30 Days of RTL Coding

-By T Tushar Shenoy

Day 7

Problem Statement: Implementing a 2-Bit Binary Multiplier using Structural Style of Implementation and to display the Test Case Status.

Theory:

A binary multiplier definition is; an electronic device or digital device or a combinational logic circuit that performs the multiplication of two binary numbers (0 and 1). The two binary numbers or the two binary inputs used in the binary multiplication are multiplicand and multiplier to get the binary product as a result.

The bit size of the multiplier and the multiplicand can be varied. But the bit size of the binary product depends on the multiplier and the multiplicand bit size. The sum of the multiplier and the multiplicand bit size is equal to the final binary product's bit size.

Function of Binary Multiplier

In the first step of the process of the binary multiplier, the partial product terms are obtained by the bit by bit multiplication, which is equal to the ANDing of two binary numbers. In the next step, all the partial product terms of each column are added together to get the final binary product output.

The logic circuit design of this multiplier varies with bit size and its complexity increases with an increase in the multiplier's bit size. It is governed by the AND gate functions when the two bits, which are to be multiplied, are fed as inputs with various bit sizes.

The main function of this multiplier is to do binary multiplication of 2 binary numbers with various bit sizes and reduce the calculation time in electronic digital systems such as computers. The binary multiplication is similar to the decimal multiplication.

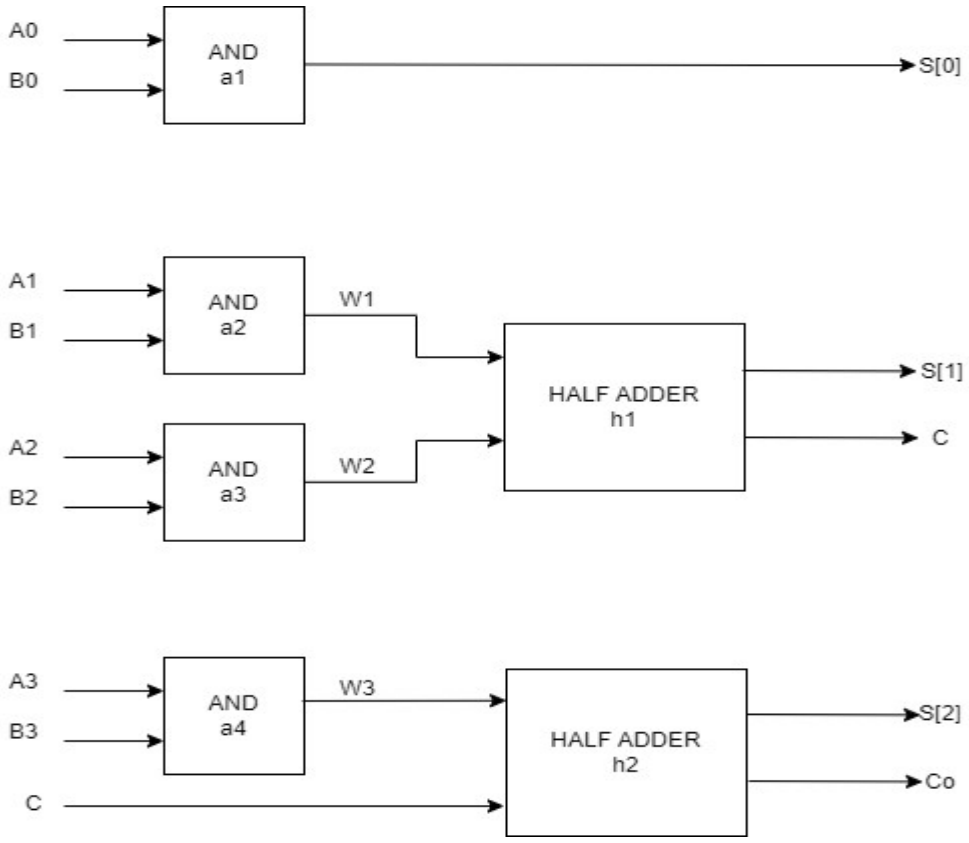


FIG: 2-Bit Multiplier

Applications

The binary multiplier applications are listed below:

- Used in computers.
- Used in mobiles.
- Used in high-speed calculators.
- Used in digital signal processors.
- Used in controlling devices.
- Used in digital communication systems.

