Indian Institute of Information Technology Vadodara

CS/IT 429

Gate and User Defined Primitives

Dr. Yash Agrawal
Visiting Faculty, IIIT Vadodara
Associate Professor, DA-IICT Gandhinagar

Primitives

- Primitives are generalized modules that already exists in Verilog.
- They can be initiated directly in other modules.
- Primitives can be of following types:
  - Gate Primitive
  - User Defined Primitive (UDP)

Gate Primitives

GATE	Mode of Instantiation	Output Port	Input Port
AND	and g1(o, i1, i2, i3, …);	o	i1, i2, i3, ….
NAND	nand g1(o, i1, i2, i3, …);	o	i1, i2, i3, ….
OR	or g1(o, i1, i2, i3, …);	o	i1, i2, i3, ….
NOR	nor g1(o, i1, i2, i3, …);	o	i1, i2, i3, ….
EXOR	xor g1(o, i1, i2, i3, …);	o	i1, i2, i3, ….
EXNOR	xnor g1(o, i1, i2, i3, …);	o	i1, i2, i3, ….
BUF	buf g1(o1, o1, o3, …., i);	o1, o2, o3, …..	i
INV	not g1(o1, o1, o3, …, i);	o1, o2, o3, …..	i

Logic Levels supported in Verilog

Logic Level/value	Interpretation
0	Logic Low
1	Logic High
x	Unknown
z	High Impedance

Truth table of Logic Gates in Verilog

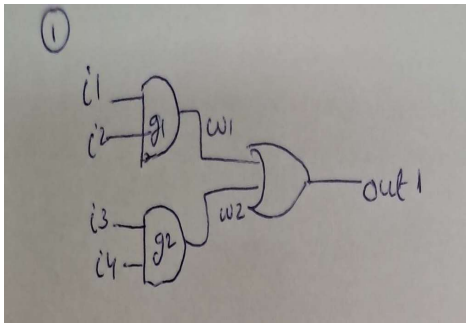
Truth table for AND Gate Primitive

		Input 2			
		0	1	x	z
Input 1	0	0	0	0	0
	1	0	1	x	x
	x	0	x	x	x
	z	0	x	x	x

Try for NAND, OR, NOR, EXOR, EXNOR, BUF and NOT Gates as well…

Contd…

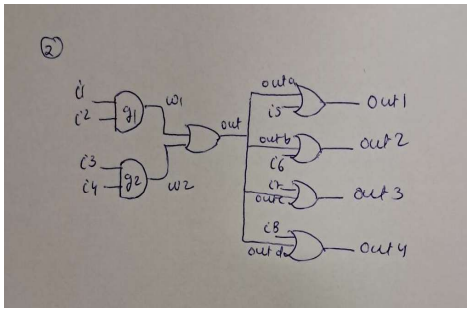
Implement the design using Gate Primitives




```
module logic1 (out1, i1, i2, i3, i4);
input i1, i2, i3, i4;
output out1;
wire w1, w2;
and g1(w1, i1, i2), g2(w2, i3, i4);
or g3(out1, w1, w2);
endmodule
```


Contd…

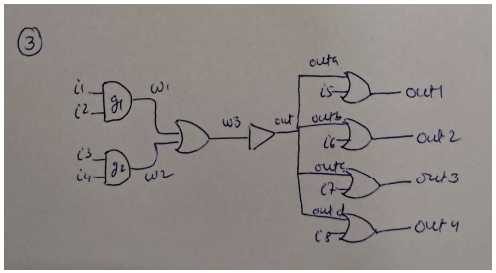
Implement the design using Gate Primitives




```
module logic2 (out1, out2, out3, out4, i1,
i2, i3, i4, i5, i6, i7, i8);

input i1, i2, i3, i4, i5, i6, i7, i8;
output out1, out2, out3, out4;
wire w1, w2, out, outa, outb, outc, outd;

and g1(w1, i1, i2), g2(w2, i3, i4);

or g3(out, w1, w2);
or g3(outa, outb, outc, outd, w1, w2);

or g4(out1, out, i5), g5(out2, out, i6),
   g6(out3, out, i7), g7(out4, out, i8);
or g4(out1, outa, i5), g5(out2, outb, i6),
   g6(out3, outc, i7), g7(out4, outd, i8);

endmodule
```


Contd…

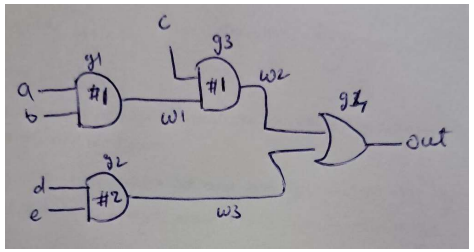
Implement the design using Gate Primitives




```
module logic3 (out1, out2, out3, out4, i1, i2,
i3, i4, i5, i6, i7, i8);

input i1, i2, i3, i4, i5, i6, i7, i8;
output out1, out2, out3, out4;
wire w1, w2, w3, out, outa, outb, outc, outd;

and g1(w1, i1, i2), g2(w2, i3, i4);

or g3(w3, w1,w2);

not g7(out,w3);
not g7(outa, outb, outc, outd, w3);

or g4(out1, out, i5), g5(out2, out, i6),
   g6(out3, out, i7), g7(out4, out, i8);
or g4(out1, outa, i5), g5(out2, outb, i6),
   g6(out3, outc, i7), g7(out4, outd, i8);

endmodule
```

