DIGITAL DESIGN

LAB-4

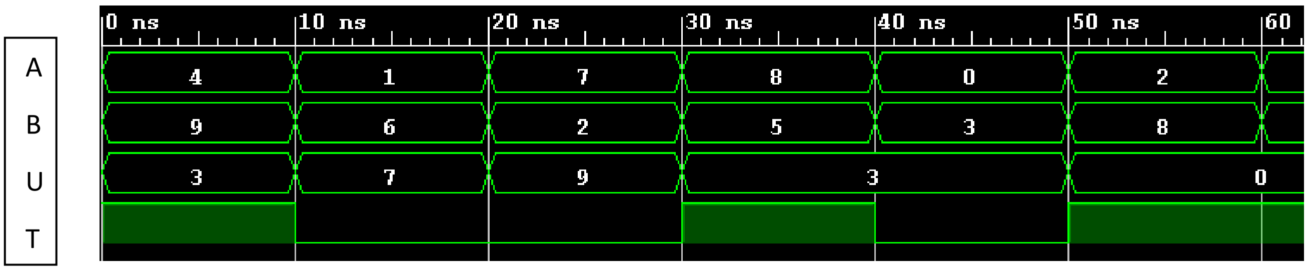
1. BCD Adder:

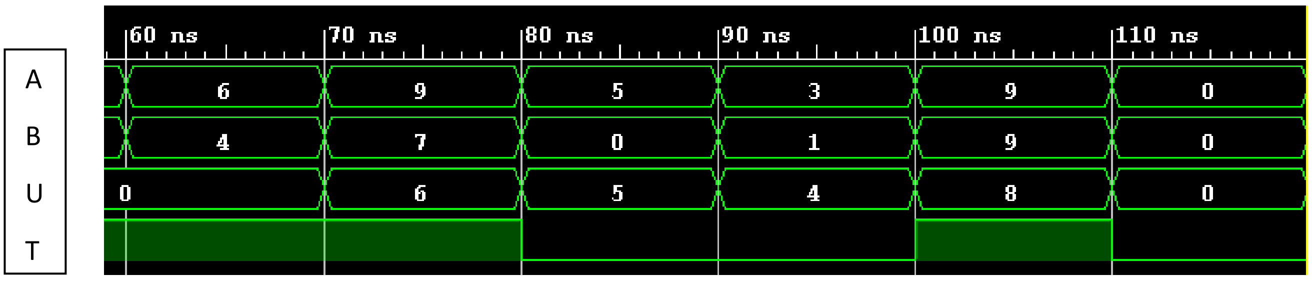
Logic Code:

```
module full_adder(A,B,C,sum,carry);...

module four_bit_adder(A,B,S,Carry);
input [3:0] A,B;
output [3:0] S;
output Carry;
reg C0=1'b0;
wire C1,C2,C3;

full_adder FA1(A[0],B[0],C0,S[0],C1);
full_adder FA2(A[1],B[1],C1,S[1],C2);
full_adder FA3(A[2],B[2],C2,S[2],C3);
full_adder FA4(A[3],B[3],C3,S[3],Carry);
endmodule

module bcd_adder(A,B,S,C);
input [3:0] A,B;
output [3:0] S;
output C;
wire [3:0] Z,X;
wire K;

four_bit_adder FBA1(A,B,Z,K);
assign C=K|(Z[3]&Z[2])|(Z[3]&Z[1]);
assign X[0]=1'b0; assign X[1]=C; assign X[2]=C; assign X[3]=1'b0;
four_bit_adder FBA2(X,Z,S);
endmodule
```

Test Bench:

```
module test_bcd_adder();
reg [3:0] A,B;
wire [3:0] S;
wire C;
bcd_adder BCDA1(A,B,S,C);

initial
begin
    A=4'd4;B=4'd9;
    #10 A=4'd1;B=4'd6;
    #10 A=4'd7;B=4'd2;
    #10 A=4'd8;B=4'd5;
    #10 A=4'd0;B=4'd3;
    #10 A=4'd2;B=4'd8;
    #10 A=4'd6;B=4'd4;
    #10 A=4'd9;B=4'd7;
    #10 A=4'd5;B=4'd0;
    #10 A=4'd3;B=4'd1;
    #10 A=4'd9;B=4'd9;
    #10 A=4'd0;B=4'd0;
end
initial #120 $finish;
endmodule
```





