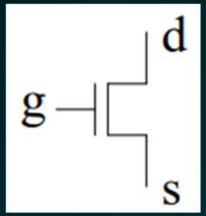
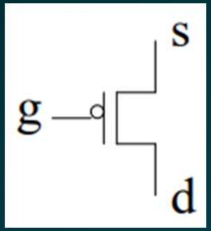
Switch-level modeli u Verilog-u

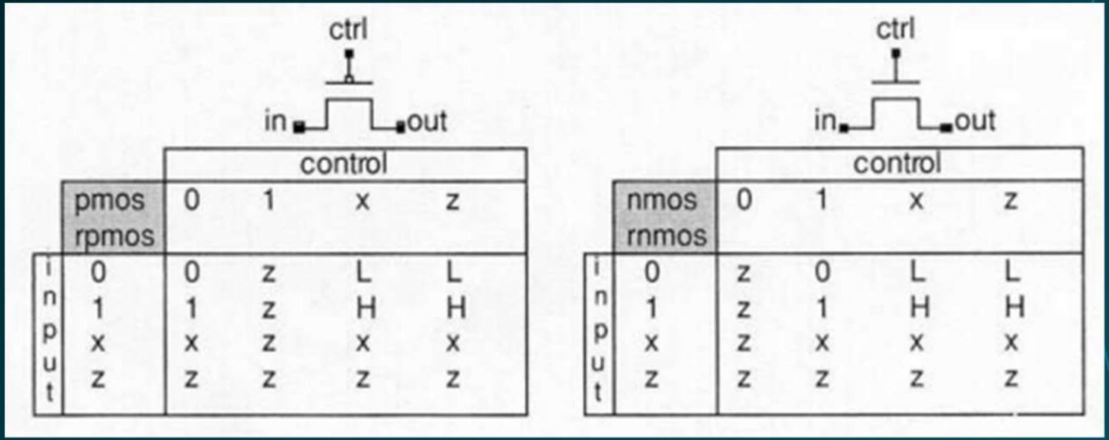
- Switch-level primitive

MOS transistor switches	MOS pull gates	MOS Bi-directional switches
nmos	pullup	tran
pmos	pulldown	tranif0
cmos		tranif1
rnmos		rtran
rpmos		tranif0
rcmos		rtranif1

MOS tranzistori






ctrl
in
out
ctrl
in
out

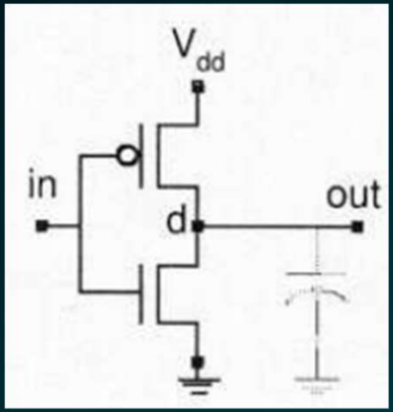
	pmos rpmos	control 0	1	x	z
input	0	0	z	L	L
	1	1	z	H	H
	x	x	z	x	x
	z	z	z	z	z

	nmos rnmos	control 0	1	x	z
input	0	z	0	L	L
	1	z	1	H	H
	x	z	x	x	x
	z	z	z	z	z

Statička CMOS kola

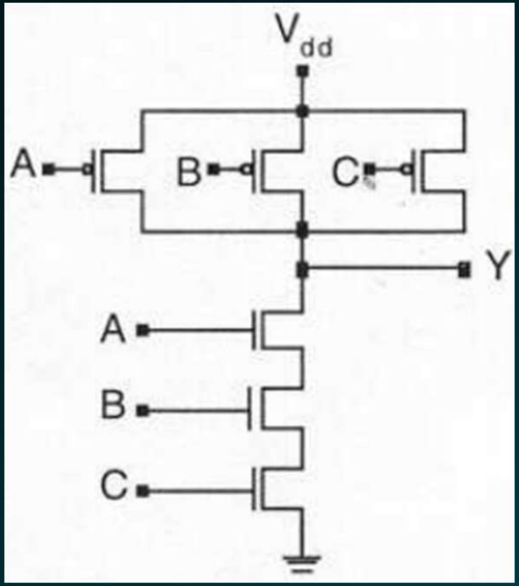
- Pull-up logika implementirana pomoću PMOS tranzistora
- Pull-down logika implementirana pomoću NMOS tranzistora
- PMOS i NMOS tranzistori se uključuju komplementarnim signalima
- Kada je tranzistor u on stanju, ponaša se kao kratak spoj, ili otpornost
- Kada je tranzistor u off stanju ponaša se kao otvorena veza
- Ne postoji put između napajanja i mase, pa je statička disipacija jednaka nuli

