1. Amplitude Shift Keying

clear;

clc;

b =[0 1 0 1 1 1 0];

n = length(b);

t = 0:.01:n;

for i = 1:n

bw(i\*100:(i+1)\*100) = b(i);

end

bw = bw(100:end);

sint = sin(2\*pi\*t);

st = bw\*sint;

subplot(3,1,1)

plot(t,bw)

grid on ;

axis([0 n -2 +2])

subplot(3,1,2)

plot(t,sint)

grid on ;

axis([0 n -2 +2])

subplot(3,1,3)

plot(t,st)

grid on ;

axis([0 n -2 +2])

2. Phase Shift Keying

b = [0 1 0 1 1 1 0];

n = length(b);

t = 0:.01:n;

for i = 1:n

if (b(i) == 0)

b\_p(i) = -1;

else

b\_p(i) = 1;

end

end

for j = 1:n

bw(j\*100:(j+1)\*100) = b\_p(j);

end

bw = bw(100:end);

sint = sin(2\*pi\*t);

st = bw.\*sint;

subplot(3,1,1)

plot(t,bw)

grid on ;

axis([0 n -2 +2])

subplot(3,1,2)

plot(t,sint)

grid on ;

axis([0 n -2 +2])

subplot(3,1,3)

plot(t,st)

grid on ; axis([0 n -2 +2])

3. Frequency Shift Keying

clear;

clc;

b = [0 1 0 1 1 1 0];

n = length(b);

t1 = 0:.01:n;

sint1=sin(2\*pi\*t1);

sint2=sin(2\*pi\*110\*t1);

for i = 1:n

bm(i\*100:(i+1)\*100) = b(i)

end

bm = bm(100:end)

+

for i = 1:length(bm)

if bm(i)==0

bm(i) = -1

end

end

% bm(bm==0)=-1

for i = 1:length(bm)

if bm(i) == -1

bw(i)=sint1(i)

else

bw(i)=sint2(i)

end

end

subplot(4,1,1)

plot(t1,bm)

grid on ;

axis([0 n -2 +2])

subplot(4,1,2)

plot(t1,sint1)

grid on ;

axis([0 n -2 +2])

subplot(4,1,3)

plot (t1, sint2);

grid on ;

axis([0 n -2 +2])

subplot(4,1,4)

plot (t1, bw);

grid on ;

axis([0 n -2 +2])

4. AM Modulation

clear;

clc;

t=0:0.0005:1;

% Equation of modulating (Message signal) signal

ym=5\*sin(2\*pi\*5\*t);

subplot(3,1,1);

plot(t,ym), grid on;

title ( ' Modulating Signal ');

xlabel ( ' time(sec) ');

ylabel (' Amplitude');

% Equation of carrier signal

yc=5\*sin(2\*pi\*100\*t);

subplot(3,1,2);

plot(t,yc), grid on;

title ( ' Carrier Signal ');

xlabel ( ' time(sec) ');

ylabel (' Amplitude');

% Amplitude modulated signal

y=ym.\*yc;

subplot(3,1,3);

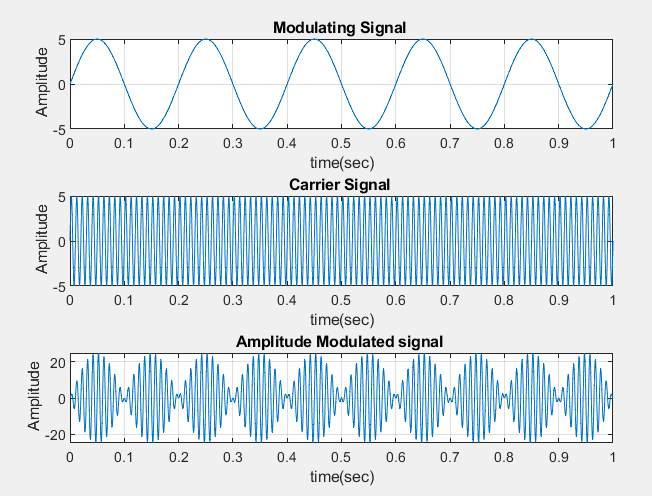
plot(t,y);

title ( ' Amplitude Modulated signal ');

xlabel ( ' time(sec) ');

ylabel (' Amplitude');

grid on



5. Phase Modulation

%Fs must be at least 2\*Fc

clear;

clc;

t=0:0.005:1;

