Section 5

Gate Level Modeling

Verilog Primitives

- Verilog primitives are basic gates built into the Verilog language.
- Reference designators are not required for Verilog primitives.
- Some of the "built-in" primitives:

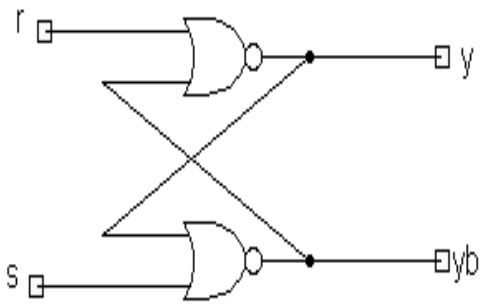
Gate Type	Functionality
and	Logical AND
or	Logical OR
not	Inverter
bufif1	Conditional buffer (enabled high);
…	…

Verilog has many "built-in" gate level primitives:

and
nand
or
nor
xor
xnor
buf
not
bufif1
bufif0
notif1
notif0

Structural (Gate Level) Modeling

- A structural model is equivalent to a schematic diagram.
- A structural model is created from existing components including Verilog "built-in" primitives, pre-defined modules, or user defined primitives.

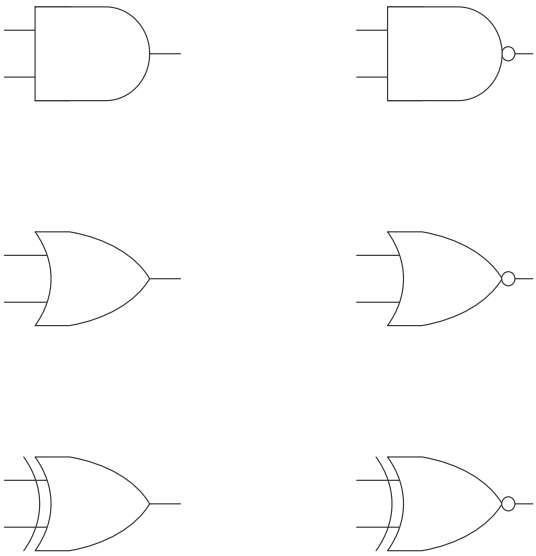



```
module rs_lat (y, yb, r, s);
input r, s;
output y, yb;

nor u1 (y, r, yb);
nor u2 (yb, s, y);

endmodule
```

Gate Types – and/or



- Verilog has a set of and/or type primitives.
- and/or gates have multiple scalar inputs and one scalar output.

```
module andorexmp (andout, nandout, orout,
                  norout, xorout, xnorout,
                  a, b);

  output andout, nandout, orout, norout,
         xorout, xnorout;
  input a, b;

  and (andout, a, b);
  nand (nandout, a, b);
  or (orout, a, b);
  nor (norout, a, b);
  xor (xorout, a, b);
  xnor (xnorout, a, b);

endmodule
```


and

or

Truth Tables – and/or

b \ a	*and* 0	1	x	z
0	0	0	0	0
1	0	1	x	x
x	0	x	x	x
z	0	x	x	x

b \ a	*or* 0	1	x	z
0	0	1	x	x
1	1	1	1	1
x	x	1	x	x
z	x	1	x	x

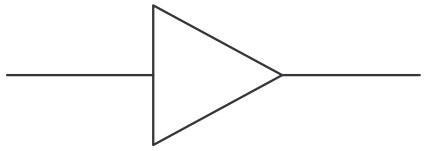
b \ a	*xor* 0	1	x	z
0	0	1	x	x
1	1	0	x	x
x	x	x	x	x
z	x	x	x	x

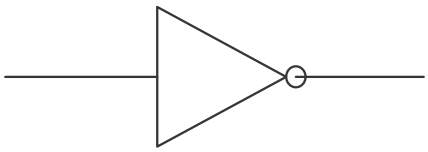
b \ a	*nand* 0	1	x	z
0	1	1	1	1
1	1	0	x	x
x	1	x	x	x
z	1	x	x	x

b \ a	*nor* 0	1	x	z
0	1	0	x	x
1	0	0	0	0
x	x	0	x	x
z	x	0	x	x

b \ a	*xnor* 0	1	x	z
0	1	0	x	x
1	0	1	x	x
x	x	x	x	x
z	x	x	x	x

Gate Types – buf/not





- Verilog has a buffer (pass-through) primitive, and an inverter primitive.
- buf/not gates have one scalar input and multiple scalar outputs.

```
module bufnotexmp (bufout, notout, in);

  output bufout, notout;

  input in;

  buf (bufout, in);
  not (notout, in);

endmodule
```


