EEE414/EEE6410: Computer Communications/Digital communications

Model Solutions for 12/13 session

1.

a.

[2 marks]

Asynchronous transmission means that there is no meaningful synchronisation between the transmitter and receiver and there may be unpredictable gaps between transmitted data characters. Used over short distances and in low data rate/low bandwidth applications such as connecting a terminal or a keyboard to a computer, MIDI (musical instrument digital interface), remote controls, etc... To enable reliable communication in this case, data is broken down into one character or one byte at a time and framed into specific patterns or format comprising extra redundant synchronisation bits since the clocks used to transmit and recover the digital signals are independent from each other. Both receiver and transmitter use master clocks that run at some multiple, usually 16, of the baud rate.

In synchronous transmission the transmitter and receiver are synchronised. Therefore there is no need for extra redundant synchronisation bits, and characters are sent as a continuous bit stream without any gaps. Synchronous transmission is hence more suited to high data rate and long distances communications. It is the most widely used means of transmission. Synchronisation is often achieved by incorporating the clock information into the transmitted data signal using a suitable data encoding technique.

b.

[7 marks]

Worst case scenario: Transmitter clock fast, Receiver Clock slow and Start bit detected a cycle late

Ftc = Fn(1+δ)= 1/Ttc (transmitter clock fast)

Frc=Fn(1- δ) = 1/Trc (receiver clock slow)

Tmid-start bit = (m/2+1)Trc (start-bit detected 1 cycle late)

T character = (N-1)xmxTrc + (m/2+1)Trc = (mN-m/2+1)Trc = NxmxTtc

(mN-m/2+1) [1/Fn(1- δ)] = mN [1/Fn(1+δ)]

mN (1- δ) = (mN-m/2+1) (1+ δ)

mN-mNδ = mN +mNδ –m/2 –m/2δ +1 + δ

In this case: m=8, N=11; so:

88-88δ= 88 +88δ –4 –4δ +1+ δ

173δ= 3 giving a tolerance δ=±1.73%

c.

[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

d.

[8 marks]

Sequence: **101100010100**

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

- A Polar Biphase code since it employs two voltage levels (polarities) and uses positive to negative and negative to positive transitions.
- Non-DC free as transitions are not uniformly compensated (even number of 1s between 0s)
- Middle bit transitions for 1s and end of bit transitions for any 2 consecutive zeros only, hence less transitions overall than comparable biphase codes as in Fig.Q.1.b.
- Less transitions results in less bandwidth requirements but also less noise immunity.
- Clock Synchronisation is difficult due to non-uniform change in position and number of transitions.
- 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:

- Biphase encoding, employs positive to negative and negative to positive transitions

- The transition at the start of the bit period is used for synchronisation, but the bits are represented by a presence (for a 0), or an absence (for a 1), of a transition in the middle of the bit period.
- Therefore incurring no latency in recovering the bits and offering improved noise immunity.
- Synchronisation and bit representation, using only 2 voltage levels, as opposed to 3 levels as in RZ type encoding. (less complex to implement)
- DC-free code (transitions compensated)
- 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.

2.
a.

i. [4 marks]
0110 = $i(x) = x^2 + x$. Next we need to find the 3 parity check bits bits generated by the generator polynomial gx)= $x^3 + x^2 + 1$ by calculating the remainder of division of $i(x)x^{n-k} = i(x)\,x^3 = x^5 + x^4$ by g(x) using modulo-2 long hand division as follows:

$x^5 + x^4$	$x^3 + x^2 + 1$
$x^5 + x^4 + x^2$	x^2
x^2 (Remainder = 100)	

Therefore the codeword is: $x^5 + x^4 + x^2$ = 0110100.

ii. [8 marks]
The error pattern = 0100000 = x^5 (2nd MSB corrupted) therefore received codeword is $x^4 + x^2$ = 0010100.

