

# Introduction to Verilog

Sukanta Pramanik

(Modified from Winter 11 slides by Omid Ardakanian)

CS 450/650 - Computer Architecture

University of Waterloo

# Outline

- Introduction
- CAD tools for Verilog
- How to write “Hello World!” in Verilog
  - Module Declaration
  - Register vs. Wire
  - Vectors vs. Arrays
  - Data Types
  - Primitives
  - Port Connection Rule
- Assignment

# Introduction (What are HDLs?)

- Definition...
- Popular HDLs
  - Verilog (Verifying Logic)
  - VHDL
- Why are HDLs essential?

# Verilog vs. VHDL

- VHDL
  - More general language
  - Not all constructs are synthesizable
- Verilog
  - Not as general as VHDL
    - but easier to learn than VHDL
  - Most constructs are synthesizable
  - Its syntax is similar to C

# CAD tools for Verilog

- Why do we need a CAD tool?
  - Fast design
  - Debugging and testing (*simulation*)
  - It is necessary for *synthesis*
- Popular examples:
  - **Modelsim**
  - Verilog-XL
  - VCS
  - Icarus Verilog

# One-bit Full Adder (full\_adder)

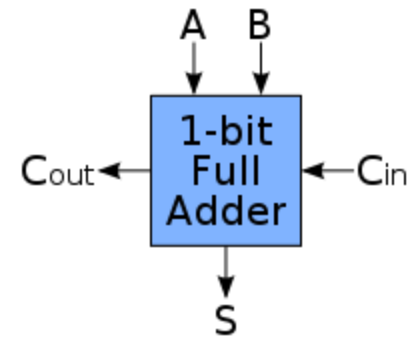
module name

port declaration (input, output, inout)

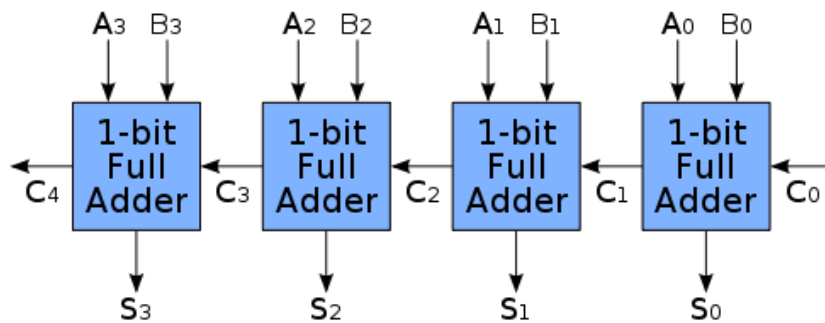
```
module full_adder(a,b,cin,sum,cout);  
    input a, b, cin;  
    output sum, cout;  
  
    assign sum = cin ^ a ^ b;  
    assign cout = ~cin & a & b | cin & (a | b);  
endmodule
```

