VLSI: Switch-Level and Gate-Level System Design

Tassadaq Hussain

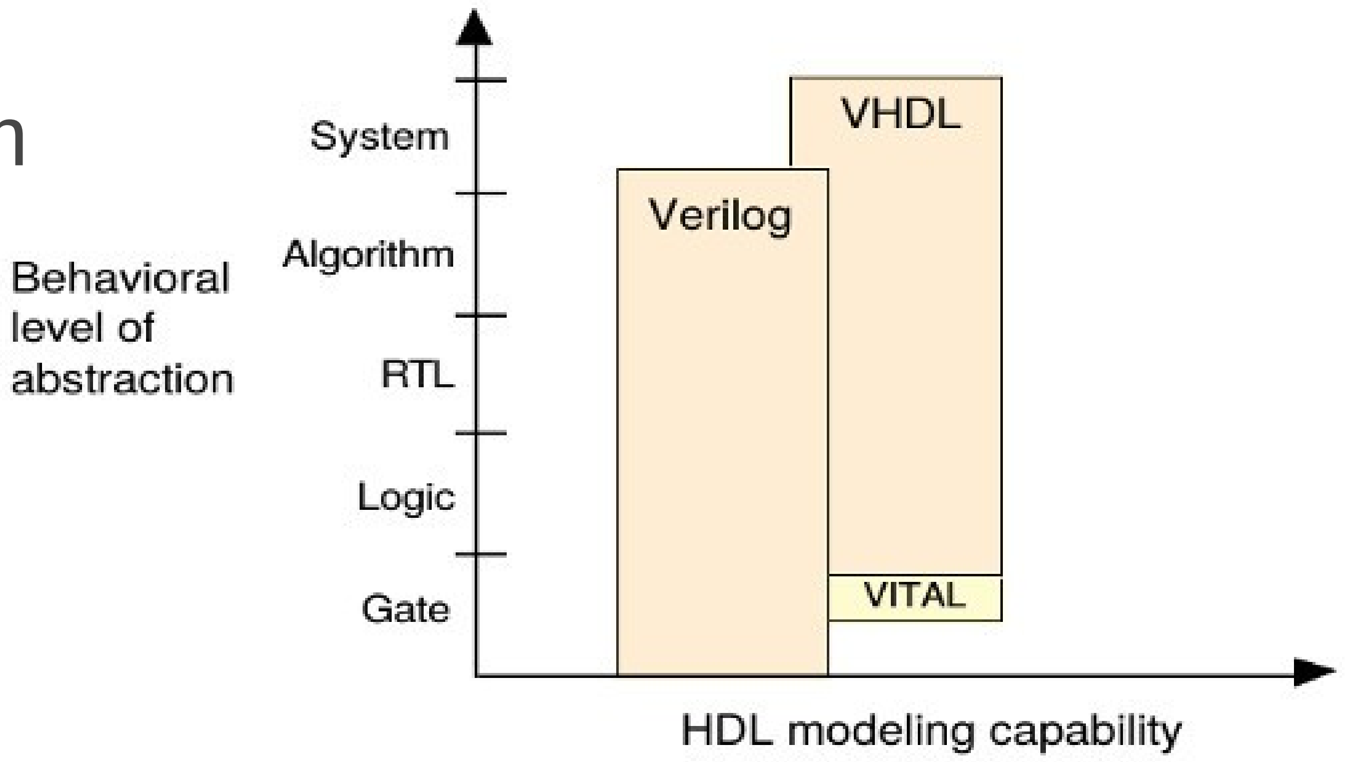
Namal University Mianwali

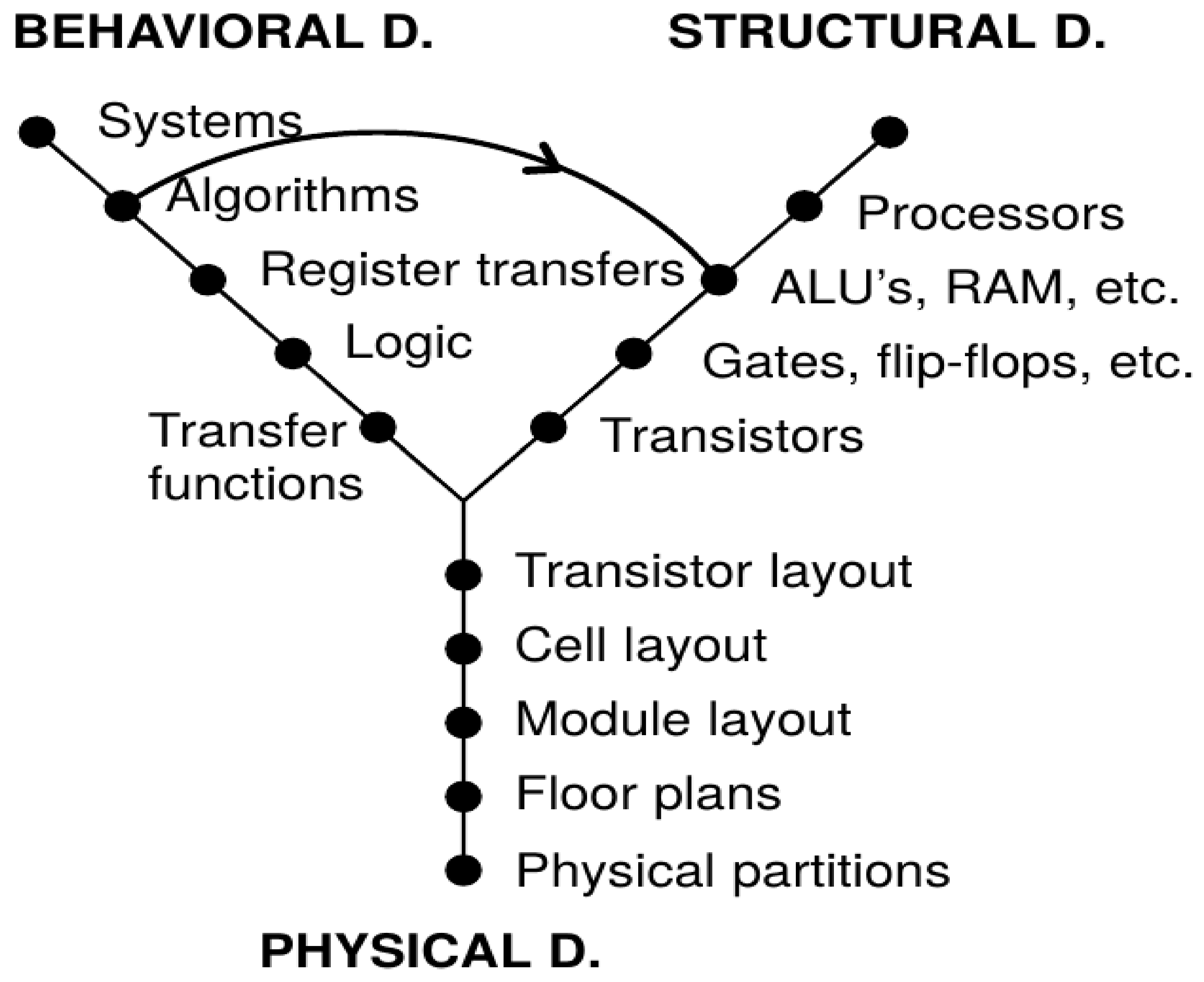
PakASIC.com