Verilog model CMOS invertora



```
module cmos_inv (out, in);
output out;
input in;
supply0 GND;
supply1 PWR;
pmos (out, PWR, in);
nmos (out, GND, in);
endmodule
```

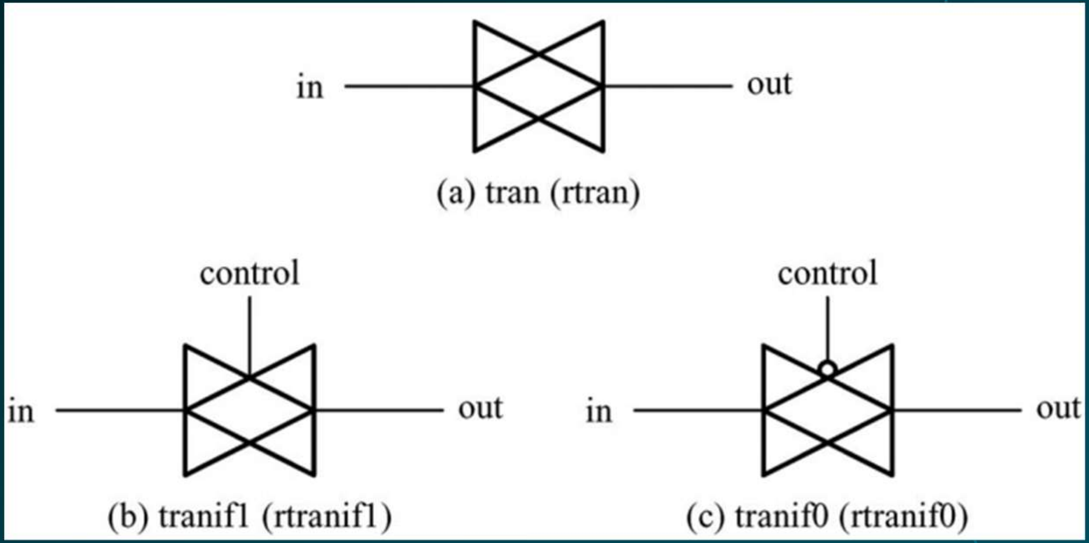
Verilog model 3-ulaznog NI kola



```
module nand_3 (Y, A,B,C);
output Y;
input A, B, C;
supply0 GND;
supply1 PWR;
wire w1,w2;
pmos (Y, PWR, A);
pmos (Y, PWR, B);
pmos (Y, PWR, C);
nmos (Y, w1, A);
nmos (w1, w2, B);
nmos (w2, GND, C);
endmodule
```

Bidirekcioni prekidači

- tran [instance_name] (in, out);
- tranif0 [instance_name] (in, out, control);
- tranif1 [instance_name] (in, out, control);



Kašnjenje : Unidirekcioni prekidači

- Specify no delay :

mos_sw [instance_name](output, input, …);

- Specify propagation delay only :

mos_sw #(prop_delay)[instance_name](output, input, …);

- Specify both rise and fall times :

mos_sw #(t_rise, t_fall)[instance_name](output, input, …);

Specify rise, fall, and turn-off times :

mos_sw #(t_rise, t_fall, t_off)[instance_name](output, input, …);

Kašnjenje : Bidirekcioni prekidači

- Ne unose kašnjenje signala koji prolazi kroz bidirekcioni prekidač. Umesto toga ovi prekidači unose turn-on, ili turn-off kašnjenja
- Specify no delay :

bdsw name [instance name](in, out, control);

