CSE-308: Digital System Design

Lecture 4

VERILOG – Part II

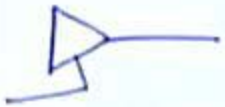
Parameters

- A *parameter* is a constant with a name.
- No size is allowed to be specified for a parameter.
  - The size gets decided from the constant itself (32-bits if nothing is specified).
- <u>Examples</u>:

```
parameter  HI = 25, LO = 5;
parameter  up = 2b'00, down = 2b'01,
           steady = 2b'10;
```

Logic Values

- The common values used in modeling hardware are:
  - 0 :: Logic-0 or FALSE
  - 1 :: Logic-1 or TRUE
  - x :: Unknown (or don't care)
  - z :: High impedance



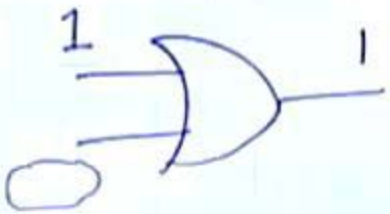
- Initialization:
  - All unconnected nets set to 'z'
  - All register variables set to 'x'

- Verilog provides a set of predefined logic gates.
  - They respond to inputs (0, 1, x, or z) in a logical way.
  - Example :: AND

0 & 0 → 0	0 & x → 0
0 & 1 → 0	1 & z → x
1 & 1 → 1	z & x → x
1 & x → x	

XOR

A	B	f
0	X	X
1	X	X
0	Z	X
1	Z	X



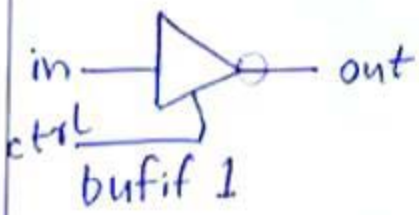
Primitive Gates

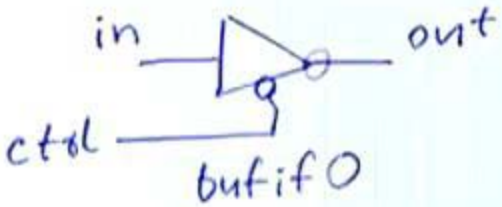
- Primitive logic gates (instantiations):

```
and   G (out, in1, in2);
nand  G (out, in1, in2);
or    G (out, in1, in2);
nor   G (out, in1, in2);
xor   G (out, in1, in2);
xnor  G (out, in1, in2);
not   G (out1, in);
buf   G (out1, in);
```

- **Primitive Tri-State gates (instantiation)**

```
bufif1  G (out, in, ctrl);
bufif0  G (out, in, ctrl);
notif1  G (out, in, ctrl);
notif0  G (out, in, ctrl);
```





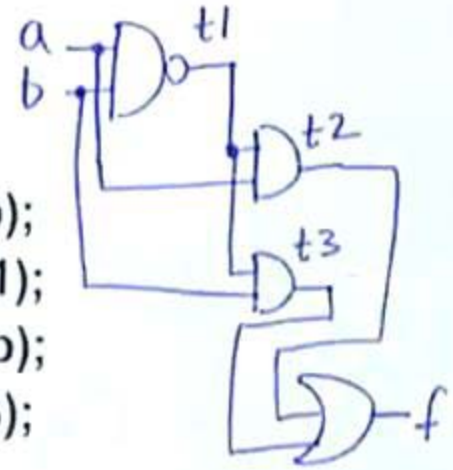
Some Points to Note

- For all primitive gates,
  - The output port must be connected to a net (a wire).
  - The input ports may be connected to nets or register type variables.
  - They can have a single output but any number of inputs.
  - An optional delay may be specified.
    - *Logic synthesis tools ignore time delays.*

and (out, in1, in2, in3, in4)

>4

```
`timescale 1 ns / 1ns
module exclusive_or (f, a, b);
        input a, b;
        output f;
        wire t1, t2, t3;
        nand #5 m1 (t1, a, b);
        and  #5 m2 (t2, a, t1);
        and  #5 m3 (t3, t1, b);
        or   #5 m4 (f, t2, t3);
endmodule
```



`timescale <ref-time-unit> / <time precision>

Ex.

