DS LAB CODES SEM 4

Exp1: Write Matlab programs for the verification of truth tables of basic logic gates and their realization using universal logic gates.

```
clear all;
clc;
a=[0;0;1;1];
b=[0;1;0;1];
not_a=[0;1];
not=~not_a;
and=a&b;
or=a|b;
disp('Result for NOT gate for input A');
disp('    A     !A');
disp([not_a,not]);
disp('Result for AND gate');
disp('    A     B     A&B');
disp([a,b,and]);
disp('Result for OR gate');
disp('    A     B     A|B');
disp([a,b,or]);
```

//using and nand and nor gate

```
clear all;
clc;
A=[0;1];
a=[0;0;1;1];
b=[0;1;0;1];
display('Using NAND gate');
c=nand(A,A);
display('Result for NOT gate for input A');
disp('    A     !A');
disp([A,c]);
c=nand(a,a);
d=nand(b,b);
e=nand(c,d);
disp('Result for OR gate');
disp('    A     B     A|B');
disp([a,b,e]);
c=nand(a,b);
d=nand(c,c);
disp('Result for AND gate');
disp('    A     B     A&B');
```

```
disp([a,b,d]);
display('Using NOR gate');
c=nor(A,A);
display('Result for NOT gate for input A');
disp('    A     !A');
disp([A,c]);
c=nor(a,b);
d=nor(c,c);
disp('Result for OR gate');
disp('    A     B    A|B');
disp([a,b,d]);
c=nor(a,a);
d=nor(b,b);
e=nor(c,d);
disp('Result for AND gate');
disp('    A     B    A&B');
disp([a,b,e]);
```

function for nand

```
function n=nand(x,y)
a=x&y;
n=~a;
end
```

function for nor

```
function n = nor(x,y)
a=x|y;
n=~a;
end
```

Exp2: Write Matlab programs for half-adder, half-subtractor, full-adder, and full-subtractor.

```
clear all;
clc;
a=[0;0;1;1];
b=[0;1;0;1];
sum=xor(a,b);
carry=a&b;
disp('Truth table for Half Adder');
disp('     A     B    Sum   carry');
disp([a,b,sum,carry]);
a=[0;0;0;0;1;1;1;1];
b=[0;0;1;1;0;0;1;1];
c=[0;1;0;1;0;1;0;1];
x=xor(a,b);
sum=xor(x,c);
z=a&b;
w=x&c;
u=w|z;
disp('Truth table for Full Adder');
disp('     A     B     C    sum   cout');
disp([a,b,c,sum,u]);
a=[0;0;1;1];
b=[0;1;0;1];
disp('Truth table for Half Subtractor');
diff=xor(a,b);
borrow=~a&b;
disp('     A     B   Diff  borrow');
disp([a,b,diff,borrow]);
a=[0;0;0;0;1;1;1;1];
b=[0;0;1;1;0;0;1;1];
c=[0;1;0;1;0;1;0;1];
x=xor(a,b);
y=xor(x,c);
z=~a&b;
w=~x&c;
u=w|z;
disp('Truth table for Full Subtractor');
disp('     A     B     C    Diff   borrow');
disp([a,b,c,y,u]);
```

