Switch-Level Models

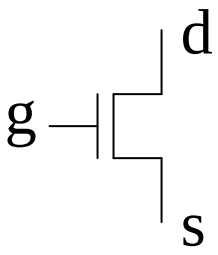
Primitives Supported by Verilog

COMBINATIONAL LOGIC GATES			SWITCH-LEVEL PRIMITIVES		
Multi-input gates	Multi-output gates	Three-state gates	MOS transistor switches	MOS Pull gates	MOS Bi-directional switches
and	buf	bufif0	nmos	pullup	tran
nand	not	bufif1	pmos	pulldown	tranif0
or		notif0	cmos		tranif1
nor		notif0	rnmos		rtran
xor			rpmos		rtranif0
xnor			rcmos		rtranif1

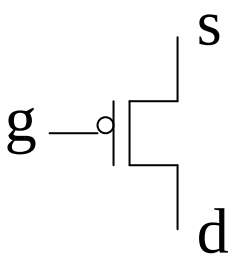
notif1

MOS Transistors

- n-channel (nmos) transistor

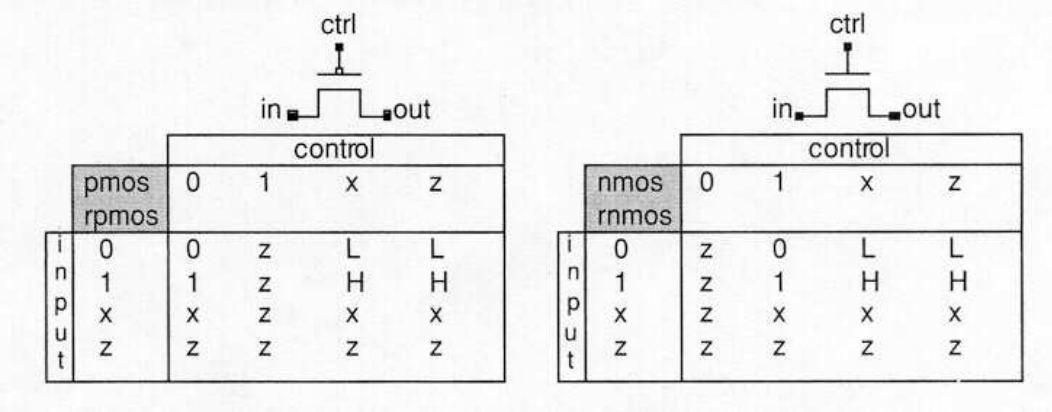


- p-channel (pmos) transistor



Primitives for MOS Transistors

- **nmos**, **pmos**
- **rnmos**, **rpmos** (resistive version)
- Terminal list: (drain, source, gate) (i.e., (output, input, control))
- Input-output relationship
  - L: 0 or z
  - H: 1 or z

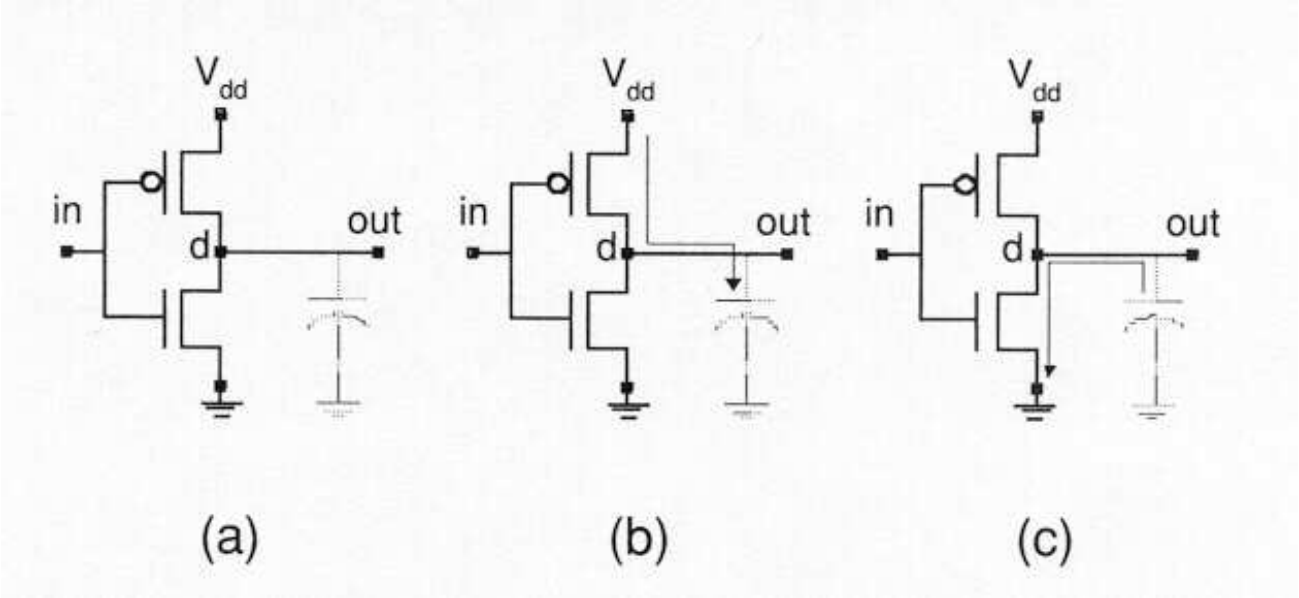


Static CMOS Circuits

- Implement pull-up logic by p-channel transistors
- Implement pull-down logic by n-channel transistors
- p-channel and n-channel transistors are turned on by complementary signal values
- When a transistors is
  - on: very low source-to-drain resistance (short circuit)
  - off: very high source-to-drain resistance (open circuit)
- No DC path between power and ground
  - static power dissipation is 0
  - power is consumed during switching of the input signal

Example: Inverter

(a) CMOS circuit, (b) in = 0, (c) in = 1



Verilog Model for CMOS Inverter

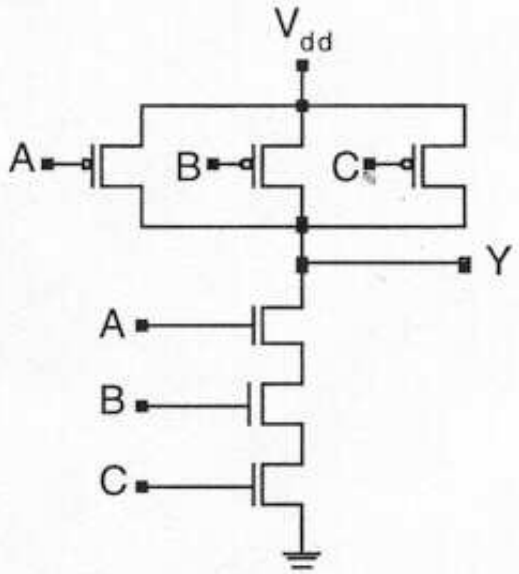
```
module cmos_inverter (inv_out, inv_in);
    output inv_out;
    input inv_in;
    supply0 GND;
    supply1 PWR;

    pmos (inv_out, PWR, inv_in);
    nmos (inv_out, GND, inv_in);
endmodule
```

