EEE6410 Typical Exam Solutions 13/14

Q.1

a) (3)

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

b) (10)

From the timing diagram in Fig.Q.1.a, the following characteristics can be extracted for the code:

- 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 .
-

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

- Biphase encoding, employs positive to negative and negative to positive transitions to represent the bits.
- transitions occur at the middle of each bit time, negative to positive transition for a 1 bit and positive to negative transition for a 0 bit
- Synchronisation and bit representation, using only 2 voltage levels, as opposed to 3 levels as in RZ type encoding. (less complex to implement, can be obtained by XNOR operation of the data and the transmitter clock. A PLL is used by the receiver to recover the clock from the transitions).
- A degree of error-detection due to absence of a transition.
- Requires higher bandwidth (due to extra transitions)
- Suitable for applications where synchronisation is more important than bandwidth.
- The code is a Manchester code and used in Ethernet

From the timing diagram in Fig.Q.1.c, the following characteristics can be extracted for the code:

- 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.
- AMI (Alternate Mark Inversion) code suitable for low bandwidth applications where synchronisation is not critical.

A good selection would contrast the codes as above and come to the conclusion that Code Q.1.a would be most suited here offering a good balance between a lower bandwidth than Q.1.b and better synchronisation (reliability) than Q.1.c.

c) (7)

Ftc = Fn(1+δ)= 1/Ttc (transmitter clock fast)
Frc=Fn(1- δ) = 1/Trc (receiver clock slow)
Tmid-start bit = (16+1)Trc = 17Trc (start-bit detected 1 cycle late)
T character = (N-1)x32xTrc + 17Trc = (32N-15)Trc = Nx32xTtc
(32N-15) [1/Fn(1- δ)] = 32N [1/Fn(1+δ)]
32N (1- δ) = (32N-15)(1+ δ)
32N-32Nδ = 32N +32Nδ -15 -15δ
64Nδ=15 +15δ
N=(15 +15δ)/64δ = 15.15/0.64= 23.67
Therefore a reliable value for m would be up to 20 bits (N=m+3).

2.

a) (5)

Need first to determine correct CRC bits using longhand division.From the received data input the 3 bits are in error are first inverted to give the correct transmitted data input pattern: 1000 = i(x) = x^3. Next we need to find the CRC bits generated by the generator polynomial gx) = p(x) = $x^3 + x+1$. This g(x) will generate a 3-bit CRC which we can find by calculating the remainder of division of $i(x)x^{n-k}$ $=i(x)\,x^3 = x^6$ by g(x) using modulo-2 long hand division as follows:

$$\begin{array}{l|l}
x^6 & x^3+x+1 \\ \hline
x^6+x^4+x^3 & x^3+x+1 \\ \hline
x^4+x^3 & \\
x^4+x^2+x & \\ \hline
x^3+x^2+x & \\
x^3+x+1 & \\ \hline
\end{array}$$

x^2+1 (Remainder = 101).

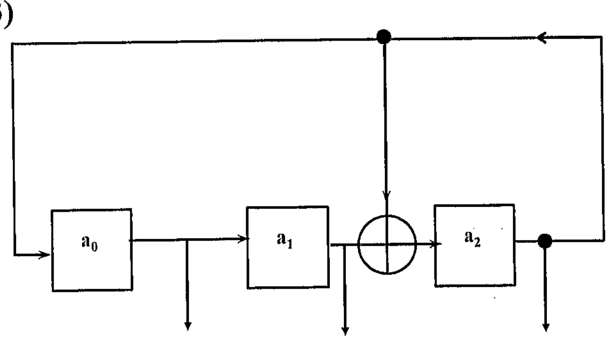
Therefore the transmitted codeword is $c(x) = x^6 + x^2 + 1$ = 1000101.