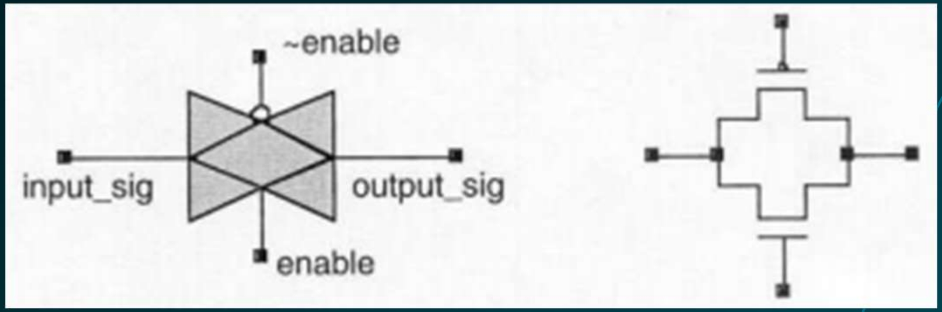
- Specify a turn-on and turn-off delay :

bdsw name #(t_on_off)[instance name](in, out,control);

Specify separately turn-on and turn-off delays :

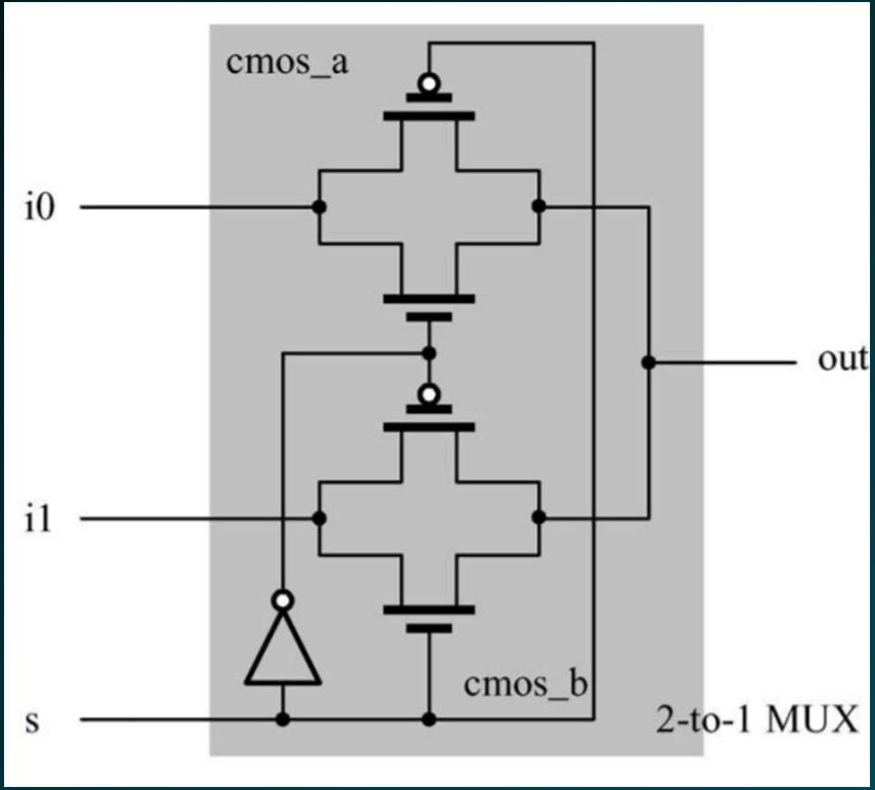
bdsw name #(t_on, t_off)[instance name](in, out, control);

