^6+ x^3$= x^3g(x) which results in an undetectable pattern given that for the 4-bit CRC, the error-detection reliability figure is 93.75%

c.
$r(x)= x^{14}+x^{11}+x^9+x^7+ x^3+ x^2+x+1$
S_1 = r(α) = $\alpha^{14}+\alpha^{11}+\alpha^9+\alpha^7+\alpha^3+\alpha^2+\alpha +1=\alpha^3 + 1+\alpha^3 + \alpha^2 + \alpha+\alpha^3 + \alpha+\alpha^3 + \alpha+ 1+\alpha^3+\alpha^2+\alpha =\alpha^3+1=\alpha^{14}$
$S_3 = r(\alpha^3) = \alpha^{42}+\alpha^{33}+\alpha^{27}+\alpha^{21}+\alpha^9+\alpha^6+\alpha^3+1 =\alpha^{12}+\alpha^3+\alpha^{12}+\alpha^6+\alpha^9+\alpha^6+\alpha^3 = \alpha^9+1=\alpha^7$

$\dfrac{S_3+S_1^3}{S_1} = \alpha^8+\alpha^{13}=\alpha^2 + 1 + \alpha^3 + \alpha^2 + 1 = \alpha^3$ and therefore $\sigma(x) = x^2 + S_1 x + \dfrac{S_3+S_1^3}{S_1} = x^2+\alpha^{14} x + \alpha^3$
Try the 4 LSB positions for errors (evaluate $\sigma(\alpha^i)$ for i=0,1,2,3)
$\sigma(\alpha^0)= 1+\alpha^{14} + \alpha^3 = 1+\alpha^3 + 1+ \alpha^3 =0$ so 1st error in the LSB position
$\sigma(\alpha)= \alpha^2+\alpha^{15} + \alpha^3 = \alpha^2 + 1+ \alpha^3 =\alpha^{13}$ so this is not a root and hence the second LSB is not erroneous
$\sigma(\alpha^2)= \alpha^4+\alpha^{16} + \alpha^3 = \alpha + 1 +\alpha+ 1+ \alpha^3 =\alpha^3$ so this is not a root and hence the third LSB is not erroneous
$\sigma(\alpha^3)= \alpha^6+\alpha^{17} + \alpha^3 = \alpha^3 + \alpha^2 +\alpha^2 + \alpha^3 =0$ this is a root so 2st error in the 4th LSB position

To correct r(x) invert the r_0 and r_3 bits to give r(x)= $x^{14}+x^{11}+x^9+x^7+x^2+x$ =(100101010000110)

Power of α	Polynomial Representation	Vector Representation $(a_3 a_2 a_1 a_0)$
-	0	0000
0	1	0001
1	α	0010
2	α^2	0100
3	α^3	1000
4	$\alpha + 1$	0011
5	$\alpha^2 + \alpha$	0110
6	$\alpha^3 + \alpha^2$	1100
7	$\alpha^3 + \alpha + 1$	1011
8	$\alpha^2 + 1$	0101
9	$\alpha^3 + \alpha$	1010
10	$\alpha^2 + \alpha + 1$	0111
11	$\alpha^3 + \alpha^2 + \alpha$	1110
12	$\alpha^3 + \alpha^2 + \alpha + 1$	1111
13	$\alpha^3 + \alpha^2 + 1$	1101
14	$\alpha^3 + 1$	1001

3.
(a) [4 marks]
In block codes k information symbols are formed into a word $(I_1, I_2, ..., I_k)$. This information word is then encoded into n codeword symbols $(C_1, C_2, ..., C_n)$.$(n > k)$.Typically these symbols are strings of bits. So a block of k information bits is converted into a block of n code bits resulting in an (n,k) block codes where $R = k/n$ is the rate of the code. Block codes have no memory and so consecutive codewords are independent. However, because we are dealing with blocks of data, buffering memory and latency overheads are always associated with block codes.
Block codes , as opposed to convolutional codes, can be cross interleaved for reliable storage of data. Block codes can be concatenated with convolutional codes or mapped together onto an iterative (turbo) configuration for higher performance over some channels.
Hard decision decoding of block codes, although involves in cases significant latency and hardware overheads, requires well defined stages and hence the process is rather mechanical. Soft-decision decoding of block codes is rather difficult. Block codes are employed, and in some cases are standard, in satellite communications, CDs, DVB.
An (n,k,m) convolutional encoder takes, a continuous stream of k information bits and produces an encoded sequence of n bits at an $R = k/n$ code rate. But each n bit codeword depends not only on the k information bits but also on m previous message blocks. The encoder has therefore memory of order m, which is rarely of more than a few bits length, hence resulting in minimal buffering and latency overheads.
Convolutional Encoders are in essence simple state machines hence very easy to implement in hardware. However, decoding is complex as it involves searching for a best fit path to recover the sent information but is more amenable to soft-decision decoding compared to block codes, which can result in better coding performance.
Hence convolutional codes are suitable for very low SNR channels, and also where transmitters use simple low power devices.

