Tassadaq Hussain

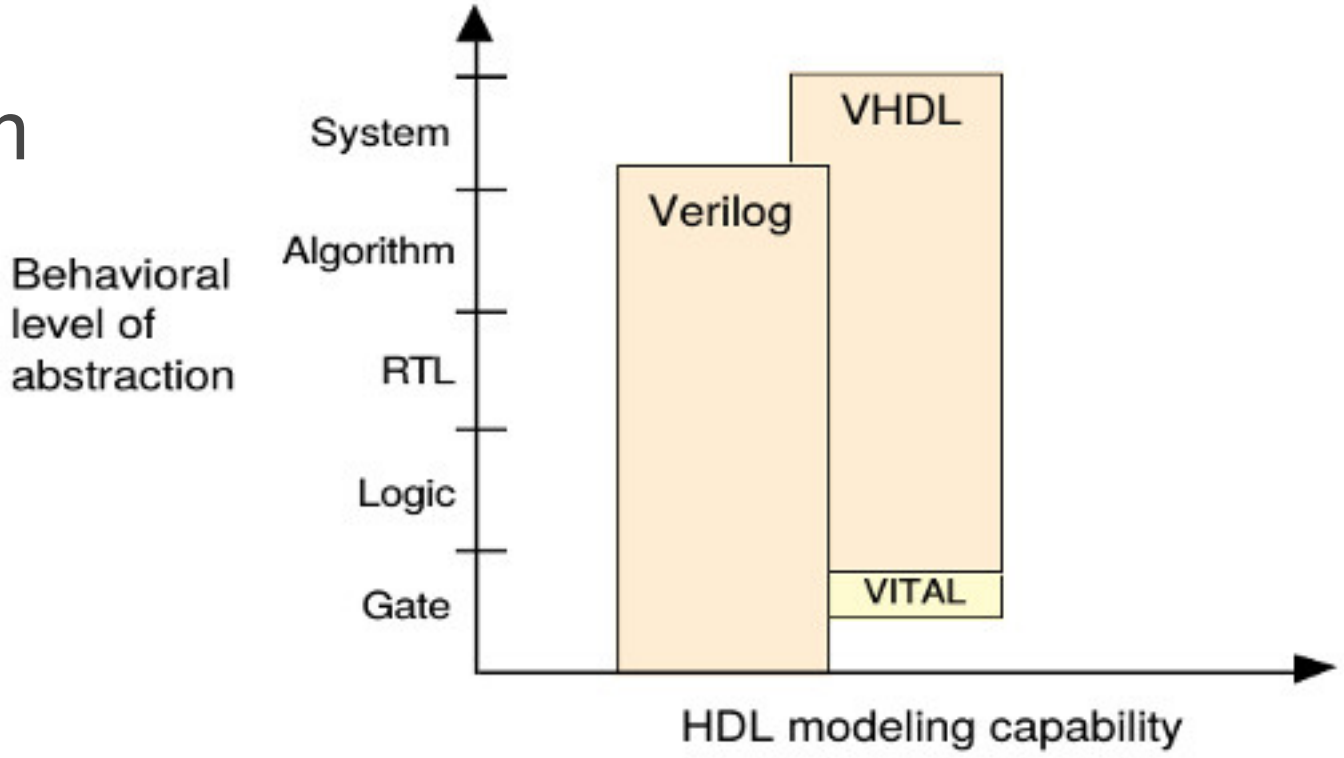
VLSI Design

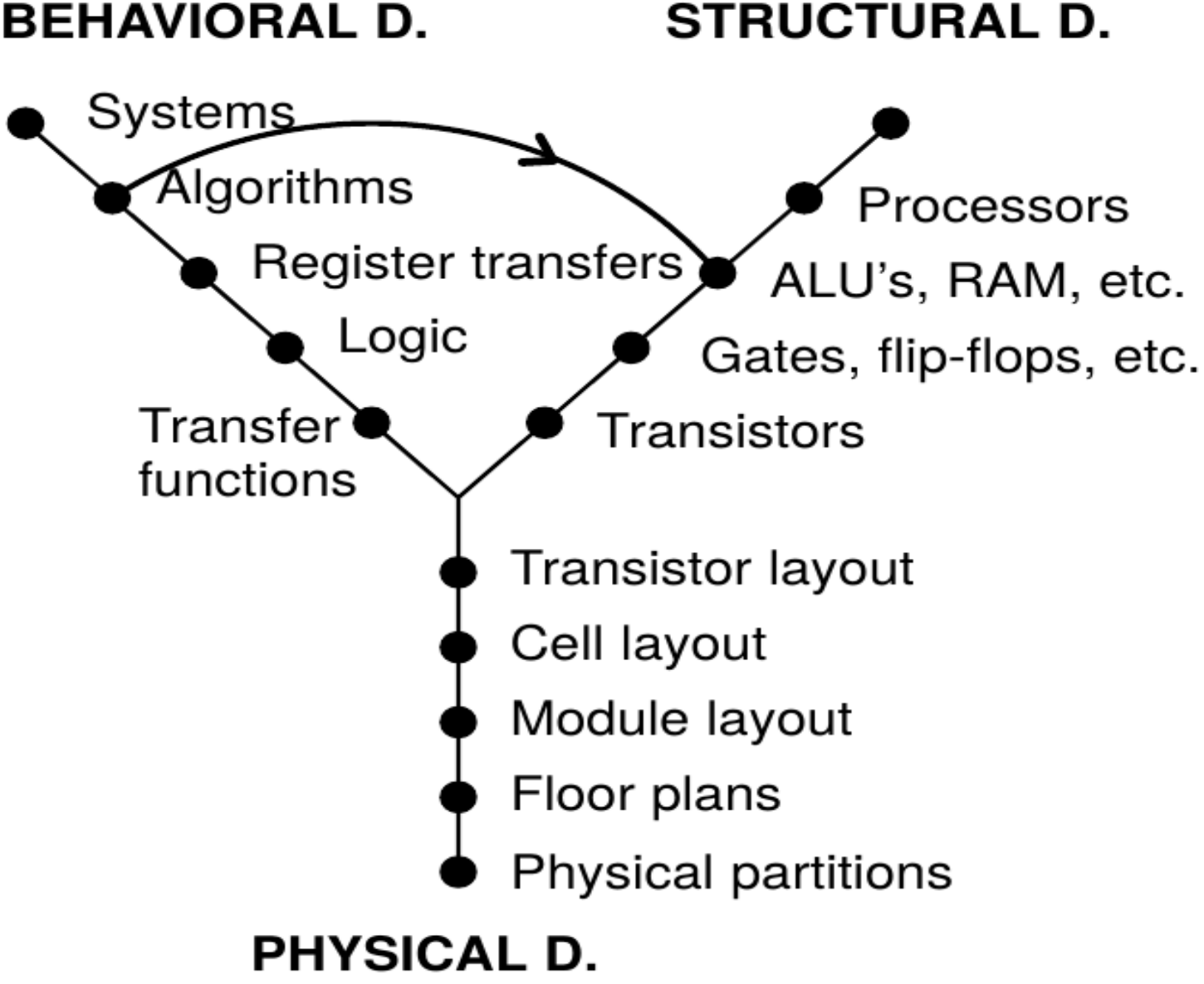
Switch Level Modeling