`timescale 10ns / 1 ns

5.2 5.2

nand #5 - - -

⇓

50ns

Hardware Modeling Issues

- The values computed can be held in
  - A 'wire'
  - A 'flip-flop' (edge-triggered storage cell)
  - A 'latch' (level-sensitive storage cell)
- A variable in Verilog can be of
  - 'net' data type
    - Maps to a 'wire' during synthesis
  - 'register' data type
    - Maps either to a 'wire' or to a 'storage cell' depending on the context under which a value is assigned.

```
module  reg_maps_to_wire  (A, B, C, f1, f2);
   input    A, B, C;
   output  f1, f2;
   wire      A, B, C;
   reg       f1, f2;
   always  @(A or B or C)
   begin
      f1 = ~(A & B);
      f2 = f1 ^ C;
   end
endmodule
```

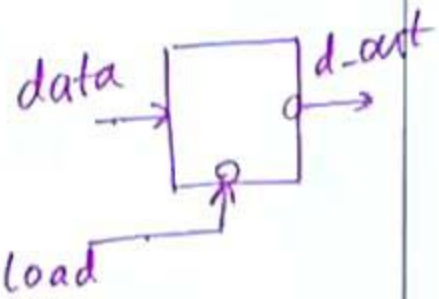
The synthesis system will generate a wire for f1

```
module  a_problem_case  (A, B, C, f1, f2);
   input    A, B, C;
   output  f1, f2;
   wire     A, B, C;
   reg      f1, f2;
   always  @(A or B or C)
   begin
      f2 = f1 ^ f2;
      f1 = ~(A & B);
   end
endmodule
```

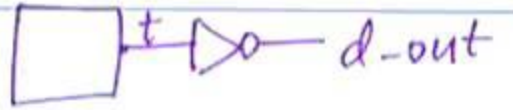
The synthesis system will not generate a storage cell for f1

```
// A latch gets inferred here
module  simple_latch  (data, load, d_out);
   input    data, load;
   output  d_out;

   always @(load or data)
   begin
      if (!load)
         t = data;
      d_out = !t;
   end
endmodule
```



Else part missing; so latch is inferred.



Verilog Operators

- Arithmetic operators

  *, /, +, -, %

- Logical operators

  ! → logical negation

  && → logical AND

  || → logical OR

- Relational operators

  >, <, >=, <=, ==, !=

- Bitwise operators

  ~, &, |, ^, ~^

- <u>Reduction operators</u> (operate on all the bits within a word)

  &, ~&, |, ~|, ^, ~^

  ➔ accepts a single word operand and produces a single bit as output

- <u>Shift operators</u>

  >>, <<

- <u>Concatenation</u> { }
- <u>Replication</u> { { } }
- <u>Conditional</u>

  <condition> ? <expression1> : <expression2>

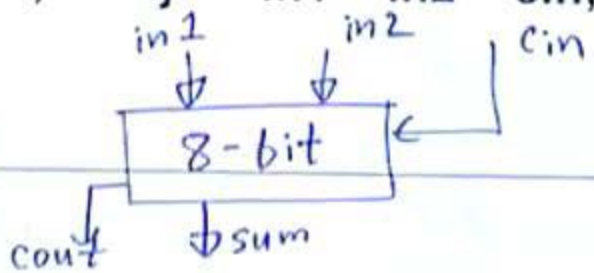
```
module operator_example (x, y, f1, f2);
   input    x, y;
   output  f1, f2;
   wire [9:0]  x, y;   wire [4:0]  f1;   wire  f2;

   assign  f1 = x[4:0] & y[4:0];
   assign  f2 = x[2] | ~f1[3];
   assign  f2 = ~& x;
   assign  f1 = f2 ? x[9:5] : x[4:0];
endmodule
```

```
// An 8-bit adder description

module parallel_adder (sum, cout, in1, in2, cin);
   input  [7:0] in1, in2;   input   cin;
   output [7:0] sum;        output  cout;

   assign  #20 {cout, sum} = in1 + in2 + cin;
endmodule
```



Some Points

- The presence of a 'z' or 'x' in a *reg* or *wire* being used in an arithmetic expression results in the whole expression being unknown ('x').
- The logical operators (!, &&, | |) all evaluate to a 1-bit result (0, 1 or x).
- The relational operators (>, <, <=, >=, ~=, ==) also evaluate to a 1-bit result (0 or 1).
- Boolean *false* is equivalent to 1'b0

  Boolean *true* is equivalent to 1'b1.

Some Valid Statements

```
assign  outp = (p == 4'b1111);
if  (load  &&  (select == 2'b01)) .......
assign   a = b >> 1;
assign   a = b << 3;
assign  f = {a, b};
assign f = {a, 3'b101, b};
assign f = {x[2], y[0], a};
assign f = { 4{a} };   // replicate four times
assign f = {2'b10, 3{2'b01}, x};
```

{ 4{=} }

10 010101 ---