To check for errors we divide the erroneous received codeword by g(x) and check the remainder. Since the errors have only affected 3 data bits, the CRC bits remain the same and the erroneous received codeword is **0011**101 = $x^6+ x^4+ x^3+ x^2+ 1$ working out the remainder as before:

$$\begin{array}{l|l}
x^4+x^3+x^2+1 & x^3+x+1 \\ \hline
x^4+x^2+x & x+1 \\ \hline
x^3+x+1 & \\
x^3+x+1 & \\ \hline
0 \text{ (Remainder = 000)} & \\
\end{array}$$

A zero remainer for an erroneous codeword! An erroneous codeword that cannot be detected by the 3-bit CRC.This is because the error pattern is a multiple of g(x) which results in an undetectable pattern. For this 3-bit CRC, the error-detection reliability figure is (1-1/8)x100 = 87.5%

b) (3)



GF(2^3) elements representation using Irreducible polynomial $p(x) = x^3 + x^2 + 1$

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	$1 + \alpha^2$	101
α^4	$1 + \alpha + \alpha^2$	111
α^5	$\alpha + 1$	011
α^6	$\alpha + \alpha^2$	110

c)

i) **[2 marks]**

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=7-3 =4=2t. This code can therefore correct up to 2 symbol errors. 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) [4]

A t-error correcting (n,k) RS code has a generator polynomial g(x) with 2t consecutive powers of a primitive element $\alpha \in GF(2^3)$, as roots : $g(x)=(x-\alpha)(x-\alpha^2)(x-\alpha^3)\ldots\ldots(x-\alpha^{2t})$
For a (7,3) RS code over GF(2^3), using $p(x) = x^3 + x^2 + 1$,
$g(x)=(x+\alpha)(x+\alpha^2)(x+\alpha^3)(x+\alpha^4) = x^4 + g_3x^3 + g_2x^2 + g_1x + g_0 = \mathbf{x^4 + \alpha^2x^3 + \alpha^3x^2 + x + \alpha^3}$

iii) (7 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

ε Where X_k is a k^{th} error locator such that $X_k = \alpha^j$, $\alpha^j \underline{\chi}$ $GF(2^m)$, $k = 1,2,...,t,$denotes an error in j^{th}

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

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

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

Need to work out first 2 syndromes:

In our case $S_1 = r(\alpha) = \alpha^5$, and $S_2 = r(\alpha^2) = \alpha^3$ giving $X_1 = \alpha^5$ (need to correct r_5 symbol)

$$Y_1 = \frac{S_1^{\,2}}{S_2} = \alpha^7 = 1$$

To correct r(x) add 1 to r_5 to give: $\alpha^3 + 1 = \alpha^2$ resulting in a corrected

$r(x) = \underline{\alpha^2}x^5 + \alpha x^4 + \alpha x^2 + \alpha^6 x + \alpha^6$

3.

a) (3)

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.

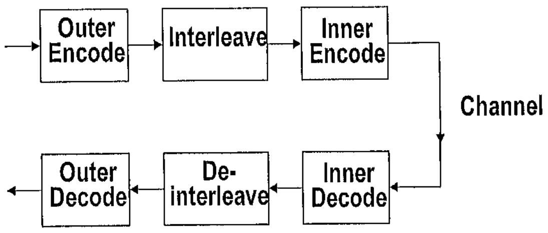
An (n,k,m) convolutional encoder takes, a continuous stream of k information bits and produces an encoded sequence of n bits at an $R = k/n$ code rate. But each n bit codeword depends not only on the k information bits but also on m previous message blocks. The encoder has therefore memory of order m, 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) (4)

Multi-stage Coding aims to create very long codes (better error correction performance) but with reasonable decoding complexity (not completely random -Shannon showed that capacity can be achieved by a long enough completely random code -) by concatenation of normally smaller constituent codes . Serial Concatenation was proposed by Forney (1966) where two codes are connected serially (as shown in the figure below), an outer code followed by an inner code; some form of interleaving necessary to match constituent codes. Both outer an inner codes could be block codes (as in product codes - DVDs); inner code could be a convolutional code (DVBs, Deep-space comms).



c) (6)

In contrast to Meggit decoding, 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

$$\sigma(x)=\prod_{k=1}^{t}(x+X_k)=x^t+\sigma_1x^{t-1}+\sigma_2x^{t-2}+.......+\sigma_t$$