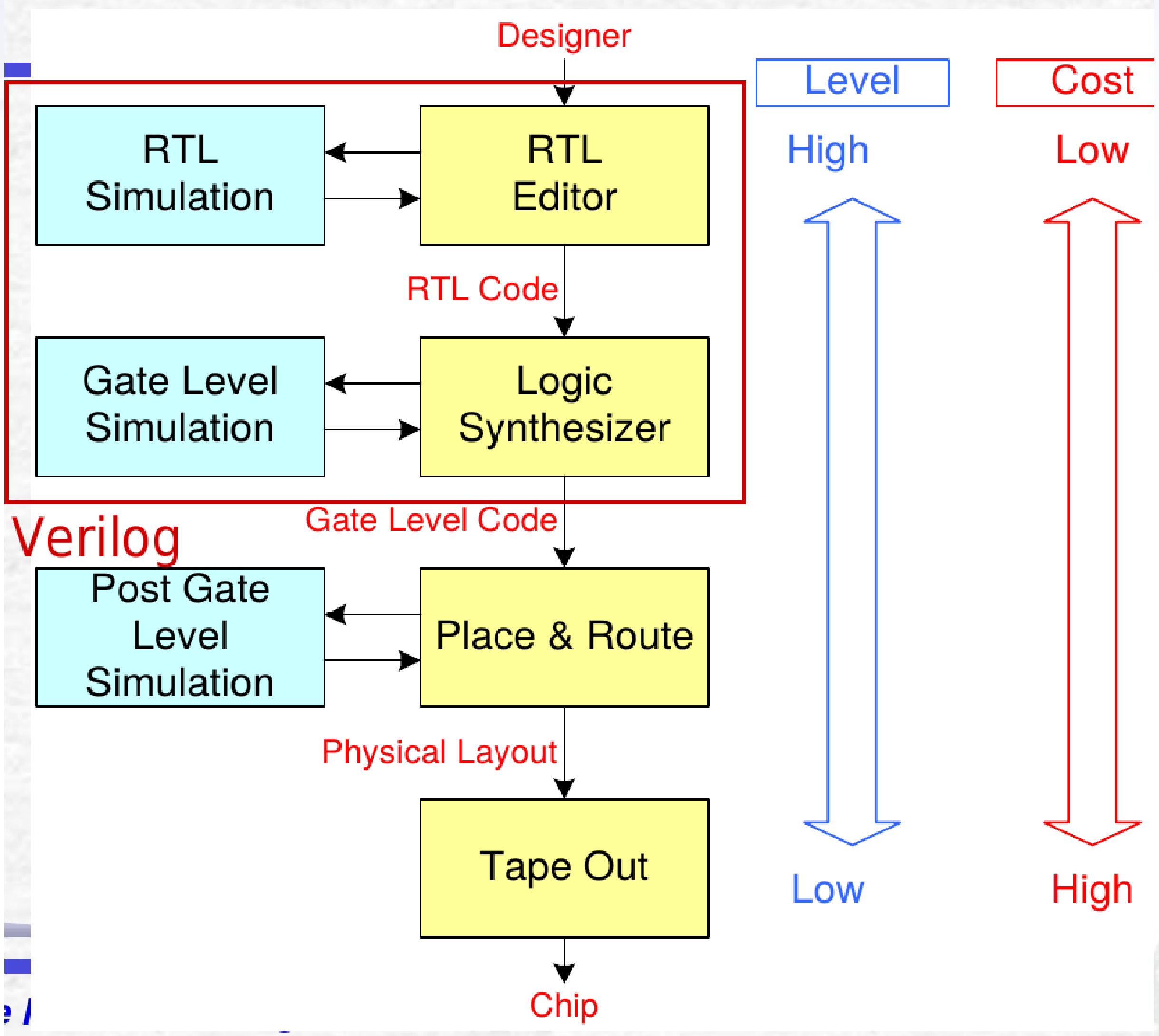
Outline

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

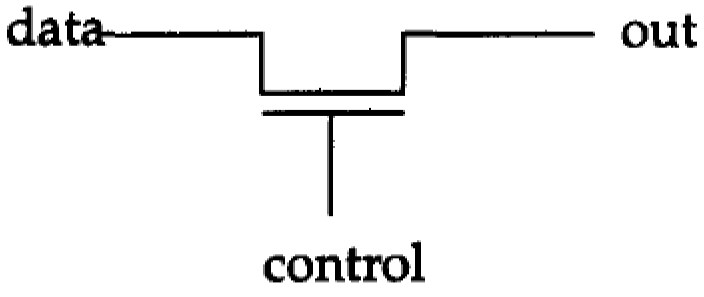
Verilog HDL Switch Level

- 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.