function for xor:

```
function n = xor(a,b)
x=a&~b;
y=~a&b;
n=x+y;
end
```

exp-3: Write Matlab programs for the design of 2-to-4 decoder and 3-to-8 decoder.

```
clear all;
clc;
x=[0;0;1;1];
y=[0;1;0;1];
O0=~x&~y;
O1=~x&y;
O2=~y&x;
O3=x&y;
disp('Verification of Truth table for 2-4 decoder');
disp('     A     B     Q0     Q1     Q2     Q3');
disp('_________________________________________');
disp([x,y,O0,O1,O2,O3]);
x=[0;0;0;0;1;1;1;1];
y=[0;0;1;1;0;0;1;1];
z=[0;1;0;1;0;1;0;1];
O0=~x&~y&~z;
O1=~x&~y&z;
O2=~x&y&~z;
O3=~x&y&z;
O4=x&~y&~z;
O5=x&~y&z;
O6=x&y&~z;
O7=x&y&z;
disp('Verification of Truth table for 3-8 decoder');
disp('     A     B     C     Q0     Q1     Q2     Q3     Q4     Q5     Q6     Q7');
disp('_________________________________________________________________________');
disp([x,y,z,O0,O1,O2,O3,O4,O5,O6,O7]);
x=[0;0;0;0;0;0;0;0;1;1;1;1;1;1;1;1];
y=[0;0;0;0;1;1;1;1;0;0;0;0;1;1;1;1];
z=[0;0;1;1;0;0;1;1;0;0;1;1;0;0;1;1];
w=[0;1;0;1;0;1;0;1;0;1;0;1;0;1;0;1];
O0=~x&~y&~z&~w;
O1=~x&~y&~z&w;
O2=~x&~y&z&~w;
O3=~x&~y&z&w;
O4=~x&y&~z&~w;
O5=~x&y&~z&w;
O6=~x&y&z&~w;
```

```
O7=~x&y&z&w;
O8=x&~y&~z&~w;
O9=x&~y&~z&w;
O10=x&~y&z&~w;
O11=x&~y&z&w;
O12=x&y&~z&~w;
O13=x&y&~z&w;
O14=x&y&z&~w;
O15=x&y&z&w;
disp('Verification of Truth table for 4-16 decoder');
disp('      A      B      C      D      Q0     Q1    Q2     Q3     Q4     Q5
Q6     Q7     Q8     Q9    Q10    Q11    Q12     Q13    Q14    Q15');
disp('_____________________________________________________________
__________________________________________________________________
____');
disp([x,y,z,w,O0,O1,O2,O3,O4,O5,O6,O7,O8,O9,O10,O11,O12,O13,O14,O15
]);
```

Exp-4: Write Matlab programs for the design of 2-to-1, 4-to-1, and 8-to-1 multiplexers.

```
clear all;
clc;
disp('2X1 Multiplexer:')
x='I0';
y='I1';
select=[0;1];
disp('   Select  Output');
disp('-----------------------');
for i=1:2
a=aamux(x,y,select(i));
disp(['       ',int2str(select(i)),'       ',a]);
end
disp('4X1 Multiplexer:')
x='I0';
y='I1';
z='I2';
w='I3';
select1=[0;0;1;1];
select2=[0;1;0;1];
disp('    S1      S0     Output');
disp('------------------------------');
for i=1:4
a=aamux(x,y,select2(i));
b=aamux(z,w,select2(i));
c=aamux(a,b,select1(i));
disp(['       ',int2str(select1(i)),'       ',int2str(select2(i)),'
',c]);
end
disp('8X1 Multiplexer:')
```

```
x='I0';
y='I1';
z='I2';
w='I3';
p='I4';
q='I5';
r='I6';
s='I7';
select2=[0;0;0;0;1;1;1;1];
select1=[0;0;1;1;0;0;1;1];
select0=[0;1;0;1;0;1;0;1];
disp('    S2     S1     S0    Output');
disp('-------------------------------------');

for i=1:8
a=aamux(x,y,select0(i));
b=aamux(z,w,select0(i));
c=aamux(p,q,select0(i));
d=aamux(r,s,select0(i));
e=aamux(a,b,select1(i));
f=aamux(c,d,select1(i));
g=aamux(e,f,select2(i));
disp(['      ',int2str(select2(i)),'      ',int2str(select1(i)),'
',int2str(select0(i)),'      ',g]);
end
```

function for aamux:

```
function output =aamux( I0,I1,select)
if(select==0)
output = I0;
elseif (select==1)
output = I1;
end
```

Exp5: Write Matlab programs for the realization of SR latch, SR flip-flop, D flip-flop, JK flip-flop.

```
s=[0;0;1;1];
r=[0;1;0;1];
q=0;
disp('Truth Table for SR LATCH : ');
disp("     S      R      Q      ~Q     CONDITION ");
disp('-----------------------------------------------');
for i=1:4
[x,y,status]=aasrlatch(s(i),r(i),q);
disp(['     ',int2str(s(i)),'      ',int2str(r(i)),'     ',x,'
',y,'     ',status]);
end
clk=[0;1;1;1;1];
disp('Truth Table for SR Flipflop : ');
disp("     clk    S      R      Q      ~Q     CONDITION ");
disp('---------------------------------------------------------');
 disp('     0      x      x     Q      ~Q       off');
for i=2:5
        s1=(clk(i)&s(i-1));
        s2=(clk(i)&r(i-1));
        [x,y,status]=aasrlatch(s1,s2,q);
        disp(['     ',int2str(clk(i)),'      ',int2str(s(i-1)),'
',int2str(r(i-1)),'     ',x,'     ',y,'     ',status]);

end
disp('Truth Table for D Flipflop : ');
disp("    clk     D     Qn+1   ~Q(n+1)    CONDITION ");
disp('-----------------------------------------');
clk=[0;1;1];
d=[0;1];
disp('     0      x       Q     ~Q         off');
for i=2:3
        [x,y,status]=aasrlatch(d(i-1),~d(i-1),q);
        disp(['     ',int2str(clk(i)),'      ',int2str(d(i-1)),'
',x,'     ',y,'       ',status]);
```

```
end
disp('Truth Table for J-K Flipflop : ');
disp("    clk      Qn      J      K     Qn+1  ~Q(n+1)  CONDITION ");
disp('-------------------------------------------------------');
clk=[0;1;1;1;1;1;1;1;1];
q=[0;0;0;0;1;1;1;1];
j=[0;0;1;1;0;0;1;1];
k=[0;1;0;1;0;1;0;1];
disp('    0       x      x      x      Q      ~Q       off');
for i=2:9
        [x,y,status]=aajkff(q(i-1),j(i-1),k(i-1));
        disp(['    ',int2str(clk(i)),'      ',int2str(q(i-1)),'
',int2str(j(i-1)),'       ',int2str(k(i-1)),'      ',x,'      ',y,'
',status]);

end
```

