

# Verilog HDL: Switch Level Modeling

**Pravin Zode**

# Outline

- Modeling style review
- Switch Level Primitives
- Bidirectional switches
- Resistive Switches
- Examples

# Introduction

- Verilog provides multiple levels of abstraction:
  - Behavioral Modeling
  - Dataflow Modeling
  - Gate-Level Modeling
  - Switch-Level Modeling (Lowest level)
- Switch-level modeling describes circuits at the transistor level using MOS switches.

# Why Use Switch-Level Modeling?

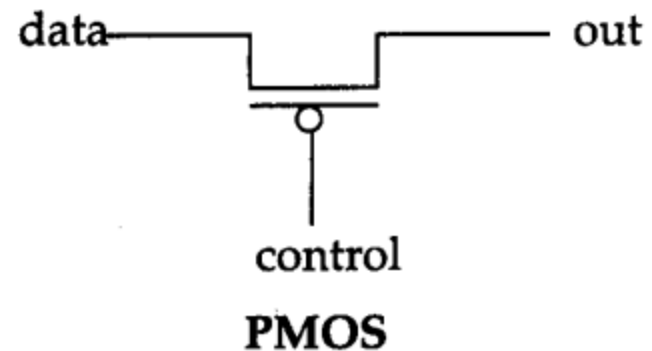
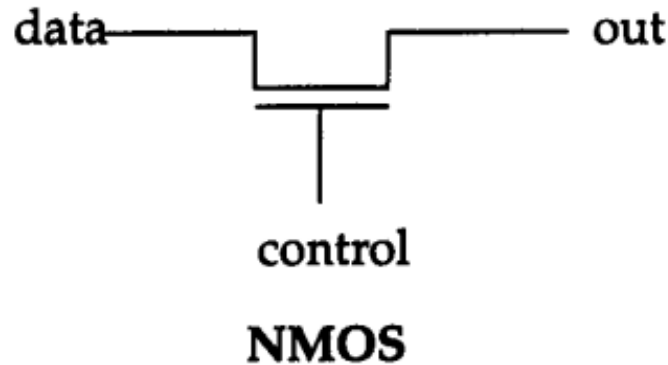
- Provides a detailed view of circuit operation.
- Helps analyze power consumption and delays.
- Useful for low-level circuit debugging.
- Mainly used for custom digital IC design and transistor-level simulations.

# Switch-Level Primitives

- Verilog provides two switch-level primitives:
  - ✓ **`nmos`** – Represents an nMOS transistor.
  - ✓ **`pmos`** – Represents a pMOS transistor.
  - ✓ **`cmos`** – Represents a complementary MOS switch.
  - ✓ **`tran`** – A bidirectional pass switch.
  - ✓ **`tranif0`** and **`tranif1`** – Conditional pass switches.

# Syntax

- `nmos (out, in, control);` // nMOS transistor
- `pmos (out, in, control);` // pMOS transistor
- - nMOS conducts when `control = 1`.
- - pMOS conducts when `control = 0`.



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

# Logic Table (PMOS & NMOS)

- The symbol L stands for 0 or z
- H stands for 1 or z

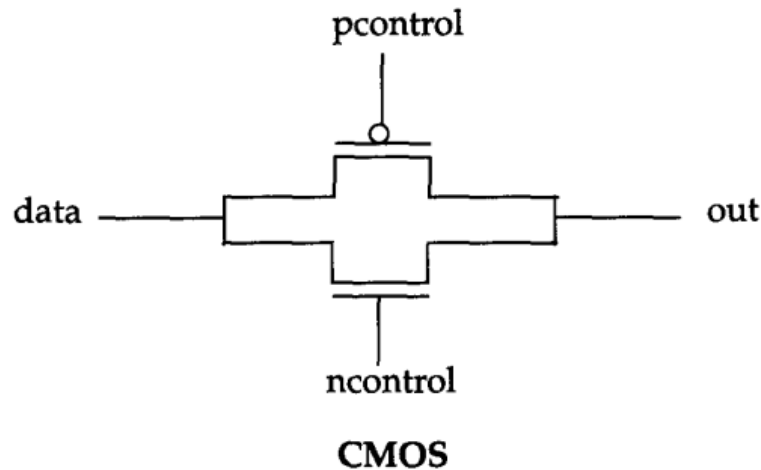
		control			
data	nmos	0	1	x	z
	0	z	0	L	L
	1	z	1	H	H
	x	z	x	x	x
	z	z	z	z	z