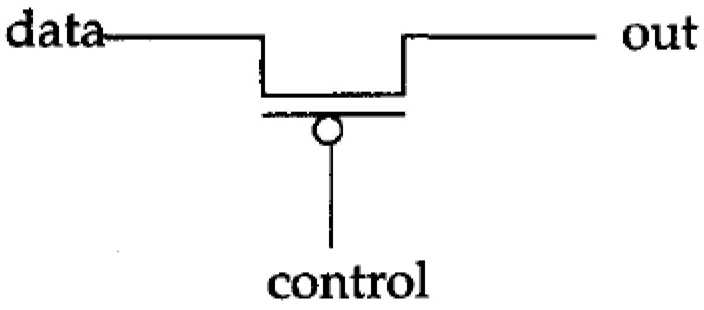
Objectives

- 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
```



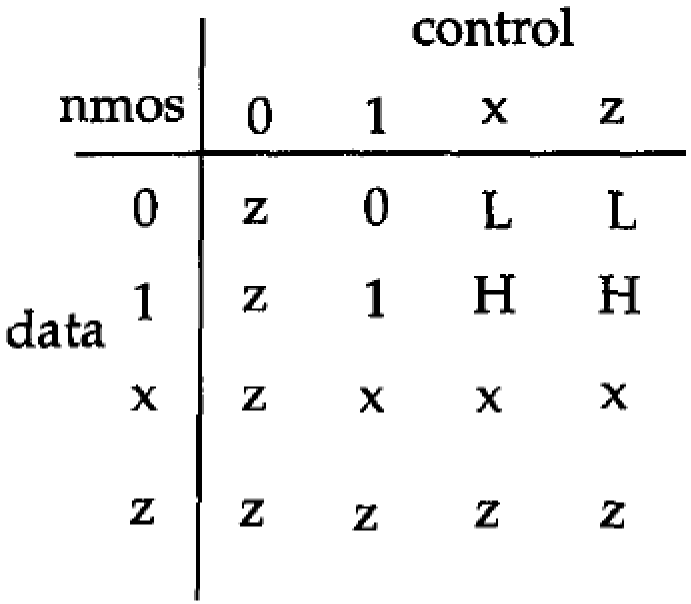
NMOS



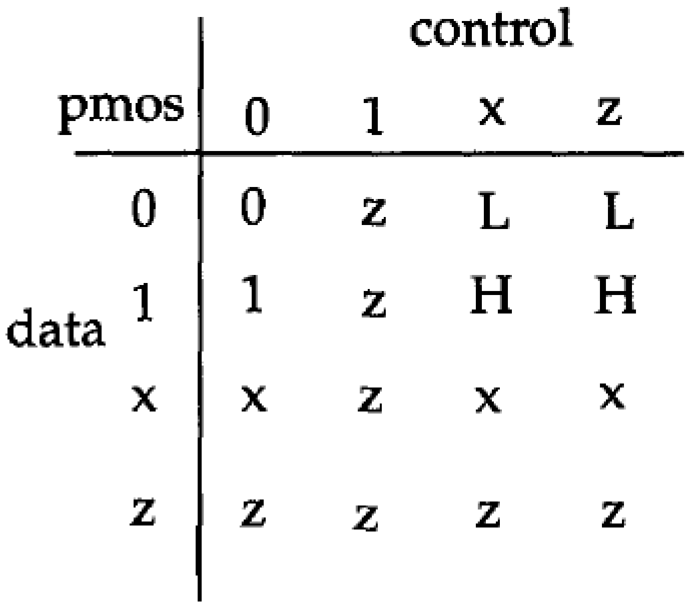
PMOS

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

Logic Tables for NMOS and PMOS

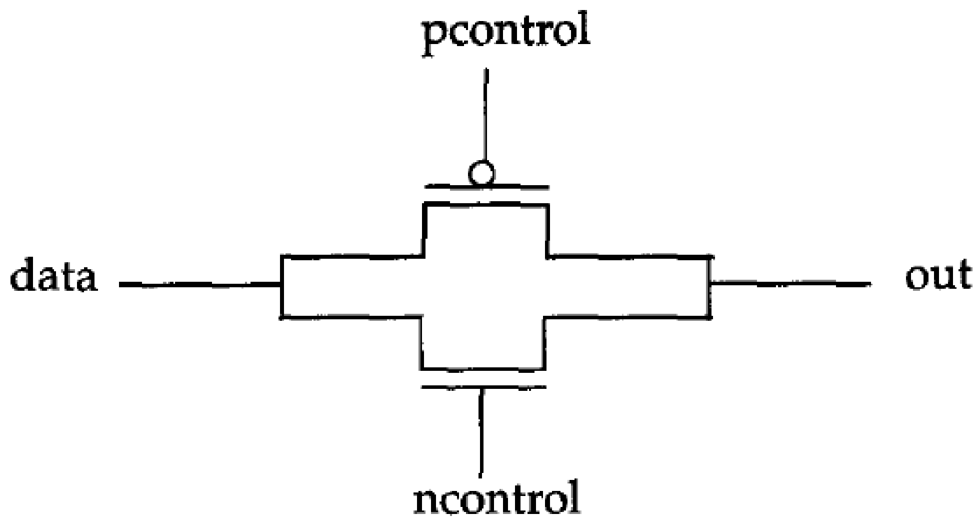


		control			
	nmos	0	1	x	z
	0	z	0	L	L
data	1	z	1	H	H
	x	z	x	x	x
	z	z	z	z	z



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

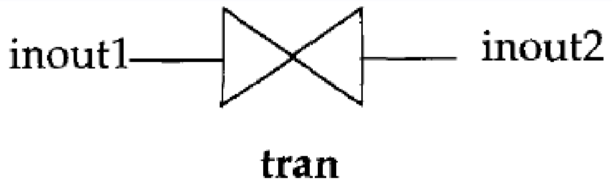
CMOS Switch

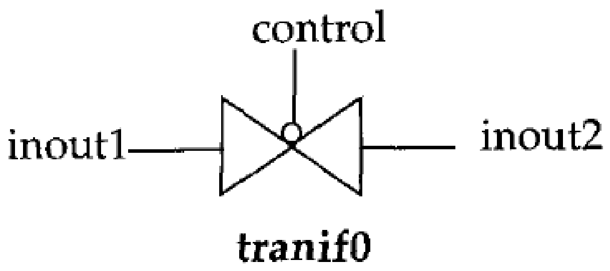


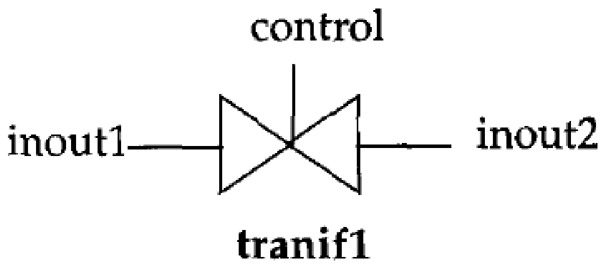
CMOS

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

Bidirectional Switches





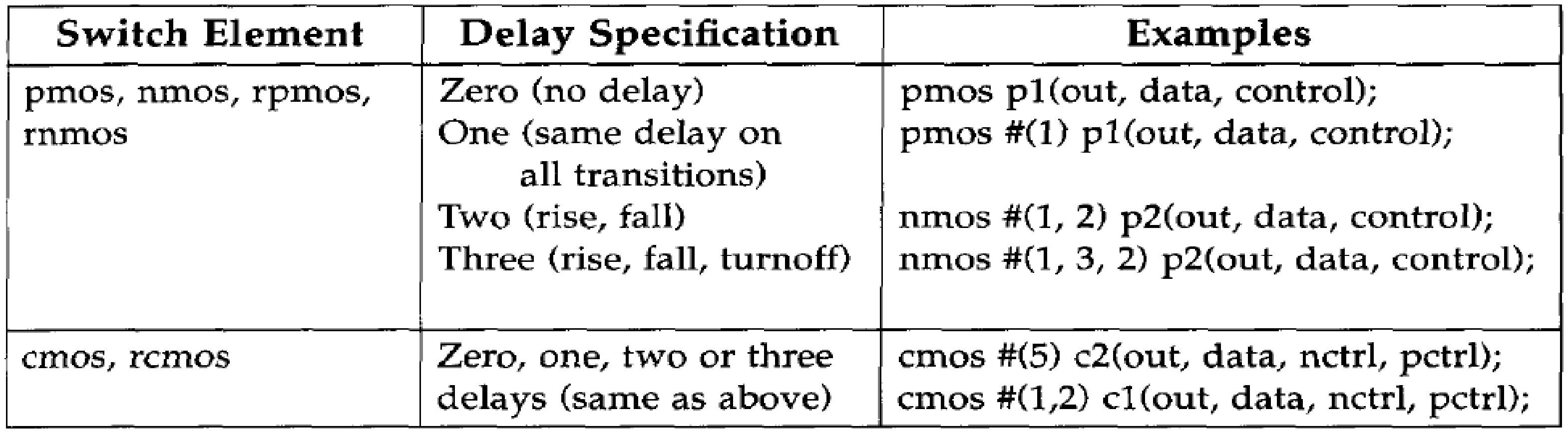


```
tran t1(inout1, inout2); //instance name t1 is optional
tranif0 (inout1, inout2, control); //instance name is not specified
tranif1 (inout1, inout2, control); //instance name is not specified
```

Power and Ground

```
supply1 vdd;
supply0 gnd;

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

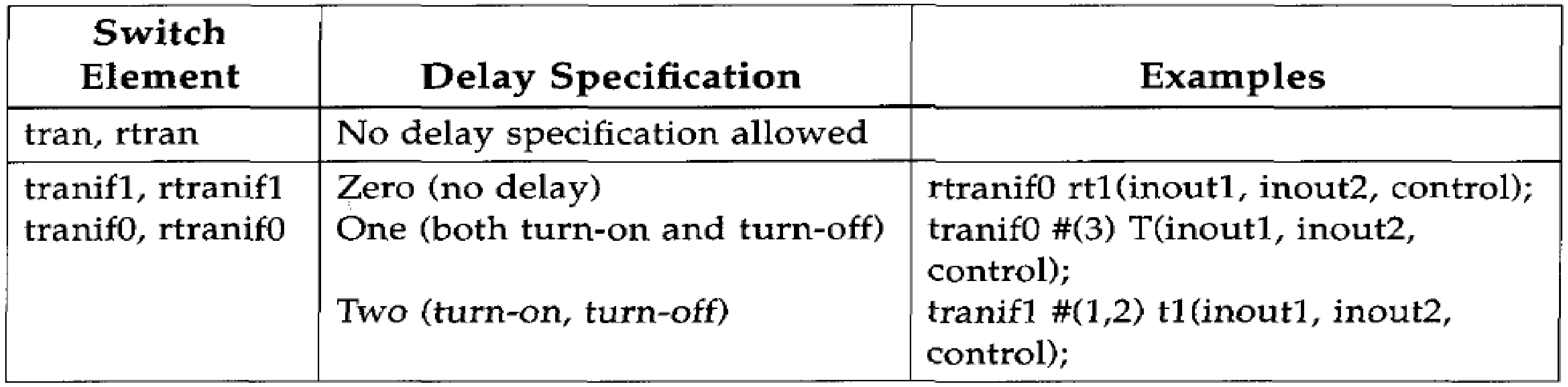
Delays



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);