Example: 3-input nand Gate

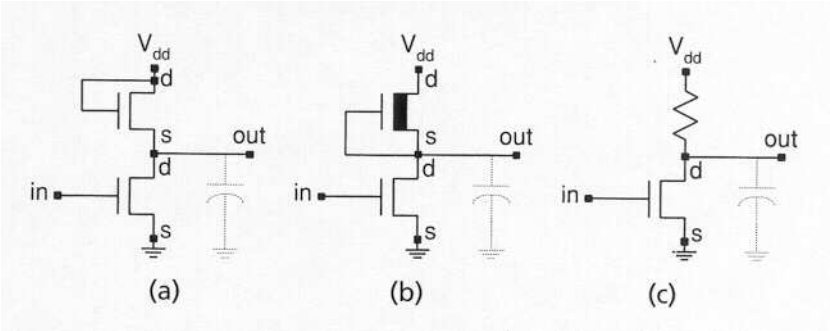
```
module nand_3 (Y, A, B, C);
  output      Y;
  input       A, B, C;
  supply0     GND;
  supply1     PWR;
  wire        w1, w2;

  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
```



Alternative Loads

- Silicon area can be reduced by replacing pull-up logic by a single transistor which acts like a resistor
  - fabricated with a single type of transistors
  - penalized by DC power consumption
- Example (using nmos)
  - enhancement-mode: gate connected to drain
  - depletion mode: gate connected to source
  - (a) enhancement-mode, (b) depletion-mode, (c) resistive pull-up



Primitives for MOS Pull Gates

- **pullup**, **pulldown**
- Both have a single port element which is to be pulled to 1 or 0
- Pull-up (pull-down) gate has a predefined strength of **pull1** (**pull0**)
- Synthesis tools sometimes use pull-up and pull-down gates to tie unused set/rest inputs on flip-flops
- Do not get confused with **tri0** and **tri1**

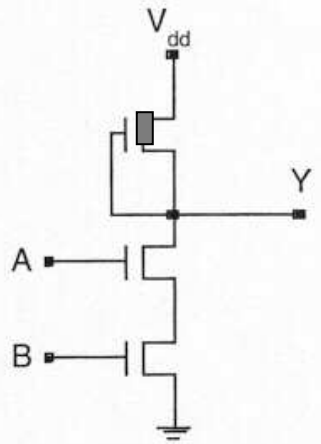
Example: 2-input nand Gate

- nmos pull-down logic + depletion load pull-up device

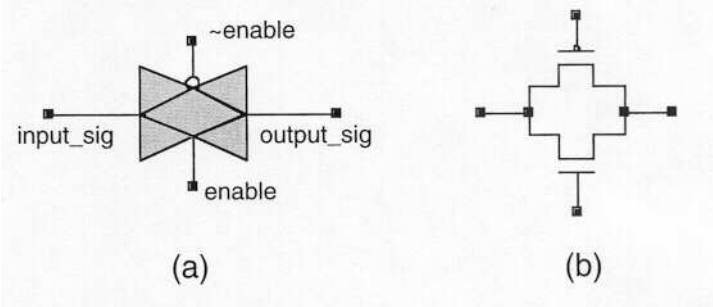
should be **wire**

```
module nmos_nand_2 (Y, A, B);
  output      Y;
  input       A, B;
  supply0     GND;
  tri         w1;

  pullup      (Y);
  nmos        (Y, w1, A);
  nmos        (w1, GND, B);
endmodule
```



CMOS Transmission Gates



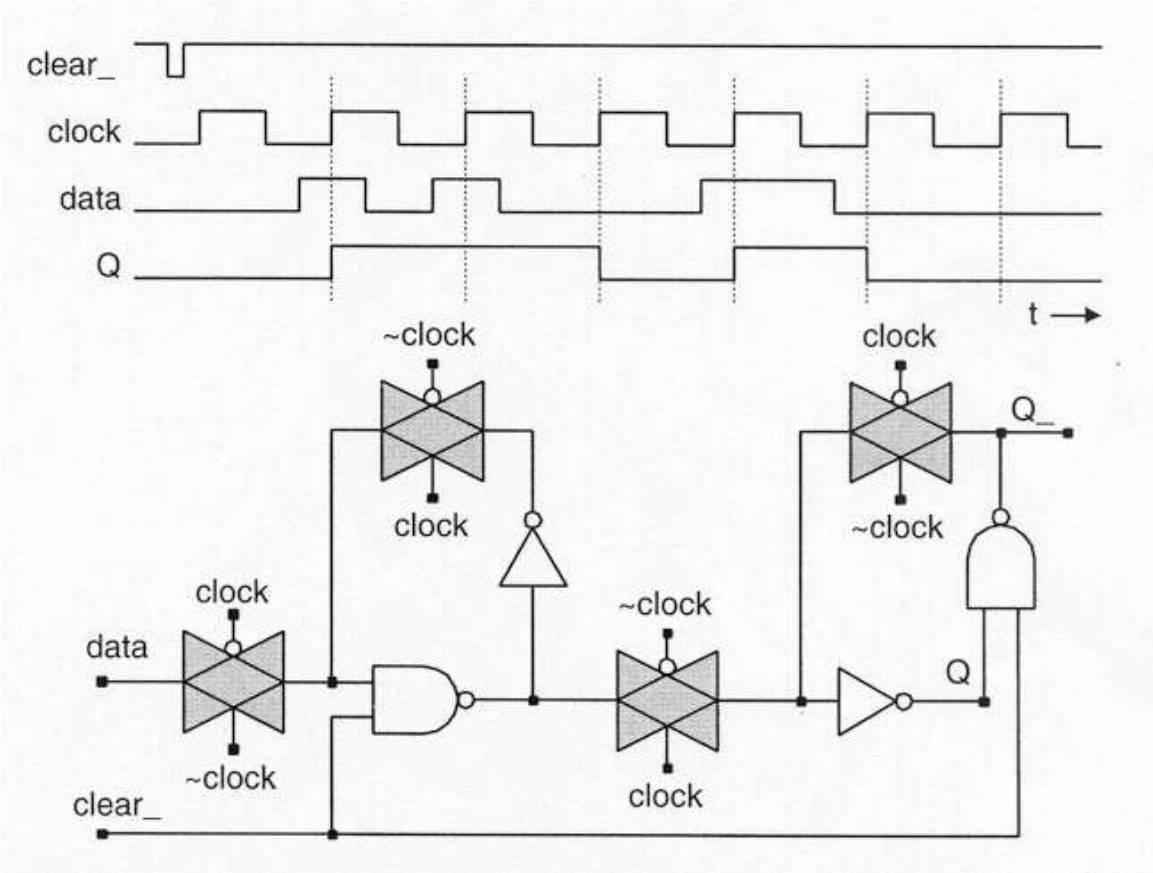
- Primitives: **cmos, rcmos** (resistive version)
- Terminal list: (output, input, enable, ~enable)
- Example (transmission gate implemented by **nmos** and **pmos** primitives)