Where X_k is a k^{th} error locator such that $X_k=\alpha^j$, $\alpha^j € GF(2^m)$, $k=1,2,...,t.,$denotes an error in j^{th} position(bit), $j=0,1,2,.....n-1,$ of the received n-bit word $r(x)$ and hence

$$S_i=e(\alpha^i)=\sum_{k=1}^{t}X_k^i \quad , \quad i=1,2,.....,2t \qquad (*)$$

using $\sigma(x)$ allows (*) to be translated into a set of linear equations

For t =1 we have $\sigma(x)=x+\sigma_1$ giving an error location $X_1=\sigma_1$

and from (*) we have $S_1=\sigma_1$ and $S_2=\sigma_1{}^2$, However for a binary code $S_2=S_1{}^2$ which gives $S_1=\sigma_1$

So the error location is simply $X_1=S_1$

In our case $S_1=r(\alpha)=\alpha^{14}+\alpha^{11}+\alpha^9+\alpha^2+\alpha=\alpha^7=X_1$

To correct r(x) invert the r_7 (the 8th LSB) to give $r(x)=x^{14}+x^{11}+x^9+x^7+x^2+x$ =(100101010000110)

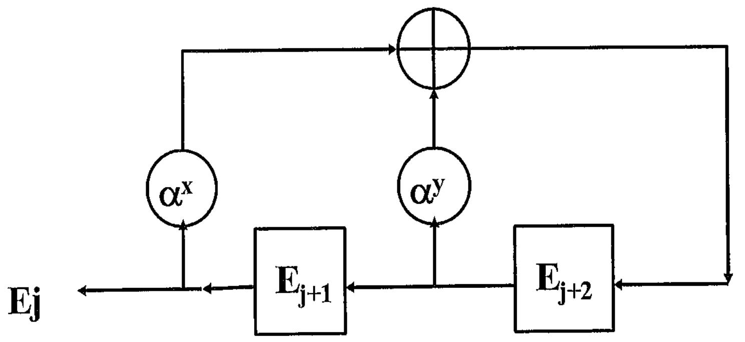
d) (7)

Ignoring all routes with more than 3 errors, the viterbi cost table is as follows:
The corrected message is hence 1 3 7 2 4 5 7 2 5 and the correct data is 101110110

Depth/Op code(decimal)	1	7	5	2	6	5	7	2	5	
0/0	1	4	6	4		5				
1/6			3	4	3					
2/3		1	5	3						
3/5			2	6	4	3			3	
4/1	0	3	5	5		4				
5/7			2	4	4		3			
6/2		2	6	2				3		
7/4			3	5	3	4				

4.
a) (6)

From C'= (α^4, α^2, α, α, α^5, α, α^2), E= (α^4, α^2, α, α, ? , ?, ?). So knowing the first 2t=4 error spectral values Ej (j-0 to 3) , we want to find the rest of the error spectrum. This can be achieved simply by using the FSR circuit shown below of length t=2 that recursively calculates the remaining 3 error spectrum values from the already known 4 values of the spectrum.



The process starts by preloading, registers Ej+1 and Ej+2 with E0 (α^4) and E1(α^2), respectively such that on the first clock cycle E0 is output followed by E1 on the next cycle. Appropriate values for α^x and α^y are then evaluated such that E2 (α)and E3(α) are output in the next 2 clock cycles. Further clocking of the circuit will generate the remaining 3 error spectrum values.

Thus we can find appropriate values for α^x and α^y ($\alpha^x = \alpha^5$, $\alpha^y = \alpha^2$) that satisfy:
$\alpha^4\alpha^x + \alpha^2\alpha^y = \alpha$
$\alpha^2\alpha^x + \alpha\alpha^y = \alpha$
recursively from above we complete the error spectrum E = (α^4, α^2, α, α, α^4 , 0, α^2)

b) (6)
In transform domain compression samples are grouped into blocks which are transformed into a domain which allows a more compact representation. Compression is achieved by then discarding the less important information

An attractive transform would have:

- a good compaction efficiency of energy
- Is invertible and separable
- Has image independent basis
- Has fast algorithms for computation and implementation

