DEPARTMENT OF ELECTRONICS AND
TELECOMMUNICATION ENGINEERING
UNIVERSITY OF MORATUWA

EN2053 COMMUNICATION SYSTEMS AND NETWORKS

GROUP ASSIGNMENT

Authors:	*Index Number:*
Bandara D.R.K.W.M.S.D.	190071B
Munasinghe M.M.R.H.	190399L
Thanushan K.	190621M

This is submitted as a partial fulfilment for the module EN2053.

November 19, 2021

* We assumed processing delays are negligible and frames do not lose in the channel througout this assignment.

(a) Encoder

This function uses modulo 2 division to generate a codeword from a given dataword.

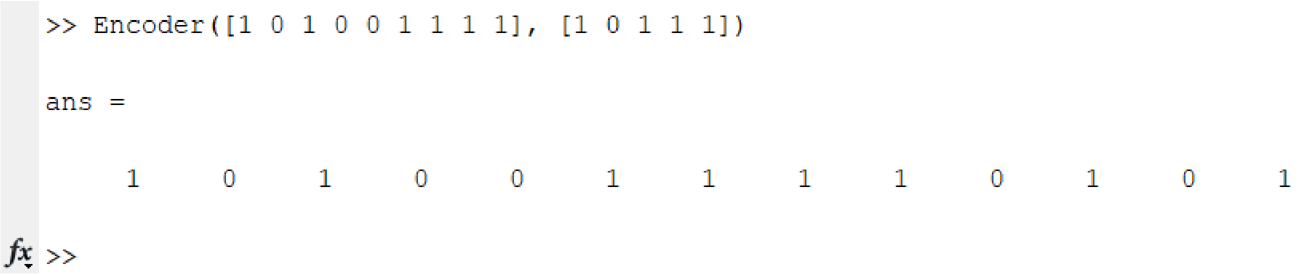
```
function [cdwrd] = Encoder(dataword,divisor)
sizeOfdivisor = size(divisor);% length of the divisor
agdataword = dataword;
%%%%%%
%add 0 s to dataword to get augmented dataword before modulo 2 division
for i=1:sizeOfdivisor(2)-1
    agdataword(end+1)=0;
end
%%%%%%%%

%%%%%%%%
%modulo 2 division
count = sizeOfdivisor(2);
count1 = 1;
word = agdataword(count1:count);
sizeOfagdataword = size(agdataword);

while true

    c = xor(word, divisor);
    g = find(c==0);
    for i=1:length(g)
        if g(i)==i
            c(1) =  [];
        end
    end
    if sizeOfdivisor(2)-length(c) > length(agdataword)-count
        remainder = [c agdataword(count+1:end)];
        for i=0: length(divisor)-1-length(remainder)-1
            remainder = [0 remainder];% add zeros to make remainder length is equal to (divisior length -1)
        end
        break;
    end
    for i = 1:sizeOfdivisor(2)-length(c)
        c(end+1) = agdataword(count+1);
        count = count +1;
    end
    word = c;
end
cdwrd = [dataword remainder]; % after get the remainder , it combined with the dataword to get the codeword
end
```

Decoder

This function uses modulo 2 division to generate the syndrome from a given codeword.