port list

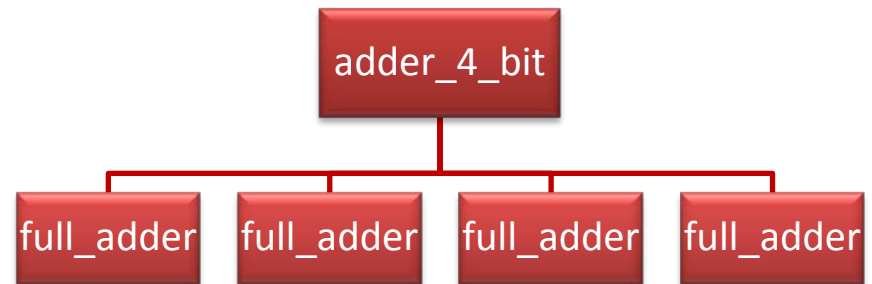
module internals



# 4-bit Ripple Carry Adder



Schematic Design of a 4-bit  
Ripple Carry Adder

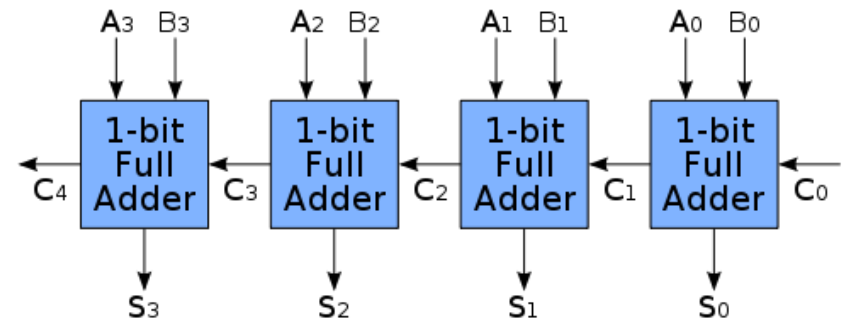


Design Hierarchy

# 4-bit Ripple Carry Adder (adder\_4\_bit)

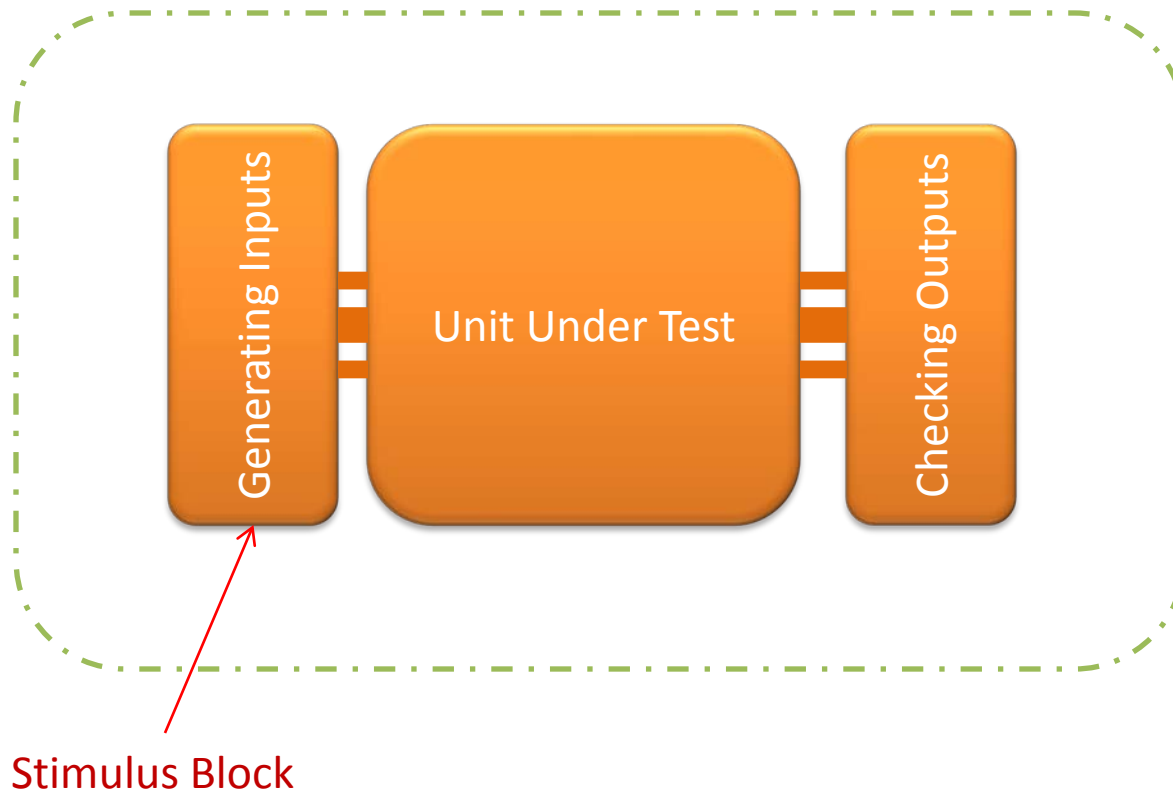
```
module adder_4_bit(x,y,cin,z,cout);  
    input [3:0] x ,y;  
    input cin;  
    output [3:0] z;  
    output cout;  
    wire [3:1] carry;  
  
    full_adder fa0(x[0],y[0],cin,z[0],carry[1]);  
    full_adder fa1(x[1],y[1],carry[1],z[1],carry[2]);  
    full_adder fa2(x[2],y[2],carry[2],z[2],carry[3]);  
    full_adder fa3(x[3],y[3],carry[3],z[3],cout);  
endmodule
```

