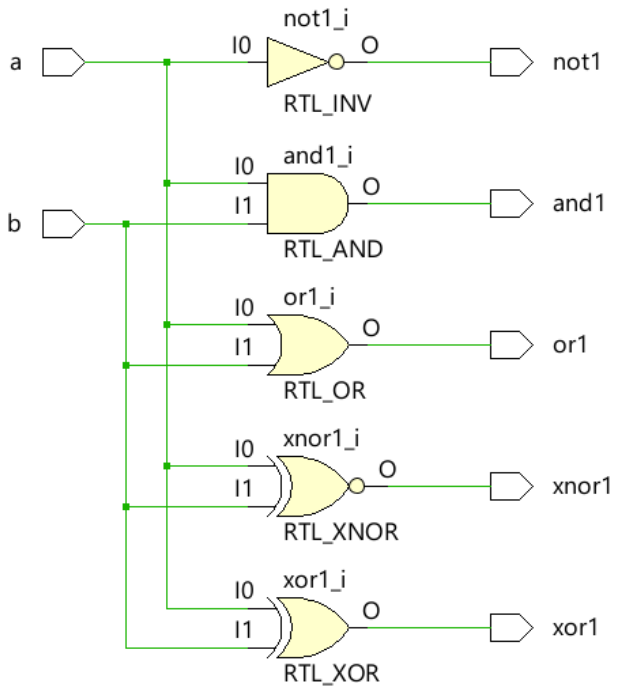
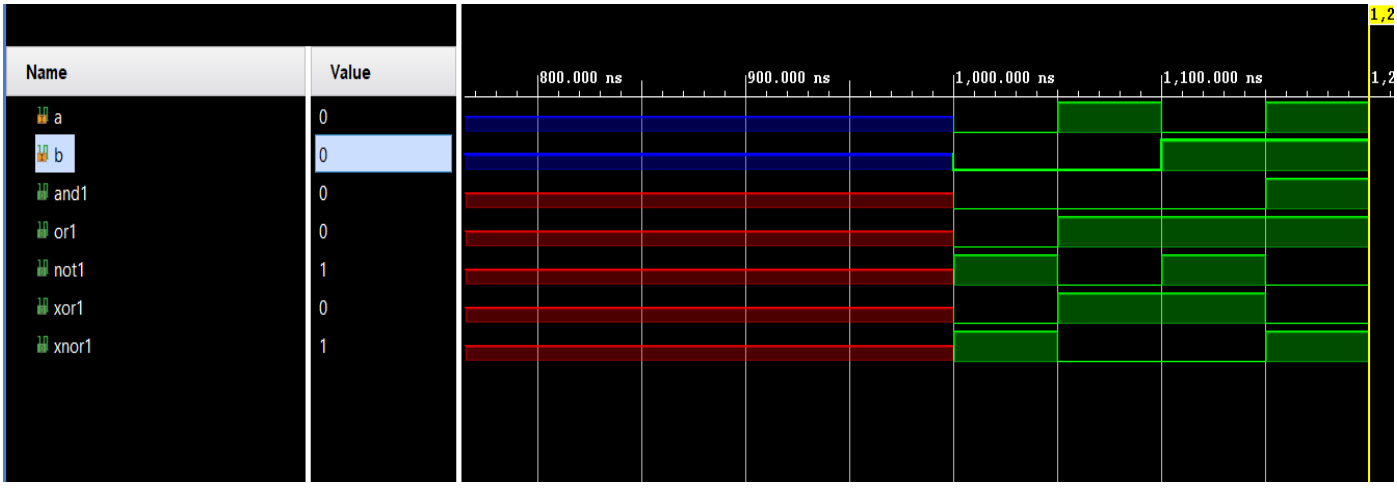
# LOGIC GATES:

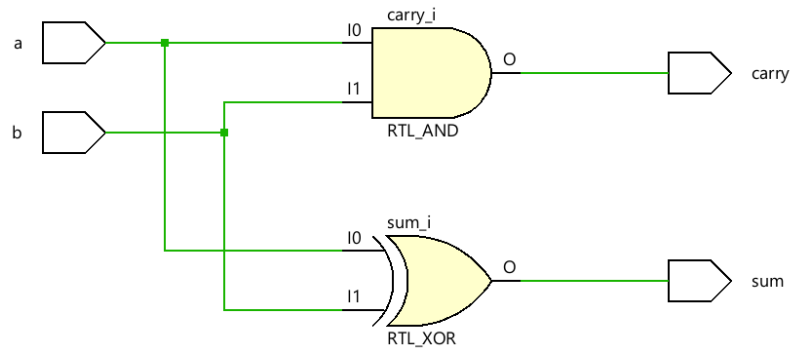
```
21
22
23      module LOGIC_GATES(
24          input a,
25          input b,
26          output and1,
27          output or1,
28          output not1,
29          output xor1,
30          output xnor1
31          );
32          //dataflow or data modelling
33          assign and1=a&b;
34          assign or1=a|b;
35          assign not1=~a;
36          assign xor1=a^b;
37          assign xnor1=a~^b;
38      endmodule
39
```