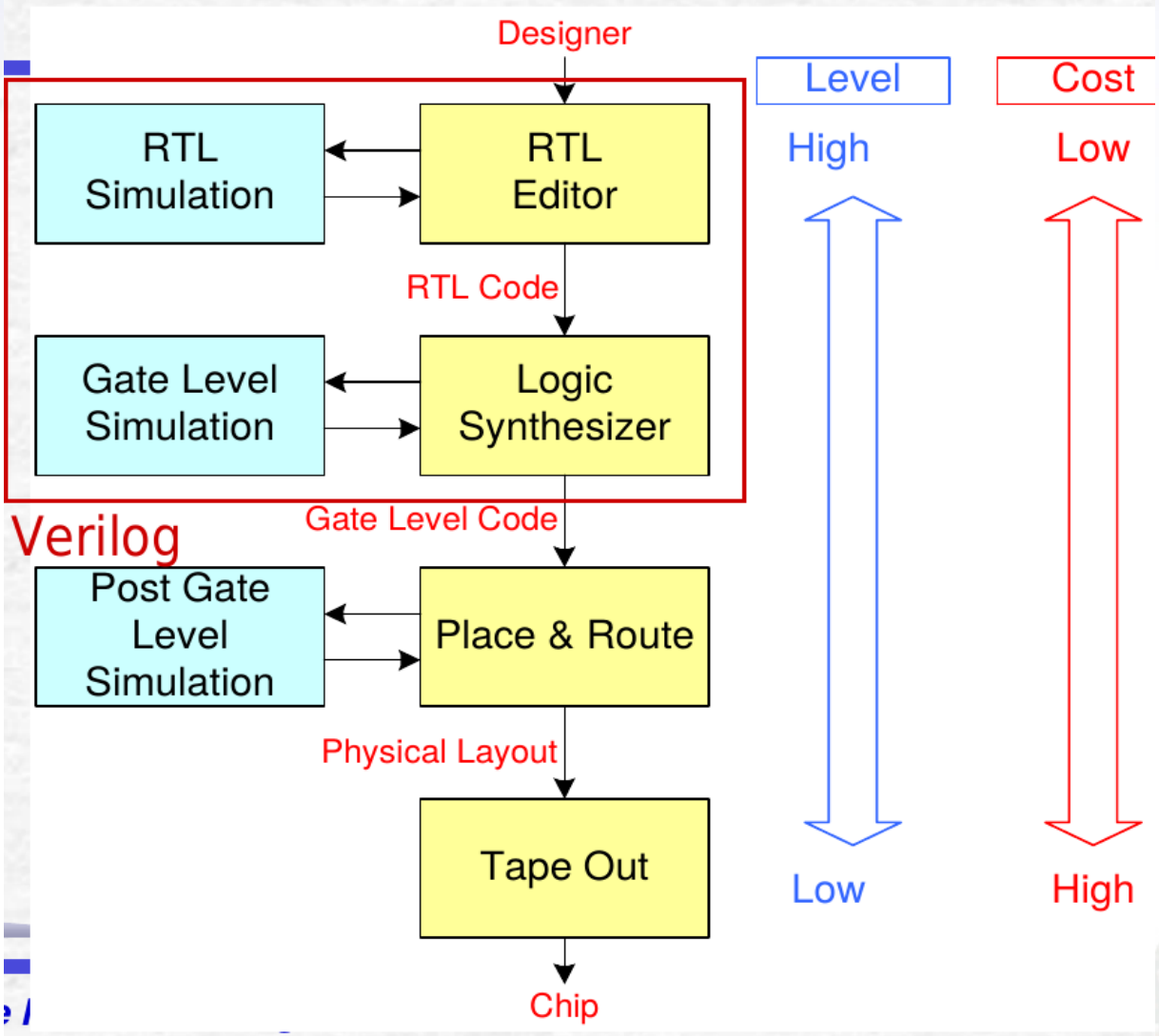
Outline

Verilog

Modelsim




BEHAVIORAL D.
STRUCTURAL D.
Systems
Algorithms
Register transfers
Logic
Transfer functions
Processors
ALU's, RAM, etc.
Gates, flip-flops, etc.
Transistors
Transistor layout
Cell layout
Module layout
Floor plans
Physical partitions
PHYSICAL D.