Verilog Code:

```
//Verilog Code for AND Gate
module andgate(a,b,o);
input a,b;
output o;
assign o=a&b;
endmodule

//Verilog Code for Half Adder
module halfadder(a,b,s,c);
input a,b;
output s,c;
assign s=a^b;
assign c=a&b;
endmodule
```

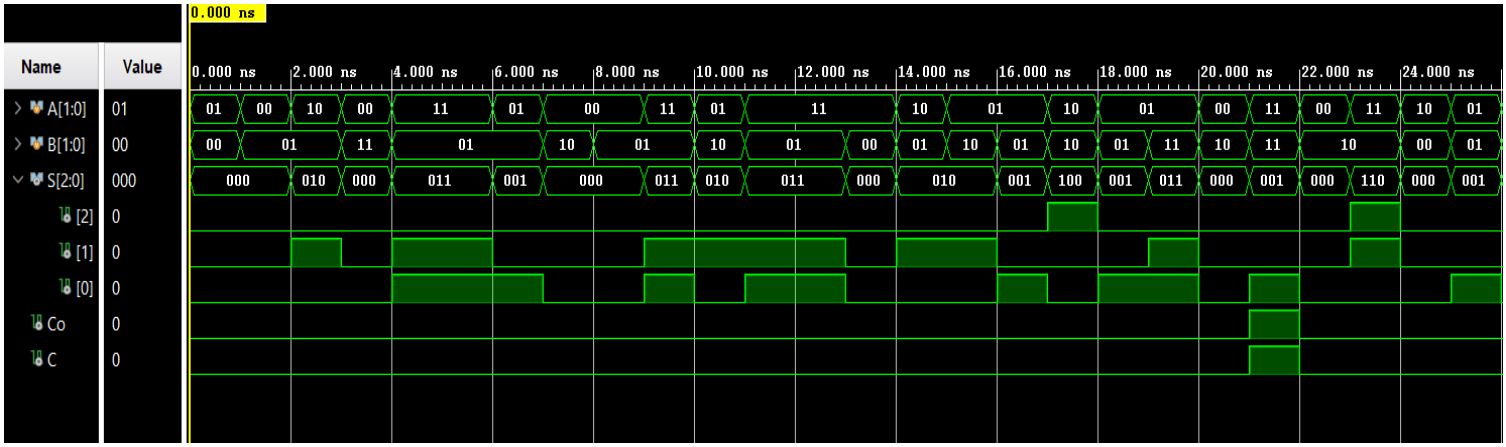
```
//Verilog Code for Two BIT Multiplier
module two_bit_multiplier(A,B,S,Co,C);
input [1:0]A,B;
output [2:0]S;
output Co,C;

//For AoBo Output
andgate a1(A[0],B[0],S[0]); /*we can also use built in AND gate Primitives*/

//For A1Bo+AoB1 Output
wire w1,w2,w3,w4;
andgate a2(A[1],B[0],w1);  /*we can also use built in AND gate Primitives*/
andgate a3(A[0],B[1],w2); /*we can also use built in AND gate Primitives*/
halfadder h1(w1,w2,S[1],C);

//For A1B1 Output
andgate a4(A[1],B[1],w3);  /*we can also use built in AND gate Primitives*/
halfadder h2(w3,C,S[2],Co);
endmodule
```

Testbench Code:

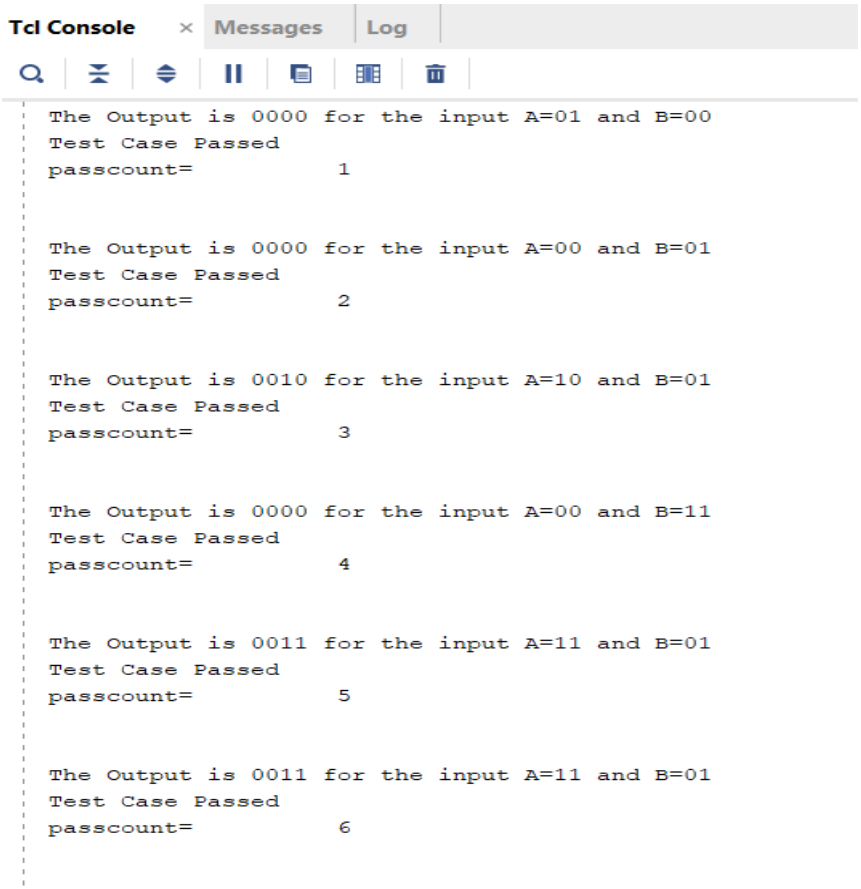
```
//Verilog Testbench Code for Two BIT Multiplier
module two_bit_multiplier_tb();

reg [1:0]A,B;
wire [2:0]S;
wire Co,C;
reg o1,o2,o3,o4;
real passcount=0,failcount=0,n=1;
real pass_percent=0;

two_bit_multiplier dut(.A(A),.B(B),.S(S),.Co(Co),.C(C));

initial begin
assign o1=A[0]&B[0];
assign o2=(A[1]&B[0])^(A[0]&B[1]);
assign o3=(A[1]&B[1])^C;
assign o4=Co;
end

initial begin
repeat(100)
begin
{A,B}=$random;
#1;
$display("The Output is %b%b%b%b for the input A=%b%b and
B=%b%b",Co,S[2],S[1],S[0],A[1],A[0],B[1],B[0]);
```

```
    if(o1==S[0]&&o2==S[1]&&o3==S[2]&&o4==Co)
    begin
     $display("Test Case Passed");
     passcount=passcount+1;
     $display("passcount=%d\n\n",passcount);
     end
 else
    begin
     $display("Test Case Failed\nThe Failed Inputs are A=%b%b and B=%b%b and the corresponding Outputs for the failed Combinations are %b%b%b%b\n",A[1],A[0],B[1],B[0],Co,S[2],S[1],S[0]);
     failcount=failcount+1;
     $display("failcount=%d\n\n",failcount);
    end
 end
$finish;
pass_percent=passcount*100/(passcount+failcount);
$display("The Pass Percentage is %0.4f percent",pass_percent);

end
endmodule
```

Simulation Output:



Console Output:



Note: If we give some delay in AND or Half adder Code we may see that few Test cases will fail.

Schematics:

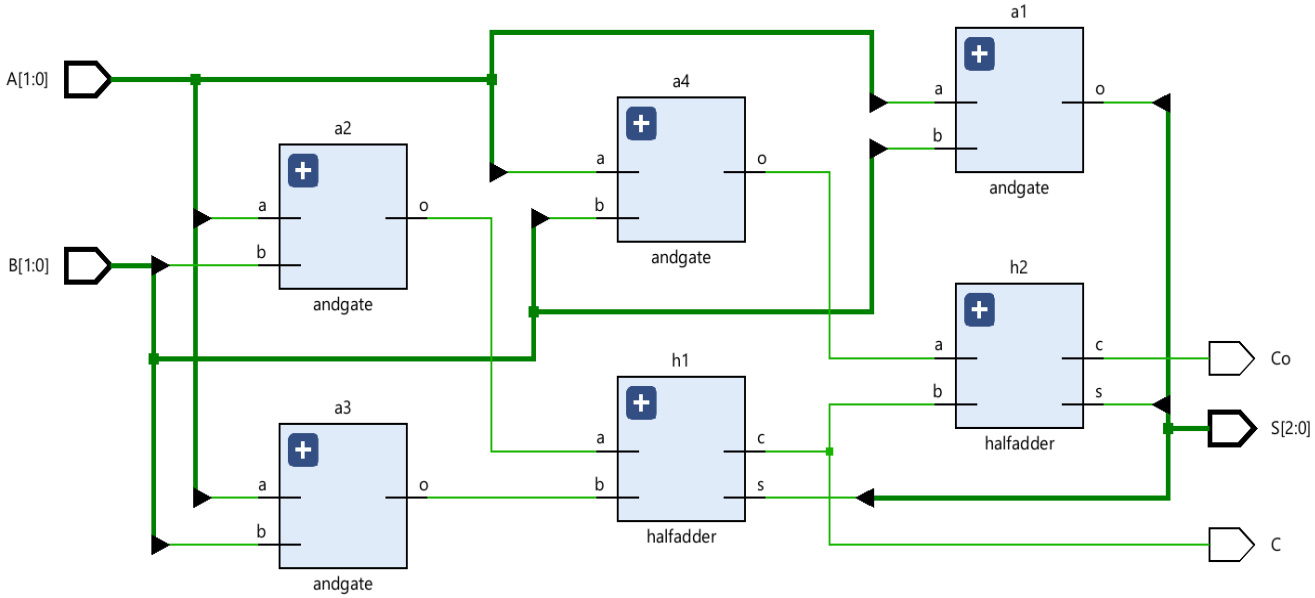


FIG: Schematic of 2 BIT Multiplier (i)

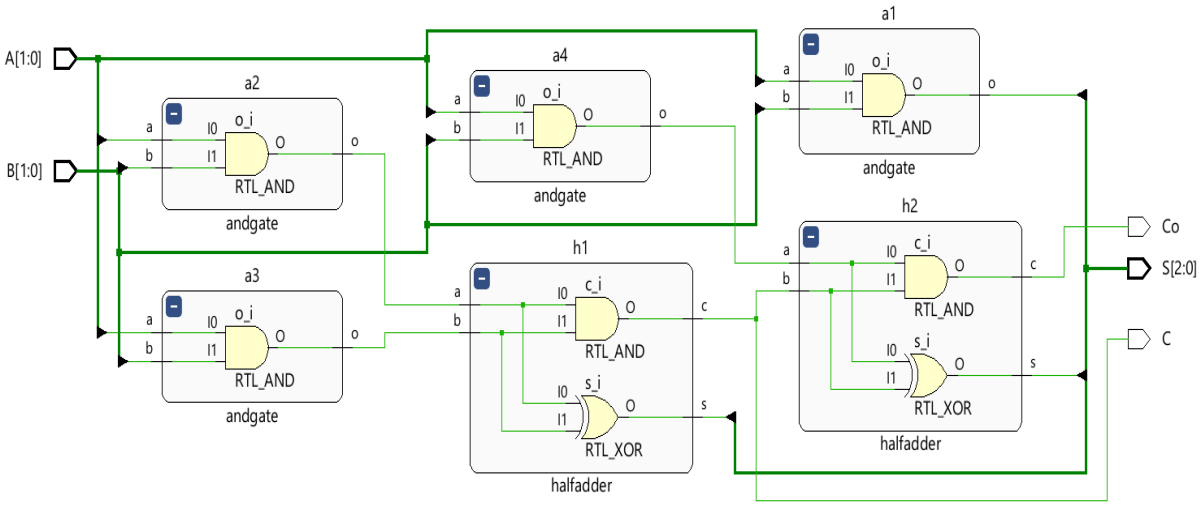


FIG: Schematic of 2 BIT Multiplier (ii)

GitHub Repository URL: - https://github.com/tusharshenoy/RTL-Day-7-2-Bit-Binary-Multiplier