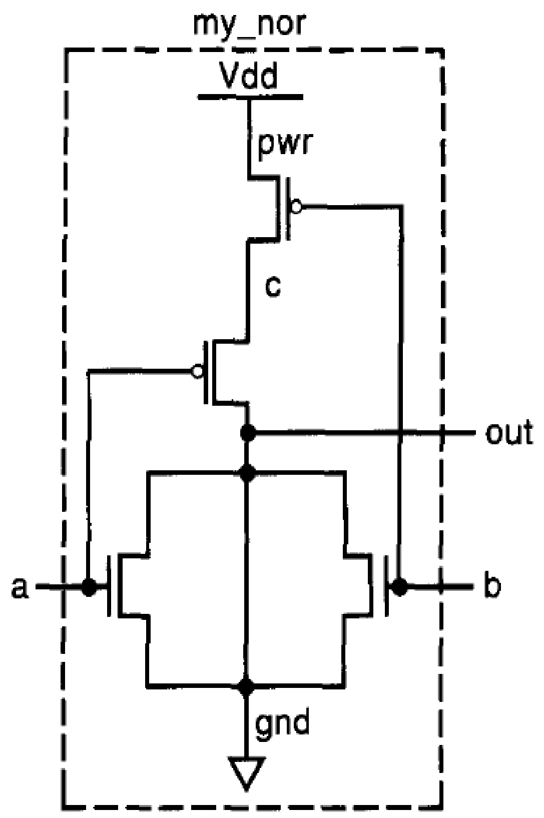


Switch Element	Delay Specification	Examples
tran, rtran	No delay specification allowed	

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

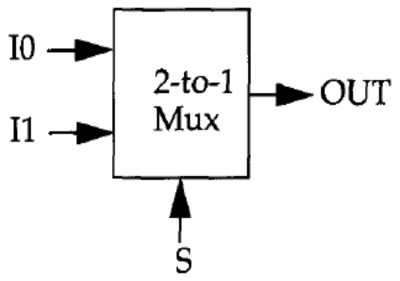
```
//Define our own nor gate, m_nor

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
```

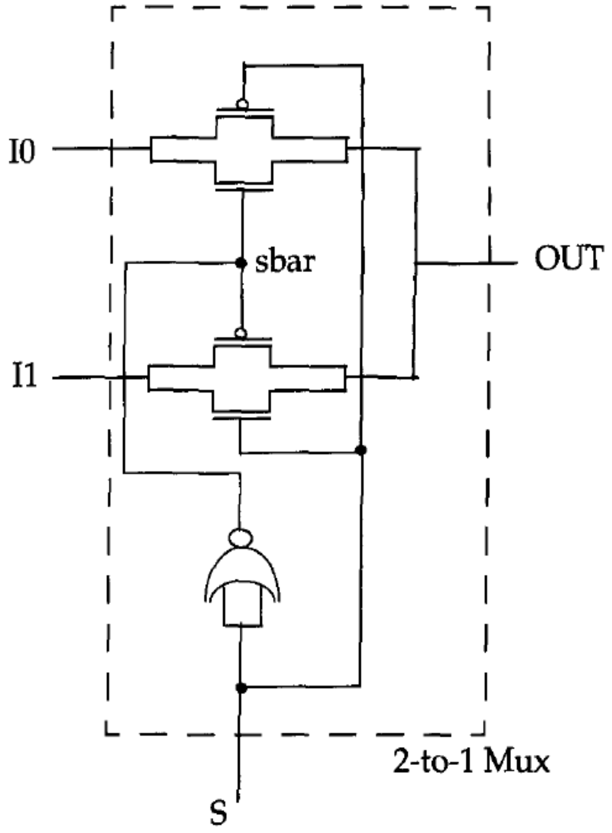


Test bench

```
//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

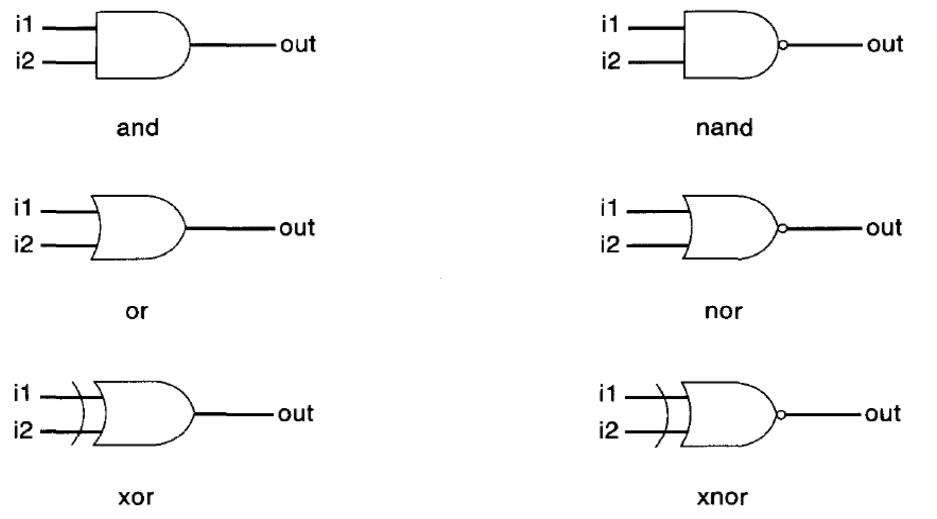


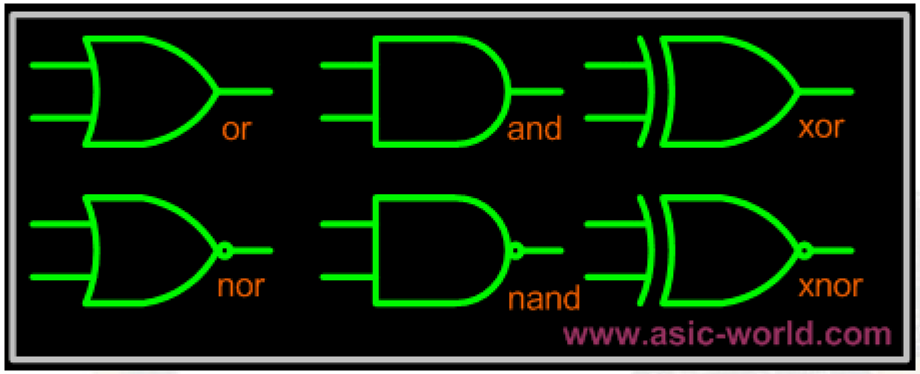


Objectives: Gate Level Design

- Identify logic gate primitives provided in Verilog.
- Understand instantiation of gates, gate symbols and truth tables for and/or and buf/not type gates.
- Understand how to construct a Verilog description from the logic diagram of the circuit.
- Describe rise, fall, and turn-off delays in the gate-level design.
- Explain min, max, and type delays in the gate-level design.

Gate Level Modeling