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

- NMOS switch conducts when its control signal is 1. If control signal is 0, the output assumes a high impedance value
- PMOS a switch conducts if the control signal is 0

# CMOS Switch

- CMOS device can be modeled with a **nmos** and a **pmos** device
- CMOS switches are declared with the keyword **cmos**

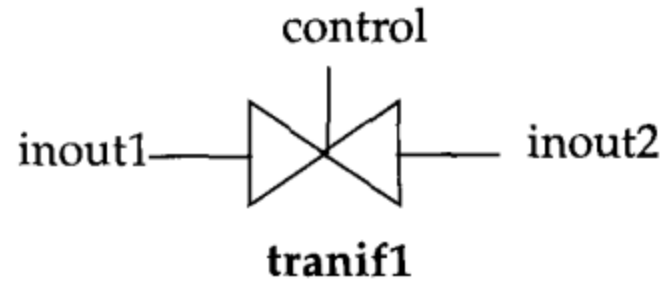
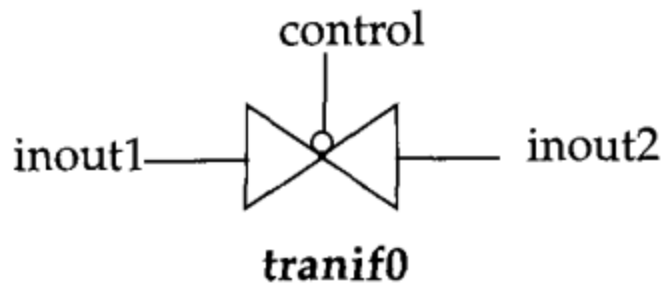
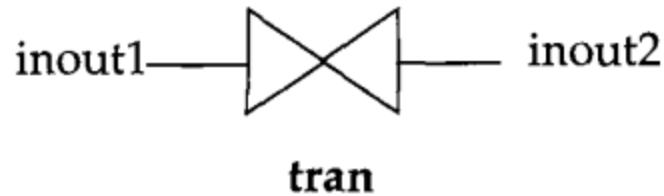


```
cmos c1(out, data, ncontrol, pcontrol); //instantiate cmos gate.  
or  
cmos (out, data, ncontrol, pcontrol); //no instance name given.
```



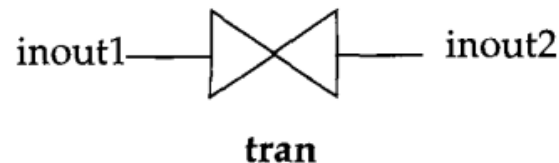
# Bidirectional Switch

- Switch that can conduct in both directions
- Signal on either side can be driver signal
- Bidirectional switches : tran, tranif0, tranif1

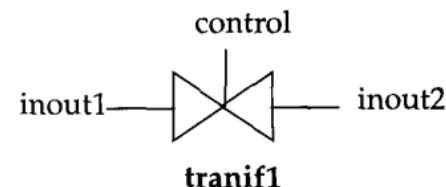
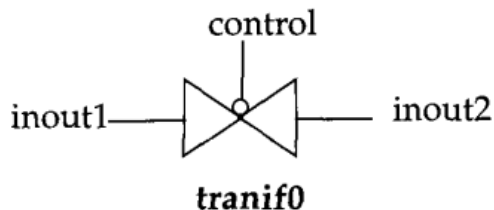


