Introduction to Digital VLSI Design

Verilog – Hierarchical Modeling Concept

Lecturer: Gil Rahav
Semester B’, EE Dept. BGU.
Freescale Semiconductors Israel
Objectives

- Understand top-down and bottom-up design methodologies for digital design
- Explain differences between modules and module instances in Verilog
- Describe four levels of abstraction – behavioral, data flow, gate level, and switch level – to represent the same module
- Describe components required for the simulation of a digital design
A top-down design methodology

- Define the top-level block and identify the sub-blocks necessary to build the top-level block. Subdivide the sub blocks until will come to leaf cells (can’t be divided)
A **bottom-up** design methodology

![Diagram of a bottom-up design methodology]

- Identify the building blocks that are available and build bigger cells using these building blocks. These cells are then used for higher-level blocks until the top-level block in the design will be built.
Verilog Hierarchical Modeling Concept

- Verilog is both behavioral and structural language

- Internals of each module can be defined at four levels of abstraction (depending on the needs of the design)
  - Behavioral or algorithmic level
  - Dataflow level
  - Gate level
  - Switch level

- The level of abstraction to describe a module can be changed without any change in the environment

- Verilog allows the designer to mix and match all four levels of abstraction in a design
Typically, a combination of top-down and bottom-up flows is used

- **Design architect** define the specifications of the top-level block

- **Logic designers** decide how the design should be structured by breaking up the functionality into blocks and sub-blocks

- **Circuit designers** are designing optimized circuits for leaf-level cells and build higher-level cells by using these leaf cells

- The flow meets at intermediate point where the switch-level circuit designers have created a library of leaf cells by using switches, and the logic level designers have designed from top-down until all modules are defined in terms of leaf cells
A module is a basic building block in Verilog

A module can be element or a collection of low-level design blocks

Elements are grouped into modules to provide common functionality that is used in many places in the design

A module provides the necessary functionality to the higher level blocks through its port interface (inputs and outputs), but hides the internal implementation

Designer can modify module internals without affecting the rest of design
Verilog Abstraction Levels

1. Behavioral level
   ✓ The highest level of abstraction provided by Verilog HDL
   ✓ A module can be implemented in terms of the desired algorithm without concern for the hardware implementation details
   ✓ Very similar to C language programming

2. RTL level
   ✓ Verilog description that is acceptable by synthesis tool.
3. **Gate level**
   - The module implemented in terms of logic gates and interconnections between these gates
   - Similar to describing a design in terms of gate-level logic diagram

4. **Switch level**
   - The lowest level of abstraction provided by Verilog HDL
   - A module can be implemented in terms of switches, storage nodes, and interconnection between them
   - Design at this level required knowledge of switch level implementation details
Verilog Abstraction Levels

- The higher (behavioral) level of abstraction makes the design more flexible and technology independent.

- As design goes lower toward switch-level, it becomes technology dependent and inflexible.
Instances in Verilog

- A **module** provide a template from which you can create actual objects.

- When a module is invoked, Verilog creates a unique object from the template.

- Each object has it’s own name, variables, parameters and I/O interface.

- The process of creating objects from a module template is called **instantiation**, and the objects are called **instances**.
// Define the top-level module called ripple carry counter.
// It instantiates 4 T-flip-flops.
module ripple_carry_counter (q, clk, reset)

input clk, reset; // I/O signals (explained later)
output [3:0] q;  // I/O signals and vector explanation

/* Four instances of the module T_FF are created. Each has a unique name. Each instance is passed a set of signals. Notice, that each instance is a copy of the module T_FF. */
T_FF tff_0 (q[0],clk,reset);
T_FF tff_1 (q[1],q[0],reset);
T_FF tff_2 (q[2],q[1],reset);
T_FF tff_3 (q[3],q[2],reset);

endmodule
Instances in Verilog (example cont.)

/* Define the module T_FF It instantiates a D-flipflop. We assumed that module D-flipflop is defined elsewhere in the design. */

module T_FF (q, q_b, clk, reset)
    input clk, reset; // I/O signals
    output q, q_b; // I/O signals and vector explanation
    wire d;

    D_FF dff_0 (q,d,clk,reset); // Instantiate D_FF. Call it dff_0.
    not n1 (q_b,q); // not gate is a Verilog primitive (explained later)

endmodule
What do you do to test a software program you write?

✓ Give it some inputs, and see if it does what you expect

Simulation tests a model of the system you wish to build
Simulation of Digital Systems

Simulation checks two properties

- functional correctness - is the logic correct?
- timing correctness - is the logic/interconnect timing correct (e.g. are the set-up times met)?

It has all the limitations of software testing

- Have I tried all the cases?
- Have I exercised every path and every option?
Simulation Algorithms

- Time driven
  - used by SPICE simulators

- Event driven
  - used by Verilog-XL, NC-Verilog, and VCS simulators

- Cycle based
  - used by UnitSIM, System-C, VERA simulators

- Demand driven
  - ideal, unattainable
Components of Simulation

- Once a design block is completed – it must be tested
- The functionality of design block can be tested by applying stimulus and checking results
- The stimulus block (also commonly called a testbench) can be written in Verilog
- Two styles of stimulus application are possible:
  - The stimulus block instantiates the design block and directly drives the signals in the design block
  - Instantiate both the stimulus and design blocks in a top-level dummy model
- It is good practice to keep the stimulus and design blocks separate
Two Styles of Stimulus Application

- The **stimulus** block instantiates the **design** block and directly drives the signals in the **design** block.

- Instantiate both the **stimulus** and design blocks in a top-level dummy model.
Hierarchical Modeling Concept Summary

- Two kinds of design methodologies are used for digital design: top-down and bottom-up. A combination of these two methodologies is used in today’s digital design. As designs become very complex, it is important to follow these structure approaches to manage the design process.

- Modules are the basic building blocks in Verilog. Modules are used in a design by instantiation. An instance of a module has a unique identity and is different from other instances of the same module. Each instance has an independent copy of the internals of the module. It is important to understand the difference between modules and instances.

- There are two distinct components in a simulation: a design block and a stimulus block. A stimulus block (usually the top-level block) is used to test the design block. There are two different styles of applying stimulus to a design block.