2. BCD Subtractor:

Logic Code:

```
module tens_complement(A,X);
input [3:0] A;
output [3:0] X;

assign X[3]=~(A[3]|A[2]) & ~(A[1]&A[0]);
assign X[2]=A[2]^(A[1]&A[0]);
assign X[1]=~(A[1]^A[0]);
assign X[0]=A[0];
endmodule

module bcd_subtractor(A,B,S);
input [3:0] A,B;
output [3:0] S;
wire [3:0] B_comp,Z,Z_comp;
wire C;

tens_complement TC1(B,B_comp);
bcd_adder BCDA2(A,B_comp,Z,C);
tens_complement TC2(Z,Z_comp);
assign S=C?Z:Z_comp;
endmodule
```

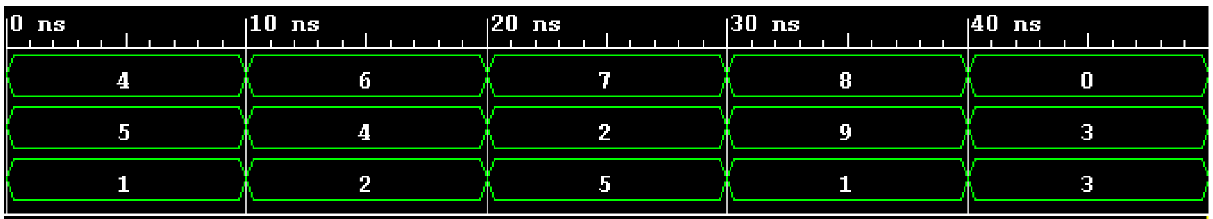
Test Bench:

```
module test_bcd_subtractor();
reg [3:0] A,B;
wire [3:0] S;
bcd_subtractor BCDS1(A,B,S);
initial
begin
    A=4'd4;B=4'd5;
    #10 A=4'd6;B=4'd4;
    #10 A=4'd7;B=4'd2;
    #10 A=4'd8;B=4'd9;
    #10 A=4'd0;B=4'd3;
    #10 A=4'd2;B=4'd6;
    #10 A=4'd1;B=4'd8;
    #10 A=4'd9;B=4'd7;
    #10 A=4'd5;B=4'd0;
    #10 A=4'd3;B=4'd1;
end
initial #100 $finish;
endmodule
```

