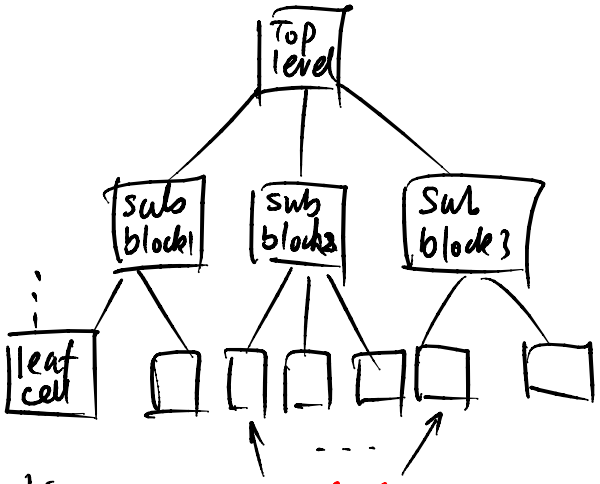
Brief intro to Verilog

⇒ Hardware descriptive language

Hierarchical Modeling Concepts



Top-down design

Bottom-up design

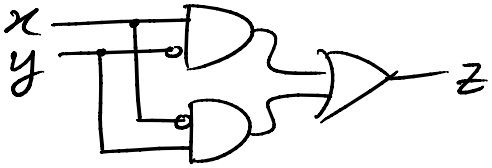
in Verilog: **module** is the basic building block. it can be an element or a collection of lower-level design blocks

4 levels of abstractions can be used to describe a **module**

→ behavior (algorithmic)
→ dataflow (data flow between logic elements) } RTL register transfer level
→ gate (logic gates)
→ switch (transistors)

no nesting modules!

Recall a logic circuit




```
module zlogic (input x, y, output z);
  assign z = (x & !y) | (!x & y);
endmodule
```

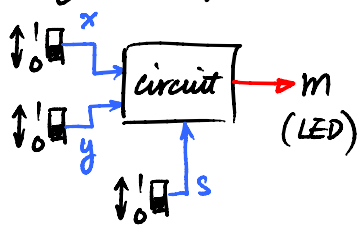
Verilog keyword (lower case only!)

declaration

AND NOT OR

2.8 Multiplexer

Design example : design a circuit that controls an LED (m) from either one of the two switches x and y. The switch selection is controlled by s.



s	
0	m(LED) controlled by SW x
1	m(LED) " " y

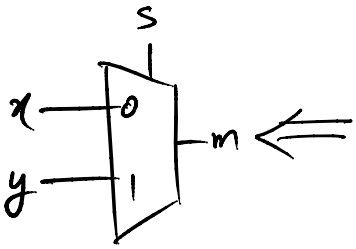
s	m
0	x
1	y

s	x	y	m
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

SOP form

$m = \bar{s}x\bar{y} + \bar{s}xy + s\bar{x}y + sxy$

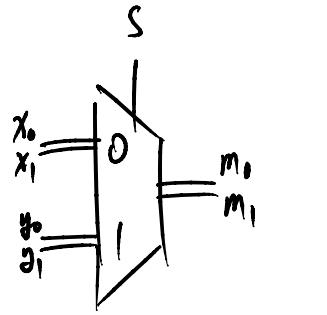
$= \bar{s}x + sy$



2-to-1 mux ⟸ 2-to-1 multiplexer

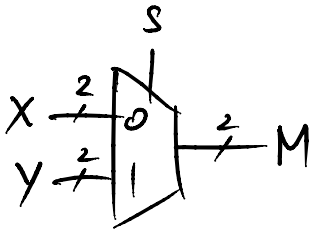
```
module mux2to1(x, y, s, m);
  input x, y, s;
  output m;
  assign m = (~s & x) | (s & y);
endmodule
```

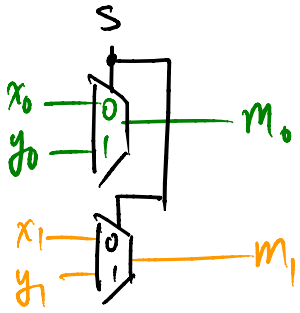
(| : OR, ~ : NOT)



2 bit 2-to-1 mux

s	$m_0 m_1$
0	$x_0 x_1$
1	$y_0 y_1$






```
module mux2bit_2to1(X, Y, s, M);
  input [1:0] X, Y;
  input s;
  output [1:0] M;
  assign M[0] = (~s & X[0]) | (s & Y[0]);
  assign M[1] = (~s & X[1]) | (s & Y[1]);
endmodule
```

→ Alternative Verilog code

Hierarchical design (i.e. use two instances of mux2to1)

```
module mux2bit_2to1(x, y, s, M);
  input [1:0] x, y ;
  input s;
  output [1:0] M;
  mux2to1 u1 (x[0], y[0], s, M[0]);
  mux2to1 u2 (x[1], y[1], s, M[1]);
endmodule
```

name given to identify one copy of mux2to1