Tutorial 3

CSE 313

Modules

```
Module Module_name ( input wire [1:0] x, output reg [3:0] y);
        //code
endmodule
```

Or

```
Module Module_name ( x, y);
Input x;
Output y;
wire [1:0] x;
reg[3:0] y;
        //code
endmodule
```

Blocks

```
always @ ( posedge clk or negedge rest)
begin
        //code
end

Initial
begin
        //code
end
```

Data types

- Nets
  - Ex:Wire,tri,wand,supply0
  - Can only be in continuous assignments (using assign keyword)
  - continuous assignments exists in parallel domains (modules)
- Variables
  - Ex: reg, int, string
  - Can only be in procedural assignments
  - procedural assignments exists in sequential domains (blocks)
- Testbench variables
  - Ex: logic, bit
  - Exist anywhere

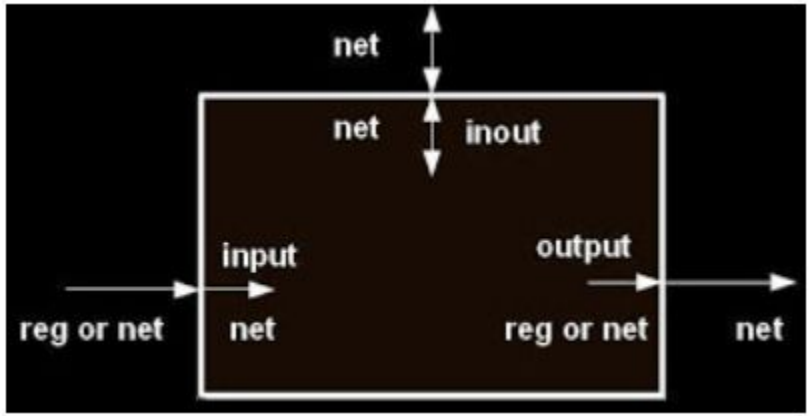
MSB rules

What is the difference between the following two declarations?

```
1 wire [7:0] w1;
2 wire [0:7] w2;
```

What is 10'had?

Port type rules



Blocking and non-blocking (procedural assignments)

. The code below starts executing a t = 0. Show all changes in a, include the time of the change and the new value.

```
1  integer a;
2  initial begin
3    a = 1;
4    #1;
5    a = 2;
6    #1;
7    a = 3;
8    #1;
9    a <= 4;
10   #1;
11   a <= 5;
12   #1;
13   #3 a = a+1;
14   #1;
15   a = #3 a+1;
16   #1;
17   a <= #3 a+1;
18   #1;
19   a = a+1;
20 end
```

Behavioral VS structural design

Write a Verilog behavioral description of a four-bit adder module. The adder should have three inputs, a, b, and c_{in}, and two outputs, sum and cout. Ports c_{in} and c_{out} are one bit, the other ports are four bits each.

```
1  module add4(sum,cout, a, b, cin);
2    output sum, cout;
3    input a, b, cin;
4
5    reg [3:0] sum;
6    reg cout;
7    wire [3:0] a, b;
8    wire cin;
9
10   always @( a or b or cin ) {cout,sum} =a+b+ cin;
11 endmodule // add4
```

```
module full_adder ( a ,b ,c ,sum ,carry );
output sum ;
output carry ;
input a ;
input b ;
input c ;

  assign sum = a ^ b ^ c;
  assign carry = (a&b) | (b&c) | (c&a);

endmodule
//----------------------------------------------
module adder_4bit ( a ,b ,sum ,carry );
output [3:0] sum ;
output carry ;
input [3:0] a ;
input [3:0] b ;

  wire [2:0]s;

  full_adder u0 (a[0],b[0],1'b0,sum[0],s[0]);
  full_adder u1 (a[1],b[1],s[0],sum[1],s[1]);
  full_adder u2 (a[2],b[2],s[1],sum[2],s[2]);
  full_adder u3 (a[3],b[3],s[2],sum[3],carry);

endmodule
```

Lab 1

. An accumulator has three inputs, *amt*, *reset*, and *clk*, and an output, *sum*. The accumulator has an internal 32-bit register which is updated as follows:

- On a positive edge of *clk* it adds *amt*, a 32-bit integer, to the register;
- On the negative edge of *clk* it places the new *sum* on its outputs (until the next negative edge).
- Whenever *reset* is high the register is set to zero and the output changes immediately.

Write a Verilog behavioral description of this module.