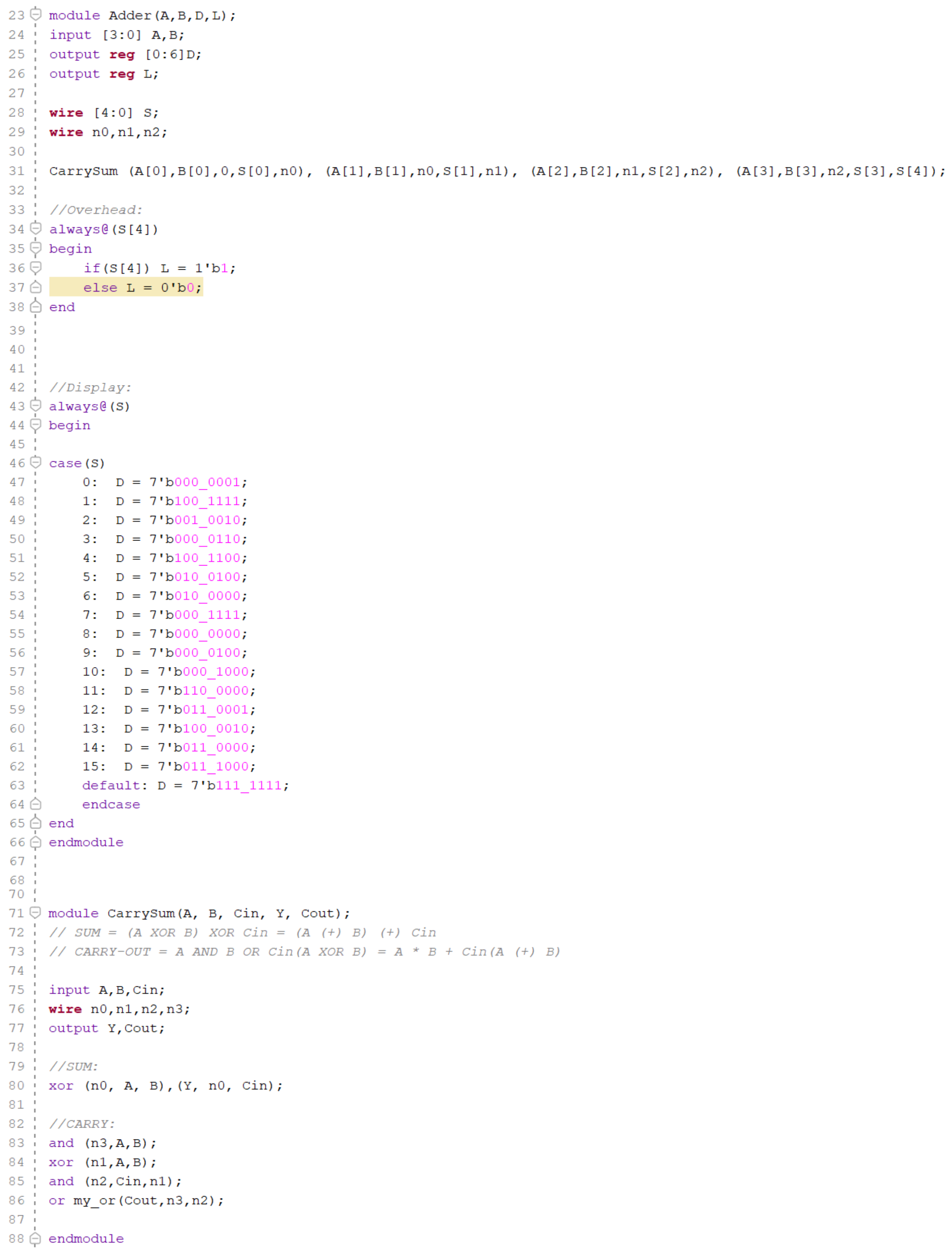
4-bit Display FullAdder



```
module Adder(A,B,D,L);
input [3:0] A,B;
output reg [0:6]D;
output reg L;

wire [4:0] S;
wire n0,n1,n2;

CarrySum (A[0],B[0],0,S[0],n0), (A[1],B[1],n0,S[1],n1), (A[2],B[2],n1,S[2],n2), (A[3],B[3],n2,S[3],S[4]);

//Overhead:
always@(S[4])
begin
    if(S[4]) L = 1'b1;
    else L = 0'b0;
end



//Display:
always@(S)
begin

case(S)
    0:  D = 7'b000_0001;
    1:  D = 7'b100_1111;
    2:  D = 7'b001_0010;
    3:  D = 7'b000_0110;
    4:  D = 7'b100_1100;
    5:  D = 7'b010_0100;
    6:  D = 7'b010_0000;
    7:  D = 7'b000_1111;
    8:  D = 7'b000_0000;
    9:  D = 7'b000_0100;
    10:  D = 7'b000_1000;
    11:  D = 7'b110_0000;
    12:  D = 7'b011_0001;
    13:  D = 7'b100_0010;
    14:  D = 7'b011_0000;
    15:  D = 7'b011_1000;
    default: D = 7'b111_1111;
    endcase
end
endmodule



module CarrySum(A, B, Cin, Y, Cout);
// SUM = (A XOR B) XOR Cin = (A (+) B) (+) Cin
// CARRY-OUT = A AND B OR Cin(A XOR B) = A * B + Cin(A (+) B)

input A,B,Cin;
wire n0,n1,n2,n3;
output Y,Cout;

//SUM:
xor (n0, A, B),(Y, n0, Cin);

//CARRY:
and (n3,A,B);
xor (n1,A,B);
and (n2,Cin,n1);
or my_or(Cout,n3,n2);

endmodule
```