4 bit counter

```
`timescale 1ns/1ps
module counter(clk,rst,m,count);
input clk,rst,m;
output reg [3:0]count;
always@(posedge clk or negedge rst)
begin
if(!rst)
count=0;
if(m)
count=count+1;
else
count=count-1;
end
endmodule
```

Test bench

```
`timescale 1ns/1ps
module counter_test;
 reg clk, rst,m;
wire [3:0] count;
Initial
begin
clk=0;
rst=0;#25;
rst=1;
end
initial
begin
m=1;
#600 m=0;
rst=0;#25;
rst=1;
#500 m=0;
end
counter counter1(clk,m,rst, count);
always #5 clk=~clk;
initial
#1400 $finish;
endmodule
```

4bit full adder

Source Code – fa.v :-

```
module full_adder( A,B,CIN,S,COUT);
 input A,B,CIN;
 output S,COUT;
 assign S = A^B^CIN;
 assign COUT = (A&B) | (CIN&(A^B));
endmodule
```

Source Code – fa_4bit.v :-

```
module four_bit_adder(A,B,C0,S,C4);
input [3:0] A,[3:0] B;
input C0;
output [3:0] S;
output C4;
wire C1,C2,C3;
 full_adder fa0 (A[0],B[0],C0,S[0],C1);
 full_adder fa1 (A[1],B[1],C1,S[1],C2);
 full_adder fa2 (A[2],B[2],C2,S[2],C3);
 full_adder fa3 (A[3],B[3],C3,S[3],C4);
endmodule
```

Test Bench – fa_test.v :-

```
module test_4_bit;
 reg [3:0] A;
 reg [3:0] B;
 reg C0;
 wire [3:0] S;
 wire C4;
 four_bit_adder dut(A,B,C0,S,C4);
 initial begin
 A = 4'b0011;B=4'b0011;C0 = 1'b0; #10;
 A = 4'b1011;B=4'b0111;C0 = 1'b1; #10;
 A = 4'b1111;B=4'b1111;C0 = 1'b1; #10;
 end
initial
#50 $finish;
endmodule
```

32 Bit alu

Source Code – Using Case Statement :

```
module alu_32bit_case(y,a,b,f);
input [31:0]a;
input [31:0]b;
input [2:0]f;
output reg [31:0]y;
always@(*)
 begin
 case(f)
 3'b000:y=a&b; //AND Operation
 3'b001:y=a|b; //OR Operation
 3'b010:y=~(a&b); //NAND Operation
 3'b011:y=~(a|b); //NOR Operation
 3'b010:y=a+b; //Addition
 3'b011:y=a-b; //Subtraction
 3'b100:y=a*b; //Multiply
 default:y=32'bx;
endcase
end
endmodule
```

Source Code - Using If Statement :

```
module alu_32bit_if(y,a,b,f);
input [31:0]a;
input [31:0]b;
input [2:0]f;
output reg [31:0]y;
always@(*)
begin
 if(f==3'b000)
 y=a&b; //AND Operation
 else if (f==3'b001)
 y=a|b; //OR Operation
 else if (f==3'b010)
 y=a+b; //Addition
 else if (f==3'b011)
 y=a-b; //Subtraction
 else if (f==3'b100)
 y=a*b; //Multiply
 else
 y=32'bx;
end
endmodule
```

Test bench :

```
module alu_32bit_tb_if;
reg [31:0]a;
reg [31:0]b;
reg [2:0]f;
wire [31:0]y;
alu_32bit_if test(.y(y),.a(a),.b(b),.f(f));
initial
begin
a=32'h00000000;
b=32'hFFFFFFFF;
#10 f=3'b000;
#10 f=3'b001;
#10 f=3'b010;
#10 f=3'b100;
end
initial
#50 $finish;
endmodule
```

```
`timescale 1ns/1ps
module jkff(j,k,clk,q,qm);
input j,k,clk;
output reg q,qm;
initial q=0;
always@(posedge clk)
begin
case({j,k})
2’b00:q=q;
2’b01:q=0;
2’b10:q=1;
2’b11:q=~q;
endacse
qm=~q;
end
endmodule
```

testbench

```
`timescale 1ns/1ps
module jktest;
reg j,k,clk;
wire q,qm;
jkff uut(j,k,clk,q,qm);
initial begin
clk=1;
j=0; k=0; #100;
j=0; k=1; #100;
j=1; k=0; #100;
j=1; k=1; #100;
end
always #5 clk=~clk;
initial
#1400 $finish;
endmodule
```