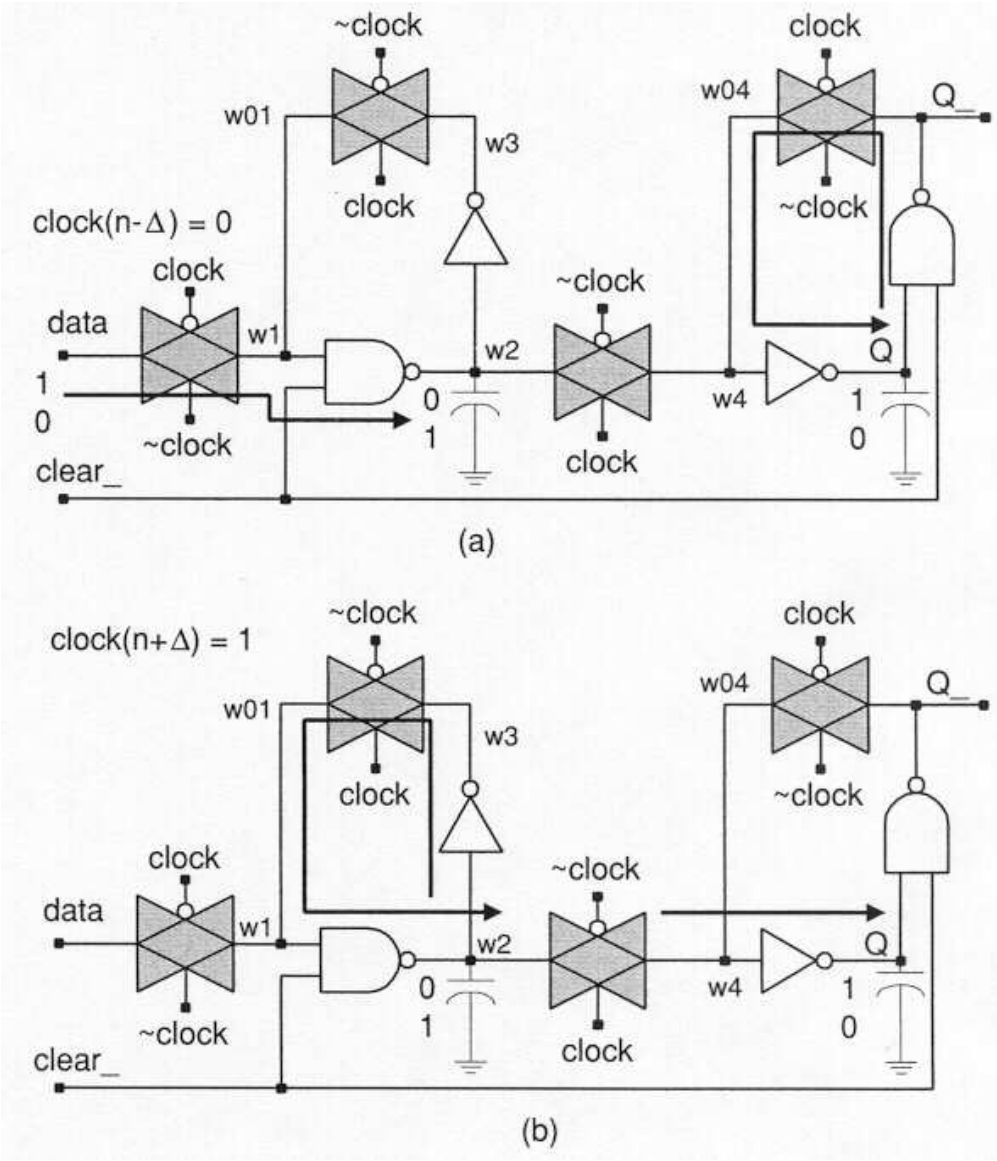
```
module Tgate_str (data_in, data_out, n_enable, p_enable);
   input data_in, n_enable, p_enable;
   output data_out;

   pmos (data_out, data_in, p_enable);
   nmos (data_out, data_in, n_enable);
endmodule
```

Example: Master-Slave D flip-flop with active-low clear

- Use transmission gates and logic gates




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

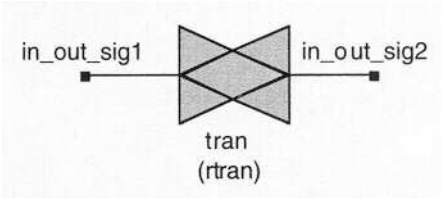
Verilog Model

```
module tgate_dflop (Q, Q_ , data, clock, clock_, clear_);
    input data, clock, clock_, clear_ ;
    output Q, Q_;
    wire w01, w1, w2, w3, w04,w4;
    assign w1 = w01;
    assign w4 = w04;
    cmos  (w1, data, clock_, clock);
    nand #1 (w2, w1, clear_);
    not #1 (w3, w2);
    cmos  (w01, w3, clock, clock_);
    cmos  (w04, w2, clock, clock_);
    not #1 (Q, w4);
    nand #1 (Q_, Q, clear_);
    cmos  (w4, Q_ , clock_, clock);
endmodule
```

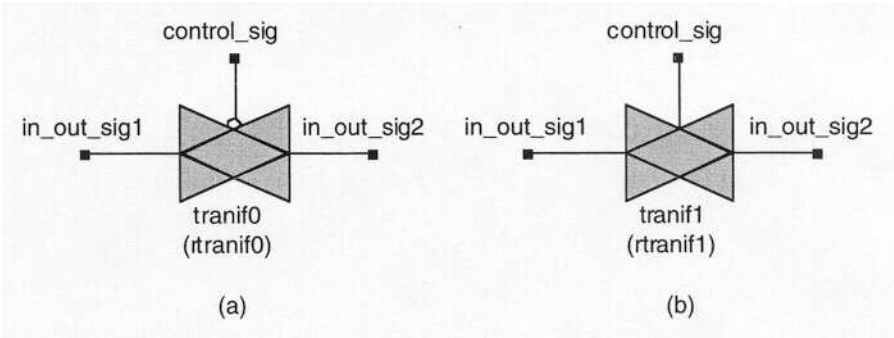
should be exchanged

Bi-Directional Switches

- 2-terminal bi-directional switches
  - primitives: **tran**, **rtran** (resistive version)



  - one terminal may be declared as **input** or **inout**, and the other as **inout** or **output**
- 3-terminal bi-directional switches
  - primitives: **tranif0**, **tranif1**, **rtranif0** (resistive version), **rtranif1** (resistive version)