Gate Delays

Incorporate the Gate delays using Gate Primitives




```
module logic4 (out1, a, ,b, c, d, e);
input a, b, c, d, e;
output out;
wire w1, w2, w3;
and #1  g1(w1, a, b);
and #2  g2(w3, d, e);
and #1  g3(w2, c, w1);
or      g4(out, w2, w3);
endmodule
```

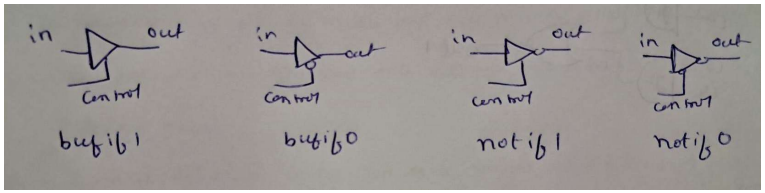
```
//and #1 g1(w1,a,b), #2 g2(w3,d,e), #1 g3(w2,c,w1);
//and #1 g1(w1,a,b),    g2(w3,d,e),     g3(w2,c,w1);
```


Gate Delays

Rise delay includes when values changes from	0 to 1 x to 1 0 to x	
Fall delay includes when values changes from	1 to 0 x to 0 1 to x	
Hold delay includes when values changes from	0 to z 1 to z x to z	Not applicable to Gate primitives

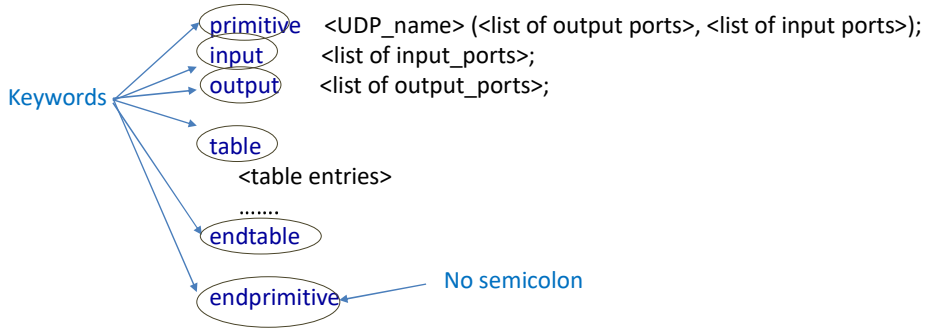
Tristate Gates

GATE	Mode of Instantiation	Output Port	Input Port
Buffer if 1	bufif1 g1(out, in, control);	out	in, control
Buffer if 0	bufif0 g1(out, in, control);	out	in, control
Invert if 1	notif1 g1(out, in, control);	out	in, control
Invert if 0	notiff0 g1(out, in, control);	out	in, control



User Defined Primitives (UDP)

Anatomy of a UDP in Verilog HDL



Contd…

Implement OR gate using Gate and UDP

Gate Primitive

```
module logic1 (out, in1, in2);

input i1, i2;
output out;

or g1(out, i1, i2);

endmodule
```

OR Gate

Input 1 \ Input 2	0	1	x	z
0	0	1	x	x
1	1	1	1	1
x	x	1	x	x
z	x	1	x	x

User Defined Primitive

```
primitive logic2 (out, in1, in2);

input i1, i2;
output out;

table
 // i1   i2   :   out;
    0    0    :    0;
    0    1    :    1;
    1    0    :    1;
    1    1    :    1;
endtable

endprimitive
```

What if…..
in1 = 1 and in2 = x

Contd…

Implement OR gate using Gate and UDP

User Defined Primitive

```
primitive logic2 (out, in1, in2);

input i1, i2;
output out;

table
 // i1   i2   :   out;
    0    0    :    0;
    0    1    :    1;
    1    0    :    1;
    1    1    :    1;
endtable

endprimitive
```

What if…..
in1 = 1 and in2 = x

User Defined Primitive

```
primitive logic3 (out, in1, in2);

input i1, i2;
output out;

table
 // i1   i2   :   out;
    0    0    :    0;
    0    1    :    1;
    1    0    :    1;
    1    1    :    1;
    1    x    :    1;
    x    1    :    1;
    x    0    :    x;
    0    x    :    x;
endtable

endprimitive
```

What if…..
in1 = 1 and in2 = x

Contd…

Implement OR gate using Gate and UDP

User Defined Primitive

```
primitive logic3 (out, in1, in2);

input i1, i2;
output out;

table
 // i1   i2   :   out;
    0    0    :    0;
    0    1    :    1;
    1    0    :    1;
    1    1    :    1;
    1    x    :    1;
    x    1    :    1;
    x    0    :    x;
    0    x    :    x;
endtable

endprimitive
```

Concept of don't care (?)

```
primitive logic4 (out, in1, in2);

input i1, i2;
output out;

table
 // i1   i2   :   out;
    0    0    :    0;
    1    ?    :    1;
    ?    1    :    1;
    x    0    :    x;
    0    x    :    x;
endtable

endprimitive
```