CMOS transmisioni gejt



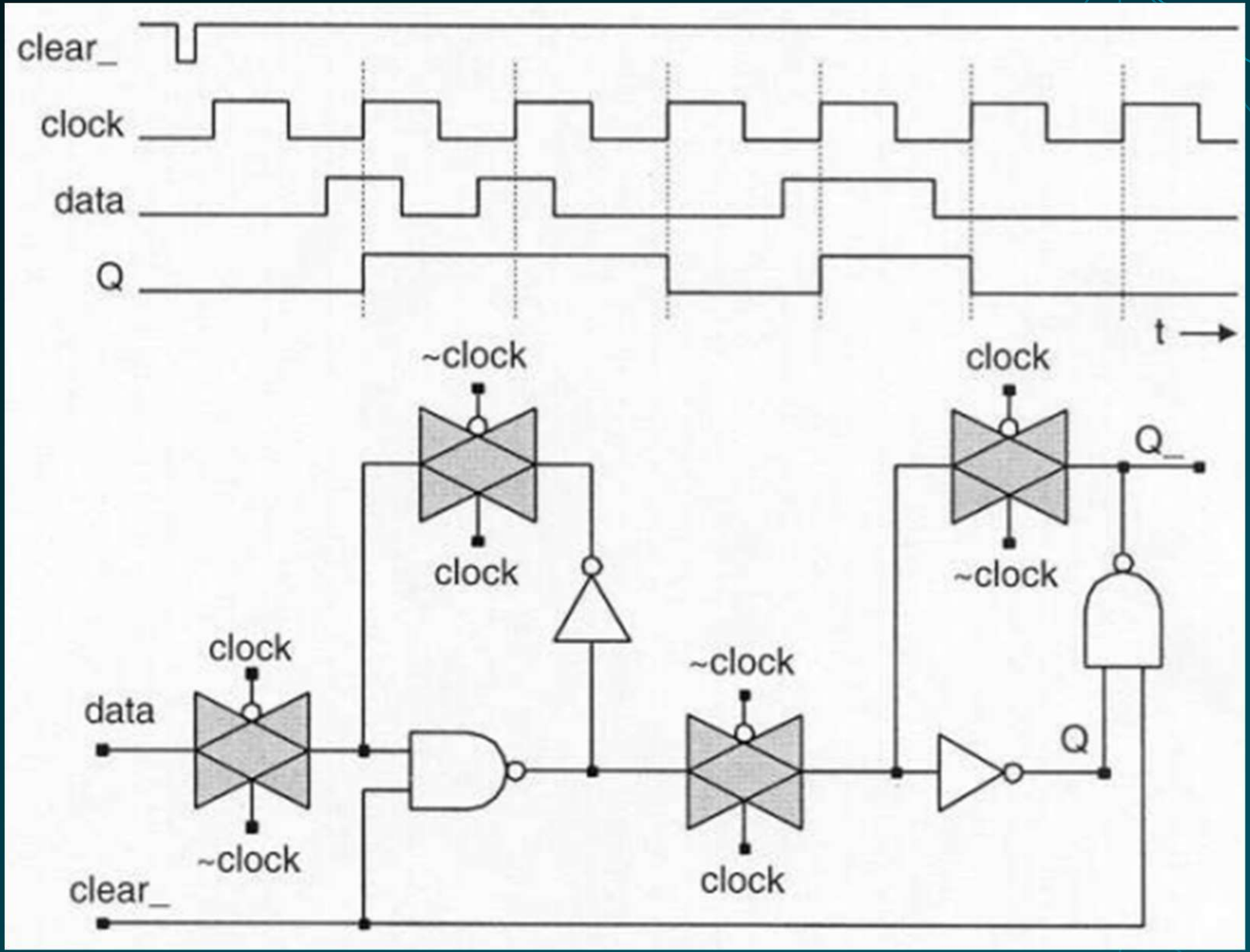
```
module Tgate (data_in, data_out,  n_ena, p_ena);
Input data_in, n_ena, p_ena;
output data_out;
pmos (data_out, data_in, p_ena);
nmos (data_out, data_in, n_ena);
endmodule
```