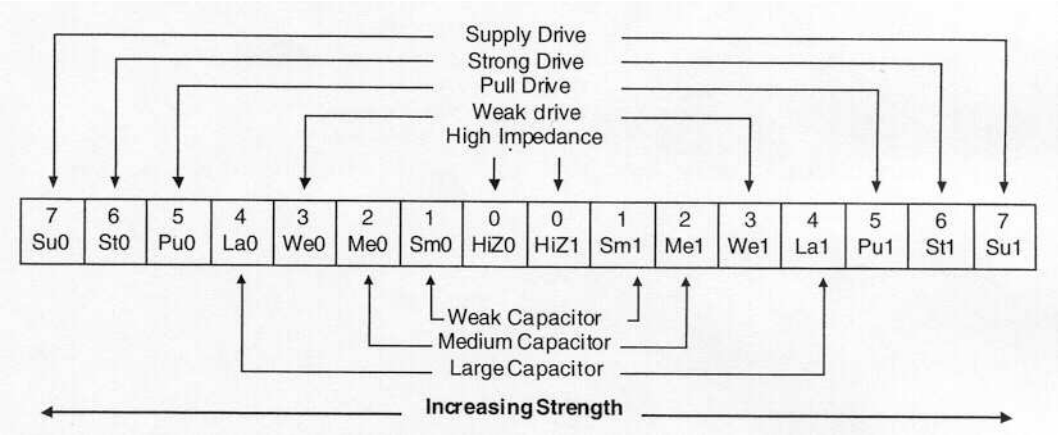


  - the third terminal is a control input

Signal Strengths

- Verilog's logic system has logic values and logic strengths
- The strength of a signal refers to the ability to act as a driver determining the resultant logic value on a net
  - gives additional information that determines the result of contending drivers on a net when multiple drives are present, or when the nets are modeled as charge storage capacitors
- The signals in gate-level models are "strong" by default

Strength Diagram



- **supply0**, **strong0**, **pull0**, **weak0**, **highz0**, **supply1**, **strong1**, **pull1**, **weak1**, **highz1** may be assigned only to nets that are outputs of continuous assignments, combinational gates, or pull gates
- **large0**, **medium0**, **small0**, **large1**, **medium1**, **small1** are "charge storage" strengths for nets of type **trireg**
- The strength of the output signal of a MOS transistor switch or bi-directional switch depends upon the switch type and the strength of the input signal (see page 25)

Strength Specifications

- Two parts: strength for 0, and strength for 1
- When the logic value is known (0 or 1), the strength is one of the strengths (1, 2, ...7)
  - The integer associated with a strength indicates relative strength, with 7 being the highest
- When the logic value is z, the strength may be HiZ0 or HiZ1
- When the logic value is ambiguous (x), the strength may be in both sides of the scale
- Use “%v” in **$display** or **$monitor** to see the strength of a signal

Driven Nets

- Combinational logic primitives or pull gates
  - gate_instantiation ::= [GATE_TYPE] [drive_strength] [delay] gate_instance {, gate_instance};
  - GATE_TYPE: combinational logic gate or pull gate
  - drive_strength: an unordered pair with one value from {**supply0**, **strong0**, **pull0**, **weak0**, **highz0**} and the other from {**supply1**, **strong1**, **pull1**, **weak1**, **highz1**}
- Continuous assignments
  - continuous_assign ::= **assign** [drive_strength] [delay] list_of_net_assignments;
- Only scalar nets may receive strength assignments
- Examples
  - **nand** (**pull1**, **strong0**) a (out, in1, in2);
  - **wire** (**pull0**, **weak1**) b = c & d;
  - **assign** (**pull1**, **weak0**) e = f;

Supply Nets

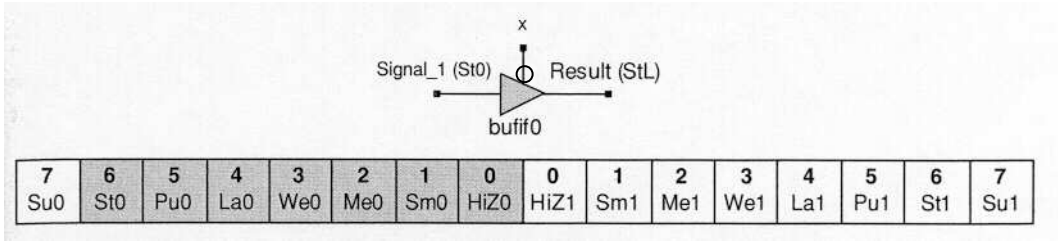
- Nets of **supply0** (**supply1)** have predefined strength of **supply0** (**supply1**)
- When a **tri0** (**tri1**) net is not driven, it has the strength of **pull0** (**pull1**)

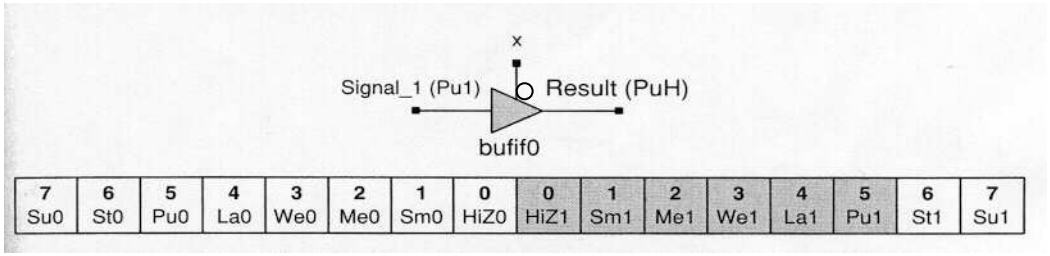
Charge Storage Nets

- net_declaration ::= **trireg** [**vectored** | **scalared**] [charge_strength] [range] list_of_net_identifiers
  - charge_strength: **small**, **medium** (default), or **large** capacitor

Ambiguous Signals

- When the strength of a signal is ambiguous, it is a member of a set of two or more strengths, but the exact strength is not known
  - a signal may have a range of strengths
- Examples



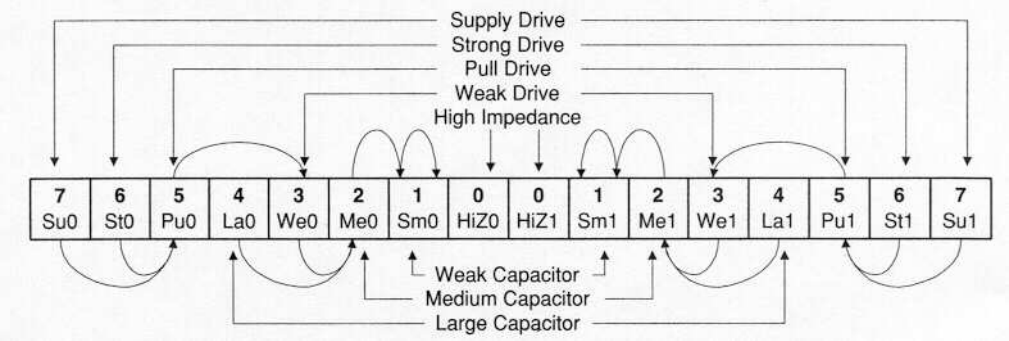


Strength Reduction by Primitives

- For a combinational primitive (except three-state gates) or pull primitive, the strength of the output is independent of the strength of the input
- For a MOS transistor switch or MOS bi-directional transistor switch, the strength of the output depends upon the the strength of the input and the switch type