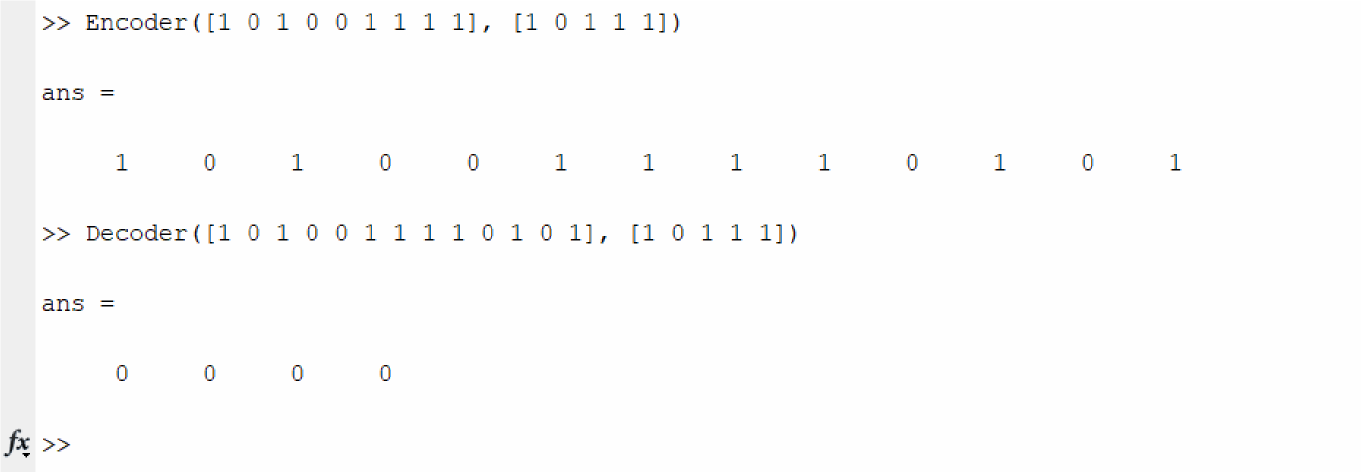
```
function [syndrome] = Decoder(codeword,divisor)
sizeOfdivisor = size(divisor);

%modulo 2 division

count = sizeOfdivisor(2);
word = codeword(1:count);
while true
    c = xor(word, divisor);
    g = find(c==0);

    for i=1:length(g)
        if g(i)==i
            c(1) =  [];
        end
    end

    if sizeOfdivisor(2)-length(c) > length(codeword)-count
        syndrome = [c codeword(count+1:end)];
        syndrome = [zeros(1, length(divisor)-length(syndrome)-1) syndrome];
        break;
    end

    for i = 1:sizeOfdivisor(2)-length(c)
        c(end+1) = codeword(count+1);
        count = count +1;
    end
    word = c;
end
end
```

(b) Encoder function has added CRC codes at the end of the dataword.



```
>> Encoder([1 0 1 0 0 1 1 1 1], [1 0 1 1 1])

ans =

     1     0     1     0     0     1     1     1     1     0     1     0     1

fx >>
```

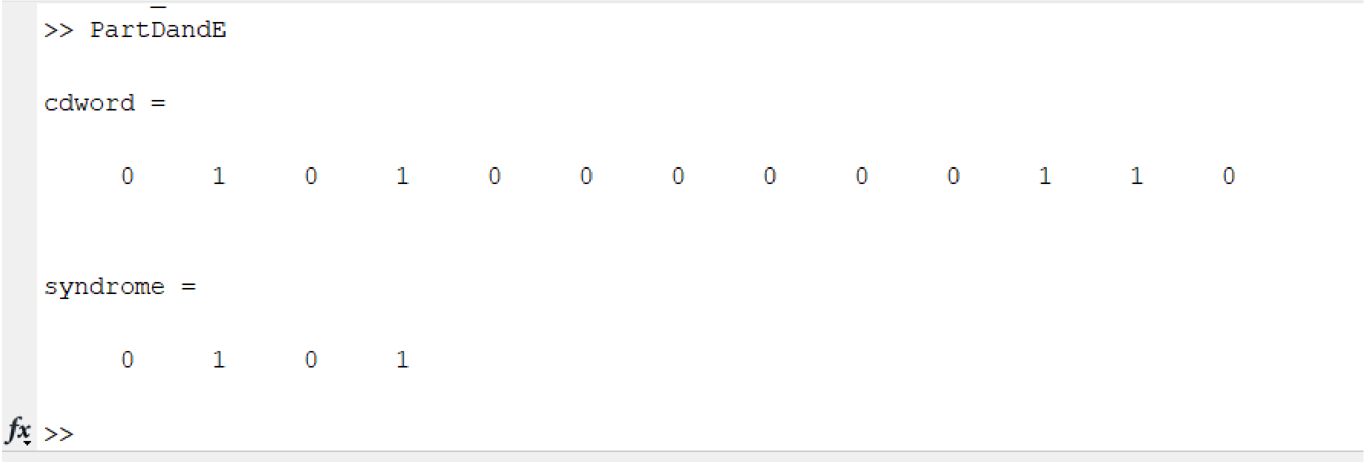
(c) Decoder function calculate the syndrome from the given codeword.
In this case, since there are no errors all the bits of the syndrome is equal to zero.



```
>> Encoder([1 0 1 0 0 1 1 1 1], [1 0 1 1 1])

ans =

     1     0     1     0     0     1     1     1     1     0     1     0     1

>> Decoder([1 0 1 0 0 1 1 1 1 0 1 0 1], [1 0 1 1 1])

ans =

     0     0     0     0

fx >>
```

(d) & (e) Since there is an error probability in binary channel the transmitted codeword has changed at the receiver. Therefore syndrome is not equal to zero.

