Verilog HDL:
Gate Level Modeling

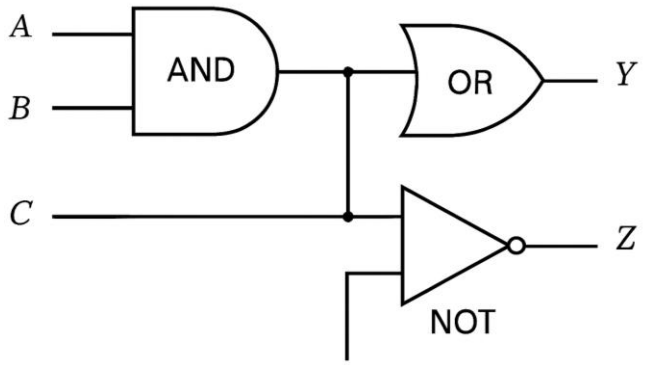
Pravin Zode

Outline

- Gate Level modeling
- Examples

Gate Level Modeling

- Represents digital circuits using basic logic gate
- Works well for small circuits with a limited number of gates
- Intuitive for designers familiar with digital logic design
- Requires manual instantiation and connection of individual gates
- Common gate primitives: `and,or,nand,nor, xor, xnor,not`



Truth Table for Gates

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

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

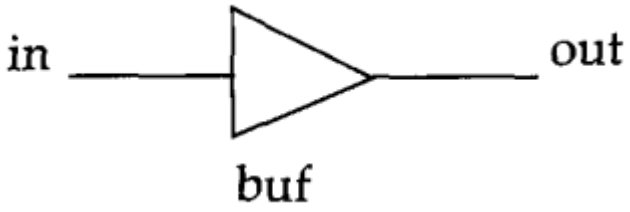
xor	i1 = 0	i1 = 1	i1 = x	i1 = z
i2 = 0	0	1	x	x
i2 = 1	1	0	x	x
i2 = x	x	x	x	x
i2 = z	x	x	x	x

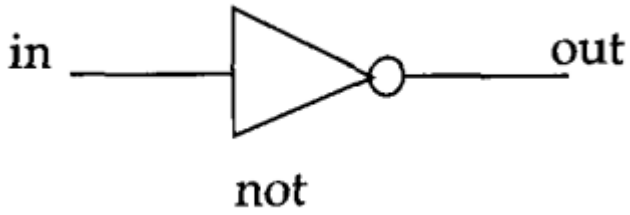
nand	i1 = 0	i1 = 1	i1 = x	i1 = z
i2 = 0	1	1	1	1
i2 = 1	1	0	x	x
i2 = x	1	x	x	x
i2 = z	1	x	x	x