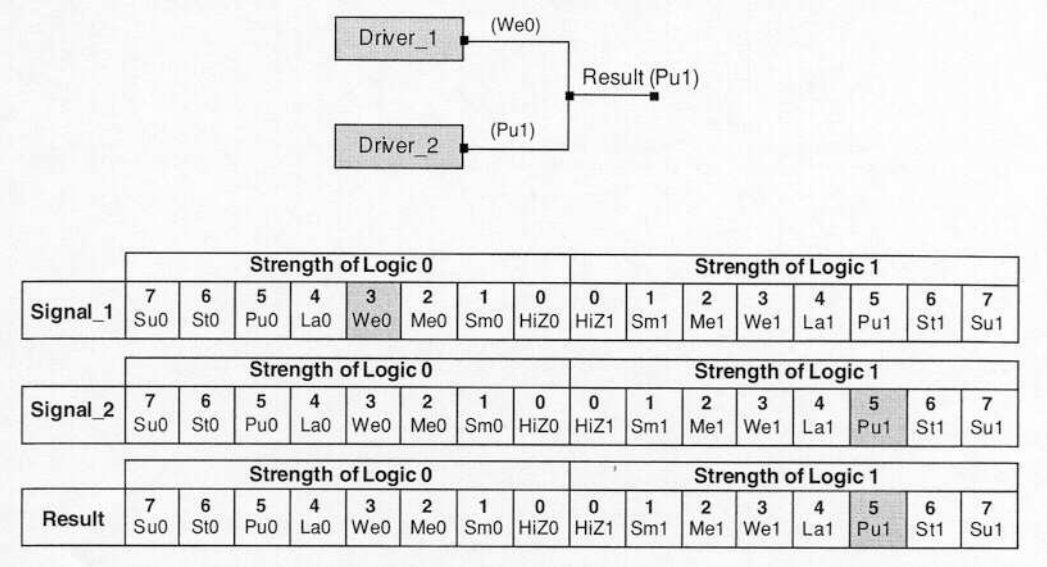
Transistor Switches and Bi-Directional Switches

- For **nmos**, **pmos**, **cmos**, **tran**, **tranif0**, or **tranif1**
  - if data input signal has strength **supply0** (**supply1**), then the strength of the output is **strong0** (**strong1**)
  - otherwise, the strength of the output remains the same as that of the input
- For a resistive switch (**rnmos**, **rpmos**, **rcmos**, **rtran**, **rtranif0**, or **rtranif1**), use the following rule



Known Strengths and Known Values

- When a net is driven by signals having known values and known strengths, the one having the greatest strength dominates
- Example



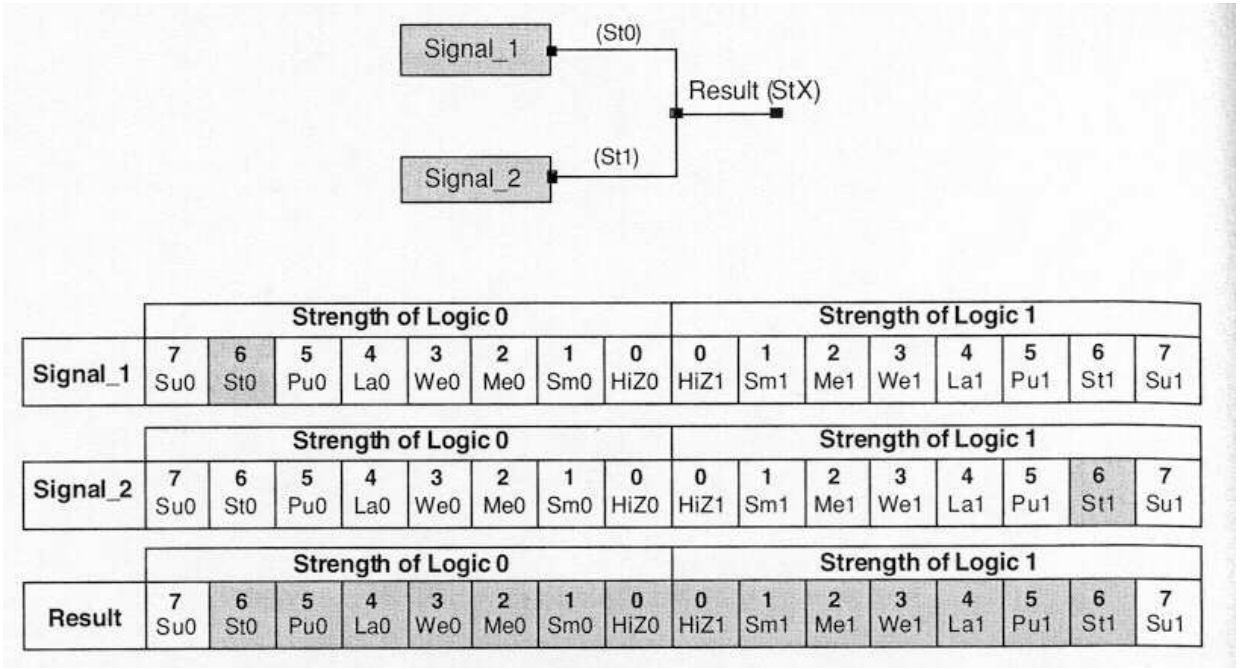
	Strength of Logic 0								Strength of Logic 1							
Signal_1	7 Su0	6 St0	5 Pu0	4 La0	3 We0	2 Me0	1 Sm0	0 HiZ0	0 HiZ1	1 Sm1	2 Me1	3 We1	4 La1	5 Pu1	6 St1	7 Su1
Signal_2	7 Su0	6 St0	5 Pu0	4 La0	3 We0	2 Me0	1 Sm0	0 HiZ0	0 HiZ1	1 Sm1	2 Me1	3 We1	4 La1	5 Pu1	6 St1	7 Su1
Result	7 Su0	6 St0	5 Pu0	4 La0	3 We0	2 Me0	1 Sm0	0 HiZ0	0 HiZ1	1 Sm1	2 Me1	3 We1	4 La1	5 Pu1	6 St1	7 Su1

Known Strengths and Known Values

- When a net is driven by signals having same value and identical strength, the net has that value and strength
- When a net is driven by signals having same value, the net has that value and the strength of the greatest of the drivers
- When a net is driven by signals having same strength but different values, the value of the net depends upon the net type
  - if the type is **wand**, **wor**, **triand** or **trior**, the value is determined by the rules associated with the net type (see page 36)
  - otherwise, the value is x, and the strength is the range of all strengths between the greatest strength of each driver