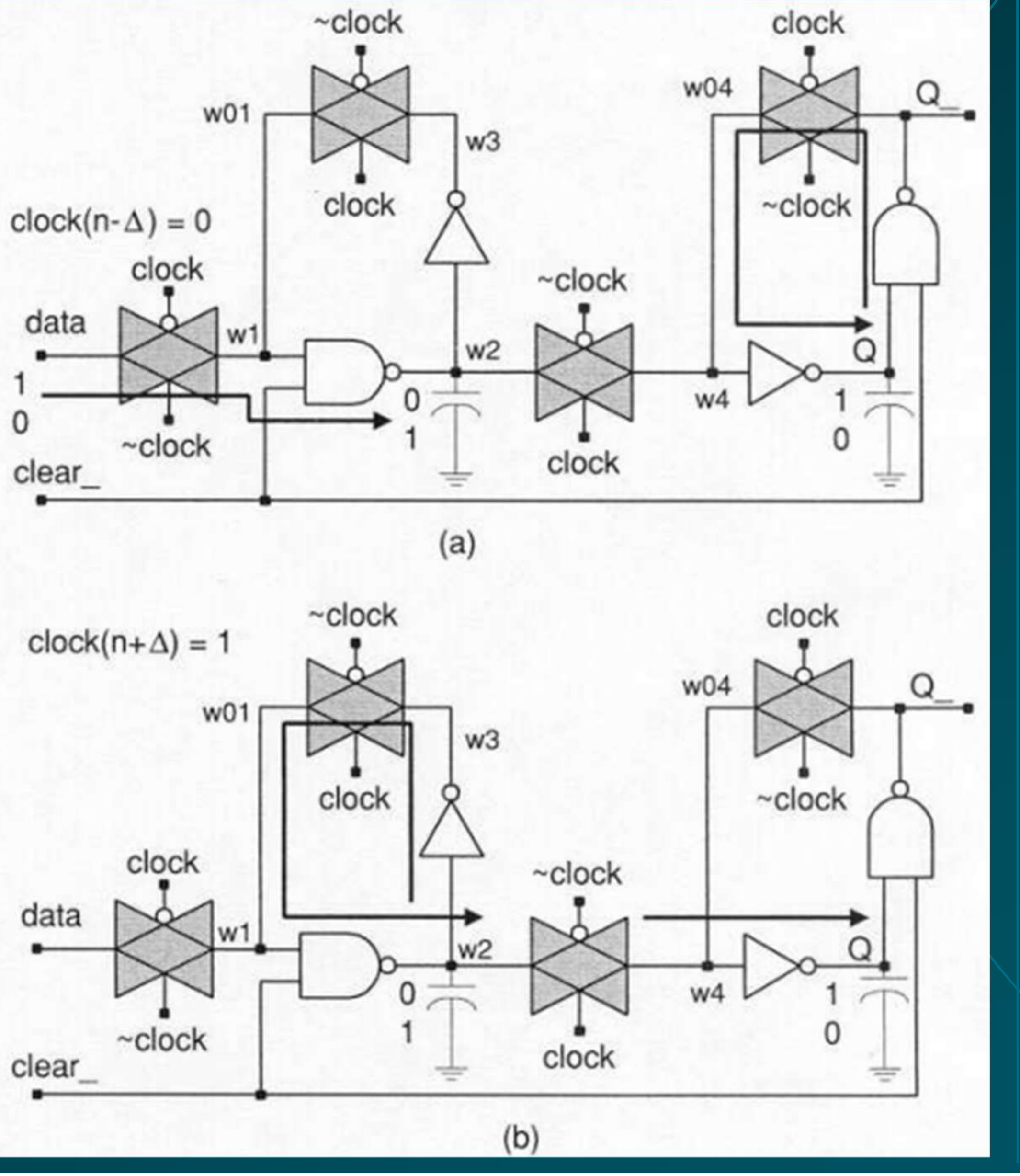
Multiplekser 2 u 1



```
module my_mux21(out,s, i0,i1)
output out;
Input s, i0, i1;
wire sbar;
not (sbar,s);
cmos(out, i0, sbar, s);
cmos(out, i1, s, sbar);
endmodule
```

Master-Slave D FF sa active-low clear-om



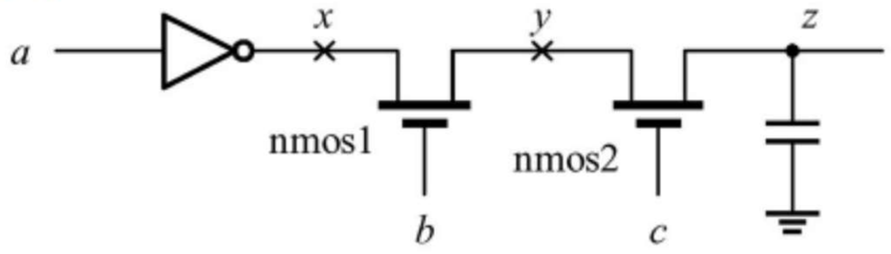
~clock
clock
w01
w3
clock(n-Δ) = 0
clock
data
w1
w2
1
0
~clock
clear_
0
1
~clock
clock
w04
Q_
~clock
Q
w4
1
0
(a)
clock(n+Δ) = 1
~clock
w01
w3
clock
clock
data
w1
w2
~clock
0
1
~clock
clear_
clock
w04
Q_
~clock
Q
w4
1
0
(b)

```
module tgate_dff (Q, Q_,  data, clk, clk_, clear_);
Input data, clk, clk_, clear_;
output Q, Q_;
wire wO1, w1, w2, w3, wO4, W4;
assign w1=wO1;
assign w4=wO4;
cmos (w1, data, clk_, clk);
nand #1 (w2,w1, clear_);
not #1 (w3,w2);
cmos (wO1, w3, clk, clk_);
cmos (w4, w2, clk, clk_);
not #1 (Q, w4);
nand #1 (Q_, Q, clear_);
cmos (wO4, Q_, clk_, clk);
endmodule
```

