```
NOT gate using Structural modeling
module not_gate_s(a,y);
input a;
output y;
not(y,a);
endmodule

NOT gate using data flow modeling
module not_gate_d(a,y);
input a;
output y;
assign y = ~a;
endmodule

NOT gate using behavioural modeling
module not_gate_b(a,y);
input a;
output reg y;
always @ (a)
y = ~a;
endmodule
```

AND Gate

AND Gate Truth Table

Input a	Input b	Output y
0	0	0
0	1	0
1	0	0
1	1	1

AND Gate Verilog Code

```
//AND gate using Structural modeling
module and_gate_s(a,b,y);
input a,b;
output y;
and(y,a,b);
endmodule

//AND gate using data flow modeling
module and_gate_d(a,b,y);
input a,b;
output y;
assign y = a & b;
endmodule

//AND gate using behavioural modeling
module nAND_gate_b(a,b,y);
input a;
output y;
always @ (a,b)
y = a & b;
endmodule
```

OR Gate

OR Gate Truth Table

Input a	**Input b**	**Output y**
0	0	0
0	1	1
1	0	1
1	1	1

OR Gate Verilog Code

```
//OR gate using Structural modeling
module or_gate_s(a,b,y);
input a,b;
output y;
or(y,a,b);
endmodule

//OR gate using data flow modeling
module or_gate_d(a,b,y);
input a,b;
output y;
assign y = a | b;
endmodule

//OR gate using behavioural modeling
module or_gate_b(a,b,y);
input a;
output y;
always @ (a,b)
y = a | b;
endmodule
```