```
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);
```

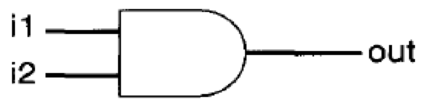
Types of Gates

And/Or Gates

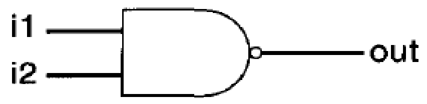
Buff/Not

And/Or Gates

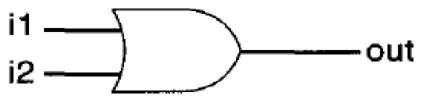
```
and      or      xor
nand     nor     xnor
```



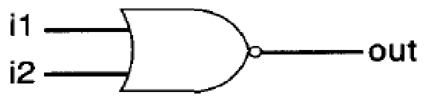
and



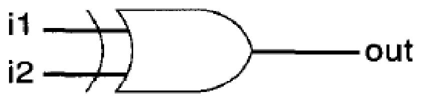
nand



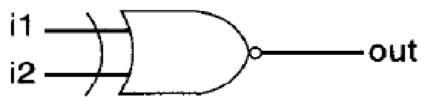
or



nor



xor

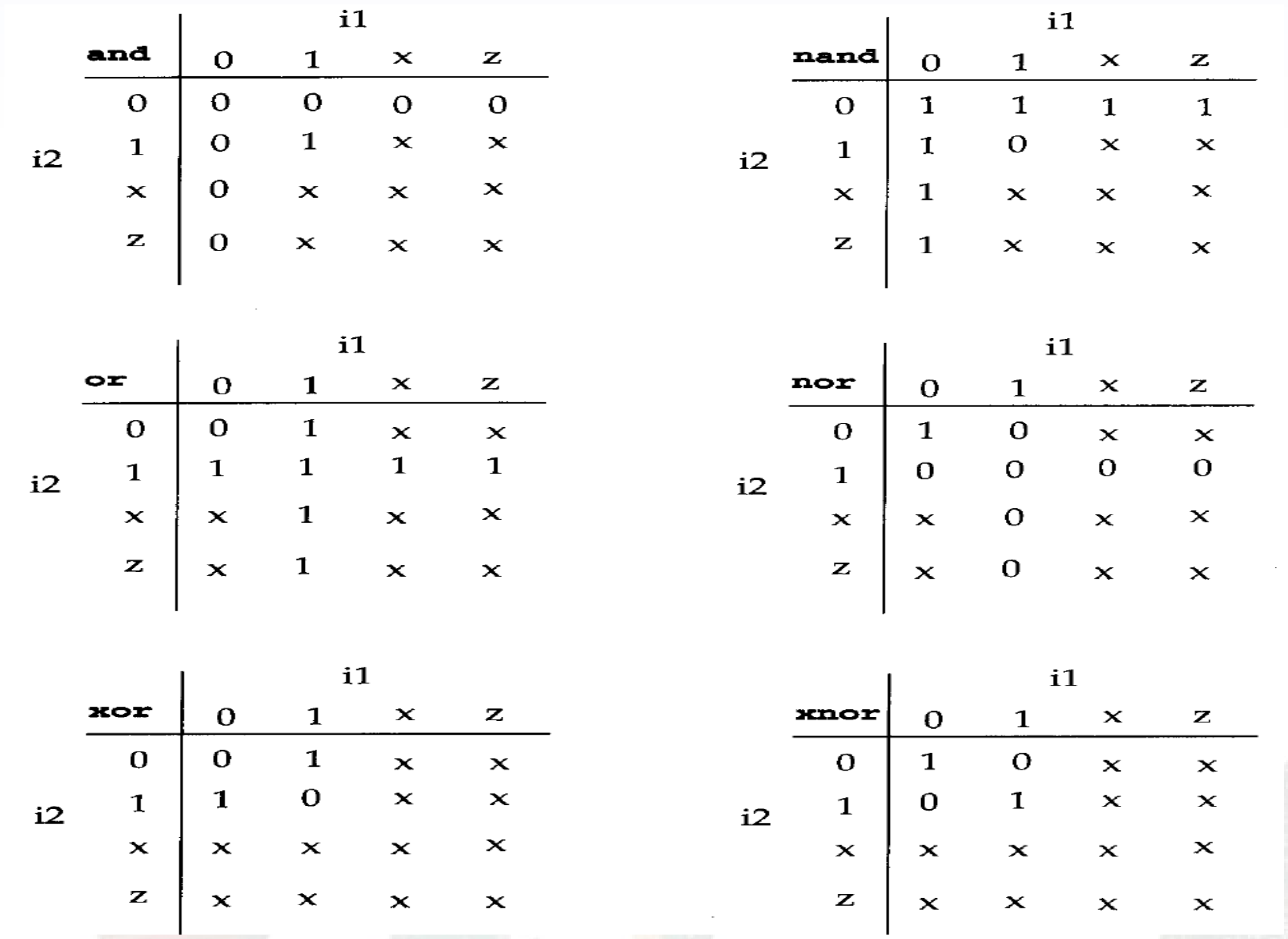


xnor

Basic Gates

```
// basic gate instantiations.
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);

// More than two inputs; 3 input nand gate
nand na1_3inp(OUT, IN1, IN2, IN3);