trireg Nets, states: Driven, Capacitive

At simulation time 0

- a, b, and c = 1
- x = 0
- y -> 0
- z -> driven state and = strong0



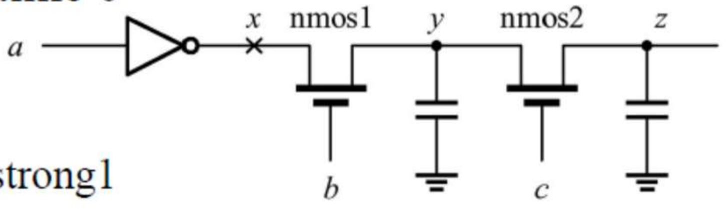
At simulation time 10

- b = 0
- y -> a high-impedance
- z -> capacitive state and = medium0

trireg Nets, states: Driven, Capacitive

At simulation time 0

- a = 0
- b = c = 1
- x, y, and z = strong1



At simulation time 10

- b = 0
- y -> capacitive state and = large1
- z -> driven state and = large1

At simulation time 20

- c = 0
- z -> capacitive state and = small1

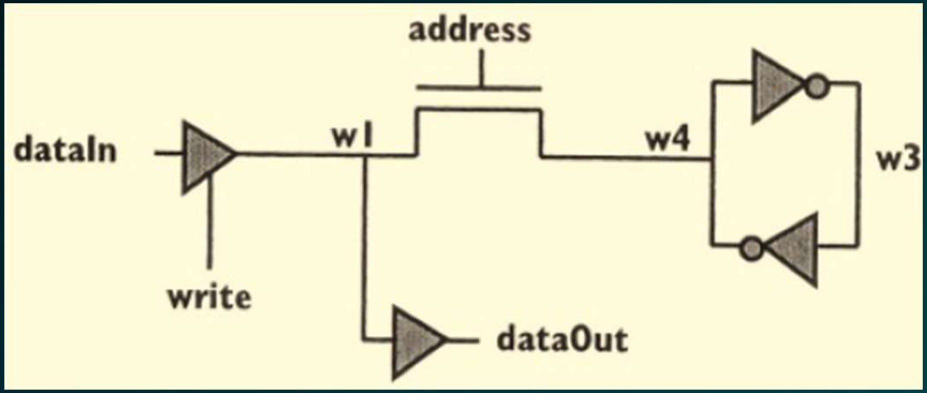
At simulation time 30

- c = 1 again,
- y and z share the charge

At simulation time 40

- c = 0
- z -> capacitive state and = small1

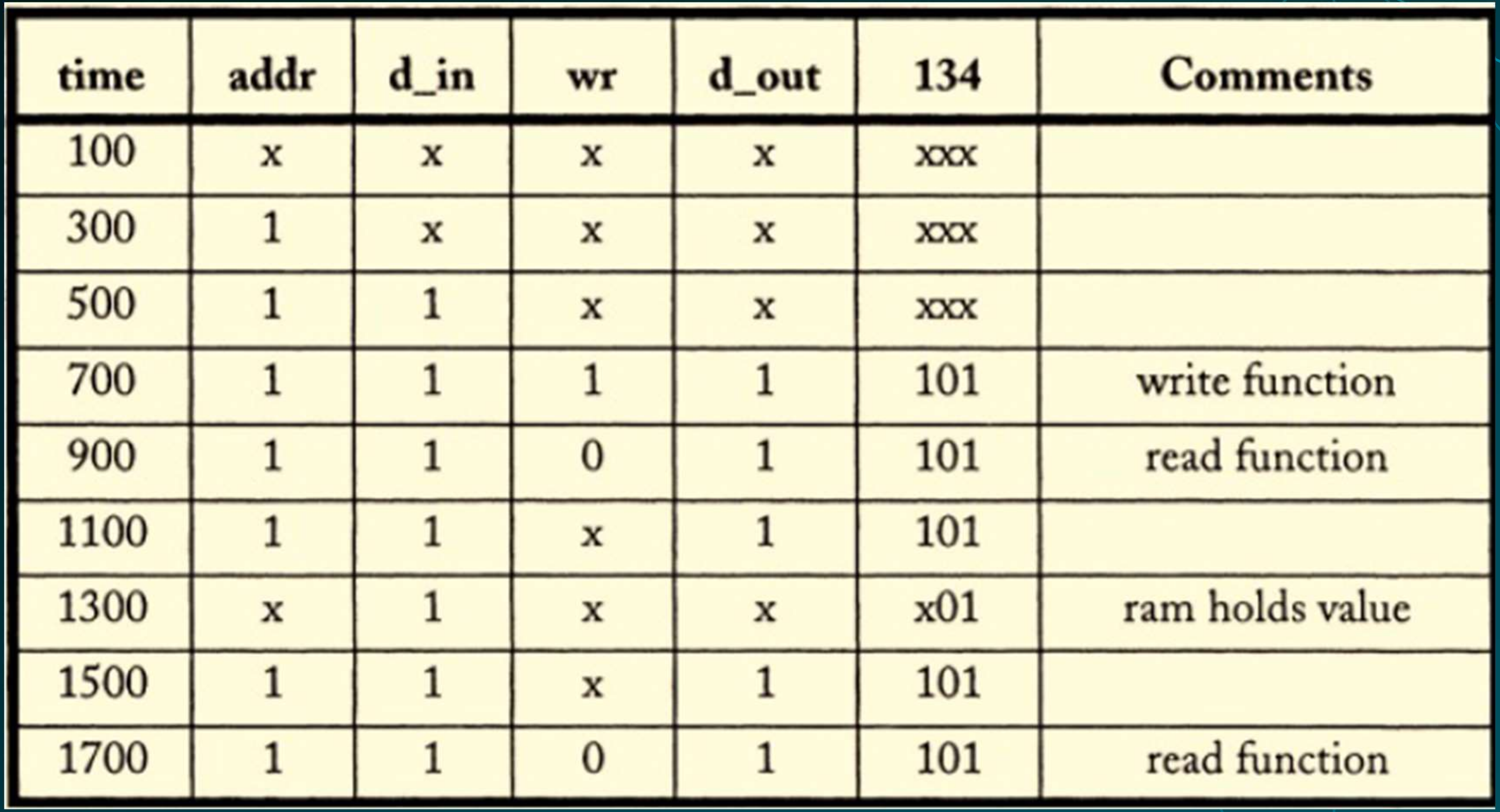
Statička RAM ćelija



```
module sram (dataOut, address, dataIn, write);
output dataOut;
input address, dataIn, write;
tri w1, w3, w4, w43;
bufif1 g1(w1, dataIn, write);
tranif1 g2(w4, w1, address);