Designer
RTL Simulation
RTL Editor
RTL Code
Gate Level Simulation
Logic Synthesizer
Verilog
Gate Level Code
Post Gate Level Simulation
Place & Route
Physical Layout
Tape Out
Chip
Level
High
Low
Cost
Low
High

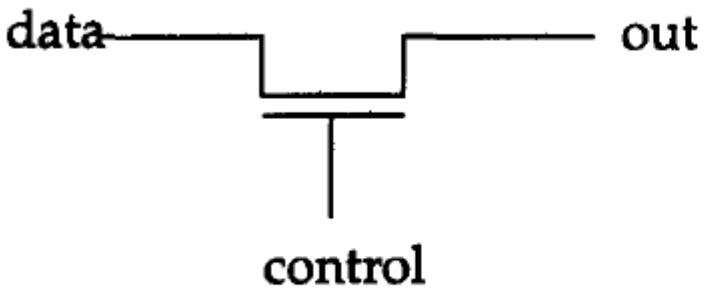
Verilog HDL Switch Level

- Verilog provides the ability to design at a MOS-transistor level.
- Design at switch level is becoming rare with the increasing complexity of circuits (millions of transistors) and with the availability of sophisticated CAD tools.
- Verilog HDL currently provides only digital design capability with logic values o, I, X, z, and the drive strengths associated with them.
- There is no analog capability. Thus, in Verilog HDL, transistors are also known switches that either conduct or are open.

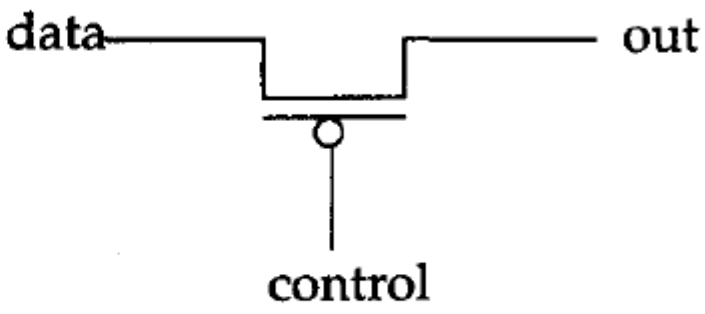
Objectives

- Describe basic MOS switches nmos, pmos, and cmos.
- Understand modeling of bidirectional pass switches, power, and ground.
- Identify resistive MOS switches.
- Explain the method to specify delays on basic MOS switches and bidirectional pass switches.
- Build basic switch-level circuits in Verilog, using available switches.

```
//MOS switch keywords
nmos                    pmos
```



NMOS



PMOS

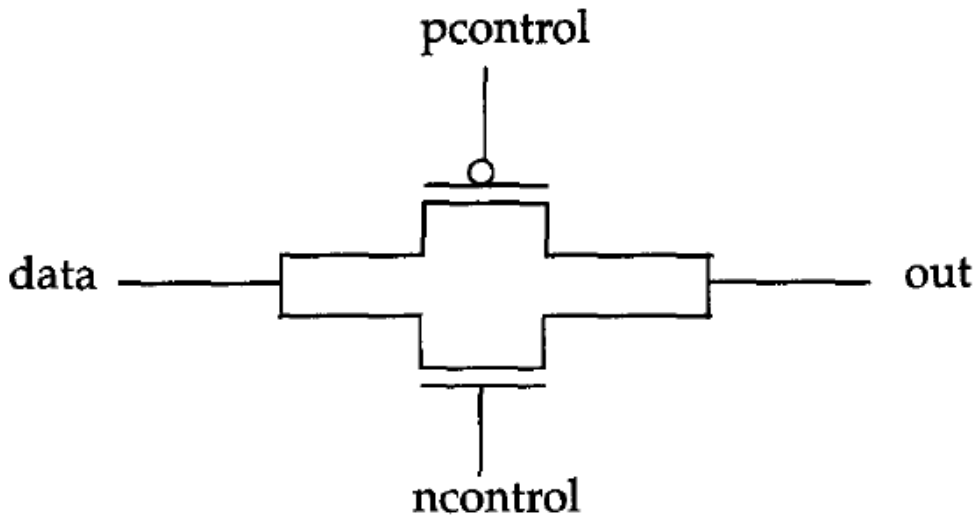
```
nmos n1(out, data, control); //instantiate a nmos switch
pmos p1(out, data, control); //instantiate a pmos switch
```