// gate instantiation without instance name
and (OUT, IN1, IN2); // legal gate instantiation
```

Truth Table of Gates



		i1			
	and	0	1	x	z
i2	0	0	0	0	0
	1	0	1	x	x
	x	0	x	x	x
	z	0	x	x	x

		i1			
	nand	0	1	x	z
i2	0	1	1	1	1
	1	1	0	x	x
	x	1	x	x	x
	z	1	x	x	x

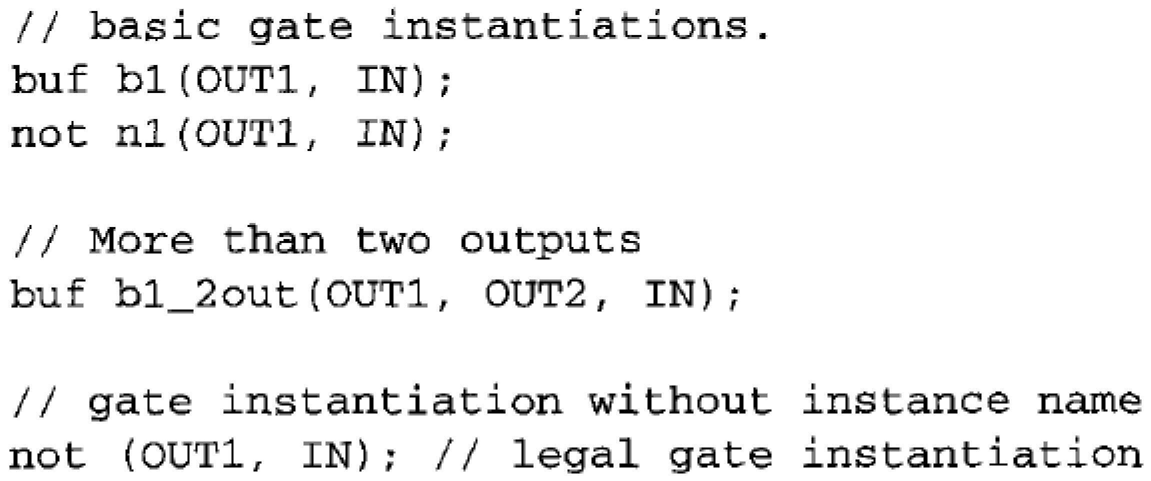
		i1			
	or	0	1	x	z
i2	0	0	1	x	x
	1	1	1	1	1
	x	x	1	x	x
	z	x	1	x	x

		i1			
	nor	0	1	x	z
i2	0	1	0	x	x
	1	0	0	0	0
	x	x	0	x	x
	z	x	0	x	x

		i1			
	xor	0	1	x	z
i2	0	0	1	x	x
	1	1	0	x	x
	x	x	x	x	x
	z	x	x	x	x

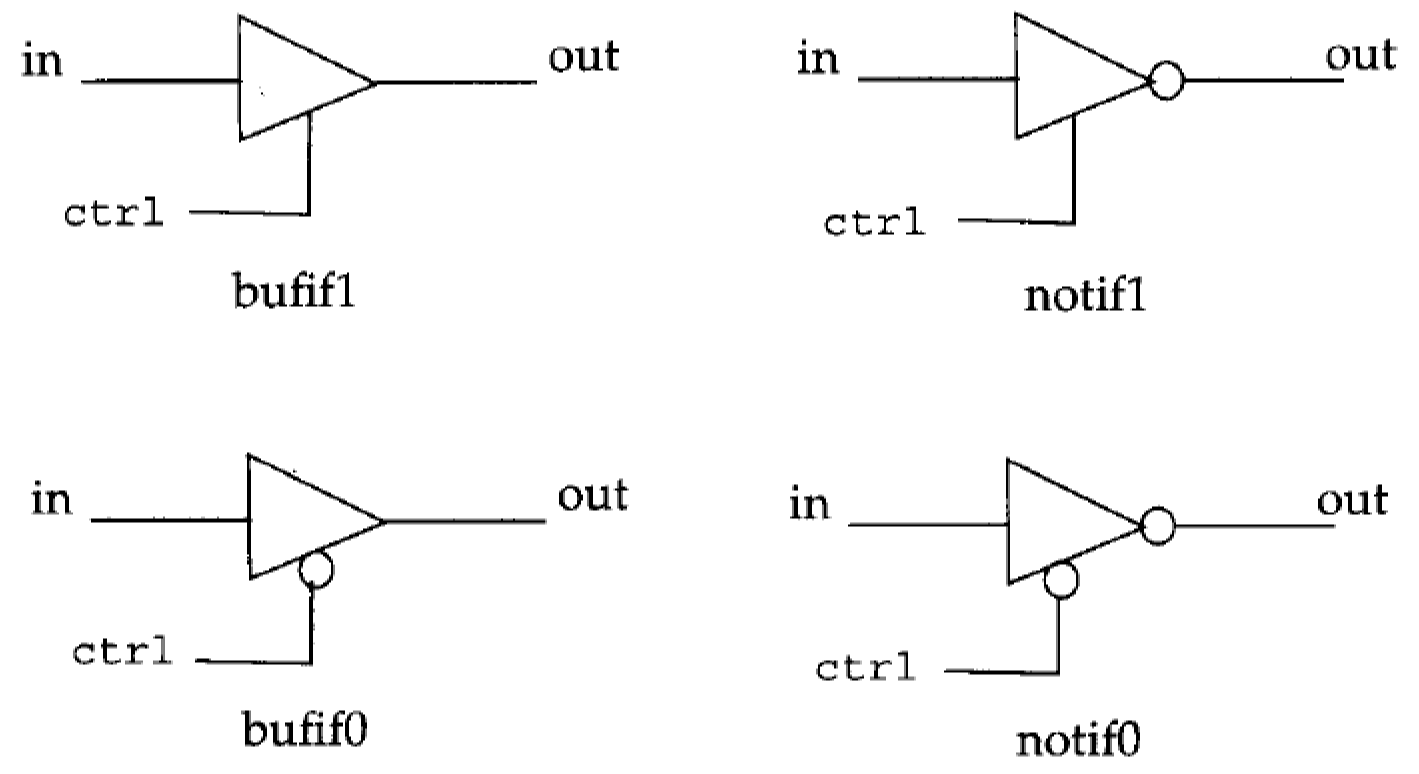
		i1			
	xnor	0	1	x	z
i2	0	1	0	x	x
	1	0	1	x	x
	x	x	x	x	x
	z	x	x	x	x

Buf/Not Gates

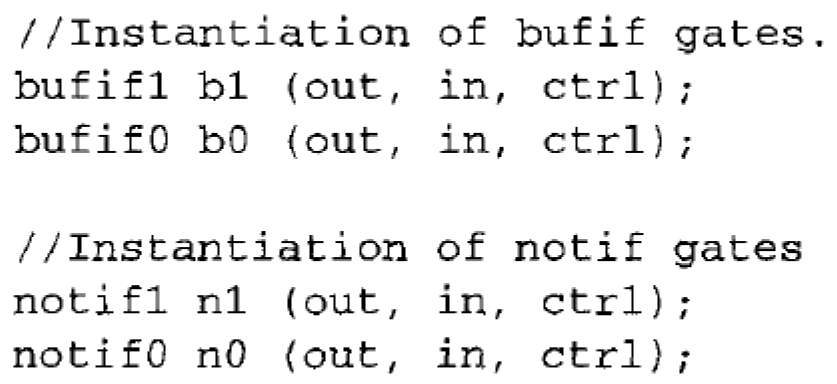


```
// basic gate instantiations.
buf b1(OUT1, IN);
not n1(OUT1, IN);

// More than two outputs
buf b1_2out(OUT1, OUT2, IN);

