DSD - This a note about Verilog in Digital System Design.

Control System (Rajshahi University)

Q: 2019.8(b): Describe the "port connection rule" used in verilog HDL.
Answer:

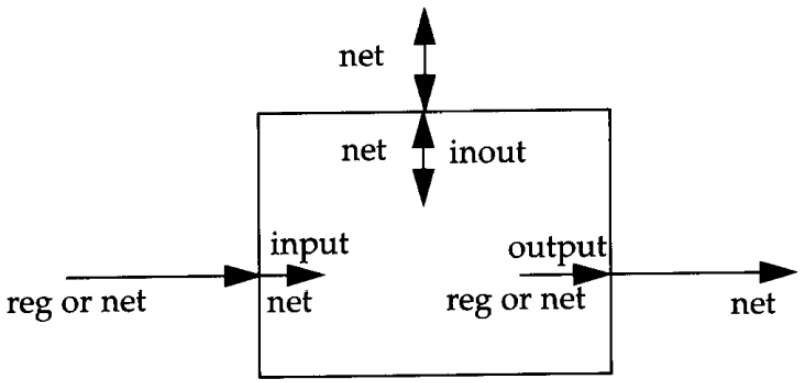


Figure: Port Connection Rules

Inputs

Internally, input ports must always be of the type *net*. Externally, the inputs can be connected to a variable which is a **`reg`** or a *net*.

Outputs

Internally, outputs ports can be of the type **`reg`** or *net*. Externally, outputs must always be connected to a *net*. They cannot be connected to a **`reg`**.

Inouts

Internally, inout ports must always be of the type *net*. Externally, inout ports must always be connected to a *net*.

Width matching

It is legal to connect internal and external items of different sizes when making inter-module port connections. However, a warning is typically issued that the widths do not match.

Unconnected ports

Verilog allows ports to remain unconnected. For example, certain output ports might be simply for debugging, and you might not be interested in connecting them to the external signals. You can let a port remain unconnected by instantiating a module as shown below.

Q: 2019.8(a): What are the "Design Block" and "Stimulus Block" used in Verilog HDL?
Answer:

Design Block:
A design block is the basic building block in verilog. Elements are grouped into design block to provide common functionality that is used by stimulus block. Design block have port interface (inputs and outputs) that allows it to perform specific operation on inputs and to produce outputs. That means design block is the complete representation of digital circuit in verilog HDL.

stimulus block:
The block which tests the design block is known as stimulus block. The functionality of the design block can be tested by applying stimulus and checking results. The stimulus block is also known as a test bench.

Q: 2018.8(a): What is verilog HDL? Briefly describe the salient features of Verilog HDL.

Answer:
Verilog HDL is a hardware description language (HDL) used to model electronic systems. It is most commonly used in the design and verification of digital circuits at the register-transfer level.

Salient features of Verilog HDL:

- Verilog HDL is a general-purpose hardware description language that is easy to learn and easy to use. It is similar in syntax to the C programming language. Designers with C programming experience will find it easy to learn Verilog HDL.
- Verilog HDL allows different levels of abstraction to be mixed in the same model. Thus, a designer can define a hardware model in terms of switches, gates, RTL, or behavioral code. Also, a designer needs to learn only one language for stimulus and hierarchical design.
- Most popular logic synthesis tools support Verilog HDL. This makes it the language of choice for designers.
- All fabrication vendors provide Verilog HDL libraries for postlogic synthesis simulation. Thus, designing a chip in Verilog HDL allows the widest choice of vendors.
- The Programming Language Interface (PLI) is a powerful feature that allows the user to write custom C code to interact with the internal data structures of Verilog. Designers can customize a Verilog HDL simulator to their needs with the PLI.

Q: 2018.8(c): Write a Verilog code to implement XOR operation.
Answer:

```
module xor(out, a, b);
        input a, b;
        output out;
        assign out = a ^ b;
endmodule
```

Q: 2015.8(b): Explain min, max and typ delays in the gate-level design with respect to Verilog HDL.
Answer:

Verilog provides an additional level of control for each type of delay mentioned above. For each type of delay—rise, fall, and turn-off—three values, *min*, *typ*, and *max*, can be specified. Any one value can be chosen at the start of the simulation. Min/typ/max values are used to model devices whose delays vary within a minimum and maximum range because of the IC fabrication process variations.