(b)

Powers of α	Polynomial	Binary ($a_2a_1a_0$)
0	0	000
1	1	001
α	α	010
α^2	α^2	100
α^3	$1+\alpha$	011
α^4	$\alpha+\alpha^2$	110
α^5	$1+\alpha+\alpha^2$	111
α^6	$1+\alpha^2$	101

i)(2marks)
A t-error correcting (n,k) RS code has : $d_{min}=2t+1$, $2t = n-k$, $n= 2^m -1$
In this case: k=3, n-k=2t=4 = number of check symbols giving RS(7,3). Each symbol is 3-bits and therefore any 2-symbol error with errors up to 3-bits/symbol can be corrected, also bursts up to 2-symbols (6-bits can also be corrected).

ii)(6 marks)

Algebraic decoding uses the the syndromes indirectly to locate the errors by defining an error locator polynomial, $\sigma(x)$ the roots, X_k, (or reciprocal roots) of which give the error locations
Where X_k is a k^{th} error locator such that $X_k = \alpha^j$, $\alpha^j \chi$ $GF(2^m)$, $k= 1,2,\ldots,t$,denotes an error in j^{th} position(bit),

$$\sigma(x) = \prod_{k=1}^{t}(x+X_k) = x^t + \sigma_1 x^{t-1} + \sigma_2 x^{t-2} + \ldots\ldots + \sigma_t$$

$j=0,1,2,\ldots..n-1$, of the received n-bit word $r(x)$, the magnitude of the error, Yj can then be evaluated using the equations

$$X_1 = \sigma_1 = \frac{S_2}{S_1}$$

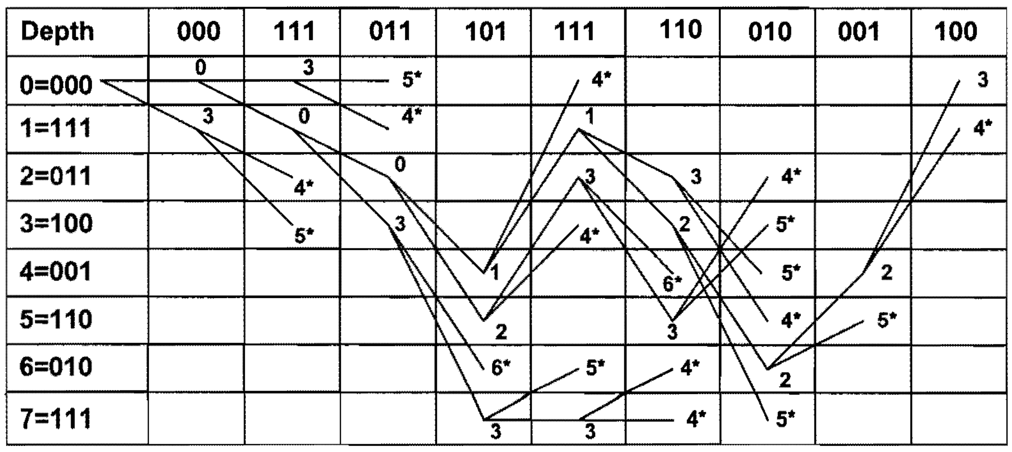
Need to work out first 2 syndromes:
In our case $S_1= r(\alpha) = \alpha^4$ and $S_2= r(\alpha^2) = \alpha^5$ giving $X_1=\alpha$ (need to correct r_1; the second LSB symbol)

$$Y_1 = \frac{S_1^2}{S_2} = \alpha^3$$

To correct r(x) add α^3 to r_1 to give: $\alpha^3 +1 = \alpha = 010$ resulting in a corrected
$r(x)= \alpha^5x^6+x^5+\alpha^2x^4+\alpha^5x^3+\alpha x^2+ \alpha x+1$ = (111001100111010**010**001).