Example

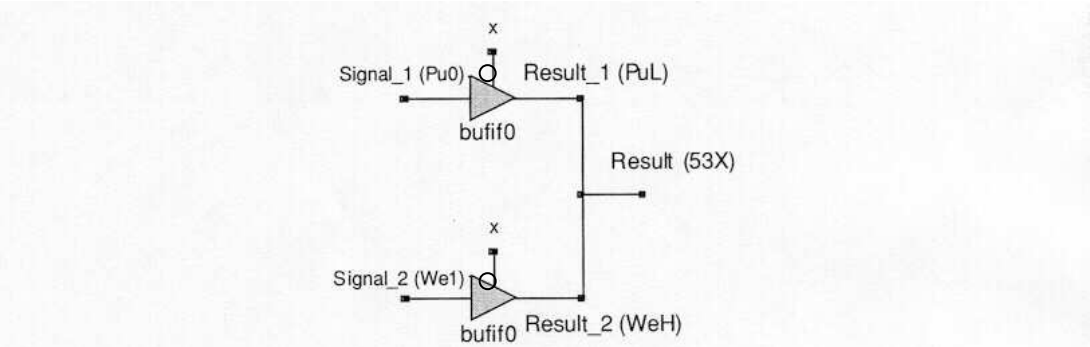
- Same strength and opposing values in two drivers => ambiguous value and strength in the driven net



	Strength of Logic 0								Strength of Logic 1							
Signal_1	7 Su0	6 St0	5 Pu0	4 La0	3 We0	2 Me0	1 Sm0	0 HiZ0	0 HiZ1	1 Sm1	2 Me1	3 We1	4 La1	5 Pu1	6 St1	7 Su1
Signal_2	7 Su0	6 St0	5 Pu0	4 La0	3 We0	2 Me0	1 Sm0	0 HiZ0	0 HiZ1	1 Sm1	2 Me1	3 We1	4 La1	5 Pu1	6 St1	7 Su1
Result	7 Su0	6 St0	5 Pu0	4 La0	3 We0	2 Me0	1 Sm0	0 HiZ0	0 HiZ1	1 Sm1	2 Me1	3 We1	4 La1	5 Pu1	6 St1	7 Su1

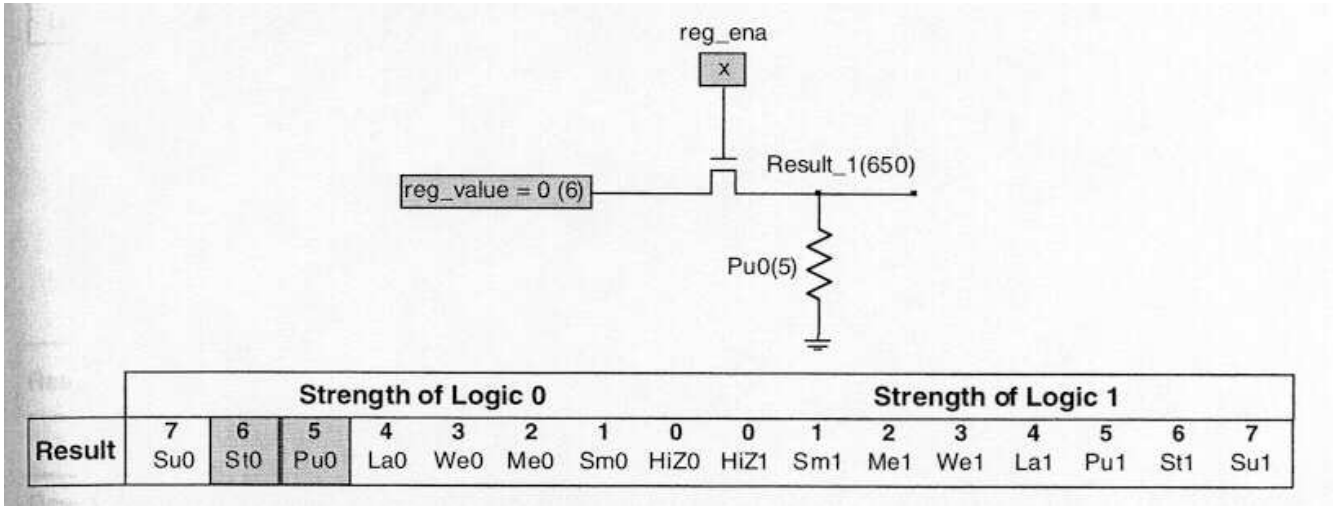
Ambiguous Strengths

- When two signals having ambiguous strengths drive the same net, the resultant strength of the net is also ambiguous and has the range including all of the strengths of the drivers
- Example



	Strength of Logic 0								Strength of Logic 1							
Result_1	7 Su0	6 St0	5 Pu0	4 La0	3 We0	2 Me0	1 Sm0	0 HiZ0	0 HiZ1	1 Sm1	2 Me1	3 We1	4 La1	5 Pu1	6 St1	7 Su1
Result_2	7 **Su0**	6 **St0**	5 **Pu0**	4 **La0**	3 **We0**	2 **Me0**	1 **Sm0**	0 **HiZ0**	0 **HiZ1**	1 **Sm1**	2 **Me1**	3 **We1**	4 **La1**	5 **Pu1**	6 **St1**	7 **Su1**
Result	7 Su0	6 St0	5 Pu0	4 La0	3 We0	2 Me0	1 Sm0	0 HiZ0	0 HiZ1	1 Sm1	2 Me1	3 We1	4 La1	5 Pu1	6 St1	7 Su1

Sub-Example

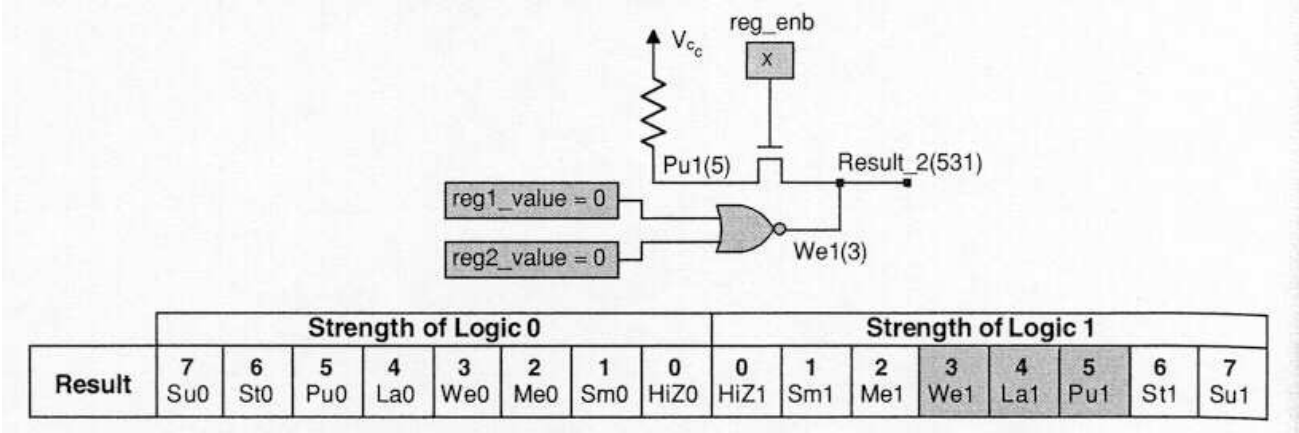


	Strength of Logic 0								Strength of Logic 1							
Result	7 Su0	6 St0	5 Pu0	4 La0	3 We0	2 Me0	1 Sm0	0 HiZ0	0 HiZ1	1 Sm1	2 Me1	3 We1	4 La1	5 Pu1	6 St1	7 Su1

```
reg reg_ena, reg_value;

wire Result_1;
pulldown (Result_1);
nmos (Result_1, reg_value, reg_ena);
```