Meggit decoding

- Generate the syndrome, determine from this if there is an error in the MSB (coefficient of x_{n-1}), if not, output rn-1
- If MSB is in error, invert rn-1 and output it, we also remove the effect of this error on the original syndrome by adding ei to the current syndrome
- The syndrome register is then shifted and the test for an error in the current MSB is made (coefficient of xn-2) and the procedure above is repeated till the LSB
- This decoding strategy takes *n* clock cycles, if a correctable error pattern exists, after *n* clock cycles the syndrome registers will contain 0 (or a zon-zero syndrome if the effect of the errors is not removed - ei is not added to the syndrome)

Therefore in Meggitt decoding we need to check against the syndrome corresponding to an error in the MSB. Assuming an MSB error, the above codeword becomes $x^6 + x^5 + x^4 + x^2$= 1110100

To find the syndrome we divide the MSB erroneous codeword by g(x) and find the remainder.

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

$x^6 + x^5 + x^4 + x^2$	$x^3 + x^2 + 1$
$x^6 + x^5 + x^3$	$x^3 + x$
$x^4 + x^3 + x^2$	
$x^4 + x^3 + x$	
$x^2 + x$	
Remainder = $x^2 + x$= 110	

First check if error in MSB
The error pattern = 0100000 = x^5 therefore received codeword is $x^4 + x^2$ = 0010100.

$x^4 + x^2$	$x^3 + x^2 + 1$

x^4+x^3+x $\quad$ $x+1$

x^3+x^2+x
x^3+x^2+1
$x+1$

Remainder = $x+1 = 110 \neq = x^2+x = 110$ therefore the error is not in the first MSB

After the first cyclic shift, the erroneous codeword becomes $x^5+x^3 = 0101000$

Check if error in current MSB

x^5+x^3	x^3+x^2+1
$x^5+x^4+x^2$	x^2+x
$x^4+x^3+x^2$	
x^4+x^3+x	
x^2+x	

Remainder = $x^2+x = 110$ which is the syndrome corresponding to an erroneous MSB; to correct, invert second MSB of erroneous codeword to obtain correct codeword $x^5+x^4+x^2 = 0110100$

c.
[~~7~~8 marks]
$r(x)= x^{13}+x^{11}+x^9+x^7+x^2+x$
$S_1 = r(\alpha) = \alpha^{13}+\alpha^{11}+\alpha^9+\alpha^7+\alpha^2+\alpha = \alpha^2$
$S_3 = r(\alpha^3) = \alpha^{39}+\alpha^{33}+\alpha^{27}+\alpha^{21}+\alpha^6+\alpha^3 = \alpha^8$

$$\frac{S_3+S_1^3}{S_1} = \alpha^6+\alpha^4 = \alpha^{12} \text{ and therefore } \sigma(x) = x^2+S_1x+\frac{S_3+S_1^3}{S_1} = x^2+\alpha^2x+\alpha^{12}$$

Try the 4 MSB positions for errors (evaluate $\sigma(\alpha^i)$ for i=14,13,12,11)
$\sigma(\alpha^{14}) = \alpha^{28}+\alpha^{16}+\alpha^{12} = 0$ so 1st error in the MSB position
$\sigma(\alpha^{13}) = \alpha^{26}+\alpha^{15}+\alpha^{12} = 0$ this is a root so 2st error in the 2nd MSB position
No need to work out the rest

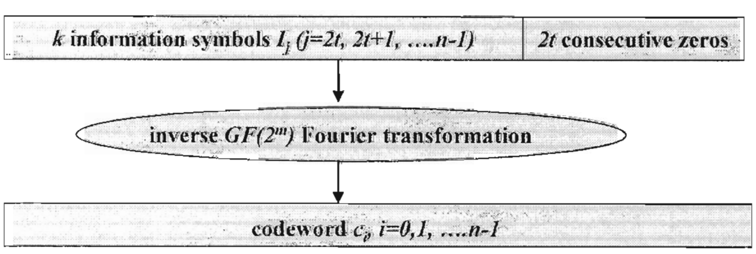
To correct r(x) invert the r_{14} and r_{13} bits to give $r(x)= x^{14}+x^{11}+x^9+x^7+x^2+x$ =(100101010000110)