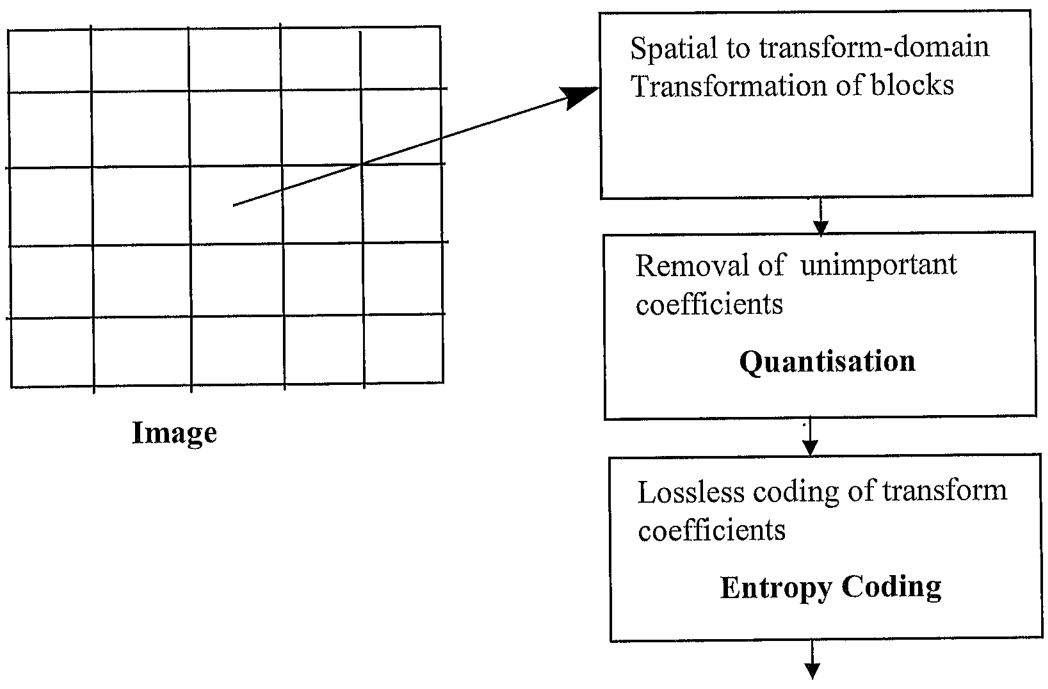


Image
Spatial to transform-domain Transformation of blocks
Removal of unimportant coefficients
Quantisation
Lossless coding of transform coefficients
Entropy Coding

Generic transform-based compression system

c)

i) (8)

After de-quantisation the compressed block doe~~s~~ not change and therefore 8=point 1D-IDCT is applied to [25, -13, 0, -1, 0, 0, 0, 0]

Applying the IDCT equation $x_n = \sqrt{\frac{2}{N}} \sum_{k=0}^{N-1} \alpha_k X_k \cos\left[\frac{(2n+1)k\pi)}{2N}\right]$ with N =8 we obtain:

$X0= 1/2[1/\sqrt{2}x\ 25 \cos 0 -13 \cos\pi/16 - \cos 3\pi/16]= 2.05 \approx 2$
$X1=1/2[1/\sqrt{2}x\ 25 \cos 0 -13 \cos 3\pi/16 - \cos 9\pi/16]= 3.53 \approx 4$
$X2=1/2[1/\sqrt{2}x\ 25 \cos 0 -13 \cos 5\pi/16 - \cos 5\pi/16]= 5.72 \approx 6$
$X3=1/2[1/\sqrt{2}x\ 25 \cos 0 -13 \cos 7\pi/16 - \cos 21\pi/16]= 7.85 \approx 8$
$X4=1/2[1/\sqrt{2}x\ 25 \cos 0 -13 \cos 9\pi/16 - \cos 27\pi/16]= 9.83 \approx 10$
$X5=1/2[1/\sqrt{2}x\ 25 \cos 0 -13 \cos 11\pi/16 - \cos 33\pi/16]= 11.96 \approx 12$
$X6=1/2[1/\sqrt{2}x\ 25 \cos 0 -13 \cos 13\pi/16 - \cos 39\pi/16]= 14.15 \approx 14$
$X7=1/2[1/\sqrt{2}x\ 25 \cos 0 -13 \cos 15\pi/16 - \cos 45\pi/16]= 15.63 \approx 16$

The result would approximate the line block [2,4,6,8,10,12,14,16] quite well.
Students to identify original line equation/patern from the results above

ii) (2)

Blocking effects (block transform)
Reduced Energy compaction efficiency for very weakly correlated data