% Equation of carrier signal

fc=10;

x=sin(2\*pi\*fc\*t);

subplot(3,1,1);

plot(t,x), grid on;% Graphical representation of Modulating signal

title ( ' Carrier signal ');

xlabel ( ' time(sec) ');

ylabel (' Amplitude');

% Equation of (modulating (mEssage) signal

fs=20;

yc=sin(2\*pi\*fs\*t);

subplot(3,1,2);

plot(t,yc), grid on;

title ( ' modulating signal ');

xlabel ( ' time(sec) ');

ylabel (' Amplitude');

% Equation of modulated signal

phasedev = pi/2;

tx = pmmod(x,fc,fs,phasedev);

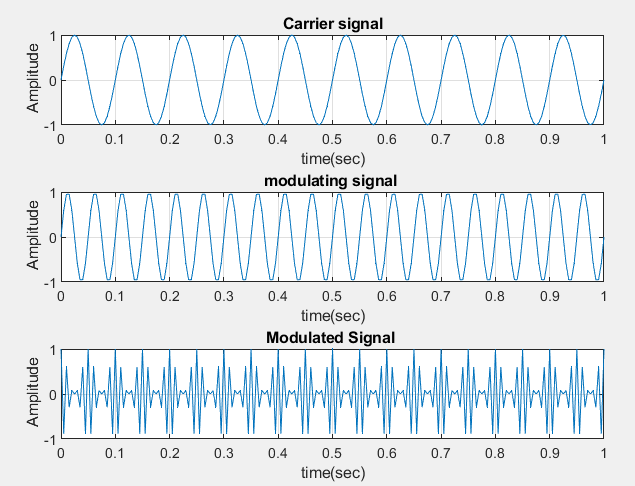
subplot(3,1,3);

plot(t,tx), grid on;

title ( ' Modulated Signal ');

xlabel ( ' time(sec) ');

ylabel (' Amplitude');



6. Unipolar NRZ Encoding

clear;

clc;

b =[1 0 1 1 1 1 0];

n = length(b);

t = 0:.01:n;

for i = 1:n

bw(i\*100:(i+1)\*100) = b(i);

end

bw = bw(100:end);

subplot(1,1,1)

plot(t,bw)

grid on ;

axis([0 n -2 +2])

7. Polar NRZ-L Encoding

b = [0 1 0 0 1 1 1 0];

n = length(b);

t = 0:.01:n;

for i = 1:n

if (b(i) == 0)

b\_p(i) = 1;

else

b\_p(i) = -1;

end

end

for j = 1:n

bw(j\*100:(j+1)\*100) = b\_p(j);

end

bw = bw(100:end);

subplot(1,1,1)

plot(t,bw)

grid on ;

axis([0 n -2 +2])

8. %Polar NRZ-1

b = [0 1 0 0 1 1 1 0];

n = length(b);

t = 0:.001:n;

for i = 1:n

if (b(i) == 0)

b\_p(i) = 1;

else

b\_p(i) = -1;

end

end

for j = 1:n

bw(j\*1000:(j+1)\*1000) = b\_p(j);

end

bw = bw(1000:end);

subplot(2,1,1)

plot(t,bw)

grid on;

lastbit = 1;

for i=1:n

if b(i)==1

x(i\*1000:(i+1)\*1000) = -lastbit;

lastbit = -lastbit;

else x(i\*1000:(i+1)\*1000) = lastbit;

end

end

x = x(1000:end);

subplot(2,1,2)

plot(t,x)

grid on;