buf

Truth Tables – buf/not

buf

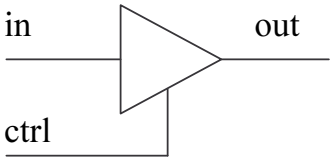
In	Out
1	1
0	0
x	x
z	x

not

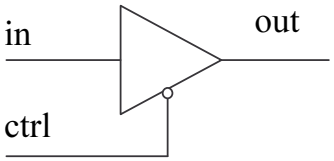
In	Out
1	0
0	1
x	x
z	x

Gate Types – conditional buf/not

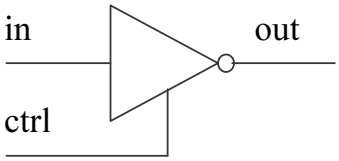
bufif1



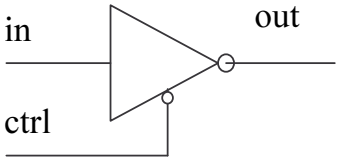
bufif0



notif1



notif0



- Verilog has 4 different built in conditional primitives.
- They can only have 3 scalar pins: Input, Output, and Control.
- The output goes to high impedance (‘z’) when the enable pin is false.

```
module condexmp (bufif1out, bufif0out,
                 notif1out, notif0out,
                 in, ctrl);

  output bufif1out, bufif0out, notif1out, notif0out;
  input in, ctrl;

  bufif1 (bufif1out, in, ctrl);
  bufif0 (bufif0out, in, ctrl);
  notif1 (notif1out, in, ctrl);
  notif0 (notif0out, in, ctrl);

endmodule
```

Truth Tables – Conditional buf/not

	ctrl				
in	***bufif1***	0	1	x	z
	0	z	0	x	x
	1	z	1	x	x
	x	z	x	x	x
	z	z	x	x	x

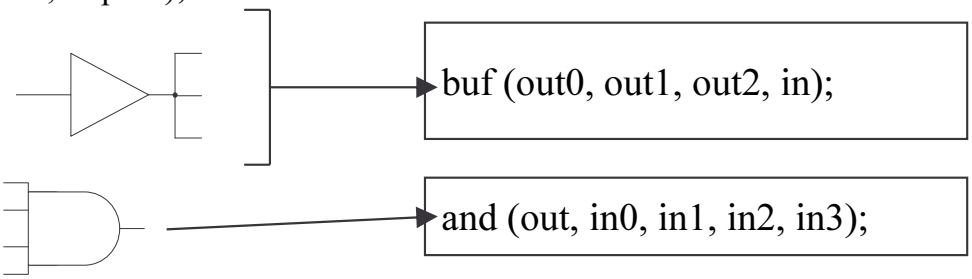
	ctrl				
in	***bufif0***	0	1	x	z
	0	0	z	x	x
	1	1	z	x	x
	x	x	z	x	x
	z	x	z	x	x

	ctrl				
in	***notif1***	0	1	x	z
	0	z	1	x	x
	1	z	0	x	x
	x	z	x	x	x
	z	z	x	x	x

	ctrl				
in	***notif0***	0	1	x	z
	0	1	z	x	x
	1	0	z	x	x
	x	x	z	x	x
	z	x	z	x	x

Primitive Pins

- Pins are expandable for Verilog primitives, except for conditional primitives (bufif1, notif1, etc.).
  - Multiple outputs, one input allowed for ***not*** and ***buf***.
  - Multiple inputs, one output allowed for ***and***/***or*** primitives.
- Syntax:
  - Multiple inputs: <***and/or*** set> <ref desg. (optional)> (<output>, <input1>, input2>,…,<inputn>);
  - Multiple outputs: <***not*** or ***buf***> <ref desg. (optional)> <output1>,<output2> …,<outputn>,<input>);



Specifying Gate Delays

- You can specify gate delays in Verilog primitives.
- Rise, Fall, and Turn-off Delays
  - Rise (‘0’ to ‘x’ or ‘1’)
  - Fall (‘1’ to ‘x’ or ‘0’)
  - Turn-off (‘1’,’0’, or ‘x’ to ‘z’)

```
bufif0 #(5) u1 (q, a, en);          // Rise=5, Fall=5, Turn-off=5
bufif0 #(5,3) u1 (q, a, en);        // Rise=5, Fall=3, Turn-off=min(5,3)=3
bufif0 #(5,4,3) u1 (q, a, en);      // Rise=5, Fall=4, Turn-off=3
```

Delay specification can only be added to primitives

Gate Delays

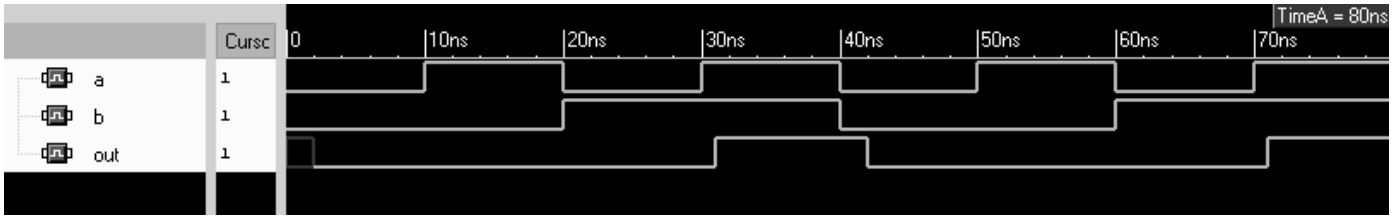
- Example: An ***and*** gate with a rise delay of 1ns and a fall delay of 2ns.

```
module andexamp (out, a, b);

  output out;
  input a, b;

  and #(1,2) (out, a, b);

endmodule
```