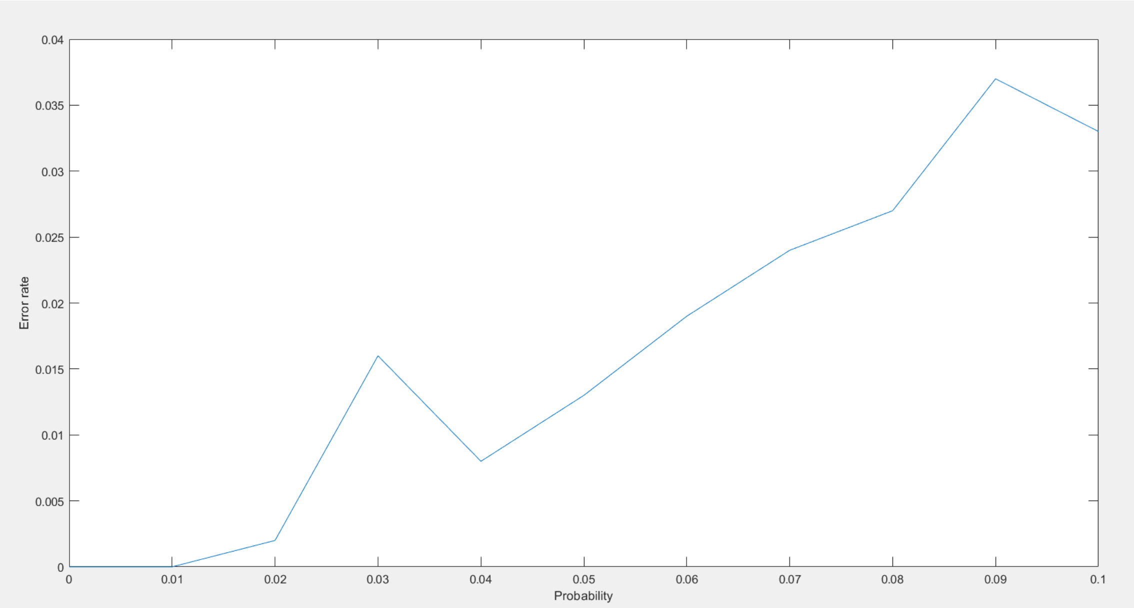
```
dataword = [1 0 1 0 0 1 1 1 1];
divisor = [1 0 1 1 1];
p = 0.5;
cdword = Encoder(dataword, divisor);%encode at sender
cdword = bsc(cdword, p);%transmit through the BSC channel
syndrome = Decoder(cdword, divisor);%decode at the reciever
display(cdword);%recived codeword at the reciever
display(syndrome);% Syndrome at the reciever
```



```
>> PartDandE

cdword =

     0     1     0     1     0     0     0     0     0     0     1     1     0


syndrome =

     0     1     0     1

fx >>
```

(f) The number of frames with transmission errors are increasing with the probability of error.

```matlab
datawords = mod(randi(2,10000,9),2); % generate 10**4 datawords with 9 bits
divisor = [1 0 1 1 1];
check = zeros(1,length(divisor)-1); % define a zero vector with the length is equal to (divisor length -1)
ErrorRates = []; %define a vector to store the error rates at differnet probabilities
prob = [];%define a vector to store the differnet probabilities
for p = 0:0.1:1 % change probability from 0 to 1
    count = 0; % variable to count codewords at reciever with errors
    for i = 1:10000
        dataword = datawords(i,:);
        cdword = Encoder(dataword, divisor);% encode for get the codeword
        cdword = bsc(cdword, p);%transmit through BSC channel
        syndrome = Decoder(cdword, divisor);% calculate syndrome at recioever
        if syndrome ~= check % check whether the syndrome is all zero or not
            count = count + 1;% if not, then increment error count by 1
        end
    end

    prob(end+1)=p;
    ErrorRates(end+1)=count/10000; % calculate error rates
end
plot(prob,ErrorRates);
```



