## 📘 Day 1: Logic Gates in Verilog Using Cadence (Xcelium)

---

### 🔹 1. Concept

Logic gates are the **building blocks** of all digital circuits.

- **AND** → Output = 1 only if both inputs are 1
- **OR** → Output = 1 if any input is 1
- **NOT** → Output = Inverts input
- **XOR** → Output = 1 if inputs differ
- **NAND / NOR** → Universal gates (can build any circuit using only these)

### 🔹 2. Design Code

l_g.v

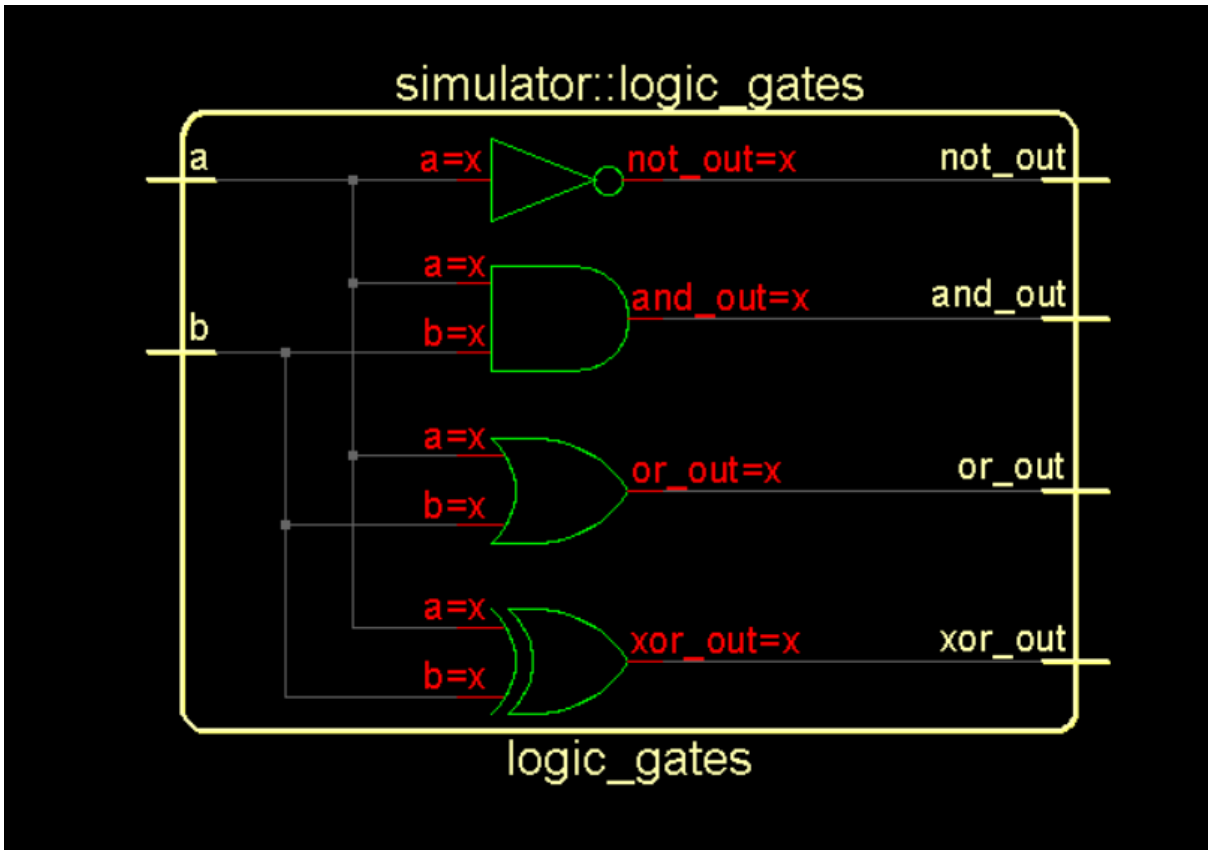
```
module logic_gates(
    input  a, b,
    output and_out, or_out, not_out, xor_out
);
    assign and_out = a & b;    // AND
    assign or_out  = a | b;    // OR
    assign not_out = ~a;       // NOT (only one input needed)
    assign xor_out = a ^ b;    // XOR
endmodule
```

### 🔹 3. Test Bench Code

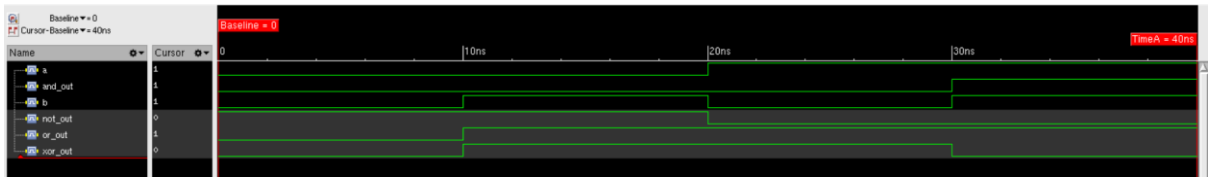
*l_g_tb.v

```
module tb_logic_gates;
    reg a, b;
    wire and_out, or_out, not_out, xor_out;
    logic_gates dut (.a(a), .b(b), .and_out(and_out),
                     .or_out(or_out), .not_out(not_out), .xor_out(xor_out));
    initial begin
        $monitor("a=%b b=%b | AND=%b OR=%b NOT(a)=%b XOR=%b",
                 a, b, and_out, or_out, not_out, xor_out);
        a=0; b=0; #10;
        a=0; b=1; #10;
        a=1; b=0; #10;
        a=1; b=1; #10;
 $finish;
    end
endmodule
```

## 🔹 4. Schematic




## 🔹 5. Wave Forms




**Day 1 Summary**

- Learned basic logic gates in Verilog
- Used assign statements for clean, optimized code
- Built testbench with all input combinations