instance  
name





# Test Bench



# Test Bench (adder\_4\_bit\_tb)

```
module adder_4_bit_tb;
    reg [3:0] x, y;
    reg cin;
    wire [3:0] z;
    wire cout;

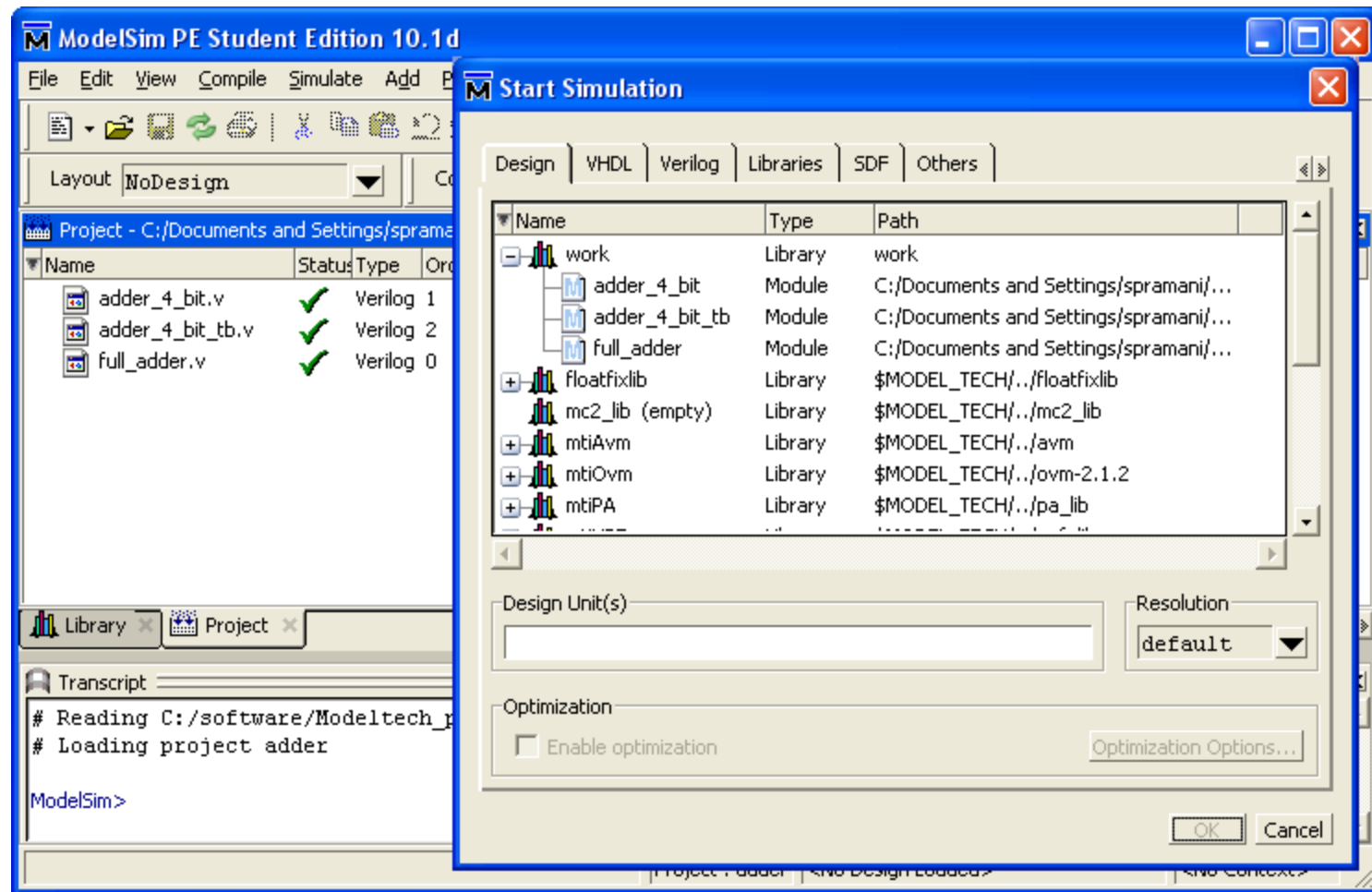
    adder_4_bit a0(x,y,cin,z,cout);

    initial begin
        x = 0; y = 0; cin = 0;
        #10 x = 4'b0101; y = 4'b0001; cin = 1'b0;
        #10 x = 4'b0101; y = 4'b1110; cin = 1'b1;
        #10 x = 0; y = 0; cin = 0;
        #10 $finish;
    end
endmodule
```

