BUFFER

RTL CODE

```
module buffer(a,b);
input a;
output b;
wire ta;
not(ta,a);
not(b,ta);
endmodule
```

TEST BENCH

```
module buffer_test;
reg a;
wire b;
buffer my_buffer(a,b);
initial
begin
a=0;#100;
a=1;#100;
end
endmodule
```

D FLIP FLOP

RTL CODE

```
module dff(d,clk,q,qb);
input d,clk;
output reg q,qb;
always@(posedge(clk))
begin
q=d;
qb=~d;
end
endmodule
```

TEST BENCH

```
module ff_test;
reg d,clk;
wire q,qb;
dff my_ff(d,clk,q,qb);
initial
clk=1'b1;
always
#5 clk=~clk;
initial
begin
d=0;#10;
d=1;#10;
d=0;#10;
d=1;#10;
```

```
end
endmodule
```

INVERTOR

RTL CODE

```
module inv(a,b);
input a;
output b;
assign b=~a;
endmodule
```

TEST BENCH

```
module inv_test;
reg a;
wire b;
inv my_inv(a,b);
initial begin
a=0; #100;
a=1; #100;
end
endmodule
```

JK FLIP FLOP

RTL CODE

```
module jkff(j,k,clk,q,qb);
input j,k,clk;
output reg q,qb;
reg kk=1'b0;
reg[1:0] t;
always@(posedge(clk))
begin
t={j,k};
case(t)
2'b00:kk=kk;
2'b01:kk=1'b0;
2'b10:kk=1'b1;
2'b11:kk=~kk;
default:;
endcase
q=kk;
qb=~q;
end
endmodule
```

TEST BENCH

```
module jkff_test;
reg j,k,clk;
wire q,qb;
```

```
jkff my_ff(j,k,clk,q,qb);
initial clk=1'b0;
always
#5 clk=~clk;
initial
begin
j=0; k=0; #10;
j=1; k=1; #20;
j=0; k=1; #10;
j=1; k=1; #20;
j=1; k=0; #10;
end
endmodule
```

MASTER SLAVE FLIP FLOP

RTL CODE

```
module ms_ff(j,k,clk,q,qb);
input j,k,clk;
output q,qb;
jk_ff u1(j,k,clk,qm,qmb);
jk_ff u2(qm,qmb,~clk,q,qb);
endmodule
```

JK FF RTL CODE

```
module jk_ff(j,k,clk,q,qb);
input j,k,clk;
output reg q,qb;
reg kk=1'b0;
reg[1:0]t;
always@(posedge(clk))
begin
t={j,k};
case(t)
2'b00:kk=kk;
2'b01:kk=1'b0;
2'b10:kk=1'b1;
2'b11:kk=~kk;
default: ;
endcase
q=kk;
qb=~q;
end
endmodule
```

TEST BENCH

```
module ms_test;
reg j,k,clk;
wire q,qb;
ms_ff my_ff1(j,k,clk,q,qb);
```

```
initial
clk=1'b0;
always
#5 clk=~clk;
initial
begin
j=0;k=0;#10;
j=0;k=1;#10;
j=1;k=0;#10;
j=1;k=1;#10;
end
endmodule
```

BASIC GATES

RTL CODE

```
module gates(a,b,y1,y2,y3,y4,y5);
input a,b;
output y1,y2,y3,y4,y5;
assign y1=~a;
assign y2=a&b;
assign y3=a|b;
assign y4=a^b;
assign y5=~(a^b);
endmodule
```

TEST BENCH

```
module basic_test;
reg a,b;
wire y1,y2,y3,y4,y5;
gates my_gates(a,b,y1,y2,y3,y4,y5);
initial
begin
a=0;b=0;
#5 a=0;b=1;
#5 a=1;b=0;
#5 a=1;b=1;
end
endmodule
```

SR FLIP FLOP

RTL CODE

```
module srff(s,r,q,qb);
input s,r;
output reg q,qb;
reg st=1'b0;
reg[1:0]k;
always@(s,r)
begin
k={s,r};
case(k)
```

```
2'b00:st=st;
2'b01:st=1'b0;
2'b10:st=1'b1;
2'b11:st=1'bz;
default:;
endcase
q=st;
qb=~q;
end
endmodule
```

TEST BENCH

```
module sr_test;
reg s,r;
wire q,qb;
srff my_ff(s,r,q,qb);
initial
begin
r=0;s=0;#10;
r=0;s=1;#10;
r=1;s=0;#10;
r=1;s=1;#10;
r=0;s=0;#10;
r=0;s=1;#10;
r=1;s=0;#10;
r=1;s=1;#10;
end
endmodule
```

T FLIP FLOP

RTL CODE

```
module tff(t,clk,q,qb);
input t,clk;
output q,qb;
jkff u1(t,t,clk,q,qb);
endmodule
```

JKFF RTL CODE

```
module jkff(j,k,clk,q,qb);
input j,k,clk;
output reg q,qb;
reg kk=1'b0;
reg[1:0] t;
always@(posedge(clk))
begin
t={j,k};
case(t)
2'b00:kk=kk;
2'b01:kk=1'b0;
2'b10:kk=1'b1;
2'b11:kk=~kk;
```

```
default:;
endcase
q=kk;
qb=~q;
end
endmodule
```

TEST BENCH

```
module ff_test;
reg t,clk;
wire q,qb;
tff my_ff(t,clk,q,qb);
initial
clk=1'b0;
always
#5 clk=~clk;
initial
begin
t=0; #10;
t=1; #10;
t=0; #10;
t=1; #10;
end
endmodule
```

UNIVERSAL GATES

RTL CODE

```
module gate(a,b,y1,y2);
input a,b;
output y1,y2;
assign y1=~(a&b);
assign y2=~(a|b);
endmodule
```

TEST BENCH

```
module gate_test;
reg a,b;
wire y1,y2;
gate my_gate(a,b,y1,y2);
initial
begin
a=0;b=0;
#5 a=0;b=1;
#5 a=1;b=0;
#5 a=1;b=1;
end
endmodule
```