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

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

Buffer and NOT Gate

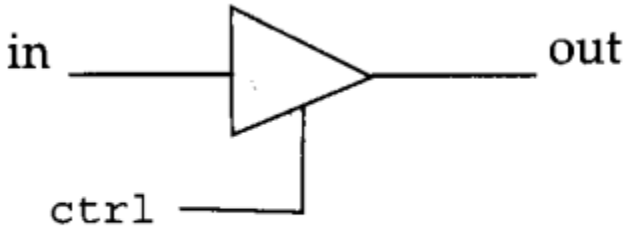




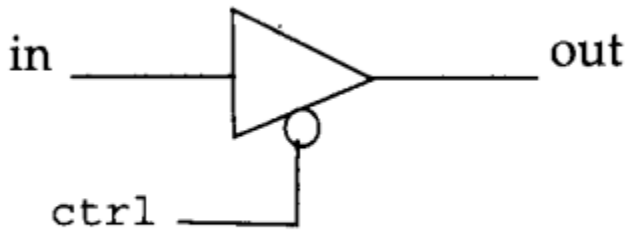
buf	in	out
	0	0
	1	1
	x	x
	z	x

not	in	out
	0	1
	1	0
	x	x
	z	x

Bufif Gate



bufif1



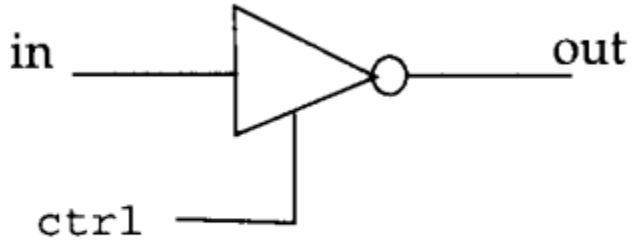
bufif0

	bufif1	ctrl 0	1	x	z
in	0	z	0	L	L
	1	z	1	H	H
	x	z	x	x	x
	z	z	x	x	x

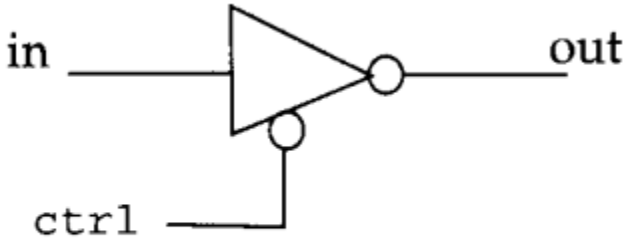
	bufif0	ctrl 0	1	x	z
in	0	0	z	L	L
	1	1	z	H	H
	x	x	z	x	x
	z	x	z	x	x

```
bufif1 b1 (out, in, ctrl);
bufif0 b0 (out, in, ctrl);
```

NOTif Gate



notif1



notif0

in \ notif1	ctrl 0	1	x	z
0	z	1	H	H
1	z	0	L	L
x	z	x	x	x
z	z	x	x	x

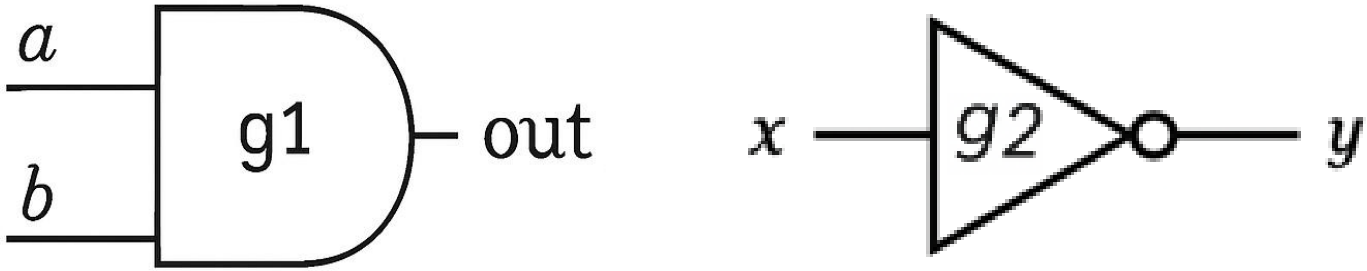
in \ notif0	ctrl 0	1	x	z
0	1	z	H	H
1	0	z	L	L
x	x	z	x	x
z	x	z	x	x

```
notif1 n1 (out, in, ctrl);
notif0 n0 (out, in, ctrl);
```

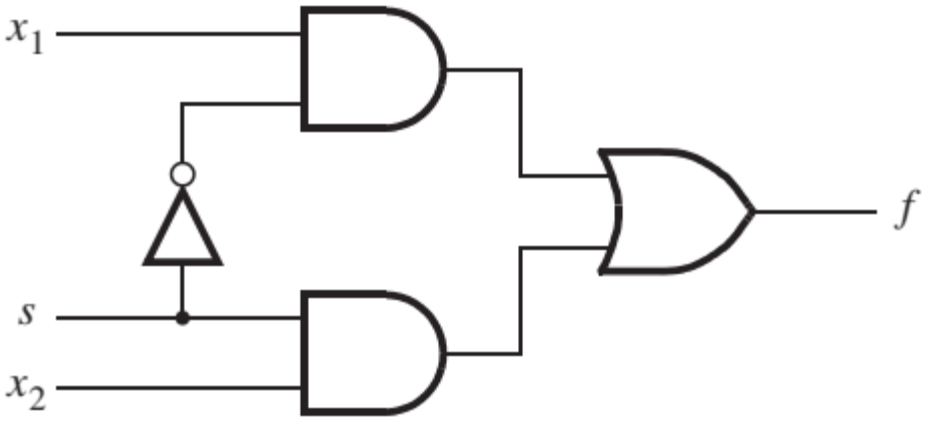
Syntax

- Gates can have one or more inputs, one output

```
and g1(out, a, b);    // out = a AND b
not g2(y, x);         // y = NOT x
```



