





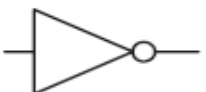
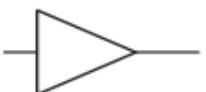
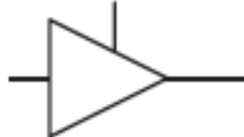
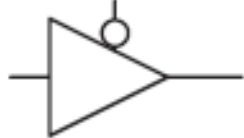
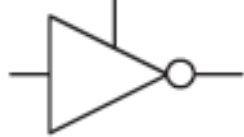
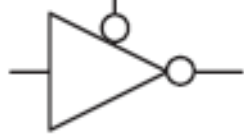


Verilog (contd..)

Verilog Gate Primitives

- Verilog provides primitive gates. One can instantiate these gates and use these gates to construct a logic circuit. This is also called Gate-level Structural Modelling.

<code>and (w, i₁, i₂ ...)</code>	
<code>nand (w, i₁, i₂ ...)</code>	
<code>or (w, i₁, i₂ ...)</code>	
<code>nor (w, i₁, i₂ ...)</code>	
<code>xor (w, i₁, i₂ ...)</code>	
<code>xnor (w, i₁, i₂ ...)</code>	
<code>not (w, i)</code>	
<code>buf (w, i)</code>	

<code>bufif1 (w, i, c)</code>	
<code>bufif0 (w, i, c)</code>	
<code>notif1 (w, i, c)</code>	
<code>notif0 (w, i, c)</code>	

Gates categorized as *n_input* gates are **and**, **nand**, **or**, **nor**, **xor**, and **xnor**. An *n_input* gate has one output, which is its left-most argument, and can have any number of inputs that may be listed as its argument separated by commas. These gates can have up to two delay parameters that can appear after the name of the gate in a set of parenthesis followed by a sharp sign. An example instantiation of a 4-input **nand** is shown here.

```
nand #(3, 5) gate1 (w, i1, i2, i3, i4);
```

In this example, t_{PLH} (low to high propagation) and t_{PHL} (high to low propagation) times are 3 and 5, respectively. Gate delays are optional, and if not included, 0 delay values are assumed

HA.v - D:/codes

```
1 module HA(a,b,s,c);  
2   input a,b;  
3   output s,c;  
4   xor g1(s,a,b);  
5   and g2(c,a,b);  
6 endmodule
```

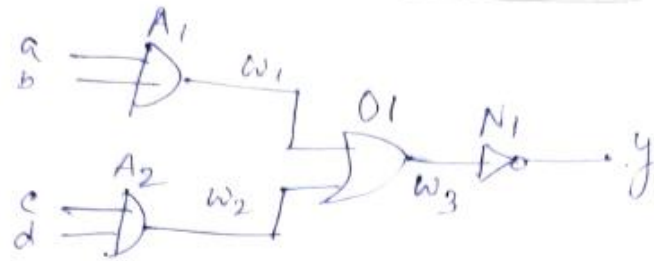
FA.v - D:/codes

```
1 module FA(a,b,cin,s,cout);  
2   input a,b,cin;  
3   output s,cout;  
4   wire s1,c1,c2;  
5   // instantiate HAs  
6   HA m1(a,b,s1,c1,);  
7   HA m2(s1,cin,s,c2,);  
8   or o1(c,c1,c2);  
9 endmodule
```

4bit_RCA.v - D:/codes

```
1 module 4bit_RCA(x,y,s,co);  
2   input [3:0] x,y; // Two 4-bit inputs  
3   output [3:0] s;  
4   output co;  
5   wire w1,w2,w3;  
6   // instantiating 4 1-bit full adders in Verilog  
7   FA u1(x[0],y[0],1'b0,s[0],w1);  
8   FA u2(x[1],y[1],w1,s[1],w2);  
9   FA u3(x[2],y[2],w2,s[2],w3);  
10  FA u4(x[3],y[3],w3,s[3],co);  
11 endmodule
```

Verilog code for AOI4 circuit



```
module logic-ckt (a,b,c,d,y);  
    input a,b,c,d;  
    output y;  
  
    wire w1,w2,w3;  
    and A1(w1,a,b);  
    and A2(w2,c,d);  
    or O1(w3,w1,w2);  
    not N1(y,w3);  
endmodule
```

module logic-ckt
(input a,b,c,d,
output y);