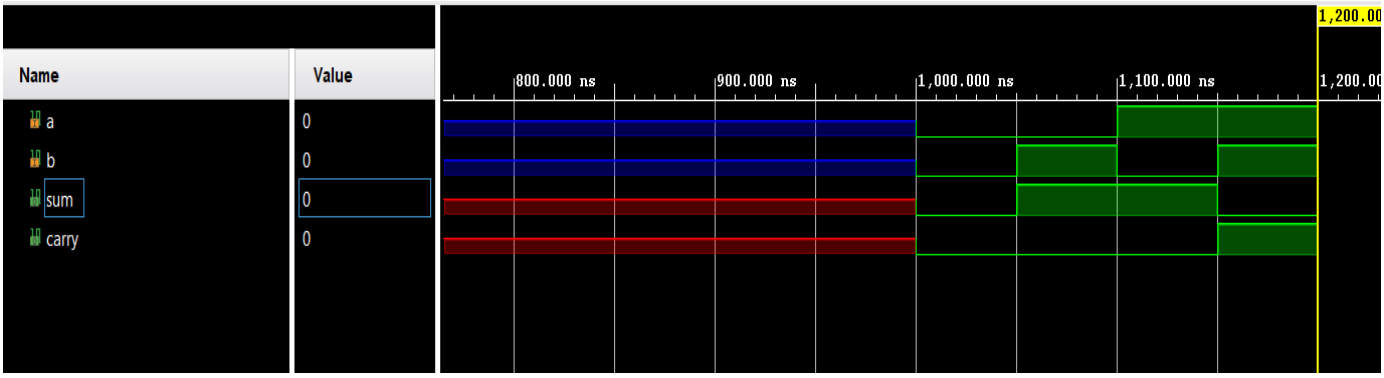


# HALF ADDER:

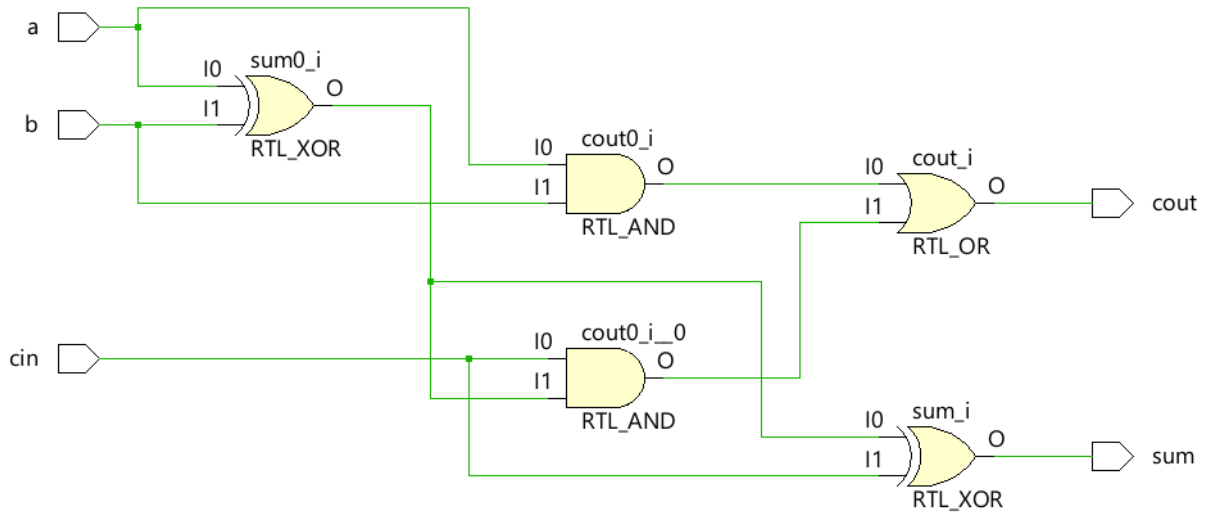
```
22
23  module HALF_ADDER(
24       input a,
25       input b,
26       output sum,
27       output carry
28       );
29       assign sum=a^b;
30       assign carry=a & b;
31  endmodule
32
```

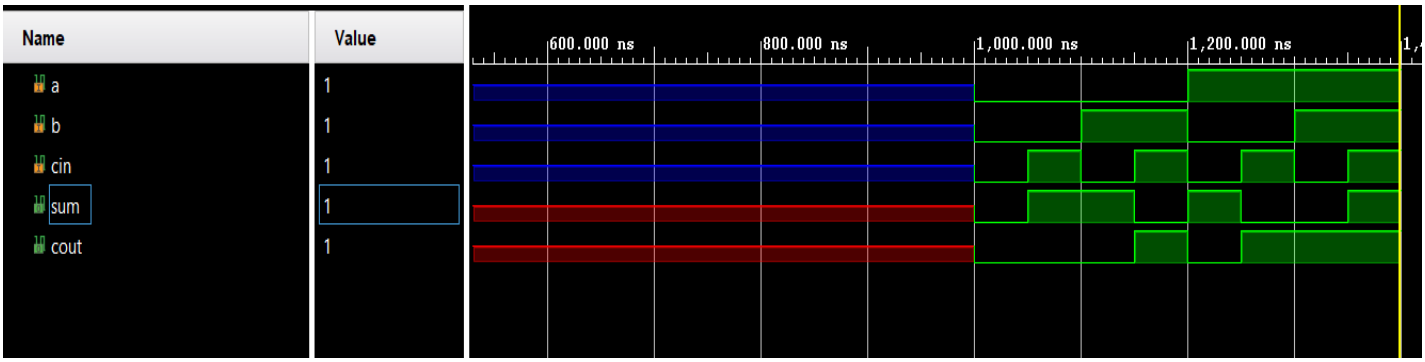


# FULL ADDER:

```
22
23  module FULL_ADDER(
24       input a,
25       input b,
26       input cin,
27       output sum,
28       output cout
29       );
30       assign sum=a^b^cin;
31       assign cout=(a&b)|(cin&(a^b));
32  endmodule
33
```



# 4*1 MULTIPLEXER:

```
22
23  module MUX_4_1(
24      input d0,
25      input d1,
26      input d2,
27      input d3,
28      input [1:0] s,
29      output y
30      );
31      reg y;
32      always@(s)
33      begin
34      case(s)
35      2'b00 : y=d0;
36      2'b01 : y=d1;
37      2'b10 : y=d2;
38      2'b11 : y=d3;
39      endcase
40      end
41  endmodule
42
```