// gate instantiation without instance name
not (OUT1, IN); // legal gate instantiation
```

Bufif / Notif

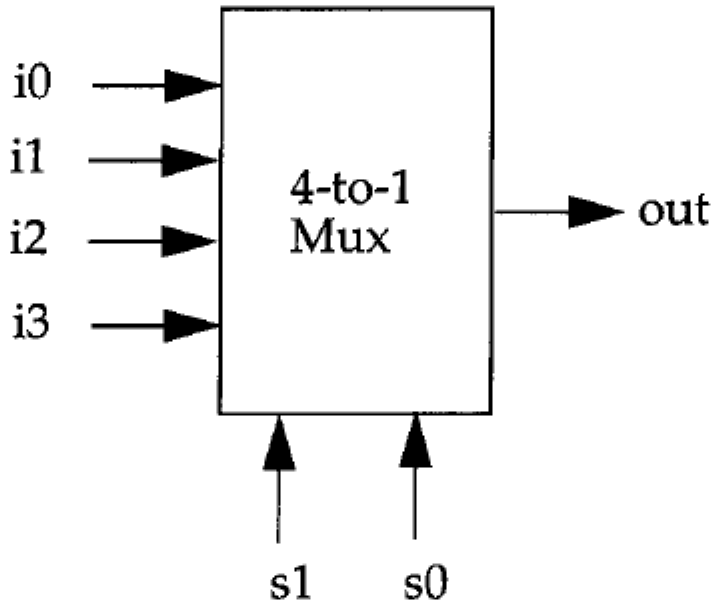

in
out
ctrl
bufif1
in
out
ctrl
notif1
in
out
ctrl
bufif0
in
out
ctrl
notif0

Bufif / Notif



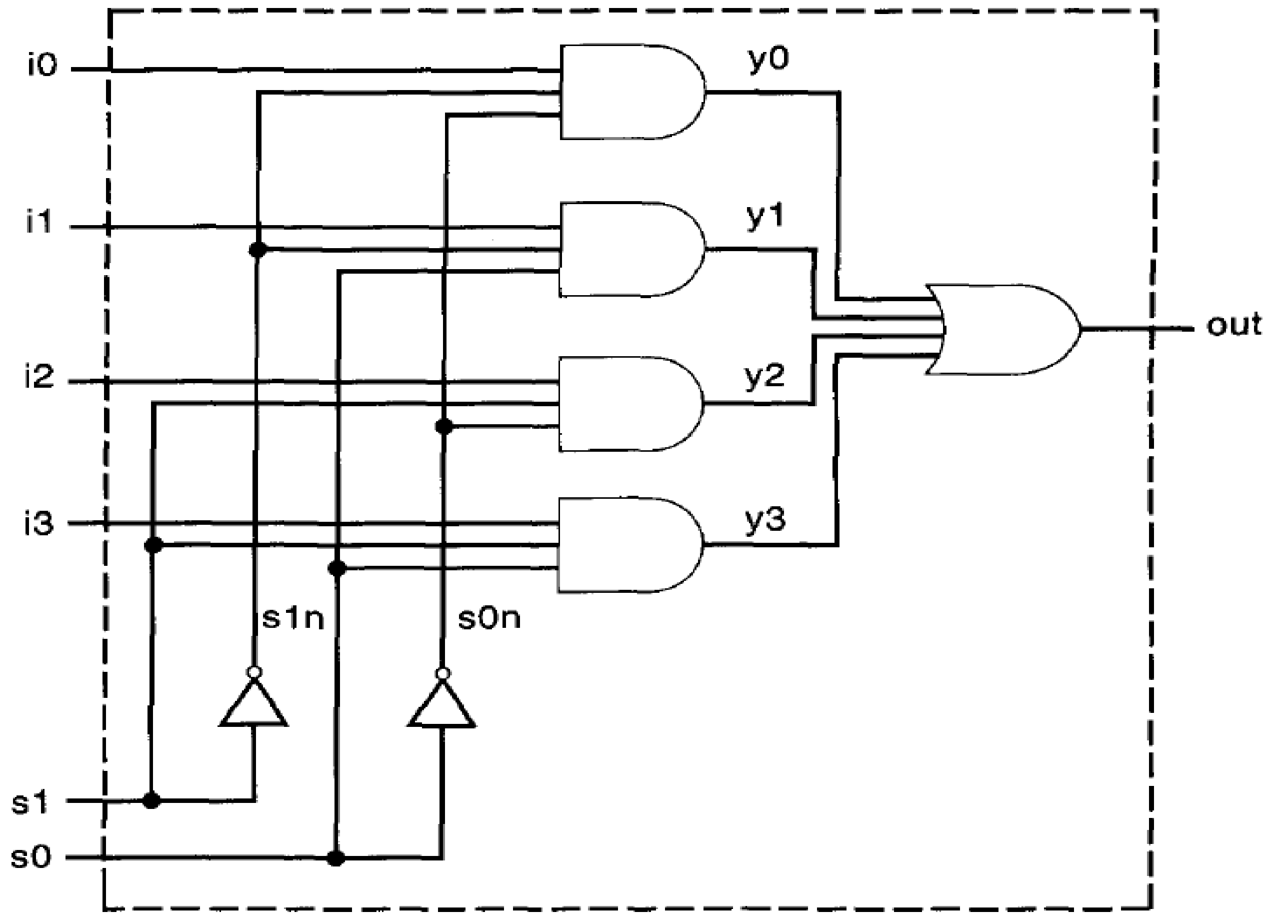
```
//Instantiation of bufif gates.
bufif1 b1 (out, in, ctrl);
bufif0 b0 (out, in, ctrl);