Function for SR latch (aasrlatch.m)

```
function [Q1,Q2,status] = aasrlatch(S,R,q)
Q1=q;
Q2=~q;
status='Memory';
if(S)
 Q2=~(Q1|S);
 Q1=~(Q2|R);
 status='Reset';
end
if(R)
 Q1=~(Q2|R);
 Q2=~(Q1|S);
 status='Set';
end
 Q1=aastr(Q1);
 Q2=aastr(Q2);
 if(S==1&&R==1)
```

```
 status='Invalid';
 end
end
```

Function for JK Flip Flop (aajkff.m)

```
function [Q1,Q2,status] = aajkff(q,J,K)
a=(J&(~q));
b=(K&q);
[Q1,Q2,status]=aasrlatch(a,b,q);
if(J==1&&K==1)
 status='Toggle';
end
end
```

Function to convert integer to string type for proper display with string type status variable used (aastr.m)

```
function x =aastr(y)
if(y==0)
 x='0 ';
end
if(y==1)
 x='1 ';
end
end
```

Exp6: Write Matlab programs for generation of elementary continuous time signals and discrete time signals.

```
figure(1);
x=linspace(-3,3,200);
z=length(x);
y=NaN(z);
for i=1:z
    if(x(i)<0)
        y(i)=0;
    else
        y(i)=1;
    end
end
plot(x,y,'m');
xlim([-3,3]);
ylim([0,2]);
xlabel('t');
ylabel('U(t)');
title('Continuous Unit Step Function')
figure(2);
x=linspace(-3,3,10);
z=length(x);
y=NaN(z);
for i=1:z
    x(i)=ceil(x(i));
    if(x(i)<0)
        y(i)=0;
    else
        y(i)=1;
    end
end
stem(x,y,'r');
xlim([-3,3]);
ylim([0,2]);
```

```
xlabel('t');
ylabel('U(t)');
title('Discrete Unit Step Function')
figure(3);
x=linspace(-5,5,200);
z=length(x);
y=NaN(z);
for i=1:z
    if(x(i)<0)
        y(i)=0;
    else
        y(i)=x(i);
    end
end
plot(x,y,'m');
xlim([-5,5]);
ylim([0,6]);
xlabel('t');
ylabel('U(t)');
title('Continuous Unit Ramp Function')
figure(4);
x=linspace(-5,5,11);
z=length(x);
y=NaN(z);
for i=1:z
    x(i)=ceil(x(i));
    if(x(i)<0)
        y(i)=0;
    else
        y(i)=x(i);
    end
end
stem(x,y,'r');
xlim([-5,5]);
ylim([0,6]);
xlabel('t');
ylabel('R(t)');
title('Discrete Unit Ramp Function')
figure(5);
x=linspace(-10,10);
y=5*sin(2*pi*0.1*x);
z=zeros(length(x));
plot(x,y,'m',x,z,'k');
xlim([-10,10]);
ylim([-6,6]);
xlabel('t');
ylabel('X(t)');
title('Continuous sinusoidal Function')
figure(6);
x=linspace(-10,10,30);
```

```
for i=1:length(x)
    x(i)=ceil(x(i));
end
y=5*sin(2*pi*0.1*x);
stem(x,y,'r');
xlim([-10,10]);
ylim([-6,6]);
xlabel('t');
ylabel('R(t)');
title('Discrete sinusoidal Function')
figure(7);
x=linspace(-2,3);
y=exp(-2*x);
plot(x,y,'m');
xlim([-2,3]);
ylim([0,8]);
xlabel('t');
ylabel('X(t)');
title('Continuous Exponential Function')
figure(8);
x=linspace(-10,10,20);
for i=1:length(x)
    x(i)=ceil(x(i));
end
y=exp(-0.3*x);
stem(x,y,'r');
xlim([-10,10]);