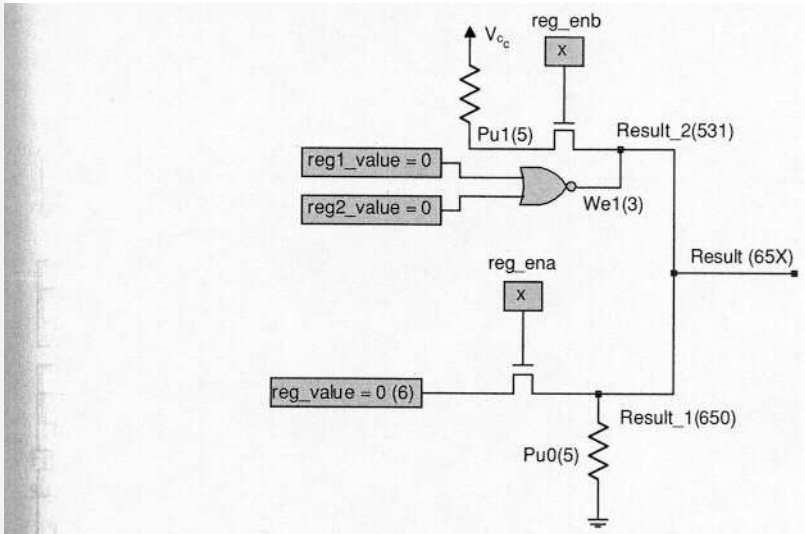
Sub-Example



	Strength of Logic 0								Strength of Logic 1							
Result	7 Su0	6 St0	5 Pu0	4 La0	3 We0	2 Me0	1 Sm0	0 HiZ0	0 HiZ1	1 Sm1	2 Me1	3 We1	4 La1	5 Pu1	6 St1	7 Su1

```
reg reg1_value, reg2_value, reg_enb;
pullup (Vcc);
wire Result_2;
nmos (Result_2, Vcc, reg_enb);
nor (strong0, weak1) (Result_2, reg1_value, reg2_value);
```

Complete Example




```
module sw3 (Result_1, Result_2, Result);
   output Result_1, Result_2, Result;
   wire Result_1, Result_2;
   reg reg1_value, reg2_value, reg_enb;
   reg reg_ena, reg_value;
   pullup (Vcc);
   pulldown (Result_1);
   nmos (Result_2, Vcc, reg_enb);
   nmos (Result_1, reg_value, reg_ena);
   nor (strong0, weak1) (Result_2, reg1_value, reg2_value);
   assign Result = Result_2;
   assign Result = Result_1;
endmodule
```

	Strength of Logic 0								Strength of Logic 1							
Result_2	7 Su0	6 St0	5 Pu0	4 La0	3 We0	2 Me0	1 Sm0	0 HiZ0	0 HiZ1	1 Sm1	2 Me1	3 We1	4 La1	5 Pu1	6 St1	7 Su1

	Strength of Logic 0								Strength of Logic 1							
Result_1	7 Su0	6 St0	5 Pu0	4 La0	3 We0	2 Me0	1 Sm0	0 HiZ0	0 HiZ1	1 Sm1	2 Me1	3 We1	4 La1	5 Pu1	6 St1	7 Su1

	Strength of Logic 0								Strength of Logic 1							
Result	7 Su0	6 St0	5 Pu0	4 La0	3 We0	2 Me0	1 Sm0	0 HiZ0	0 HiZ1	1 Sm1	2 Me1	3 We1	4 La1	5 Pu1	6 St1	7 Su1

Combining a Signal Having Known Strength and Known Value with a Signal Having Ambiguous Strength

- Step 1: the result includes the strengths of the ambiguous signal that are greater than or equal to the strength of the unambiguous signal
- Step 2: if the unambiguous signal and the ambiguous signal now have different logic values, the result includes the strength of the unambiguous signal and the intermediate values in the gap between the strengths taken from Step 1 and the strength of the unambiguous signal