//Instantiation of notif gates
notif1 n1 (out, in, ctrl);
notif0 n0 (out, in, ctrl);
```

Example

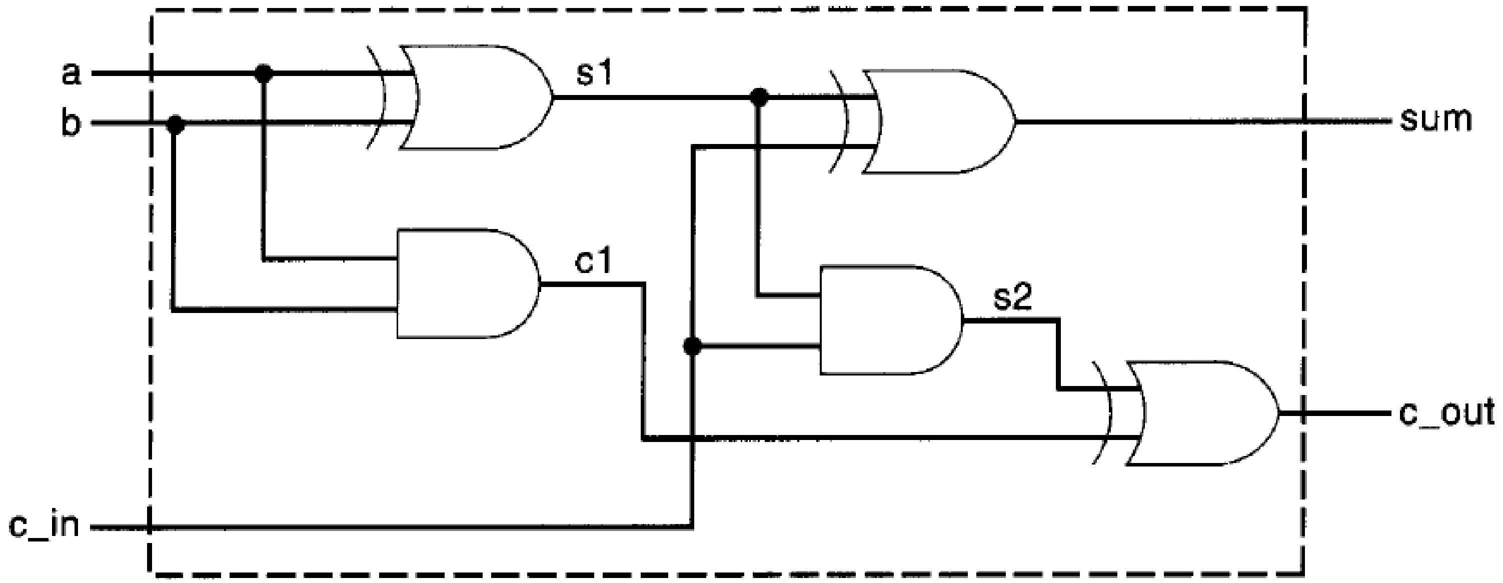


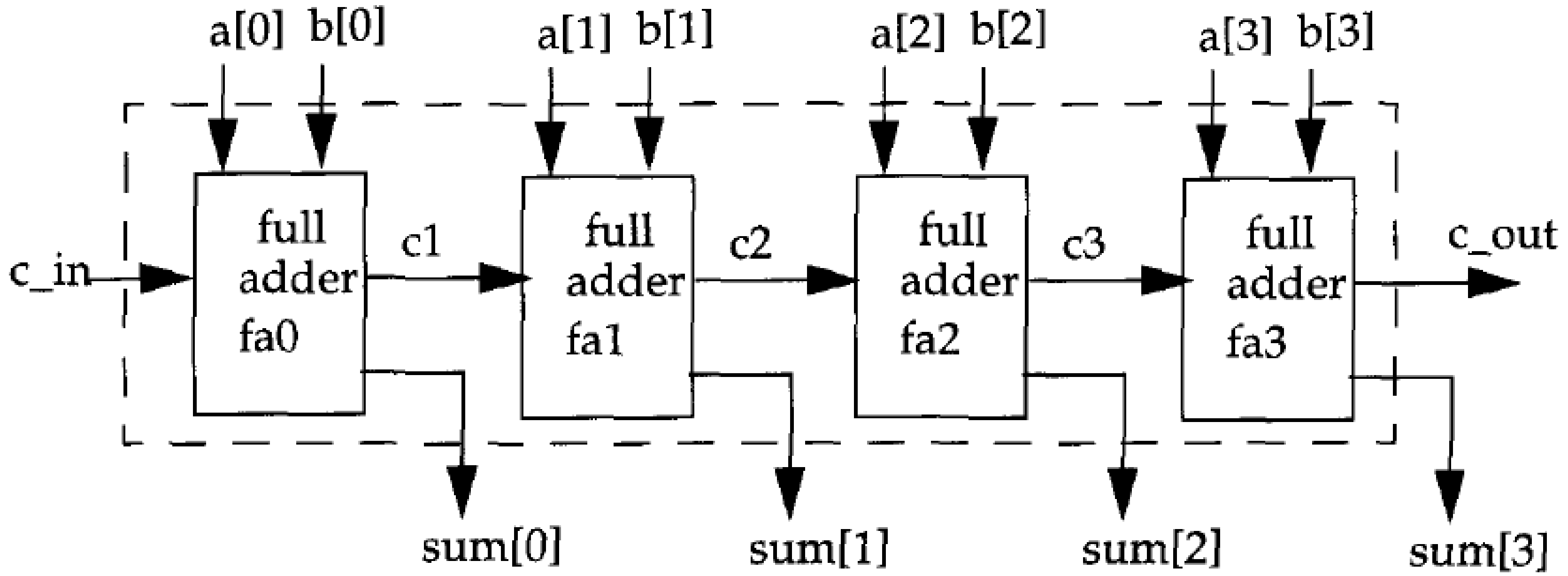
s1	s0	out
0	0	I0
0	1	I1
1	0	I2
1	1	I3

Gate Level Diagram



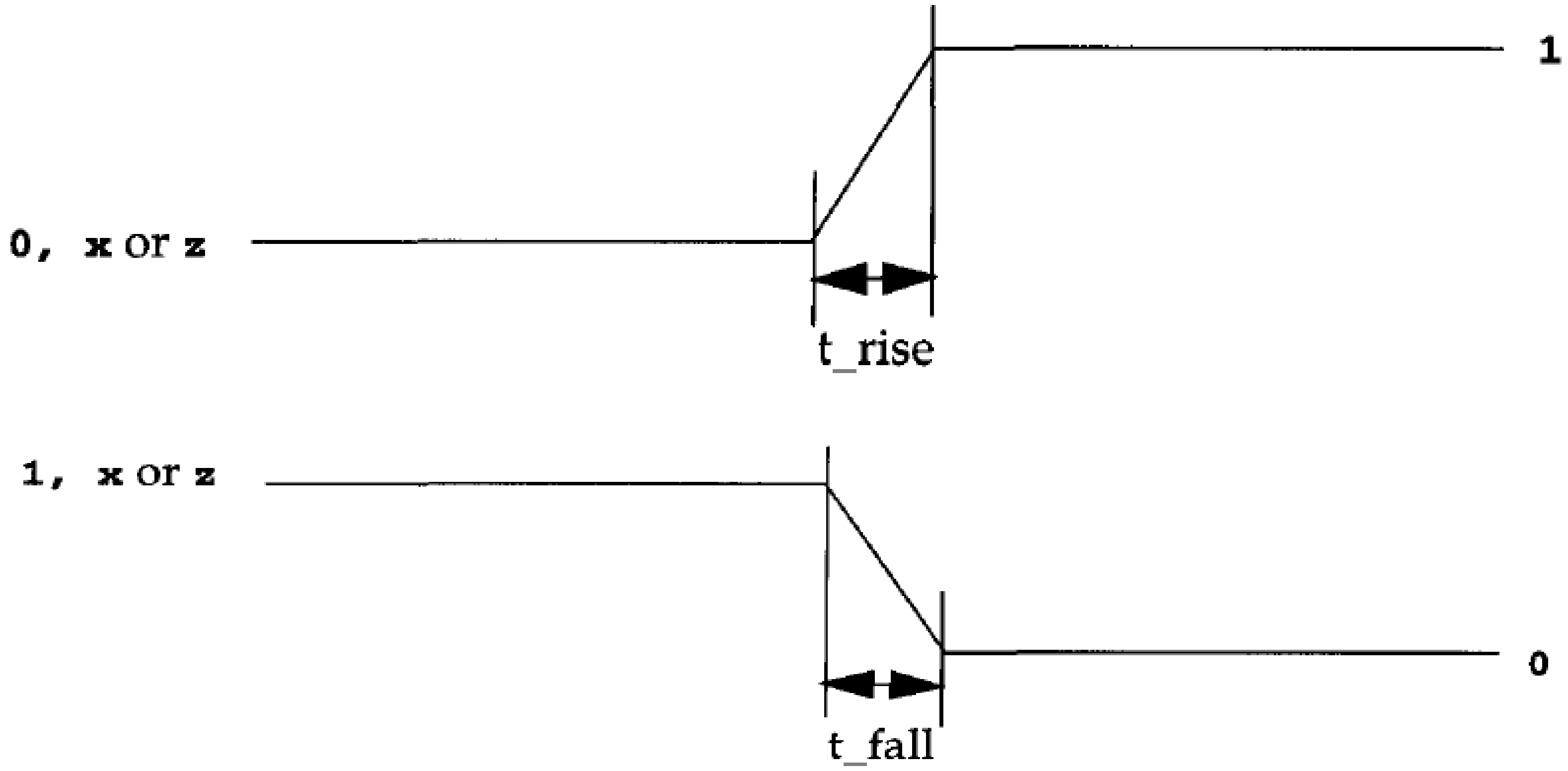
4 Bit Adder




a[0] b[0]
a[1] b[1]
a[2] b[2]
a[3] b[3]
c_in
full adder fa0
c1
full adder fa1
c2
full adder fa2
c3
full adder fa3
c_out
sum[0]
sum[1]
sum[2]
sum[3]

Gate Delays

Rise, Fall, and Turn-off Delays



Min/Typ/Max Value

```
// One delay
// if +mindelays, delay= 4
// if +typdelays, delay= 5
// if +maxdelays, delay= 6
and #(4:5:6) a1(out, i1, i2);

// Two delays
// if +mindelays, rise= 3, fall= 5, turn-off = min(3,5)
// if +typdelays, rise= 4, fall= 6, turn-off = min(4,6)
// if +maxdelays, rise= 5, fall= 7, turn-off = min(5,7)
and #(3:4:5, 5:6:7) a2(out, i1, i2);

// Three delays
// if +mindelays, rise= 2 fall= 3 turn-off = 4
// if +typdelays, rise= 3 fall= 4 turn-off = 5
// if +maxdelays, rise= 4 fall= 5 turn-off = 6
and #(2:3:4, 3:4:5, 4:5:6) a3(out, i1,i2);
```

Turn-off Delay

The Turn-off delay is associated with a gate output transition to z from another value (0, 1, x).

Design Gate/Switch Level

Inverter

Comparative

Counter

Multiplexer

Adder

Logic Indicator