Chapter 2

# Data Types

## 2.1 Introduction

SystemVerilog offers many improved data structures compared with Verilog. Some of these were created for designers but are also useful for testbenches. In this chapter you will learn about the data structures most useful for verification.

SystemVerilog introduces new data types with the following benefits.

- Two-state: better performance, reduced memory usage
- Queues, dynamic and associative arrays and automatic storage: reduced memory usage, built-in support for searching and sorting
- Unions and packed structures: allows multiple views of the same data
- Classes and structures: support for abstract data structures
- Strings: built-in string support
- Enumerated types: code is easier to write and understand

## 2.2 Built-in Data Types

Verilog-1995 has two basic data types: variables (**`reg`**) and nets, that hold four-state values: 0, 1, Z, and X. RTL code uses variables to store combinational and sequential values. Variables can be unsigned single or multi-bit (**`reg [7:0] m`**), signed 32-bit variables (**`integer`**), unsigned 64-bit variables (**`time`**), and floating point numbers (**`real`**). Variables can be grouped together into arrays that have a fixed size. All storage is static, meaning that all variables are alive for the entire simulation and routines cannot use a stack to hold arguments and local values. A net is used to connect parts of a design such as gate primitives and module instances. Nets come in many flavors, but most designers use scalar and vector wires to connect together the ports of design blocks.

SystemVerilog adds many new data types to help both hardware designers and verification engineers.

### 2.2.1 The logic type

The one thing in Verilog that always leaves new users scratching their heads is the difference between a **`reg`** and a **`wire`**. When driving a port,