Digital Logic Design

Ryan Hou

2024-12-21

Table of contents

Preface

This is my notes on digital logic design.

Resources

Some relevant resources:

- EECS 270 - Logic Design (University of Michigan)
- Digital Design and Computer Architecture (ETH Zurich)

Textbooks:

- J. F. Wakerly, Digital Design: Principles and Practices, 4th ed., Prentice-Hall.
- J. P. Hayes, Introduction to Digital Logic Design, Addison-Wesley.
- C. H. Roth, Jr., Fundamentals of Logic Design.
- R. H. Katz, Contemporary Logic Design, Prentice-Hall.
- D. Thomas, P. Moorby, The Verilog Hardware Description Language.

1 Introduction

1.1 Perspective

Figure 1: Digital Logic Design in the Computing Stack

> Note 1: Definition - Digital
>
> **Digital** signals can only take on discrete binary values in one of two states: 0 (low, false) or 1 (high, true).

This notebook focuses on the design of **digital circuits**. We study both the logic/math used to build digital systems (Boolean algebra), as well as the circuit design implications (transistors, timing, etc).

1.2 Analog vs Digital

In contrast to the discrete **digital** signals, **analog** signals are *continuous*. Signals from the physical world are inherently analog (e.g. sound, light, temperature, voltage). However, modern computing systems are primarily digital because of several key advantages:

- Reliability: Provides more noise resistance since it operates at low or high levels
- Digitized signals can represent analog values with good precision given enough digits
- Ease of data storage, transmission, and compression
- Digital circuit components are more cost-effective and scalable compared to analog components

1.3 (Very) High-Level Digital Circuit Design Flow

Insert high-level design flow figure here

The design flow of a digital circuit starts off with a problem statement or design specification. Digital circuits are then described by the designer in a **Hardware Description Language (HDL)**, most commonly **Verilog/SystemVerilog** or **VHDL**. The design is then simulated with a **testbench**, which feeds the design with test inputs. During **simulation**, we can use tools to inspect the state of the signals in the circuit to analyze, debug, and evaluate the design. At this point, such a **behavioral** description of the design merely describes the functionality and not yet its physical implementation. **Synthesis** maps the behavioral description of the design into **netlist** of standard cells, which indicates the physical mapping to circuit components. The **place and route** process then physically places the netlist of cells and routes the wires to connect the components, generating a hardware implementation.

This notebook will cover basic Verilog. For more in-depth notes on Verilog/SystemVerilog, please refer to my other notebook.

2 Binary Basics

2.1

3 Boolean Algebra

3.1

4 Combinational Logic

4.1

5 Timing

5.1

6 Sequential Logic

6.1

7 Finite State Machines

7.1

8 Digital Arithmetic

8.1

9 Memories

10 Summary

In summary...

References