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)

RS encoding can be performed in the frequency domain using a *GF* Fourier transform, this is possible because the *2t* consecutive powers of α roots in time domain correspond to *2t* consecutive zeros in frequency domain. Therefore *k* information symbols I_j ($j=2t, 2t+1, \ldots n-1$) can be encoded by letting $C_j = 0, j=0,\ldots 2t-1$ and $C_j = I_j, j=2t,2t+1, \ldots n-1$ then perform an inverse $GF(2^m)$ Fourier transformation on C_j to generate the codeword c_i, $i=0,1, \ldots n-1$



RS decoding can also be performed in the frequency domain by recursive extension of the error spectrum as follows:

1 Apply forward *GF* Fourier Transform to the received word to obtain the original frequency domain message

2 If the message is not corrupted, *2t* consecutive zeros will appear in the transformed word and the information symbols is retrieved

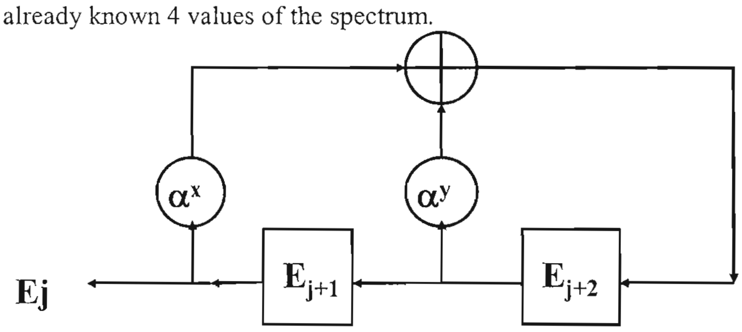
3 In the case of errors, some or all of the *2t* zero positions will be non-zero. The complete error spectrum is then obtained by recursive extension from the *2t* error spectrum.

4 A time domain complete error polynomial is then obtained by inverse *GF* Fourier Transform of the error spectrum. The time domain error polynomial is added (*XORed*) with the received message for correction

5 A forward Fourier Transform is then applied to the corrected message to obtain the original frequency domain message

b. (**4 marks**)

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 error spectrum values from the already known 4 values of the spectrum.



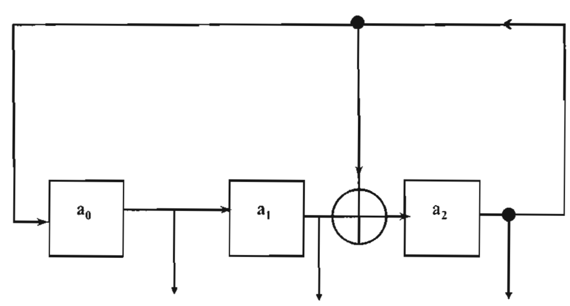
The process starts by preloading, registers Ej+1 and Ej+2 with E0 and E1, respectively such that on the first clock cycle E0 is output followed by E1 on the next cycle. Appropriate values for α^x and α^y are then evaluated such that E2 and E3 are output in the next 2 clock cycles. Further clocking of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

c.

i. **[3 marks]**

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

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



ii. [**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 and 2t =4 =n-3, therefore n=7 and the code is (7,3) RS code. This code can 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.

