Textbook: Verilog® HDL 2nd. Edition
Samir Palnitkar
Prentice-Hall, Inc.

INSTRUCTOR: CHING-LUNG SU
E-mail: kevinsu@yuntech.edu.tw
Chapter 2

Hierarchical Modeling Concepts
Outline of Chapter 2

2.1 Design Methodologies
2.2 4-bit Ripple Carry Counter
2.3 Modules
2.4 Instances
2.5 Components of a Simulation
2.6 Example
2.7 Summary
2.1 Design Methodologies

2.2 4-bit Ripple Carry Counter

2.3 Modules

2.4 Instances

2.5 Components of a Simulation

2.6 Example

2.7 Summary
2.1 Design Methodologies

- Digital Design Methodologies

- Top-down vs. Bottom-up design methodologies
- Top-down Methodology: Designers define the top-level block and identify the sub-blocks necessary to build the top-level block. We divide the sub-modules until we come to leaf cells, which are the cells that cannot further be divided.
- Bottom-up Methodology: Designers first identify the building blocks that are available to us. We build bigger cells, using these blocks. These cells are then used for higher-level blocks until we build the top-level block.
2.1 Design Methodologies

- Top-down Design Methodology
2.1 Design Methodologies

- **Bottom-up Design Methodology**

  - Top-level Block
    - Sub-block 1
      - Leaf Cell
    - Sub-block 1
      - Leaf Cell
    - Sub-block 1
      - Leaf Cell

  - Sub-block 1
    - Leaf Cell
  - Sub-block 1
    - Leaf Cell
  - Sub-block 1
    - Leaf Cell
2.2 4-bit Ripple Carry Counter

2.1 Design Methodologies
2.2 4-bit Ripple Carry Counter
2.3 Modules
2.4 Instances
2.5 Components of a Simulation
2.6 Example
2.7 Summary
2.2 4-bit Ripple Carry Counter

- 4-Bit Ripple Carry Counter

![电路图](image_url)
2.2 4-bit Ripple Carry Counter

- Negative Edge-triggered Toggle Flipflop
2.2 4-bit Ripple Carry Counter

Design Hierarchy

- Ripple Carry Counter
  - T_FF (tff0)
    - D_FF
    - Inverter Gate
  - T_FF (tff1)
    - D_FF
    - Inverter Gate
  - T_FF (tff2)
    - D_FF
    - Inverter Gate
  - T_FF (tff3)
    - D_FF
    - Inverter Gate

Top-down

Bottom-up
2.3 Modules

2.1 Design Methodologies
2.2 4-bit Ripple Carry Counter
2.3 Modules
2.4 Instances
2.5 Components of a Simulation
2.6 Example
2.7 Summary
2.3 Modules

- **Module**
  - Module: Verilog Keyword
  - Module: The basic building block in Verilog
  - Module of lower-level design: Elements are grouped into modules to provide common functionality that is used at many places in design.
  - Module of higher-level design: A module provides the necessary functionality to the higher-level block through its port interface (inputs and outputs), but hides the internal implementation.
2.3 Modules

- **Module Example**

![Diagram showing a module tree with Ripple Carry Counter at the top, followed by T_FF (tff0), T_FF (tff1), T_FF (tff2), and T_FF (tff3) modules, each containing D_FF and Inverter Gate modules.](image-url)
Module in Verilog Files

XXX.v File

```
module <module_name> (<module_terminal_list>)
...

<module internal>
...

endmodule
```

Keyword
2.3 Modules

- **T-flipflop Module in Verilog File**

XXX.v File

```verilog
module T_FF (q, clock, reset)
...
<function of T-flipflop>
...
endmodule
```

Diagram:
- T_FF module
- D_FF module
- Clock input
- Reset input
- Q output
2.3 Modules

- **4 Abstraction levels in Verilog**

  - **Behavioral or Algorithmic Level**: A module can be implemented in terms of the desired design algorithm without concern for the hardware implementation details.

  - **Dataflow Level**: 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.

  - **Switch Level**: A module can be implemented in terms of switches, storage nodes, and the interconnections between them.
2.3 Modules

- **Verilog Supports**

- Verilog allows the designer to mix and match all four levels of abstractions in a design.
- The (Register Transfer Level) RTL is a combination of behavioral and dataflow constructs for synthesis tools.
- The **higher Level of Abstraction**: The more flexible and technology-independent.
- The **lower Level of Abstraction**: The more technology-dependent and inflexible.
2.4 Instances

2.1 Design Methodologies
2.2 4-bit Ripple Carry Counter
2.3 Modules
2.4 Instances
2.5 Components of a Simulation
2.6 Example
2.7 Summary
2.4 Instances

Module Design vs. Instance

Ripple Counter Circuit

XXX.v

module ripple_counter ()
...
T_FF tff0 (? )
T_FF tff1 (? )
T_FF tff2 (? )
T_FF tff3 (? )
endmodule

XXX.v

module T_FF ()

endmodule
2.4 Instances

Example

Module TFF

```verilog
define module TFF(q, clk, reset);
output q;
input clk, reset;
wire d;
DFF dff0(q, d, clk, reset);
not n1(d, q);
endmodule
```

Single-Line Comments

```
// not is a Verilog provided primitive.
```

Module ripple_carry_counter

```verilog
define module ripple_carry_counter(q, clk, reset);
output [3:0] q;
input clk, reset;
// 4 instances of the module TFF are created.
TFF tff0(q[0], clk, reset);
TFF tff1(q[1], q[0], reset);
TFF tff2(q[2], q[1], reset);
TFF tff3(q[3], q[2], reset);
endmodule
```

4 TFF Instances
Verilog Illegal: Nesting Module

// This file is not simulatable. It is intended merely
// to demonstrate that module nesting is illegal.

// Define the top level module called ripple carry
// counter. It is illegal to define the module T_FF inside this module.
module ripple_carry_counter(q, clk, reset);
output [3:0] q;
input clk, reset;
module T_FF(q, clock, reset);
<module T_FF internals>
endmodule
endmodule

// END OF ILLEGAL MODULE NESTING

module T_FF(q, clock, reset);
<module T_FF internals>
endmodule

// ILLEGAL MODULE NESTING
2.1 Design Methodologies
2.2 4-bit Ripple Carry Counter
2.3 Modules
2.4 Instances
2.5 Components of a Simulation
2.6 Example
2.7 Summary
2.5 Components of a Simulation

- Circuit Design vs. Stimulus Block

Stimulus Block

Input Test Signals

Clock, Reset, ...

Block Design

ripple_carry_counter

Test Outputs