not (pull0, pull1) g3(w3, w4), g4(w4, w3);
buf g5(dataOut, wl);
endmodule
```

```
module wave_sram; //waveform for testing the static RAM cell
#(parameter d = 100);
wire dataOut;
reg address, dataIn, write;
sram cell (dataOut, address, dataIn, write);
initial begin
#d dis;
#d address = 1; #d dis;
#d dataIn = 1; #d dis;
#d write = 1; #d dis;
#d write = 0; #d dis;
#d write = 'bx; #d dis;
#d address = 'bx; #d dis;
#d address = 1; #d dis;
#d write = 0; #d dis;
end
task dis; //display the circuit state
"addr=%v d_In=%v write=%v d_out=%v", address, dataIn, write, dataOut, "
(134)=%b%b%b", cell.wl, cell.w3, cell.w4, " w134=%v %v %v", cell.w1, cell.w3,
cell.w4);
endtask
endmodule
```

Testbench-statički RAM



time	addr	d_in	wr	d_out	134	Comments
100	x	x	x	x	xxx	
300	1	x	x	x	xxx	
500	1	1	x	x	xxx	
700	1	1	1	1	101	write function
900	1	1	0	1	101	read function
1100	1	1	x	1	101	
1300	x	1	x	x	x01	ram holds value
1500	1	1	x	1	101	
1700	1	1	0	1	101	read function