Verilog

Introduction

- Originally intended for simulation and verification of digital circuits
- With the addition of synthesis capability, it has become popular for use in design entry for CAD systems
  - These CAD systems are used to synthesize Verilog into a hardware implementation of the described circuit

Representation of Digital Circuits

- When using CAD tools to synthesize a logic circuit, the designer can provide the initial description of the circuit in several ways, one way being Verilog source code
- The Verilog compiler translates this code into a logic circuit
- Verilog allows the designer to describe a desired circuit in many ways
  - One way is to use Verilog constructs that describe of the circuit in terms of circuit elements
- A larger circuit defined by writing code that connects such elements together
  - This approach is referred to as the *structural* representation of logic circuits
- Another possibility is to describe a circuit more abstractly by using logic expressions and Verilog programming constructs that define the desired behavior of the circuit, but not its actual structure in terms of gates
  - This is called the *behavioral* representation

Structural Specification of Logic Circuits

Gates

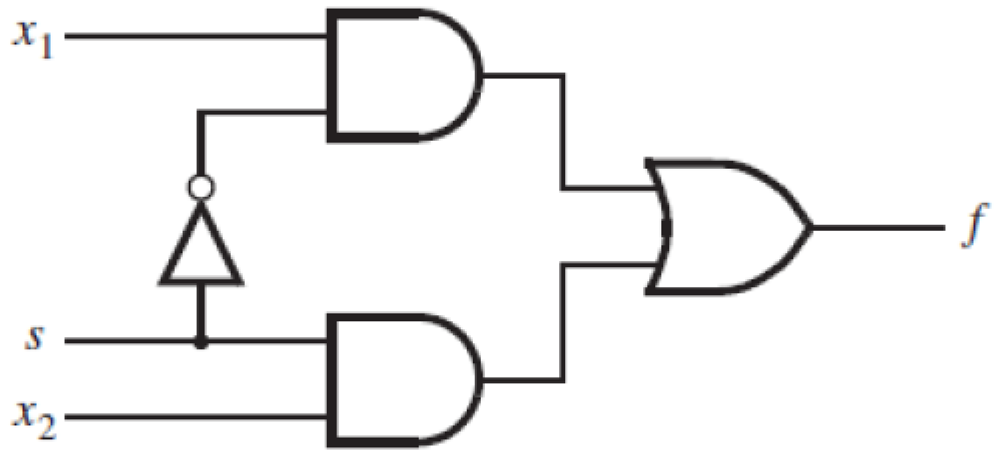
- Verilog includes a set of *gate-level primitives* that correspond to commonly-used logic gates
- A gate is represented by indicating functional name, output, and inputs

Examples

- Two input AND gate with output y and inputs x_1, x_2
  - `and(y, x1, x2)`
- Four input OR gate
  - `or(y, x1, x2, x3, x4)`
- One input NOT gate implementing $y = \overline{x}$
  - `not(y, x)`

Syntax

Example



```
module example1(x1, x2, s, f):
    input x1, x2, s;
    output f;
    not(k, s);
    and(g, k, x1);
    and(h, s, x2);
    or(f, g, h);
endmodule
```

This is a multiplexer

Model Sim

-