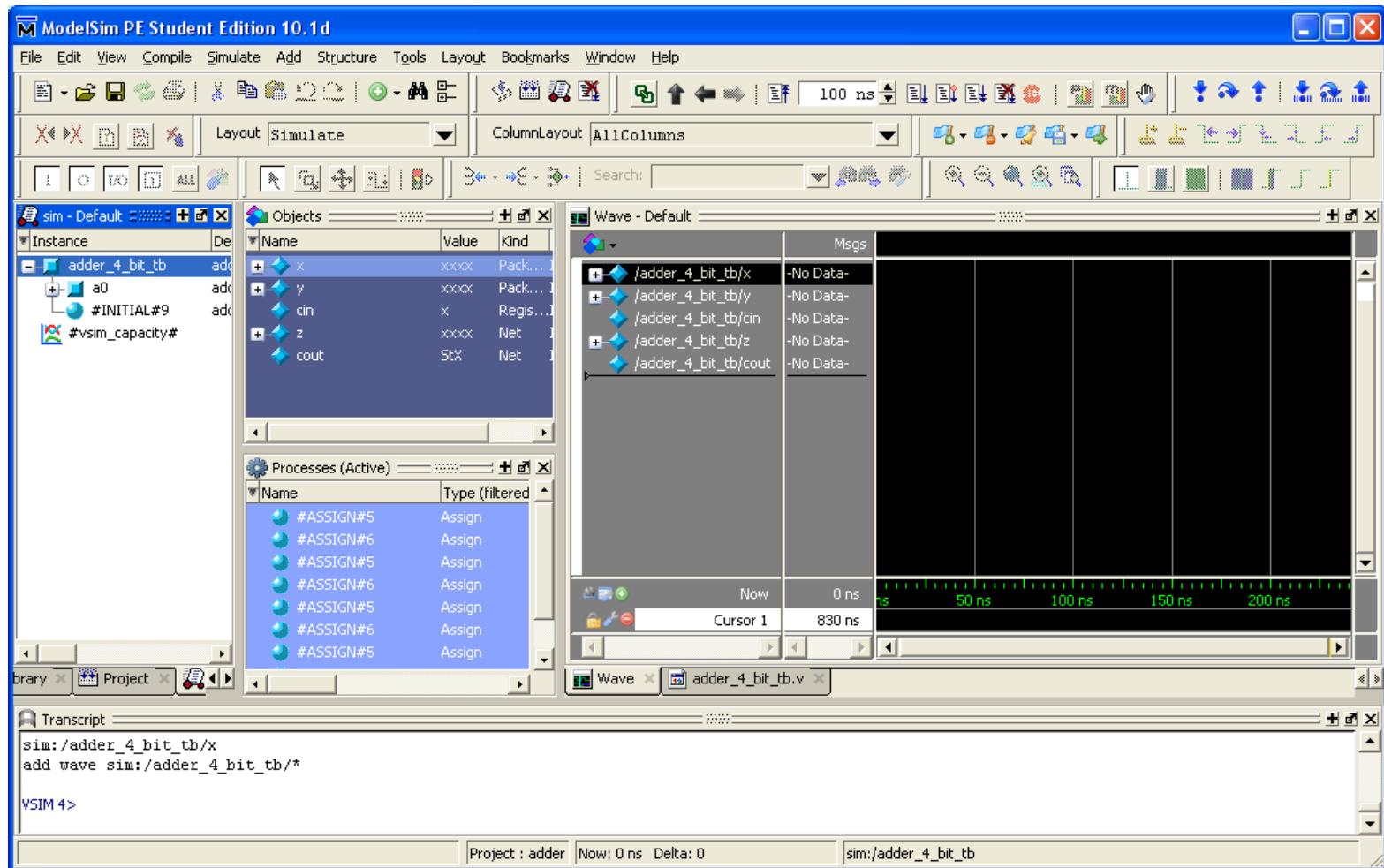
timing →

# Putting All Together in ModelSim

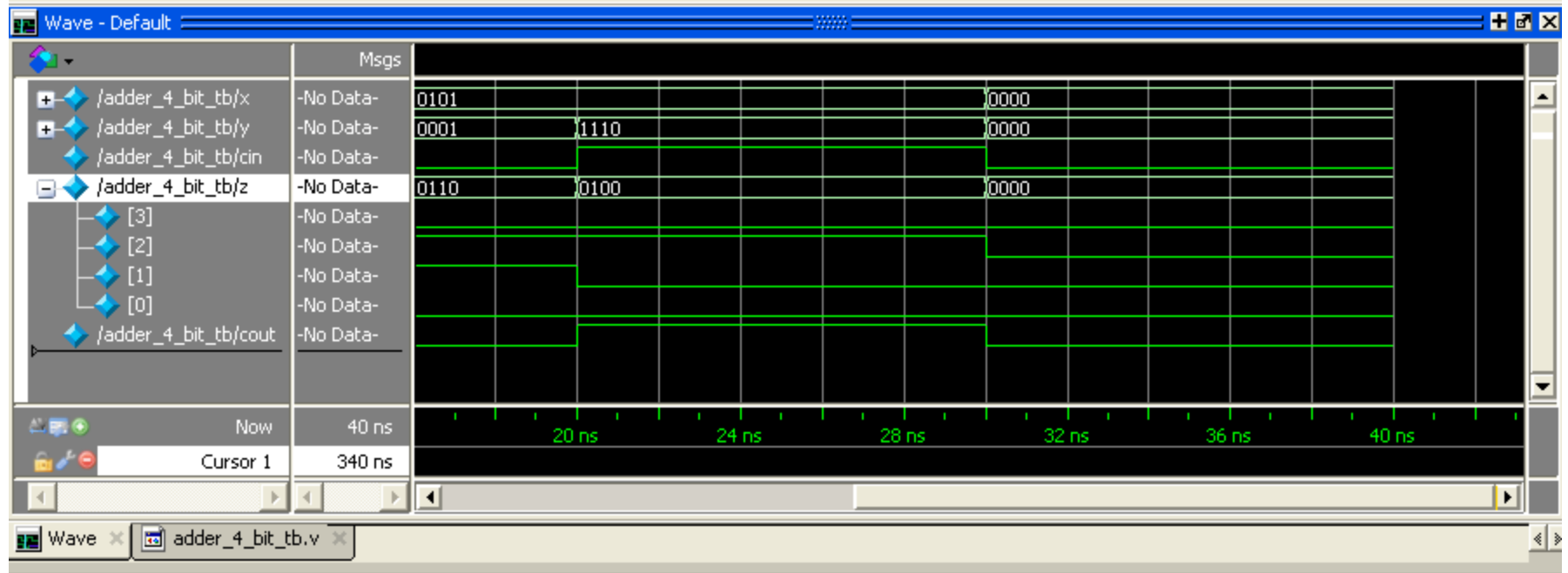
## Simulate Menu -> Start Simulation



# Putting All Together in ModelSim



# Putting All Together in ModelSim



# Register vs. Wire

- wire
  - Represents connection between components
  - Its value is determined by the value of its drivers
  - Its default value is z
- reg
  - Retains its value until next assignment
  - Its default value is x