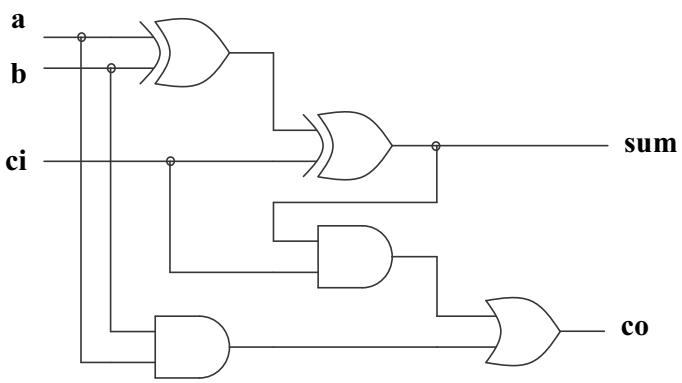


Gate Delays (cont.)

- Verilog allows three delay values to be specified as a triplet using colons (:) as separators. (i.e., (min:typ:max))
- The value used is specified at run time as a command line argument (+mindelays, +typdelays, +maxdelays)

```
bufif0 #(4:5:6) u1 (q, a, en);
bufif0 #(4:5:6, 2:3:4) u1 (q, a, en);
bufif0 #(4:5:6, 2:3:4, 1:2:3) u1 (q, a, en);
```

Primitive Instantiation




```
module addbit (a, b, ci, sum, co);
  input a, b, ci;
  output sum, co;
    xor #(2) u1 (n1, a, b);
    xor #(2,1) u2 (sum, n1, ci);
    and (strong1, weak0) #(1, 1.5) u3 (n2, a, b);
    and u4 (n3, sum, ci);
    or   u5 (co, n2, n3);
endmodule
```

Module Instantiation

- A module instantiation must have an instance name
- In positional mapping, the port order is the same is in the module declaration
- In name mapping, no order is required.

```
module add4bits (result, carry, r1, r2, ci);
…
  addbit u1 (r1[0], r2[0], ci, result[0], c1);                          // positional mapping
  addbit u2 (.a(r1[1]), .ci(c1), .b(r2[1]), .sum(result[1]), .co(c2));  // named mapping
  addbit u3 (r1[2], , c2, result[2], c3);                               // one port left unconnected
  addbit u4 (.a(r1[3]), .sum(result[3]), .co(carry));                   // named with two unconnected ports
endmodule
```

Module Instantiation – Array of Instances

- An array of instances can be created using one base instance.
- The port names must be the same throughout the instance array.
- Useful for bussed ports. The instance "value" will match the vector bit in the instance array.

```
module busena (Q, A, EN);
  input [3:0] A;
  input EN;
  output [3:0] Q;
  wire [3:0] A, Q; wire EN;
  bufif0 u[3:0] (Q, A, EN);
endmodule
```

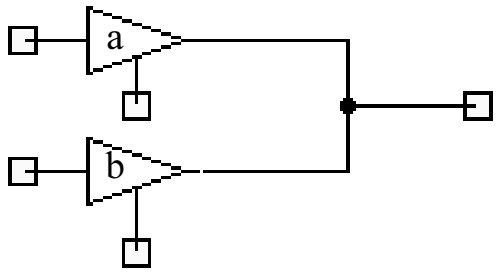
=

```
module busena (Q, A, EN);
  input [3:0] A;
  input EN;
  output [3:0] Q;
  wire [3:0] A, Q; wire EN;
 bufif0 u3 (Q[3], A[3] ,EN[3]);
 bufif0 u2 (Q[2], A[2] ,EN[2]);
 bufif0 u1 (Q[1], A[1], EN[1]);
 bufif0 uO (Q[0], A[0], EN[0]);
endmodule
```

Conflict Resolution

- There are 8 signal strengths in Verilog
  - supply, strong, pull, large, weak, medium, small, highz
- Supply is the strongest and highz is the weakest.
- Use the %v format specifier to view signal strength.
  - $monitor ("output = %v", signal);

```
bufif1 (strong0, weak1) a (out1, a, b);
bufif1 (pull0, strong1) b (out2, c, d);
```



a output	b output	output
strong1	strong0	strongX
pull0	weak1	pull0
pull0	strong0	strong0

See Appendix A of Verilog HDL by Palnitkar book for more details

Review

- Which primitives allow multiple inputs but only one output?
- Which primitives allow only two inputs and one output?
- Which primitives allow multiple outputs but only one input?
- How do you model path delays? Are path delays synthesizable?
- How are signal strength declarations useful in Verilog?