Logic Tables for NMOS and PMOS

nmos data	control 0	1	x	z
0	z	0	L	L
1	z	1	H	H
x	z	x	x	x
z	z	z	z	z

pmos data	control 0	1	x	z
0	0	z	L	L
1	1	z	H	H
x	x	z	x	x
z	z	z	z	z

CMOS Switch

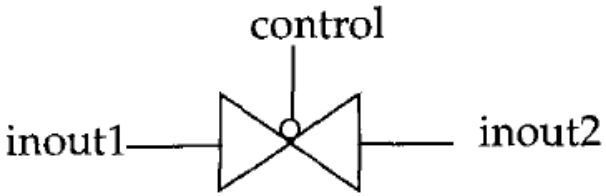


CMOS

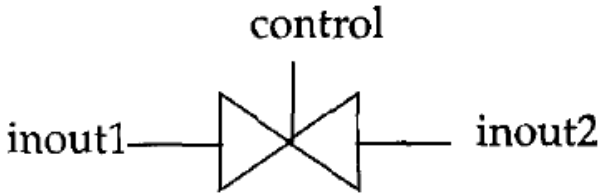
```
cmos c1(out, data, ncontrol, pcontrol);//instantiate cmos gate.
                or
cmos (out, data, ncontrol, pcontrol); //no instance name given.
```

Bidirectional Switches

tran



tranif0



tranif1

```
tran t1(inout1, inout2); //instance name t1 is optional
tranif0 (inout1, inout2, control); //instance name is not specified
tranif1 (inout1, inout2, control); //instance name is not specified
```

Power and Ground

```
supply1 vdd;
supply0 gnd;

assign a = vdd; //Connect a to vdd
assign b = gnd; //Connect b to gnd
```

Delays

Switch Element	Delay Specification	Examples

pmos, nmos, rpmos, rnmos	Zero (no delay)
	One (same delay on all transitions)
	Two (rise, fall)
	Three (rise, fall, turnoff)
		pmos p1(out, data, control);
		pmos #(1) p1(out, data, control);
		
		nmos #(1, 2) p2(out, data, control);
		nmos #(1, 3, 2) p2(out, data, control);

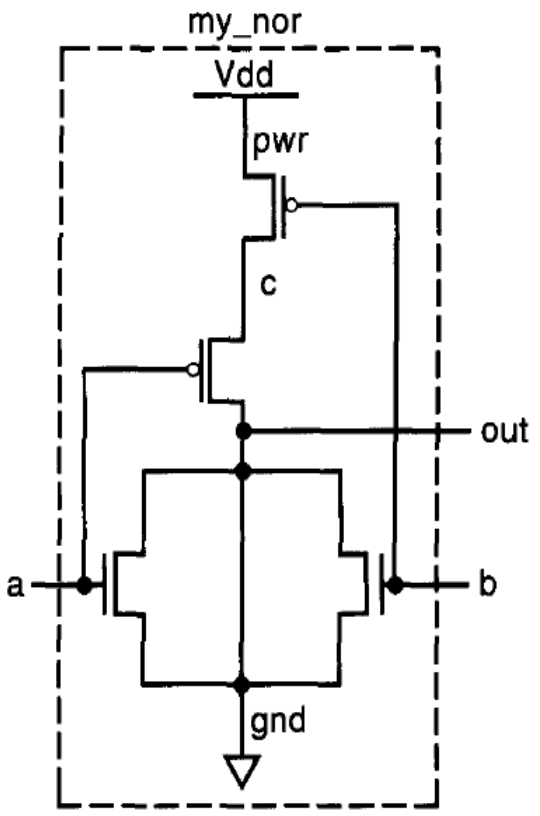
cmos, rcmos	Zero, one, two or three delays (same as above)	cmos #(5) c2(out, data, nctrl, pctrl); cmos #(1,2) c1(out, data, nctrl, pctrl);