Min value

The min value is the minimum delay value that the designer expects the gate to have.

Typ val

The typ value is the typical delay value that the designer expects the gate to have.

Max value

The max value is the maximum delay value that the designer expects the gate to have.

Min, typ, or max values can be chosen at Verilog run time. Method of choosing a min/typ/max value may vary for different simulators or operating systems. (For Verilog-XL™, the values are chosen by specifying options **+maxdelays**, **+typdelay**, and **+mindelays** at run time. If no option is specified, the typical delay value is the default). This allows the designers the flexibility of building three delay values for each transition into their design. The designer can experiment with delay values without modifying the design.

```
// One delay
// if +mindelays, delay= 4
// if +typdelays, delay= 5
// if +maxdelays, delay= 6
and #(4:5:6) a1(out, i1, i2);

// Two delays
// if +mindelays, rise= 3, fall= 5, turn-off = min(3,5)
// if +typdelays, rise= 4, fall= 6, turn-off = min(4,6)
// if +maxdelays, rise= 5, fall= 7, turn-off = min(5,7)
and #(3:4:5, 5:6:7) a2(out, i1, i2);

// Three delays
// if +mindelays, rise= 2 fall= 3 turn-off = 4
// if +typdelays, rise= 3 fall= 4 turn-off = 5
// if +maxdelays, rise= 4 fall= 5 turn-off = 6
and #(2:3:4, 3:4:5, 4:5:6) a3(out, i1,i2);
```

Q: 2016.7(a): Describe the abstraction levels of Verilog HDL.
Answer:

- **Behavioral or algorithmic level**
  This is the highest level of abstraction provided by Verilog HDL. A module can be implemented in terms of the desired design algorithm without concern for the hardware implementation details. Designing at this level is very similar to C programming.
- **Dataflow level**
  At this level the module is designed by specifying the data flow. The designer is aware of how data flows between hardware registers and how the data is processed in the design.
- **Gate level**
  The module is implemented in terms of logic gates and interconnections between these gates. Design at this level is similar to describing a design in terms of a gate-level logic diagram.
- **Switch level**
  This is the lowest level of abstraction provided by Verilog. A module can be implemented in terms of switches, storage nodes, and the interconnections between them. Design at this level requires knowledge of switch-level implementation details.

Q: 2016.7(b): Write the input-output connection rules of Verilog HDL with example.
Answer:

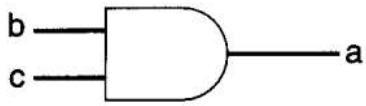
Inputs

Internally, input ports must always be of the type *net*. Externally, the inputs can be connected to a variable which is a **`reg`** or a *net*.

Outputs

Internally, outputs ports can be of the type **`reg`** or *net*. Externally, outputs must always be connected to a *net*. They cannot be connected to a **`reg`**.

Example:



Here input ports (b, c) must always be declared of the type *net.* But they can be connected to a variable which is a *reg* or *net.*

Output port 'a' must be connected to a net. But it can be declared as as *reg* or *net.*

Q: 2016.7(c): What is the difference between the following two lines of Verilog code?
#5 a = 5;
a = #5 b;

Answer:

#5 a = 5;
Wait five time units before doing the action for "a = 5;".
The value assigned to 'a' will be 5 after 5 time units.

a = #5 b;
The value of b is calculated and stored in an internal temporary register. After five time units, assign this stored value to a.

Q: 2017.8(a): What is a hardware description language? What are the requirements of a good HDL?

Answer:
Hardware description language (HDL) is a specialized computer language used to describe the structure and behavior of electronic circuits, and most commonly, digital logic circuits.
It is used for circuit verification and simulation, for timing analysis, for test analysis (testability analysis and fault grading) and for logic synthesis.

Requirements of a good HDL:

1. General purpose
2. Easy to learn and easy to use.
3. Allows different levels of abstraction to be mixed in the same model.
4. Supported by most popular logic synthesis tools.
5. Post-logic-synthesis simulation libraries by all fabrication vendors