**##1 bit Full Adder**

module FullAdder(a,b,cin,S,Cout) ;

input a,b,cin ;

output S,Cout ;

assign S = a^b^cin ;

assign Cout = (a&b) | (b&cin) | (cin&a) ;

endmodule

**##4 bit Full Adder**

module Adder\_4bit(A,B,Cin,sum,cout) ;

input [3:0]A,B ;

input Cin ;

output [3:0]sum ;

output cout ;

wire[2:0]C ;

FullAdder fa1 (A[0],B[0],Cin,sum[0],C[0]) ;

FullAdder fa2 (A[1],B[1],C[0],sum[1],C[1]) ;

FullAdder fa3 (A[2],B[2],C[1],sum[2],C[2]) ;

FullAdder fa4 (A[3],B[3],C[2],sum[3],cout) ;

endmodule

**##8x1 MUX**

module MUX\_8x1(W,S,F) ;

input [2:0]S ;

input [7:0]W ;

output F ;

reg F ;

always @ (S or W)

begin

case(S)

0: F=W[0] ;

1: F=W[1] ;

2: F=W[2] ;

3: F=W[3] ;

4: F=W[4] ;

5: F=W[5] ;

6: F=W[6] ;

7: F=W[7] ;

endcase

end

endmodule

**##1x8 DeMUX**

module DeMux\_ifelse(w,s,f) ;

input [2:0]s ;

input f ;

output [7:0]w ;

reg [7:0]w;

always @ (f or s)

begin

if (f==0)

w = 8'b 00000000;

else if (s==0)

w = 8'b 00000001;

else if (s==1)

w = 8'b 00000010;

else if (s==2)

w = 8'b 00000100;

else if (s==3)

w = 8'b 00001000;

else if (s==4)

w = 8'b 00010000;

else if (s==5)

w = 8'b 00100000;

else if (s==6)

w = 8'b 01000000;

else if (s==7)

w = 8'b 10000000;

end

endmodule

**## 1101**

module one\_one\_zero\_one(clk,reset,w,z);

input clk,reset,w;

output z;

reg [2:0]state;

reg z;

parameter A=3'b000,B=3'b001,C=3'b010,D=3'b011,E=3'b100;

always @ (posedge clk,posedge reset)

begin

if(reset==1)

state<=A;

else

begin

case(state)

A:

begin

if(w==1) state <= B;

else state <= A;

end

B:

begin

if(w==1) state <= C;

else state <= A;

end

C:

begin

if(w==1) state <= C;

else state <= D;

end

D:

begin

if(w==1) state <= E;

else state <= A;

end

E:

begin

if(w==1) state <= C;

else state <= A;

end

endcase

end

end

always @ (posedge clk,posedge reset)

begin

if(reset)

z<=0;

else if(state==E)

z<=1;

else z<=0;

end

endmodule

**##Full Adder FSM**

module Full\_adder(clk,reset,a,b,s,c);

input clk,reset,a,b;

output s,c;

reg [1:0] state;

reg s,c;

parameter A = 2'b00,B = 2'b01,C = 2'b10,D = 2'b11;

always @ (posedge clk,posedge reset)

begin

if (reset==1)

state<=A;

else

begin

case (state)

A:

begin

if(a==0 && b==0) state<=A;

else if (a==0 && b==1) state<=B;

else if (a==1 && b==0) state<=B;

else state<=C;

end

B:

begin

if(a==0 && b==0) state<=A;

else if (a==0 && b==1) state<=B;

else if (a==1 && b==0) state<=B;

else state<=C;

end

C:

begin

if(a==0 && b==0) state<=B;

else if (a==0 && b==1) state<=C;

else if (a==1 && b==0) state<=C;

else state<=D;

end

D:

begin

if(a==0 && b==0) state<=B;

else if (a==0 && b==1) state<=C;

else if (a==1 && b==0) state<=C;

else state<=D;

end

endcase

end

end

always @ (posedge clk,posedge reset)

begin

if (reset)

begin

s<=0;

c<=0;

end

else if (state==A)

begin

s<=0;

c<=0;

end

else if (state==B)

begin

s<=1;

c<=0;

end

else if (state==C)

begin

s<=0;

c<=1;

end

else if (state==D)

begin

s<=1;

c<=1;

end

end

endmodule

**##Vending Machine**

module vending\_machine(clk,reset,d,n,z);

input clk,reset,d,n;

output z;

reg [2:0] state;

reg z;

parameter A = 3'b000,B = 3'b001,C = 3'b010,D = 3'b011,E = 3'b100;

always @ (posedge clk,posedge reset)

begin

if (reset==1)

state<=A;

else

begin

case (state)

A:

begin

if(d==0 && n==0) state<=A;

else if (d==0 && n==1) state<=B;

else if (d==1 && n==0) state<=C;

end

B:

begin

if(d==0 && n==0) state<=B;

else if (d==0 && n==1) state<=C;

else if (d==1 && n==0) state<=D;

end

C:

begin

if(d==0 && n==0) state<=C;

else if (d==0 && n==1) state<=D;

else if (d==1 && n==0) state<=E;

end

D:

begin

if(d==0 && n==0) state<=A;

else if (d==0 && n==1) state<=B;

else if (d==1 && n==0) state<=C;

end

E:

begin

if(d==0 && n==0) state<=B;

else if (d==0 && n==1) state<=C;

else if (d==1 && n==0) state<=D;

end

endcase

end

end

always @ (posedge clk,posedge reset)

begin

if (reset)

z<=0;

else if (state==D || state==E)

z<=1;

else z<=0;

end

endmodule