9. % Bipolat AMI

b = [0 1 0 0 1 0];

n = length(b);

t = 0:.001:n;

for i = 1:n

if (b(i) == 0)

b\_p(i) = 0;

else

b\_p(i) = 1;

end

end

for j = 1:n

bw(j\*1000:(j+1)\*1000) = b\_p(j);

end

bw = bw(1000:end);

subplot(2,1,1)

plot(t,bw)

grid on;

lastbit = 1;

for i=1:n

if b(i)==1

x(i\*1000:(i+1)\*1000) = -lastbit;

lastbit = -lastbit;

else x(i\*1000:(i+1)\*1000) = 0;

end

end

x = x(1000:end);

subplot(2,1,2)

plot(t,x)

grid on;

10: Pseudoternary

Assignment

11. Manchester Encoding

b = [0 1 0 0 1 0];

n = length(b);

t = 0:.001:n;

for i = 1:n

if (b(i) == 0)

b\_p(i) = -1;

else

b\_p(i) = 1;

end

end

for j = 1:n

bw(j\*1000:(j+1)\*1000) = b\_p(j);

end

bw = bw(1000:end);

subplot(2,1,1)

plot(t,bw)

grid on;

for i=1:n

if b(i)==1

x(i\*1000:(i+0.5)\*1000) =-1 ;

else x(i+0.5\*1000:(i+1)\*1000) = 1;

end

end

x = x(1000:end);

subplot(2,1,2)

plot(t,x)

grid on;

12: Differential Manchester

Assignment

**Checksum:**

#include<bits/stdc++.h>

using namespace std;

string add(string a, string b)

{

string sum;

int carry= 0;

for(int i=7; i>=0; i--)

{

if(carry== 0)

{

if(a[i]=='1' && b[i]=='1')

{

sum.push\_back('0'), carry= 1;

}

else if(a[i]=='1' || b[i]=='1')

{

sum.push\_back('1');

}

else{

sum.push\_back('0');

}

}

else{

if(a[i]=='1' and b[i]=='1')

{

sum.push\_back('1');

}

else if(a[i]=='1' or b[i]=='1')

{

sum.push\_back('0');

}

else{

sum.push\_back('1'),carry= 0;

}

}

}

reverse(sum.begin(),sum.end());

if(carry)return add(sum,"00000001");

return sum;

}

string complement(string s)

{

for(int i=0; i<s.length(); i++)

{

if(s[i]=='1')

{

s[i]='0';

}

else{

s[i]='1';

}

}

return s;

}

int main()

{

string a, b;

cout<< "Enter 8 bit code for a and b : ";

cin>> a>> b;

string checksum = complement(add(a,b));

cout<<a<<" "<<b<<" "<<checksum<<endl;

string x,y,z;

cout<< "Enter receiver receive: ";

cin>> x>>y>>z;

string res= add(add(x,y),z);

int error=0;

for(auto c: res){

if(c=='0')

error = 1;

}

if(error)

cout<< "NOT CORRECT";

else

cout<< "CORRECT";

}

**CRC:**

#include<bits/stdc++.h>

using namespace std;

string calculate\_crc(const string& m,const string& d)

{

int m\_l= m.length();

int d\_l= d.length();

string extented\_m= m+ string(d\_l-1,'0');

for(int i=0; i<m\_l; i++)

{

if(extented\_m[i]=='1')

{

for(int j=0; j<d\_l; j++)

{

extented\_m[i+j]=(extented\_m[i+j]==d[j])? '0':'1' ;

}

}

}

return extented\_m.substr(m\_l);

}

string add\_crc\_to\_message(const string& m,const string& crc)

{

return m+crc ;

}

bool verify\_crc(const string &received\_m, const string &d)

{

string remainder= calculate\_crc(received\_m,d);

return remainder.find('1')== string::npos;

}

int main()

{

string m,d;

cout<< "Enter message: ";

cin>> m;

cout<< "Enter divisor: ";

cin>> d;

string crc= calculate\_crc(m,d);

string data\_w\_crc= add\_crc\_to\_message(m, crc);

cout<< "Data to be sent: "<< data\_w\_crc<< endl;

string r;

cout<< "Enter receiver : ";

cin>> r;

bool is\_data\_correct = verify\_crc(r,d);

cout<< "Receiver receive original data: "<< boolalpha<< is\_data\_correct<< endl;

}