# Bidirectional Switch

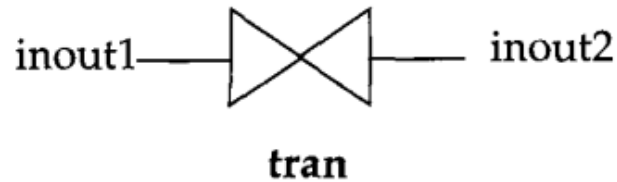
- The tran switch acts as a buffer between the two signals **inout1** and **inout2**. Either inout1 or inout2 can be the driver signal



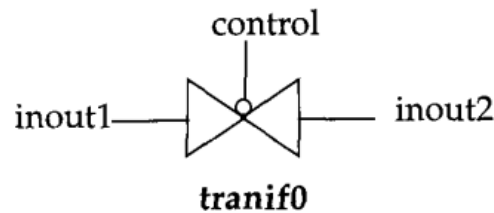
- tranif0** switch connects if the control signal is logical 0. If the control signal is 1, the nondriver signal gets a high impedance
- The driver signal retains value from its driver
- The **tranif1** switch conducts if the control signal is a 1



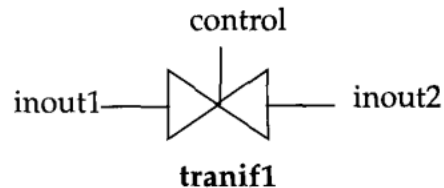
# Syntax : Bidirectional Switch



```
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

- Power and Ground sources are essential in transistor-level designs
- Verilog uses `supply1` and `supply0` keywords to represent power (Vdd) and ground (Vss), respectively.
- **supply1:**
  - Represents power (Vdd)
  - Continuously drives a logical 1 onto a net during simulation
- **supply0:**
  - Represents ground (Vss)
  - Continuously drives a logical 0 onto a net during simulation.
- These sources help in biasing transistors and ensuring correct functionality in low-level circuit modeling.

# Power and Ground

```
supply1 vdd;  
supply0 gnd;
```

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

# Resistive Switches

- MOS, CMOS, and bidirectional switches can be modeled as resistive devices in Verilog
- Resistive switches **simulate higher impedance** between source and drain compared to regular switches
- They are useful for modeling weaker signal paths or realistic analog behavior
- Verilog provides resistive switch primitives by prefixing "r" to the corresponding regular switch keywords:
  - nmos → rnmos
  - pmos → rpmos
  - cmos → rcmos
  - tran → rtran
  - tranif1 → rtranif1

# Difference : Regular & Resistive Switches

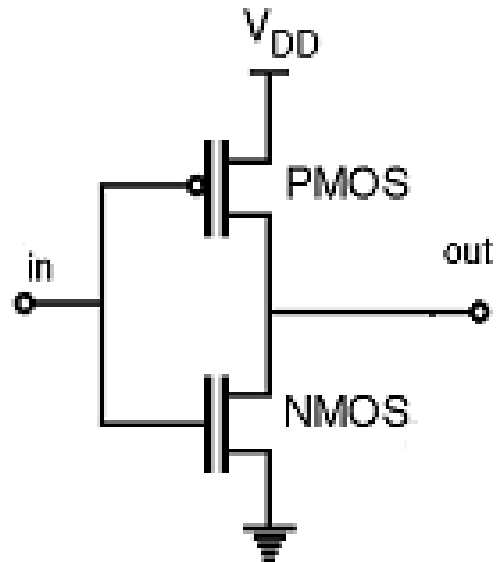
- There are two key differences between regular and resistive switches: Source-to-drain impedance & Signal strength propagation
- Regular switches:
  - Have low source-to-drain impedance
  - Pass signals without reducing their strength
  - Retain input signal strength at the output (except when input is of supply strength, which becomes strong at the output)
- Resistive switches:
  - Have high source-to-drain impedance
  - Reduce signal strength when signals pass through
  - Cause signal degradation, simulating real-world resistance behavior

# Delay Specification

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);

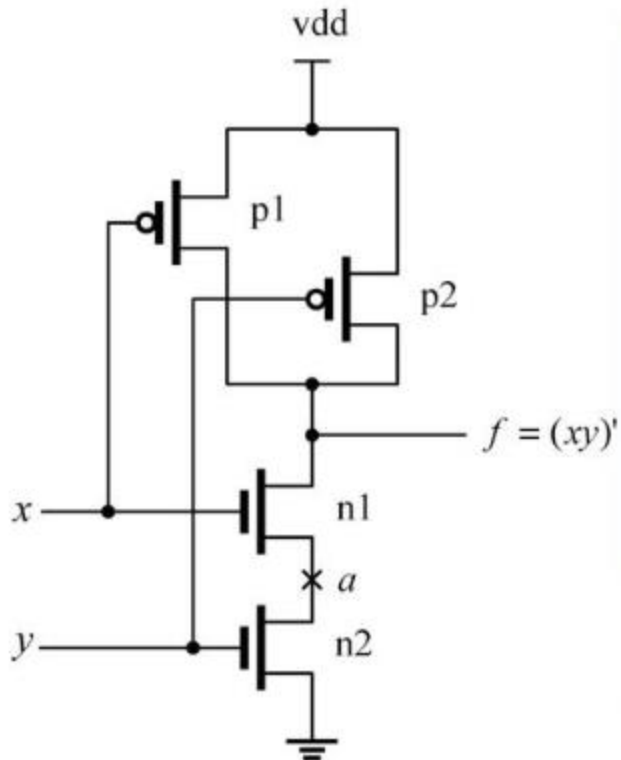


# Example : CMOS Inverter

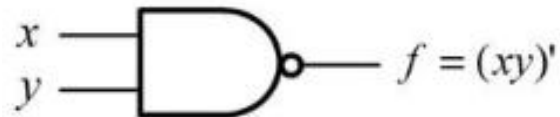