A

B

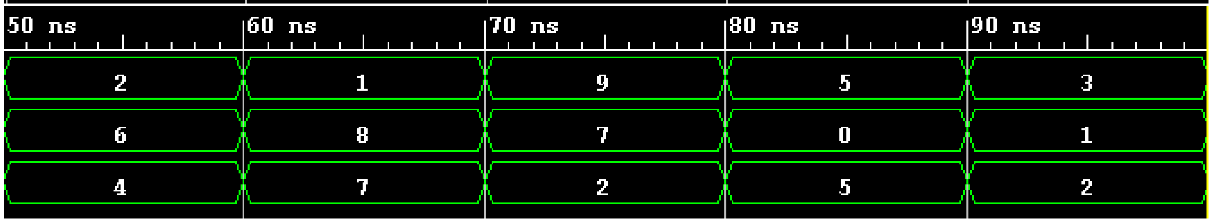
Diff



A

B

Diff



3. Multiplier:

Logic Code:

```
module binary_to_bcd(A,X,Y);
input [5:0] A;
output [3:0] X,Y;
wire [3:0] P,Q,U,V,W;
wire [2:0] M,N;
wire O1,O2,C1,Ca,Sa,L,D1,D2,J;

assign U[3]=A[5]&A[4]; assign U[2]=~A[5]&A[4]; assign U[1]=A[5]^A[4]; assign U[0]=1'b0;
four_bit_adder FBAa(A[3:0],U,P,C1);
assign O1=C1|(P[3] & (P[2]|P[1]));
assign V[3]=1'b0; assign V[2]=O1; assign V[1]=O1; assign V[0]=1'b0;
four_bit_adder FBAb(P,V,Q);
assign O2=Q[3] & (Q[2]|Q[1]);
assign W[3]=1'b0; assign W[2]=O2; assign W[1]=O2; assign W[0]=1'b0;
four_bit_adder FBAc(Q,W,X);

assign N[2]=1'b0;
assign N[0]=O1^O2;
assign N[1]=O1&O2;
assign M[0]=A[5]^A[4];
assign J=A[5]&A[4];
assign M[1]=J^A[5];
assign M[2]=J&A[5];
assign Y[0]=N[0]^M[0];
assign D1=M[0]&N[0];
full_adder FAa(M[1],N[1],D1,Y[1],D2);
full_adder FAb(M[2],N[2],D2,Y[2],Y[3]);
endmodule
```

```verilog
module multiplier(A,B,M,N);
input [2:0] A,B;
output [3:0] M,N;
wire [5:0] P;
wire [2:0] Y,Z;
wire [2:1] X;
wire C1,S1,C2,S2,C3,C4,C5;

assign P[0]=A[0]&B[0];assign X[1]=A[1]&B[0];assign X[2]=A[2]&B[0];
assign Y[0]=A[0]&B[1];assign Y[1]=A[1]&B[1];assign Y[2]=A[2]&B[1];
assign Z[0]=A[0]&B[2];assign Z[1]=A[1]&B[2];assign Z[2]=A[2]&B[2];
full_adder FAa(X[1],Y[0],1'b0,P[1],C1);
full_adder FAb(X[2],Y[1],C1,S1,C2);
full_adder FAc(1'b0,Y[2],C2,S2,C3);
full_adder FAd(S1,Z[0],1'b0,P[2],C4);
full_adder FAe(S2,Z[1],C4,P[3],C5);
full_adder FAf(C3,Z[2],C5,P[4],P[5]);
binary_to_bcd BTB(P,M,N);
endmodule
```

Test Bench:

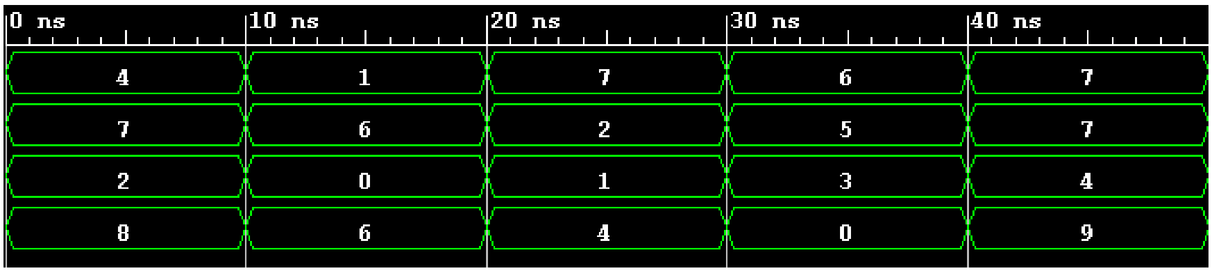
```verilog
module test_multiplier();
reg [2:0] A,B;
wire [3:0] Tens,Ones;
multiplier M1(A,B,Ones,Tens);

initial
begin
    A=3'd4;B=3'd7;
    #10 A=3'd1;B=3'd6;
    #10 A=3'd7;B=3'd2;
    #10 A=3'd6;B=3'd5;
    #10 A=3'd7;B=3'd7;
    #10 A=3'd2;B=3'd5;
    #10 A=3'd6;B=3'd4;
    #10 A=3'd0;B=3'd3;
    #10 A=3'd5;B=3'd5;
    #10 A=3'd3;B=3'd1;
end
initial #100 $finish;
endmodule
```

A

B

Tens

Unit

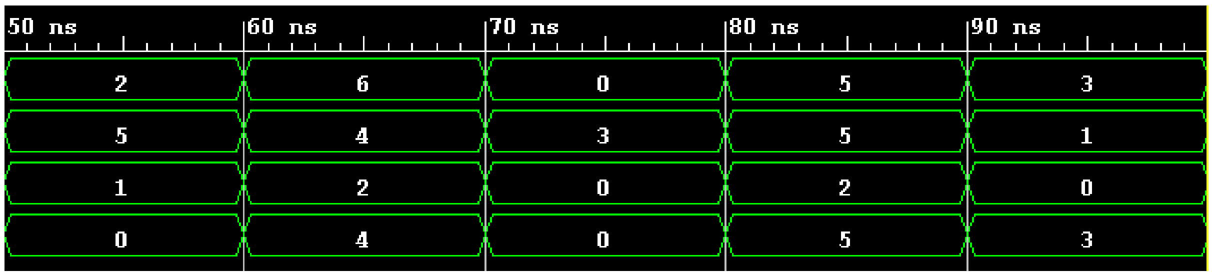


A

B

Tens

Unit


50 ns
60 ns
70 ns
80 ns
90 ns
2 6 0 5 3
5 4 3 5 1
1 2 0 2 0
0 4 0 5 3