There are three main ways to describe hardware circuits which produce a “signal”, “electrical node”, “word”, etc. (whatever you like to call it) inside a module definition:

- The assign command which describes a wire
- The always command which describes a reg
- Instantiate a module which has wires connected to its outputs

All of these must be declared at the module definition level—not inside each other (e.g., a module instance can not be declared inside an always block)
All of these signal types may be used as outputs in a module definition:

- **wire**
- **reg**

Another possibility which is typically uncommon is for an **input** to pass directly to a module **output** port.

```verilog
module module_name (port_name_list);

assign statement

always block

module instance

endmodule
```
Describing Hardware

- There are three main possible inputs to a module instance:
  - A **wire**
  - A **reg**
  - An **input** into the module (technically still a wire)

- The output of a module instance is always a **wire**, at least for this class

```verilog
module module_name (port_name_list);

endmodule
```
Describing Hardware

- In the same way, there are three main possible “inputs” to an assign statement:
  - A wire
  - A reg
  - An input into the module (technically still a wire)

- Example:
  ```
  input a;
  wire b;
  reg c;

  wire x;
  assign x = a & b | c;
  ```
Describing Hardware

• In the same way, there are three main possible “inputs” to an always block:
  - A wire
  - A reg
  - An input into the module (technically still a wire)

• Example:
  ```
  input a;
  wire b;
  reg c;

  reg x;
  always @(*) begin
    x = a & b | c;
  end
  ```