(g) Stop and Wait ARQ algorithm

```
clc;
clear all;
divisor = [1 0 0 0 0 0 1 1 1];%pre define the divisor for encoder and decoder
check = zeros(1,length(divisor)-1);
fprintf("Enter the following inputs\nNumber of frames should be in between 0 and 256\nNumber of bytes should be 2(no isssue if it increased, but it will take lot of time)\nProbabilities should be in between 0 and 1\n")
frames = input("Enter the number of frames(0-256): ");
len = input("Enter the frame length(number of data bytes)(0-2): ");
p = input("Enter the error probability of forward direction: ");
p1 = input("Enter the error probability of feedback direction: ");% get inputs from user for number of frames, number of bytes in a frame
%error probability for forward and feedback directions
datawords = randi([0,1],frames,len*8);%create datawords list with a given number of bytes of data

SN = 0;
RN = 0;% first, sequence number and request number is equal to zero
NumOfTransmissions = 0;% first number of transmission is equal to zero
ACK = [zeros(1, 8-length(Bin(RN))) Bin(RN) datawords(SN+1,:)];% for initialize the transmission, create an artificial ACK frame
%with request number 0 and fed it in to the sender
cdwrd = Encoder(ACK,divisor);
trcdwrd = cdwrd;
% for initialize the transmission, create an artificial ACK frame
%with request number 0 and fed it in to the sender
fprintf("Initialized transmission !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!\n");
while true
    %% sender side

    if Decoder(trcdwrd, divisor) == check
        if (SN==  Dec(trcdwrd(:, 1:8)))||(SN+1 ==Dec(trcdwrd(:, 1:8)));
            SN = Dec(trcdwrd(:, 1:8));%length(trcdwrd)-length(datawords(SN,:))-length(divisor)-1+2));
        end
    end
    %first, sender decode the ACK frame from the reciever and check for
    %errors. if there is error in ACK frame, sender transmit previous frame
    %again. if there is no any detectable error in ACK frame, sender
    %extract the request number from the frame(fist 8 bits) and
```

```
     convert it
    %into decimel for find the frame number which is request by
     recuiver
    %and transmit it

    fprintf("Transmitter is requested by reciever for frame %d
    and transmitted it\n", SN);
    frame = [zeros(1, 8-length(Bin(RN))) Bin(SN) datawords(SN
    +1,:)];%creat frame with requested frame number by reciever
    codeword = Encoder(frame,divisor);%Encode that frame

    trmtdcodeword = bsc(codeword,p);%transmit through channel
    NumOfTransmissions = NumOfTransmissions + 1;%increment
    number of transmission count by 1
    %%
    %% Reciver side
    syndrome = Decoder(trmtdcodeword, divisor);%reciever find
    syndrme ana check it for errors
    if (syndrome == check) & (Dec(trmtdcodeword(:, 1:8)) == RN)
        fprintf("Correctly recieved frame %d in reciever\n", SN
    );
        %SN = SN + 1;
        RN = SN + 1;
    else
        fprintf("Not Correctly recieved frame %d in reciever\n
    ", SN);
        RN = SN;
    end
    %if there is an error in frame transmitted by Tx, reciever
    send ACK
    %with exsisiting RN and if there is no any error, reciever
    send ACK to
    %Tx with next RN

    ACK = [zeros(1, 8-length(Bin(RN))) Bin(RN) datawords(SN
    +1,:)];%create frame before transmit
    fprintf("Reciever sent a request to transmitter for frame %
    d\n", RN);
    if RN == frames
        fprintf("All the frames have successfully transmitted.\
    nTransmission Ended
    !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!");
        break;
    end
    cdwrd = Encoder(ACK,divisor);%encode and transmit
    trcdwrd = bsc(cdwrd, p1);
    %%
end
display(NumOfTransmissions);
```

Assume timeout delay is equal to 150 μs and assume timeout delay is never occured since there are no any frame lost or abnormal delays in the channel during the transmission.

(h) When error probability is 2 x 10^{-4}, we did 6 trials and the results are shown below.

Trial	**Number of Transmission**
1	260
2	263
3	264
4	266
5	261
6	261

We took the average value of above values as the expected number of retransmissions for a codeword.

Expected number of retransmissions for a codeword = 1.0234

(i) When error probability is 6 x 10^{-4}, we did 6 trials and the results are shown below.

Trial	**Number of Transmission**
1	267
2	267
3	275
4	265
5	266
6	274

We took the average value of above values as the expected number of retransmissions for a codeword.

Expected number of retransmissions for a codeword = 1.0508

The expected number of retransmissions when $p = 6 \times 10^{-4}$ is more than that of when $p = 2 \times 10^{-4}$. Therefore the number of retransmissions increases when the error probability increases.

(j) Efficiency of the Stop and Wait ARQ when p = 2 × 10^{-4}

$$= \frac{25}{25+25+15+15} * \frac{1}{1.0234} * 100\%$$

$$= 30.53\%$$

Efficiency of the Stop and Wait ARQ when p = 6 × 10^{-4}

$$= \frac{25}{25+25+15+15} * \frac{1}{1.0508} * 100\%$$

$$= 29.70\%$$

All code files are uploaded to the moodle submission portal

****