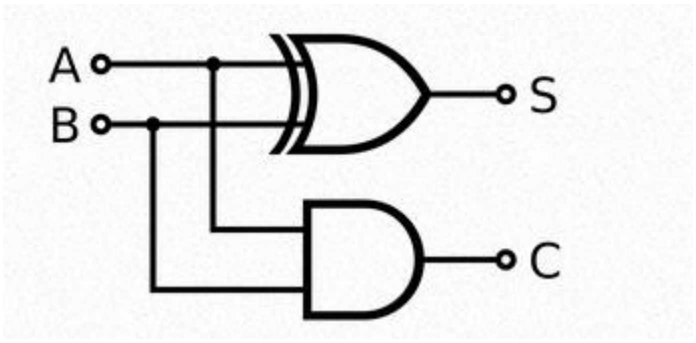
ECE 526L Lab2

Part-A

In this portion of the Lab, first design a half adder and named half_adder.v
For a half adder you need to implement on gate level by using one xor and one and gate.



```
module half_adder(
  input a,    // Input 'a'
  input b,    // Input 'b'
  output s,   // Output 's' (Sum)
  output c    // Output 'c' (Carry)
);
.
.
.

endmodule
```

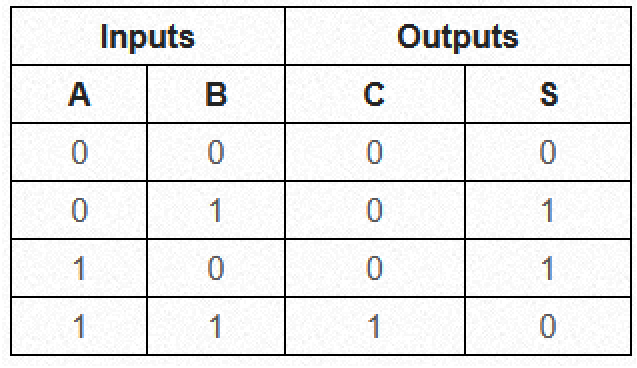
//TB

Then write test bench that covers all inputs values and name the module as half_adder_tb.v

Then use the command

```
vcs -debug_access+all half_adder.v half_adder_tb.v
```

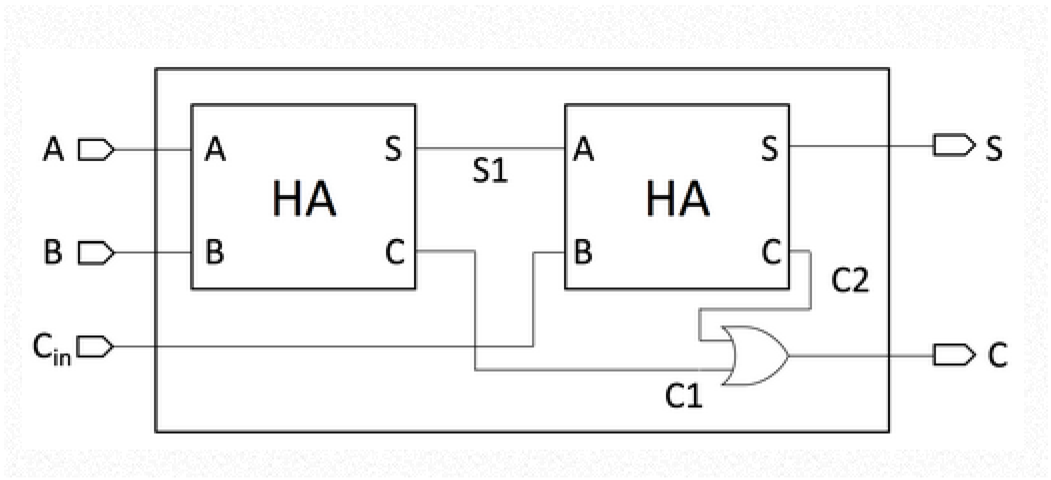
Simulate the code and cover all cases to test all possible combinations of and print out all test cases on consol.



Inputs		Outputs	
A	B	C	S
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	0

Part-B

In this part we will use the half adder from Part-A to make a full adder.(by instantiation of half_adder) and name it full_adder.v



To make a full adder we will use 2 half adders and a OR primitive

Then write test bench that covers all inputs values and name the module as full_adder_tb.v

Then use the command

```
vcs -debug_access+all  half_adder.v full_adder.v full_adder_tb.v
```

Simulate the code and cover all cases to test all possible combinations of and print out all test cases on consol.

a	b	Cin	Sum	cout	
0	0	0	0	0	
0	0	1	0	1	
0	1	0	0	1	
0	1	1	1	0	
1	0	0	0	1	
1	0	1	1	0	
1	1	0	1	0	
1	1	1	1	1	

```
module full_adder(
    A,  //input A
    B,  //input B
    C,  //input C
    Sum,
    Carry_out
    );
    input A;
    input B;
     input C;
    output Sum;
     output Carry_out;

-----------------------------TB------------

module tb_full_adder;

    // Inputs
    reg Data_in_A;
    reg Data_in_B;
    reg Data_in_C;

    // Outputs
    wire Data_out_Sum;
    wire Data_out_Carry;
    // Instantiate the Unit Under Test (UUT)
    full_adder uut (


end module;
```