Gate Level Primitive

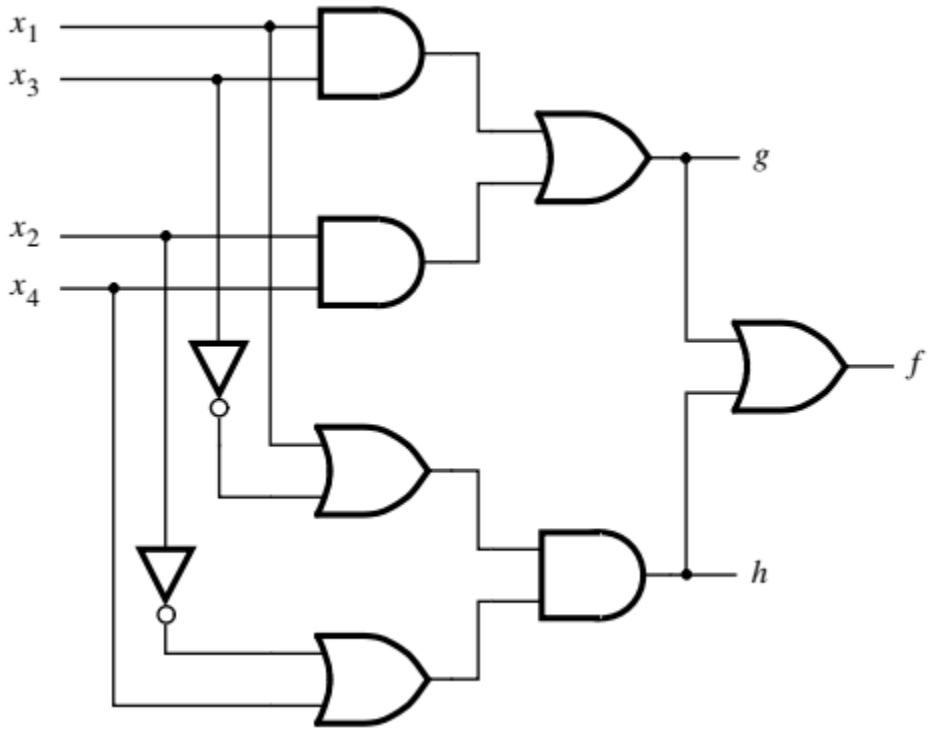



```
module example1 (x1, x2, s, f);
    input  x1, x2, s;
    output  f;

    not (k, s);
    and (g, k, x1);
    and (h, s, x2);
    or (f, g, h);

endmodule
```

Gate Level Primitive



$$g = x_1x_3 + x_2x_4$$
$$h = (x_1 + \bar{x}_3)(\bar{x}_2 + x_4)$$
$$f = g + h$$

```
module example2 (x1, x2, x3, x4, f, g, h);
  input x1, x2, x3, x4;
  output f, g, h;

  and (z1, x1, x3);
  and (z2, x2, x4);
  or (g, z1, z2);
  or (z3, x1, ~x3);
  or (z4, ~x2, x4);
  and (h, z3, z4);
  or (f, g, h);

endmodule
```

Summary

- **Structural modeling style** – design described using basic logic gates (and, or, not, xor, buf, bufif, notif, etc.)
- **Represents actual hardware connections** – closer to physical circuits (netlists).
- **Supports strength modeling** – logic strengths (strong, weak, supply) and signal states (0, 1, x, z).
- **Useful for small circuits**, switches, tri-state buffers (bufif, notif).
- **Limitations – Not practical** for large designs (tedious, error-prone).

Thank you !

Happy Learning