## Verilog Code

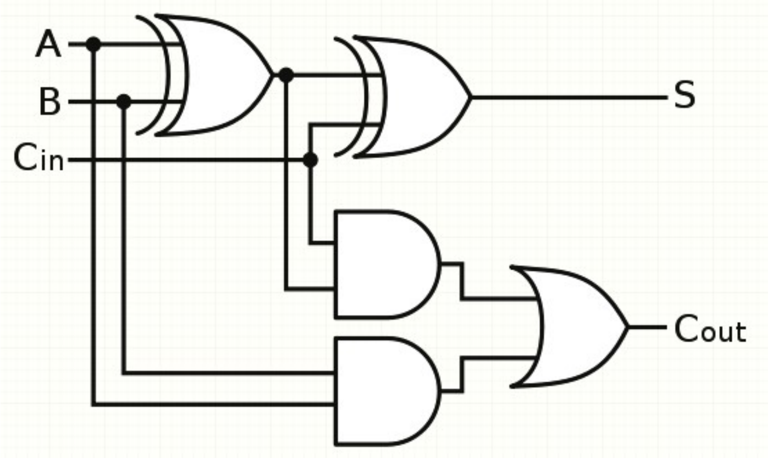
```
module full_adder(
   input A, B, Cin,
   output Sum, Cout
);
   assign Sum  = A ^ B ^ Cin;       // XOR operation
   assign Cout = (A & B) | ((A ^ B) & Cin);
endmodule
```

## Test Bench Code

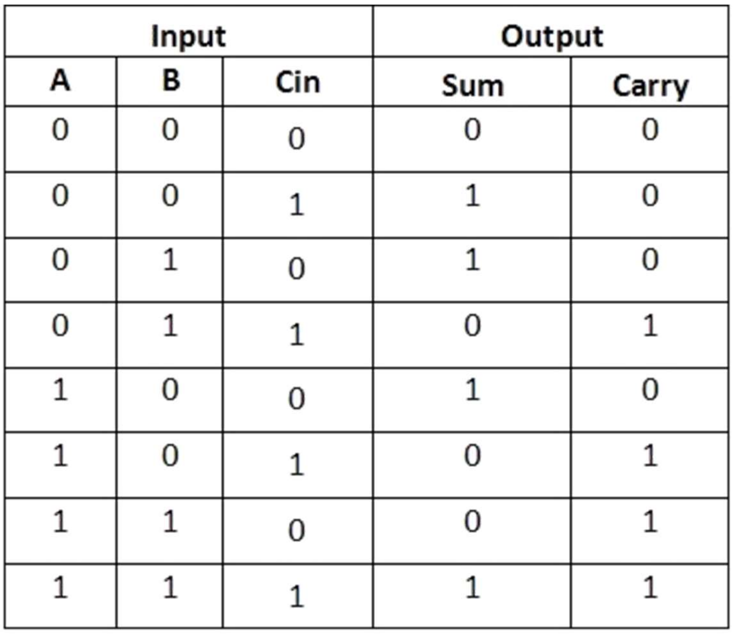
```
module tb_adder;
   reg A, B, Cin;
   wire Sum, Cout;
   // Instantiate Full Adder
   full_adder dut (
      .A(A),
      .B(B),
      .Cin(Cin),
      .Sum(Sum),
      .Cout(Cout)
   );
   initial begin
      A=0; B=0; Cin=0; #10;
      $display("A=%b B=%b Cin=%b -> Sum=%b Cout=%b", A,B,Cin,Sum,Cout);
      A=0; B=0; Cin=1; #10;
      $display("A=%b B=%b Cin=%b -> Sum=%b Cout=%b", A,B,Cin,Sum,Cout);
      A=0; B=1; Cin=0; #10;
```

```
    $display("A=%b B=%b Cin=%b -> Sum=%b Cout=%b", A,B,Cin,Sum,Cout);

    A=1; B=1; Cin=1; #10;

    $display("A=%b B=%b Cin=%b -> Sum=%b Cout=%b", A,B,Cin,Sum,Cout);

    $finish;

  end

endmodule
```

## Circuit

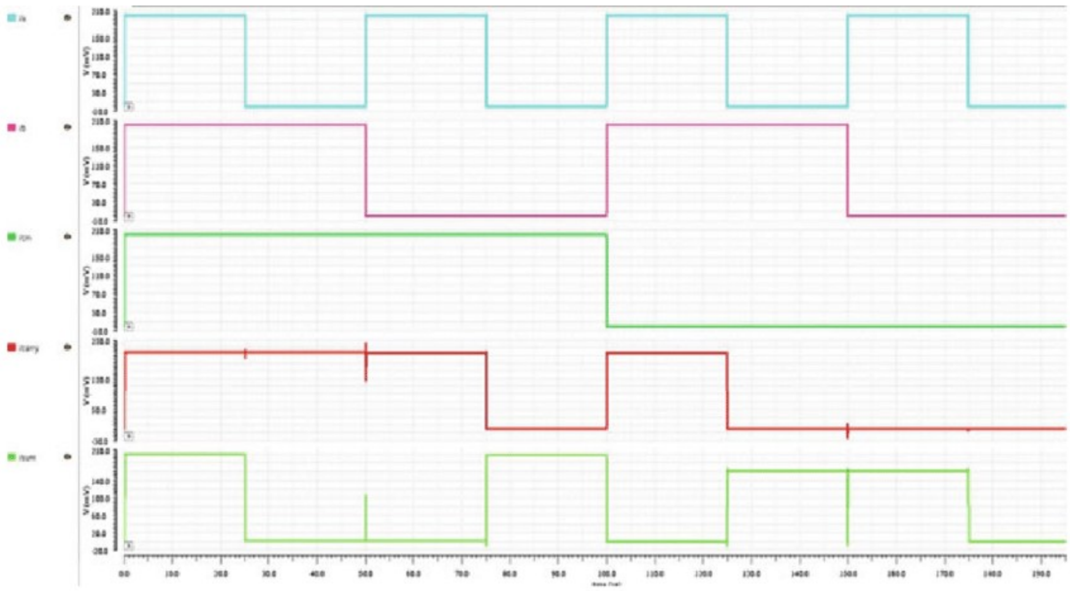


### Truth Table



| Input | | | Output | |
|---|---|---|---|---|
| A | B | Cin | Sum | Carry |
| 0 | 0 | 0 | 0 | 0 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 0 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 0 | 1 | 0 | 1 |
| 1 | 1 | 0 | 0 | 1 |
| 1 | 1 | 1 | 1 | 1 |

## Output Wave Forms


/a
/b
/cin
/carry
/sum
V (mV)