ylim([0,10]);
xlabel('t');
ylabel('R(t)');
title('Discrete Exponential Function')
figure(9);
x=linspace(-3,3,200);
z=length(x);
y=NaN(z);
a=2;
for i=1:z
    if(x(i)>=-a/2 && x(i)<=a/2)
        y(i)=1/a;
    else
        y(i)=0;
    end
end
plot(x,y,'m');
xlim([-3,3]);
ylim([0,1]);
xlabel('t');
ylabel('Rect(t)');
title('Continuous Rect Function')
figure(10);
```

```
x=linspace(-3,3,10);
z=length(x);
y=NaN(z);
a=2;
for i=1:z
    x(i)=ceil(x(i));
    if(x(i)>=-a/2 && x(i)<=a/2)
        y(i)=1/a;
    else
        y(i)=0;
    end
end
stem(x,y,'r');
xlim([-3,3]);
ylim([0,1]);
xlabel('t');
ylabel('Rect(t)');
title('Discrete Rect Function')
figure(11);
x=linspace(-8,8,200);
y=sinc(2*pi*0.1*x);
z=zeros(length(x));
plot(x,y,'m',x,z,'k');
xlim([-8,8]);
ylim([-0.5,1.5]);
xlabel('t');
ylabel('X(t)');
title('Continuous Sinc Function')
figure(12);
x=linspace(-8,8,30);
for i=1:length(x)
    x(i)=ceil(x(i));
end
y=sinc(2*pi*0.1*x);
stem(x,y,'r');
xlim([-8,8]);
ylim([-0.5,1.5]);
xlabel('t');
ylabel('X(t)');
title('Discrete Sinc Function')
figure(13);
x=linspace(-8,8,200);
y=x.^2;
z=zeros(length(x));
plot(x,y,'m',z,y,'k');
xlim([-8,8]);
ylim([0,40]);
xlabel('t');
ylabel('X(t)');
title('Continuous Parabola Function')
```

```
figure(14);
x=linspace(-8,8,30);
for i=1:length(x)
    x(i)=ceil(x(i));
end
y=x.^2;
stem(x,y,'r');
xlim([-8,8]);
ylim([0,40]);
xlabel('t');
ylabel('X(t)');
title('Discrete Parabola Function')
figure(15);
t=linspace(-8,8,200);
a=3;
b=2;
x=a*cos(2*pi*0.1*t);
y=b*sin(2*pi*0.1*t);
z=zeros(length(x));
plot(x,y,'m',z,y,'k',x,z,'k');
xlim([-3,3]);
ylim([-3,3]);
xlabel('t');
ylabel('X(t)');
title('Continuous ellipse Function')
figure(16);
t=linspace(-8,8,30);
for i=1:length(t)
    t(i)=ceil(t(i));
end
a=3;
b=2;
x=a*cos(2*pi*0.1*t);
y=b*sin(2*pi*0.1*t);
stem(x,y,'r');
xlim([-3,3]);
ylim([-3,3]);
xlabel('t');
ylabel('X(t)');
title('Discrete ellipse Function')
```

Exp7: Write Matlab program to study the sampling and reconstruction process.

```
clc;
f=2000;
T=1/f;
sf=100000;
l=1/sf;
t=-T:l:T;
y=5*sin(2*pi*f*t);
z=zeros(length(t));
figure(1);
plot(t,y,'m',t,z,'k');
ylim([-6,6]);
title('original signal')
xlabel('t (sec)');
ylabel('x(t)');
nlim=T/l;
n =-nlim:1:nlim;
y = 5*sin(2*pi*f*l*n);
figure(2);
stem(n,y,'.','r');
ylim([-6,6]);
title('signal after sampling')
xlabel('n');
ylabel('x[n]');
yr=0;
```

```
for k=0:2*nlim
 val=k:-1:-(length(t)-1)+k;
 yr=yr+y(k+1)*sinc(val);
end
figure(3);
plot(t,yr,t,z,'k');
ylim([-6,6]);
title('Reconstructed signal')
xlabel('t');
```

