More Verilog

Exercise 1: Is a signal named **$\overline{\text{overload}}$** active-high or active-low? Is there an overload if this signal is high? What if the signal was named **overload**?

$\overline{\text{overload}}$ → active-low L ⇒ True H ⇒ F (no overload)

overload → active-high L = False H ⇒ T (overload!)

Exercise 2: Come up with active-high and an active-low names for a signal that is at 3 V when a door is open and 0 V when the door is closed.

3V ⇒ H & open is T H ⇒ T active high
name is open

closed is F H ⇒ F active low
name is $\overline{\text{closed}}$

0V ⇒ L & open is F L ⇒ F active high
name is open

0V ⇒ L & closed is T L ⇒ T active low
name is $\overline{\text{closed}}$

Exercise 3: If $\overline{\texttt{D}}$ is a word and $\overline{\texttt{D[0]}}$ is low, is the word an even or odd number?

if $\overline{D[0]}$ is low it is true because it is active-low.

if it is true, the numerical value would be 1.

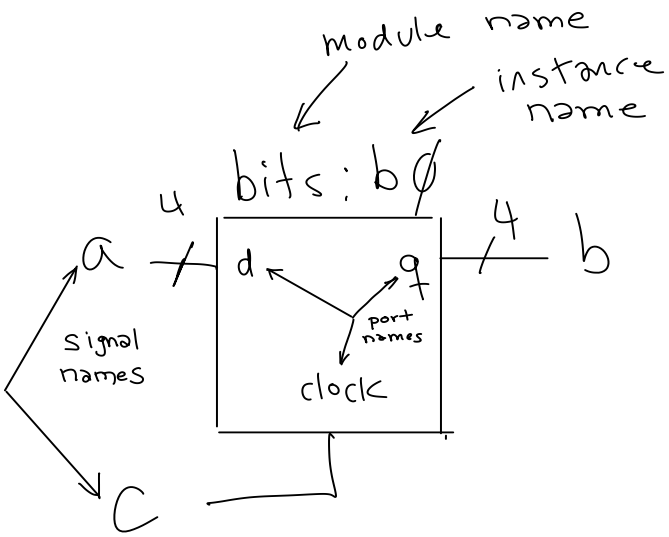
bit 0 would be the least-significant bit.

if the L.S. bit is 0 it is an <u>even</u> number.

Exercise 4:

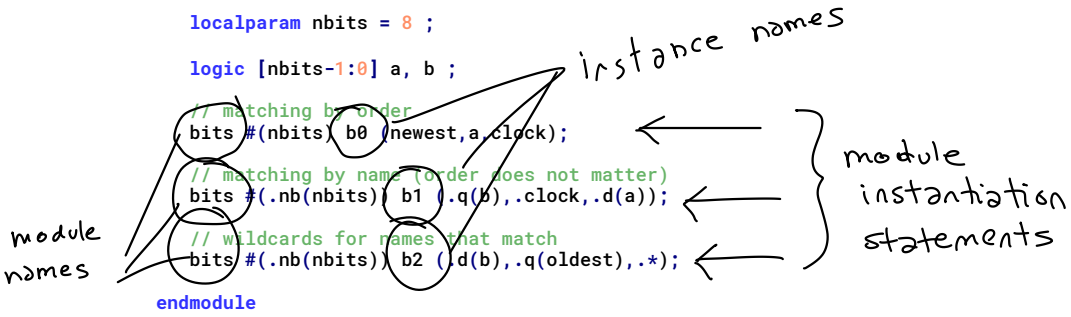
```
bits #(4) b0 (a,b,c) ;
```

Draw a diagram for this instantiation of the **bits** module. Label the module, instance, signal and port names as in the diagram above.



Exercise 5:

```
module sr3bytes
   (
    input  logic [7:0] newest,
    output logic [7:0] oldest,
    input  logic clock
    ) ;

   localparam nbits = 8 ;

   logic [nbits-1:0] a, b ;

   // matching by order
   bits #(nbits) b0 (newest,a,clock);

   // matching by name (order does not matter)
   bits #(.nb(nbits)) b1 (.q(b),.clock,.d(a));

   // wildcards for names that match
   bits #(.nb(nbits)) b2 (.d(b),.q(oldest),.*);

endmodule
```



Identify the module instantiation statements in the code above. For each one, what is the instantiated module's name? The instance name?

Exercise 6: What are the values of the following expressions: `|4'b0001`, `^4'b1001`, `&4'b1111`, `&4'b1110`?

|4'b0001 = 0|0|0||1 → 1'b1 or-reduction

^4'b1001 → 1^0^0^1 → 1'b0 xor-reduction

1
1
0

&4'b1111 → 1&1&1&1 = 1

&4'b1110 → 0

} and-reduction

Exercise 7:

```
// concatenation:
logic [3:0] x = { 2'b00, 2'b11 };

// replication (z=8'b1010_1010):
logic [7:0] y = { 4{2'b10} };

// array literal
logic [0:1] [3:0] z = '{ 4'b0011, 4'b1010 };
```

	dimensions	values
x	4	4'b0011
y	8	8'b10101010
z	2x4	{4'b0011, 4'b1010}

What are the dimensions and initial values of **x**, **y**, and **z** in the examples above?