iii. **[7 marks]**

with $\boldsymbol{m(x)} = \alpha = (1, 0, \alpha)$: Append $2t=4$ consecutive zeros before the $k=3$ information symbols to get: $Cj=(0,0,0,0, \alpha, 0, 1)$, $j=0,1,\ldots 6$; then perform an inverse $GF(2^3)$ Fourier Transform on Cj to obtain the time-domain codeword ci, $i=0,1,..6$

$$ci=\sum_{j=0}^{6} Cj\,\alpha^{-ij} \qquad \text{for } i=0,1, \ldots 6$$

$c0= C0\,\alpha^0 + C1\,\alpha^0 + C2\,\alpha^0 + C3\,\alpha^0 + C4\,\alpha^0 + C5\,\alpha^0 + C6\,\alpha^0 = C4\,\alpha 0 + + C6\,\alpha^0 = \alpha + 1 = \alpha^5$

$c1= C0\,\alpha^0 + C1\,\alpha^{-1} + C2\,\alpha^{-2} + C3\,\alpha^{-3} + C4\,\alpha^{-4} + C5\,\alpha^{-5} + C6\,\alpha^{-6} = C4\,\alpha^{-4} + C6\,\alpha^{-6} = \alpha^{-3} + \alpha^{-6} = \alpha^4 + \alpha = 1 + \alpha + \alpha^2 + \alpha = \alpha^3$

$c2= C4\alpha^{-8} + C6\,\alpha^{-12} = 1+\alpha^2 = \alpha^3$

$c3= C4\,\alpha^{-12} + C6\,\alpha^{-18} = \alpha^3 + \alpha^3 = 0$

$c4= C4\,\alpha^{-16} + C6\,\alpha^{-24} = \alpha^6 + \alpha^4 = 1$

$c5= C4\alpha^{-20} + C6\,\alpha^{-30} = \alpha^2 + \alpha^5 = \alpha^4$

$c6= C4\alpha^{-24} + C6\,\alpha^{-36} = \alpha^5 + \alpha^6 = \alpha^3$

Giving $\mathbf{c} = (\alpha^5, \alpha^3, \alpha^3, \mathbf{0}, \mathbf{1}, \alpha^4, \alpha^3)$

4.

(a) [3 marks]

In block codes k information symbols are formed into a word $(I_1, I_2, \ldots, I_k)$. This information word is then encoded into n codeword symbols $(C_1, C_2, \ldots, C_n)$.$(n > k)$.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) [8 marks]

i. [6 marks]
The sequence **00 01 01 00 00 11 10 01 00** is received at the Viterbi decoder with no more than than 3 bit errors..

Depth	00	01	01	00	00	11	10	01	00
0=00	0	1	2	4*	3	5*	4*		
1=11			3	3	4*	3	4*	4*	5*
2=10		2	3	3	4*	3	3		
3=01				2	3	4*	5*	3	
4=11	2	1	2	3	5*	3	4*		
5=00			3	5*	2	5*	4*	4*	3
6=01		4*	1	4*	4*	3	5*		
7=10				2	3	4*	3	5*	

* : stop more than 3 errors

Corrected data is therefore: **011011101**

ii. [2 mark]
In decoding convolutional codes, further information provided by the receiver can be used to advantage to determine the most likely code in situations where standard decoding fails. This is known as soft-decision decoding.

c) [2 marks]
In transform based 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.
The DCT is attractive in this respects since it:

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

ii) (7 marks)

$$\begin{bmatrix} 200 & 50 & 0 & 0 \\ 20 & 0 & 0 & 0 \\ 5 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

Need to work out the 2D-IDCT:

Using a row-column decomposition first we work out the 1-D IDCT along the rows:

we get :

$$\begin{bmatrix} 132.6 & 113.5 & 86.4 & 67.3 \\ 10 & 10 & 10 & 10 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

then we apply another 1-D IDCT but this time along the columns of the above result as we need only to obtain the first 2 rows of the original data we need only to work out the 1-D IDCT of the first 2 columns here that is :

$$\begin{bmatrix} 72.8 & 69 & 63.6 & 59.8 \\ 63.3 & 59.4 & 54 & 50.2 \\ * & * & * & * \\ * & * & * & * \end{bmatrix}$$