```
module fullAdder(A, B, cin, S, cout);
        input A, B, cin;
        output S, cout;
        assign cout = (A & B) | ((A ^ B) & cin);
        assign S = A ^ B ^ cin;
endmodule

module fourBitAdder(A, B, cin, S, cout);
        input [3:0] A;
        input [3:0] B;
        input cin;

        output [3:0] S;
        output cout;

        // wire [3:0] ci;
        wire c1;
        wire c2;
        wire c3;

        fullAdder fa1(
                .A(A[0]),
                .B(B[0]),
                .S(S[0]),
                .cin(cin),
                .cout(c1)
                //.cout(ci[0])
        );
        fullAdder fa2(
                .A(A[1]),
                .B(B[1]),
                .S(S[1]),
                .cin(c1),
                //.cin(ci[0]),
                .cout(c2)
                //.cout(ci[1])
        );
        fullAdder fa3(
                .A(A[2]),
                .B(B[2]),
                .S(S[2]),
                //.cin(ci[2]),
                .cin(c2),
                .cout(c3)
                //.cout(ci[2])
        );
        fullAdder fa4(
                .A(A[3]),
                .B(B[3]),
                .S(S[3]),
```

```
                .cin(c3),
                //.cin(ci[2]),
                .cout(cout)
                //.cout(cout)
        );
endmodule

// main call to Adder
module ripple(SW, LEDR);
        input [9:0] SW;
        output [9:0] LEDR;

        fourBitAdder fa(
                .A(SW[3:0]),
                .B(SW[7:4]),
                .S(LEDR[3:0]),
                .cin(SW[8]),
                .cout(LEDR[8])
        );
endmodule
```