COUNTER MODULE

```
//`timescale 1ns/1ps
module counter(clk,m,rst,count);
input clk,m,rst;
output reg[15:0]count;
always@(posedge clk or negedge rst)
begin
if(!rst)
        count=0;
else if(m)
        count=count+1;
else
        count=count-1;
end
endmodule
```

TESTBENCH

```
//`timescale 1ns/1ps
module counter_test;
reg clk,m,rst;
wire [3:0]count;
counter g1(clk,m,rst,count);
initial
begin
        clk=0;
        m=0;
        rst=0;
#10     rst=1;
#15     m=1;
#100    m=0;
#100    m=1;
#100    m=0;
end
always #5 clk=!clk;
initial
begin
#1400 $finish;
end
endmodule
```

FULL ADDER LIBRARY

```
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
```

FULL ADDER MODULE

```
module four_bit_adder(A,B,C0,S,C4);
input [3:0]A,B;
input C0;
output [3:0]S,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
```

TESTBENCH

```
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
```

ALU 32-BIT MODULE

```
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;
        3'b001:         y=a|b;
        3'b010:         y=~(a&b);
        3'b011:         y=~(a|b);
        3'b010:         y=a+b;
        3'b011:         y=a-b;
        3'b100:         y=a*b;
        default:y=32'bx;
endcase
end
endmodule
```

TESTBENCH

```
module alu_32bit_tb_case;
reg [31:0]a;
reg [31:0]b;
reg [2:0]f;
wire [31:0]y;
alu_32bit_case test2(.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
```

D FLIP-FLOP MODULE

```
module dff(q,qbar,d,clk);
output q,qbar;
input d,clk;
reg temp;
initial temp<=1'b0;
always @(posedge clk)
begin
	if (d==1'b0)
		temp<=1'b0;
	else
		temp<=1'b1;
end
assign q=temp;
assign qbar=~temp;
endmodule
```

TESTBENCH

```
module dff_test;
wire q,qbar;
reg d,clk;
dff d1(q,qbar,d,clk);
task display;
	begin
		$display("time=%0d",$time,"ns","input=",d,clk,"output=",q,qbar);
	end
endtask
initial
begin
	clk=1'b0;
end
always
#50clk=~clk;
initial
begin
	d=1'b0;#100;display;
	d=1'b1;#100;display;
end
initial
begin
	#500 $finish;
end
endmodule
```

D LATCH MODULE

```
//`timesacle 1ns/1ps
module dlatch(q,qb,d,en,reset);
output reg q;
output qb;
input d,en,reset;
always@(posedge en)
begin
	if(reset==1'b1)
		q<=1'b0;
	else
		q<=d;
end
assign qb=~q;
endmodule
```

TESTBENCH

```
module dlatch_test;
reg en,reset,d;
wire q,qb;
ffd uut(.en(en),.reset(reset),.d(d),.q(q),.qb(qb));
initial
begin
	en=0; reset=1;
end
always
#5 en=~en;
initial
begin
	#10; reset=0;
	#10 d=0;
	#10 d=1;
	#10;
	#10 d=0;
	#10 d=1;
end
initial #80 $finish;
initial $monitor ($time,"%b %b %b %b",en,reset,d,q,qb);
endmodule
```

JK FLIP-FLOP MODULE

```
module jkff(jk,clk,q,qb);
input clk;
input [1:0]jk;
output q,qb;
reg q=0,qb=1;
always@(posedge clk)
begin
        case(jk)
                2'b00:q=q;
                2'b01:q=0;
                2'b10:q=1;
                2'b11:q=~q;
        endcase
        qb=~q;
end
endmodule
```

TESTBENCH

```
module jkff_test;
reg [1:0]jk;
reg clk;
wire q;
wire qb;
jkff uut(.jk(jk),.clk(clk),.q(q),.qb(qb));
initial
begin
        clk=0;
        forever #5 clk=~clk;
end
initial
begin
        #10 jk=00;
        #10 jk=01;
        #10 jk=10;
        #10 jk=11;
end
initial
begin
        #100 $finish;
end
endmodule
```

J K LATCH MODULE

```
module jklatch(jk,en,q,qb);
input en;
input [1:0]jk;
output q,qb;
reg q=0,qb=1;
always@(posedge en)
begin
        case(jk)
                2'b00:q=q;
                2'b01:q=0;
                2'b10:q=1;
                2'b11:q=~q;
        endcase
        qb=~q;
end
endmodule
```

TESTBENCH

```
module jklatch_test;
reg [1:0]jk;
reg en;
wire q;
wire qb;
jkff uut(.jk(jk),.en(en),.q(q),.qb(qb));
initial
begin
        en=0;
        forever #5 en=~en;
end
initial
begin
        #10 jk=00;
        #10 jk=01;
        #10 jk=10;
        #10 jk=11;
end
initial
begin
        #100 $finish;
end
endmodule
```

S R FLIP-FLOP MODULE

```
module srff(q, qb, s, r, clk);
input s, r, clk;
output q, qb;
wire a, b;
nand(a, clk, s);
nand(b, clk, r);
nand(q, qb, a);
nand(qb, q, b);
endmodule
```

TESTBENCH

```
module srff_test;
wire q, qb;
reg s, r, clk;
srff s1(q, qb, s, r, clk);
task display;
        begin
                $display("time=%d", $time, "ns", "input=", s, r, clk, "output=", q, qb);
        end
endtask

initial
        begin
                clk=1'b0;
        end
always
#50 clk=~clk;

initial
        begin
                s = 1'b0; r=1'b1; #100; display;
                s = 1'b0; r=1'b0; #100; display;
                s = 1'b1; r=1'b0; #100; display;
                s = 1'b1; r=1'b1; #100; display;
        end
initial
        begin
                #500 $finish;
        end

endmodule
```

S R LATCH MODULE

```
module srlatch(q, qb, s, r, en);
input s, r, en;
output q, qb;
wire a, b;
nand(a, en, s);
nand(b, en, r);
nand(q, qb, a);
nand(qb, q, b);
endmodule
```

TESTBENCH

```
module srlatch_test;
wire q, qb;
reg s, r, en;
srff s1(q, qb, s, r, en);
task display;
        begin
                $display("time=%d", $time, "ns", "input=", s, r, en, "output=", q, qb);
        end
endtask

initial
        begin
                en=1'b0;
        end
always
#50 en=~en;

initial
        begin
                s = 1'b0; r=1'b1; #100; display;
                s = 1'b0; r=1'b0; #100; display;
                s = 1'b1; r=1'b0; #100; display;
                s = 1'b1; r=1'b1; #100; display;
        end
initial
        begin
                #500 $finish;
        end

endmodule
```