Exp 8: Write Matlab program to study the Quantization process of Sinusoid Signals.

```
clc;
f=100;
sf = 15000;
T = 1/sf;
t = 0:T:0.02;
y=5*sin(2*pi*f*t);
y1=5*sin(2*pi*f*t);
mn=min(y);
mx=max(y);
l=2^3;
s=(mx-mn)/l;
for j=mn:s:mx
    y(y<=j+s& y>=j)=((2*j)+s)/2;
end
figure(1);
stem(y,'.','r');
title('Sampled Signal')
xlabel('Samples Number')
ylabel('Samples')
figure(2);
e=y1-y;
```

```
plot(e,'m');
xlabel('Samples Number')
ylabel('Quantization Error')
title('Quantization Error')
figure(3);
plot(y,e,'.');
xlabel('Input signal')
ylabel('Quantization error')
title('Quantization error v/s Input Signal')
```

Exp9: Write Matlab programs to study the binary phase shift keying (BPSK) and frequency shift keying modulation (FSK) process.

```
%BPSK signal
clc;
clear all;
T=1;
t=0:0.01:T;
f=2;
c=sqrt(2/T)*sin(2*pi*f*t);
n=15;
a=rand(1,n);
x=0;
y=T;
for i=1:n
t=[x:0.01:y];
if a(i)>0.5
a(i)=1;
b=ones(1,length(t));
```

```
else
a(i)=0;
b=-1*ones(1,length(t));
end
subplot(5,1,1);
stem(a,'*','m');
xlabel('n');
ylabel('b(n)');
title('binary data');
msg(i,:)=b;
mul_sig(i,:)=c.*b;
subplot(5,1,2);
axis([0 n -2 2]);
plot(t,msg(i,:),'r');
xlabel('t');
ylabel('m(t)');
title('message');
hold on;
subplot(5,1,4);
plot(t,mul_sig(i,:));
title('BPSK signal');
xlabel('t');
ylabel('s(t)');
hold on;
x=x+1.01;
y=y+1.01;
end
hold off
subplot(5,1,3);
plot(t,c);
xlabel('t');
ylabel('c(t)');
title('carrier signal');
x=0;
y=T;
for i=1:n
t=[x:0.01:y];
x=sum(c.*mul_sig(i,:));
if x>0
dm(i)=1;
else
dm(i)=0;
end
x=x+1.01;
y=y+1.01;
end
subplot(5,1,5);
stem(dm);
xlabel('n');
ylabel('b(n)');
```

```
title('dmulated data');




%BFSK Signal
clc;
clear all;
T=1;
x=2;
Y=5;
t=0:(T/100):T;
c1=sqrt(2/T)*sin(2*pi*x*t);
c2=sqrt(2/T)*sin(2*pi*Y*t);
n=15;
m=rand(1,n);
a=0;
b=T;
for i=1:n
t=[a:(T/100):b];
if m(i)>0.5
m(i)=1;
m_s=ones(1,length(t));
invm_s=zeros(1,length(t));
else
m(i)=0;
m_s=zeros(1,length(t));
invm_s=ones(1,length(t));
end
subplot(6,1,1);
stem(m,'*','m');
xlabel('n');
ylabel('b(n)');
title('binary data');
msg(i,:)=m_s;
fs1(i,:)=c1.*m_s;
fs2(i,:)=c2.*invm_s;
fsk=fs1+fs2;
subplot(6,1,2);
axis([0 n -2 2]);
plot(t,msg(i,:),'r');
title('message signal');
xlabel('t');
ylabel('m(t)');
hold on;
subplot(6,1,5);
plot(t,fsk(i,:));
title('FSK signal');
xlabel('t');
```

```
ylabel('s(t)');
hold on;
a=a+(T+.01);
b=b+(T+.01);
end
hold off
subplot(6,1,3);
plot(t,c1);
xlabel('t');
ylabel('c1(t)');
title('carrier signal-1');
subplot(6,1,4);
plot(t,c2);
xlabel('t');
ylabel('c2(t)');
title('carrier signal-2');
a=0;
b=T;
for i=1:n
t=[a:(T/100):b];
x=sum(c1.*fs1(i,:));
x2=sum(c2.*fs2(i,:));
x=x-x2;
if x>0
dm(i)=1;
else
dm(i)=0;
end
a=a+(T+.01);
b=b+(T+.01);
end
subplot(6,1,6);
stem(dm,'*','m');
xlabel('n');
ylabel('b(n)');
title(' demodulated data');
```