Switch Element	Delay Specification	Examples
tran, rtran	No delay specification allowed	

tranif1, rtranif1
tranif0, rtranif0
	Zero (no delay)
	One (both turn-on and turn-off)
	
	Two (turn-on, turn-off)
		rtranif0 rt1(inout1, inout2, control);
		tranif0 #(3) T(inout1, inout2, control);
		tranif1 #(1,2) t1(inout1, inout2, control);

Example
CMOS NOR Gate

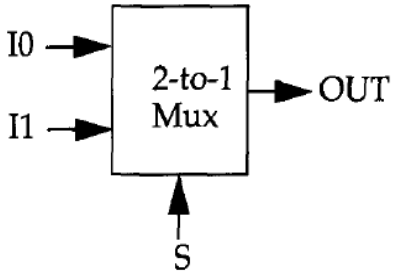
```
//Define our own nor gate, m_nor

module my_nor (out, a, b);
output out;
input a, b;
//internal wires
wire c;
//set up power and ground lines
supply1 pwr;
//pwr is connected to Vdd (power supply)
supply0 gnd;
//gnd is connected to Vss(ground)
//instantiate pmos switches
pmos (c, pwr, b);
pmos (out, c, a);
//instantiate nrnos switches
nmos (out, gnd, a);
nmos (out, gnd, b);
endmodule
```

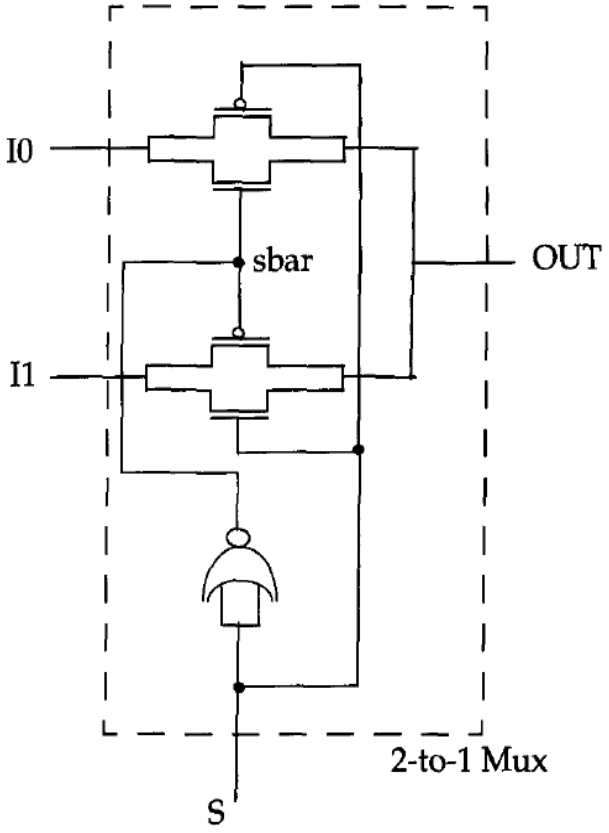


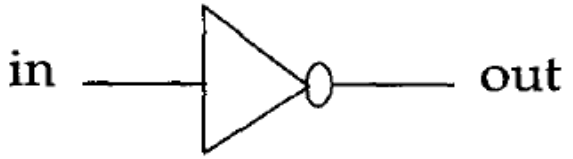
Test bench

```
//stimulus to test the gate
module stimulus;
reg A, B;
wire OUT;
//instantiate the my_nor module
my_nor n1(OUT, A, B);
//Apply stimulus
initial
begin
//test all possible combinations
A=1'b0; B = 1'b0;
#5 A=1'b0; B=1'b1;
#5 A=1'b1; B=1'b0;
#5 A=1'b1; B=1'b1;
end
//check results
initial
$monitor("Time", $time, " OUT %b, A %b, B %b", OUT, A, B);
endmodule
```