```
1 module inverter(input in, output out);  
2     wire w;  
3     supply1 Vdd;  
4     supply0 Gnd;  
5     pmos (out, Vdd, in); // pMOS pulls output HIGH when input is LOW  
6     nmos (out, Gnd, in); // nMOS pulls output LOW when input is HIGH  
7 endmodule
```

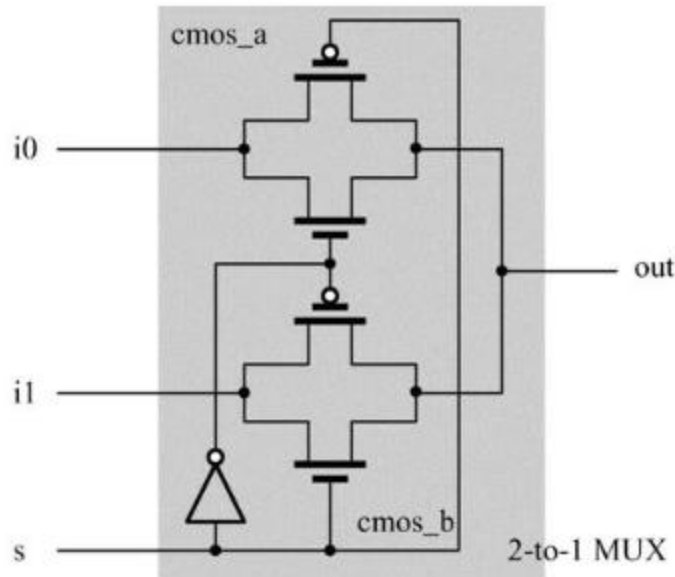
# Example : NAND Gate



```
module my_nand (input x, y, output f);  
  supply1 vdd;  
  supply0 gnd;  
  wire a;  
  // NAND gate body  
  pmos p1 (f, vdd, x);  
  pmos p2 (f, vdd, y);  
  nmos n1 (f, a, x);  
  nmos n2 (a, gnd, y);  
endmodule
```



# Example : 2:1 Mux



```
module my_mux (out, s, i0, i1);  
  output out;  
  input  s, i0, i1;  
  //internal wire  
  wire sbar; //complement of s  
  
  not (sbar, s);  
  //instantiate cmos switches  
  cmos (out, i0, sbar, s);  
  cmos (out, i1, s, sbar);  
endmodule
```



**Thank you !**

**Happy Learning**