# Vectors vs. Arrays

- Vectors
  - To define multiple width wires or registers
  - Definition:
    - `wire/reg [msb_index : lsb_index] data_id;`
- Arrays
  - It is possible to have reg, integer and time arrays
  - Definition:
    - `<data_type> <var_name>[start_idx : end_idx];`

# Data Types in Verilog

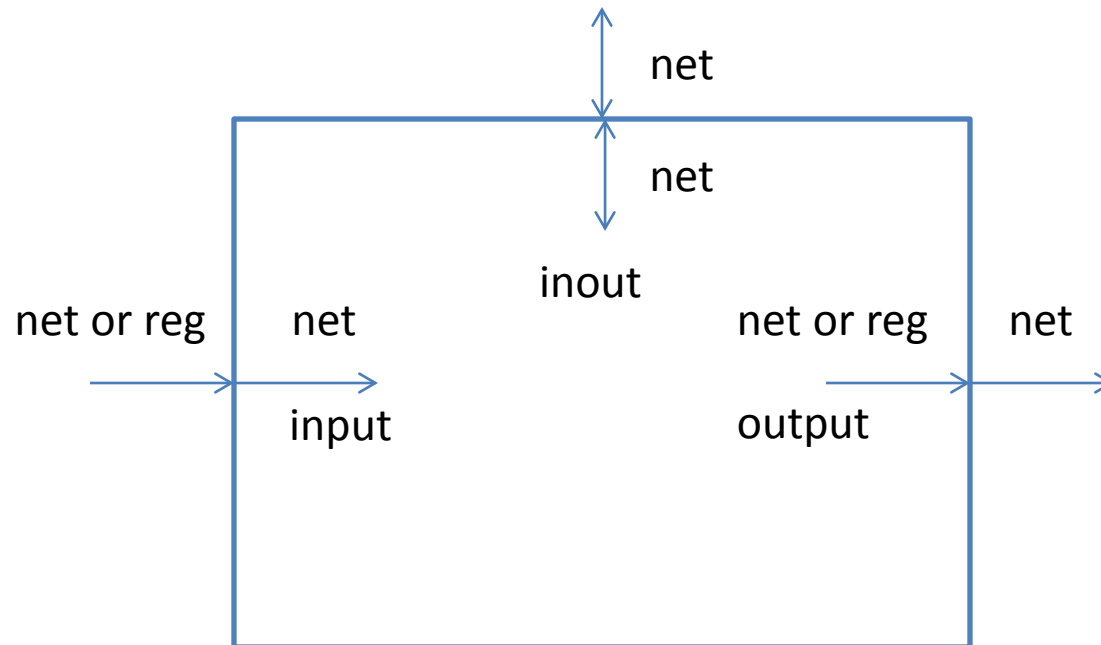
- Net
- Register
- Integer
- Real
- Time
- ...



# Primitives

- and/nand
  - or/nor
  - xor/xnor
  - buf/not
- 
- Example: and a1(out, in1, in2)

# Port Connection Rules



# Some Resources

- Verilog Tutorial Website

<http://www.electrosofts.com/verilog/>

- ModelSim Tutorial

[http://cseweb.ucsd.edu/classes/wi12/cse141L-a/Media/modelsim\\_tut.pdf](http://cseweb.ucsd.edu/classes/wi12/cse141L-a/Media/modelsim_tut.pdf)

- Links and the slides will be posted in the course homepage

<http://www.student.cs.uwaterloo.ca/~cs450>