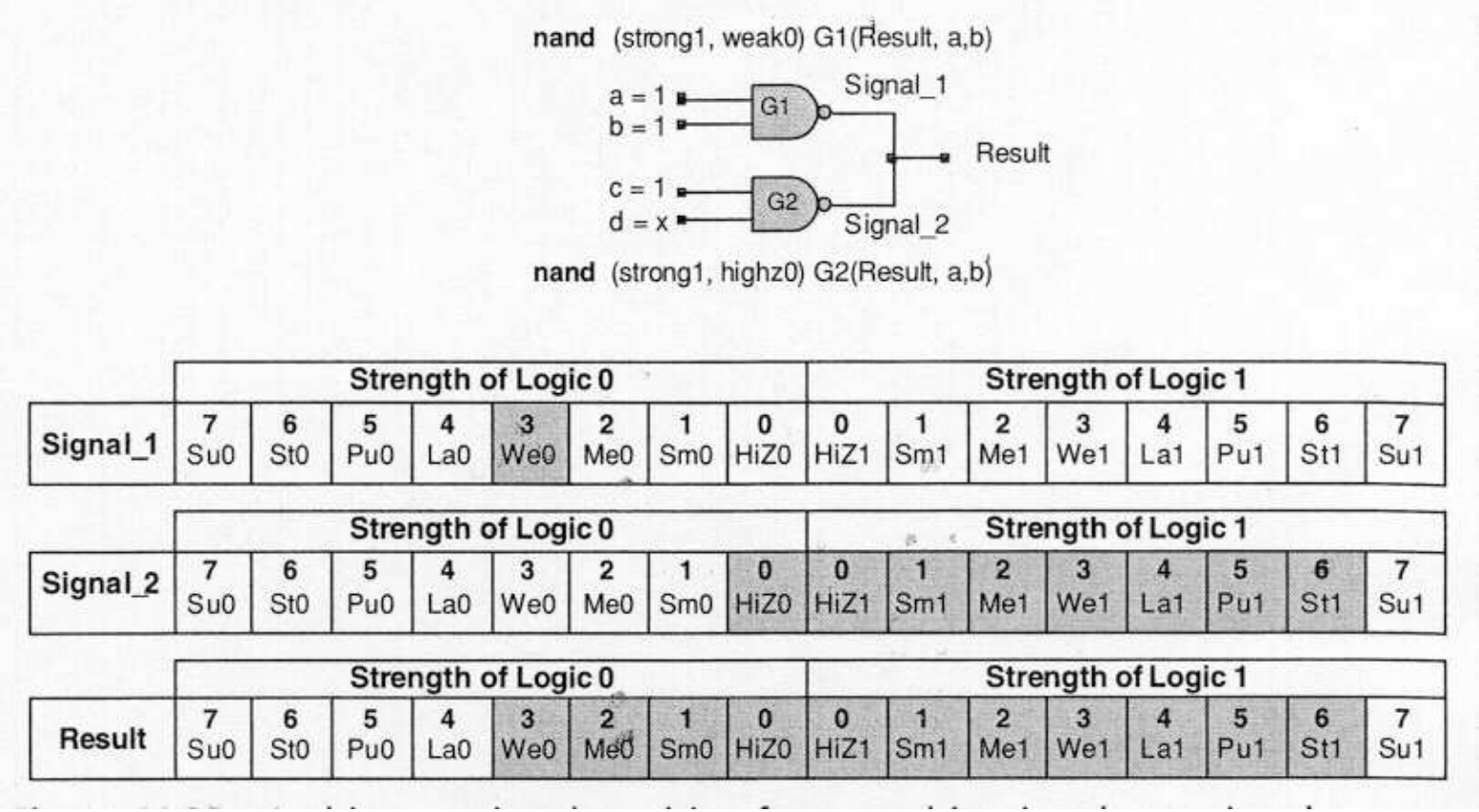
Examples

	Strength of Logic 0								Strength of Logic 1							
Signal_1	7 Su0	6 St0	5 Pu0	4 La0	3 We0	2 Me0	1 Sm0	0 HiZ0	0 HiZ1	1 Sm1	2 Me1	**3 We1**	4 La1	5 Pu1	6 St1	7 Su1
Signal_2	7 Su0	6 St0	5 Pu0	4 La0	3 We0	2 Me0	1 Sm0	**0 HiZ0**	0 HiZ1	1 Sm1	2 Me1	3 We1	4 La1	5 Pu1	**6 St1**	7 Su1
Result	7 Su0	6 St0	5 Pu0	4 La0	3 We0	2 Me0	1 Sm0	0 HiZ0	0 HiZ1	1 Sm1	2 Me1	**3 We1**	4 La1	5 Pu1	6 St1	7 Su1

	Strength of Logic 0								Strength of Logic 1							
Signal_1	7 Su0	**6 St0**	5 Pu0	4 La0	3 We0	2 Me0	1 Sm0	0 HiZ0	0 HiZ1	1 Sm1	2 Me1	3 We1	4 La1	5 Pu1	6 St1	7 Su1
Signal_2	7 Su0	6 St0	5 Pu0	**4 La0**	3 We0	2 Me0	1 Sm0	0 HiZ0	0 HiZ1	1 Sm1	2 Me1	3 We1	4 La1	5 Pu1	6 St1	7 Su1
Result	7 Su0	**6 St0**	5 Pu0	**4 La0**	3 We0	2 Me0	1 Sm0	0 HiZ0	0 HiZ1	1 Sm1	2 Me1	3 We1	4 La1	5 Pu1	6 St1	7 Su1

	Strength of Logic 0								Strength of Logic 1							
Signal_1	7 Su0	**6 St0**	5 Pu0	4 La0	3 We0	2 Me0	1 Sm0	0 HiZ0	0 HiZ1	1 Sm1	2 Me1	3 We1	4 La1	5 Pu1	6 St1	7 Su1
Signal_2	7 Su0	6 St0	5 Pu0	4 La0	3 We0	2 Me0	1 Sm0	0 HiZ0	0 HiZ1	1 Sm1	2 Me1	**3 We1**	4 La1	5 Pu1	6 St1	7 Su1
Result	7 Su0	6 St0	5 Pu0	4 La0	3 We0	2 Me0	1 Sm0	0 HiZ0	0 HiZ1	1 Sm1	2 Me1	3 We1	4 La1	5 Pu1	6 St1	7 Su1

Example



	Strength of Logic 0								Strength of Logic 1							
Signal_1	7 Su0	6 St0	5 Pu0	4 La0	3 We0	2 Me0	1 Sm0	0 HiZ0	0 HiZ1	1 Sm1	2 Me1	3 We1	4 La1	5 Pu1	6 St1	7 Su1

	Strength of Logic 0								Strength of Logic 1							
Signal_2	7 Su0	6 St0	5 Pu0	4 La0	3 We0	2 Me0	1 Sm0	0 HiZ0	0 HiZ1	1 Sm1	2 Me1	3 We1	4 La1	5 Pu1	6 St1	7 Su1

	Strength of Logic 0								Strength of Logic 1							
Result	7 Su0	6 St0	5 Pu0	4 La0	3 We0	2 Me0	1 Sm0	0 HiZ0	0 HiZ1	1 Sm1	2 Me1	3 We1	4 La1	5 Pu1	6 St1	7 Su1

Signal Strengths and Wired Logic

- Nets of type **wand**, **wor**, **triand**, and **trior** automatically resolve contention between drivers having the same strength
  - **wand** and **triand** (**wor** and **trior**) produce a logic value 0 (1) if any driver is 0 (1)
- Example (wired-and)

	Strength of Logic 0								Strength of Logic 1							
Signal_1	7 Su0	6 St0	5 Pu0	4 La0	3 We0	2 Me0	1 Sm0	0 HiZ0	0 HiZ1	1 Sm1	2 Me1	3 We1	4 La1	5 Pu1	6 St1	7 Su1
Signal_2	7 Su0	6 St0	5 Pu0	4 La0	3 We0	2 Me0	1 Sm0	0 HiZ0	0 HiZ1	1 Sm1	2 Me1	3 We1	4 La1	5 Pu1	6 St1	7 Su1
Result	7 Su0	6 St0	5 Pu0	4 La0	3 We0	2 Me0	1 Sm0	0 HiZ0	0 HiZ1	1 Sm1	2 Me1	3 We1	4 La1	5 Pu1	6 St1	7 Su1

Signal Strengths and Wired Logic

- When a net of type **wand**, **wor**, **triand**, or **trior** is driven by drivers having different strengths, the simulator considers all possible combinations of strengths between drivers and retains the dominant results, while taking into account the predefined wire behavior of the net
- Example

	Strength of Logic 0								Strength of Logic 1							
Signal_1	7 Su0	6 St0	5 Pu0	4 La0	3 We0	2 Me0	1 Sm0	0 HiZ0	0 HiZ1	1 Sm1	2 Me1	3 We1	4 La1	5 Pu1	6 St1	7 Su1
Signal_2	7 Su0	6 St0	5 Pu0	4 La0	3 We0	2 Me0	1 Sm0	0 HiZ0	0 HiZ1	1 Sm1	2 Me1	3 We1	4 La1	5 Pu1	6 St1	7 Su1
wand Result	7 Su0	6 St0	5 Pu0	4 La0	3 We0	2 Me0	1 Sm0	0 HiZ0	0 HiZ1	1 Sm1	2 Me1	3 We1	4 La1	5 Pu1	6 St1	7 Su1
wor Result	7 Su0	6 St0	5 Pu0	4 La0	3 We0	2 Me0	1 Sm0	0 HiZ0	0 HiZ1	1 Sm1	2 Me1	3 We1	4 La1	5 Pu1	6 St1	7 Su1