c)

Set up a cost table for Viterbi decoding of received sequence as follows:
000 111 011 101 111 110 010 001 100



* : stop more than 3 errors

Corrected data is therefore: **010011000**

4.

a) [2 marks]

Data in many applications

- Has high spatial, temporal and spectral correlation
- can be reduced to an acceptable level, from a perceptual point of view
- can be of fractal nature having high-level features that exhibit spatial and temporal redundancy

b)

i) [3 marks]

Attractive features of DCT :

Has good compaction efficiency of energy
Is invertible and separable
Has image independent basis
Has fast algorithms for computation and implementation
Potential limitations:
Blocking effects (block transform)
Reduced Energy compaction efficiency for for weakly correlated

ii) (2 marks)

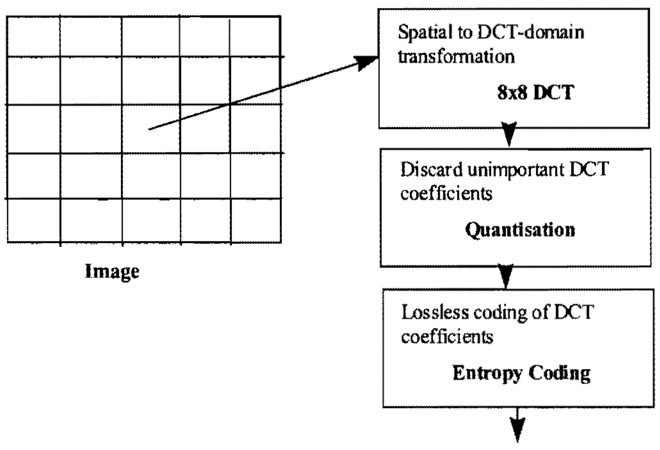


Image
Spatial to DCT-domain transformation
8x8 DCT
Discard unimportant DCT coefficients
Quantisation
Lossless coding of DCT coefficients
Entropy Coding

Generic DCT-based coding system

c) [6 marks]

Using row-column decomposition:

For all rows:

DCT0 = $1/\sqrt{2}$ ($10/\sqrt{2}$ cos 0 + $10/\sqrt{2}$ cos 0 + $10/\sqrt{2}$ cos 0 +$10/\sqrt{2}$ cos 0) = 20
DCT1 = 0
DCT2 = 0
DCT3 = 0

resulting in 1-D DCT matrix:

$$\begin{array}{cccc} 20 & 0 & 0 & 0 \\ 20 & 0 & 0 & 0 \\ 20 & 0 & 0 & 0 \\ 20 & 0 & 0 & 0 \end{array}$$

The 2-D DCT is obtained by applying the 1-D DCT to this matrix along the columns resulting in the following 2-D DCT matrix in:

$$\begin{matrix} 40 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{matrix}$$

This shows that all of the energy is concentrated in this case in the DC coefficient. The compression efficiency of the DCT is improved with highly correlated data (most energy is concentrated in fewer coefficients)

d)

First perform Zig-Zag encoding on the qunatised DCT coefficients to obtain

$$\begin{bmatrix} 47 & 25 & -18 & 15 & -12 & -3 & 0 & -2 \\ 14 & 11 & 0 & -2 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

AC codewords starting with first AC coefficient:
Value =25, run/size=0/5, Huffman code=11010, amplitude=11001, hence corresponding codeword=1101011001 (10 bits).
Value=-18, run/size=0/5, Huffman code=11010, amplitude= one's complement of 10010=01101, hence corresponding codeword=1101001101 (10bits)
Similarly the codewords for the remaining AC coefficients are therefore
15 →10111111 (8bits); -12 → 10110011 (8 bits); -3→run/size=0/2 →0100 (4 bits)
-2→run/size=1/2→1101101 (7 bits); 14→run/size=0/4 →10111110 (8bits);
11→run/size=0/4→10111011(8bits);-2→run/size=1/2 →1101101(7bits);
1→run/size=0/1 →001 (3 bits); EOB→run/size=0/0 →1010 (4 bits);

Number of bits taken by AC coefficients = 77 bits

Total number of bits (dc+ac) = 5 + 77 = 82 bits, therefore average bit rate = 82/64 = 1.28 bit/pixel giving a compression rate of 8/1.36 = 6.25