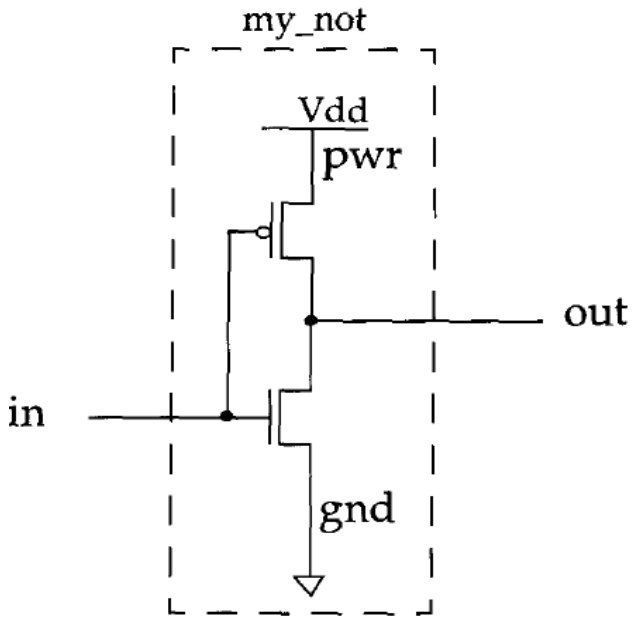
2 to 1 Mux



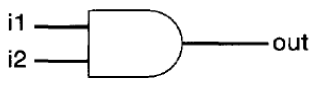




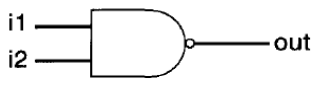
my_not



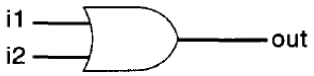
Gate Level Modeling



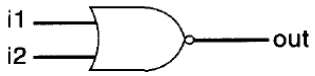
and



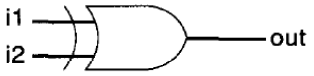
nand



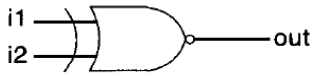
or



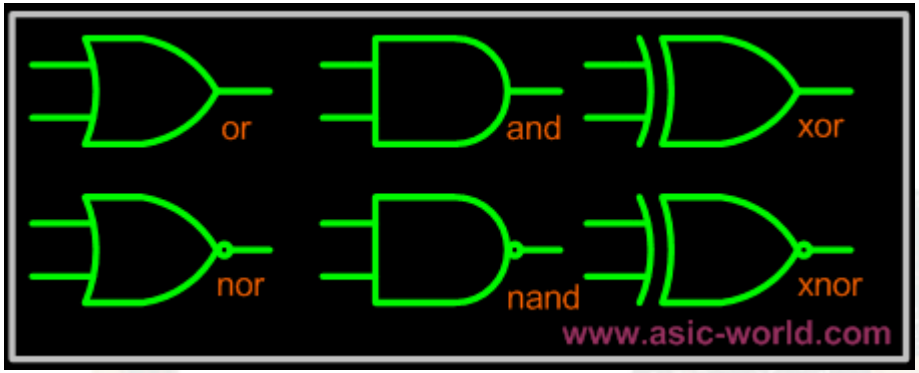
nor



xor



xnor




```
and       or       xor
nand      nor      xnor
```

Basic Gates Instantiation

```
and a1(OUT, IN1, IN2);
nand na1(OUT, IN1, IN2);
or or1(OUT, IN1, IN2);
nor nor1(OUT, IN1, IN2);
xor x1(OUT, IN1, IN2);
xnor nx1(OUT, IN1, IN2);
```