Digital Logic Design

Ryan Hou

2024-12-21

Table of contents

Preface **3**

Resources **4**

1 Introduction **5**
1.1 Perspective 5
1.2 (Very) High-Level Digital Circuit Design Flow 6

2 Binary Basics **7**
2.1 Analog vs Digital 7
2.2 Why Binary? 7
2.3 Data Encoding 7
2.3.1 Conversions 8
2.3.2 Hex - Octal - Binary 9
2.3.3 ASCII 9

3 Boolean Algebra **10**

4 Combinational Logic **11**
4.2 Transistors 11
4.3 Transistors to Gates 11
4.4 Transistor Scaling 11

5 Timing **12**

6 Sequential Logic **13**

7 Finite State Machines **14**

8 Digital Arithmetic **15**

9 Memories **16**

10 Summary **17**

References **18**

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

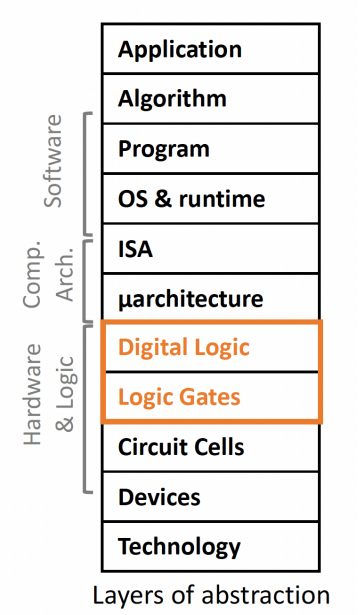

Figure 1.1: Digital Logic Design in the Computing Stack (figure from EECS270-W24)

> **Note 1: Definition - Digital**
>
> **Digital** signals represent information as *discrete* values, typically binary values where two valid states exist: 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 (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 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

2.2 Why Binary?

- Storing/transmitting binary values is much easier than three or more values
- Binary switches are easier, more robust, and more noise tolerant in circuit implementation

Note that digital binary!

2.3 Data Encoding

Numbers are encoded in a system using digits and powers of a base number. In simpler terms, each position of a number represents a quantity. And the digit in each position indicates how many of that quantity there are in the number.

A **bit** is a binary digit. The total number of integers that can be represented with n bits is 2^n.

The maximum (unsigned) decimal number that can be represented with n bits is Max Value $= 2^n - 1$. This range can be generalized to other bases:

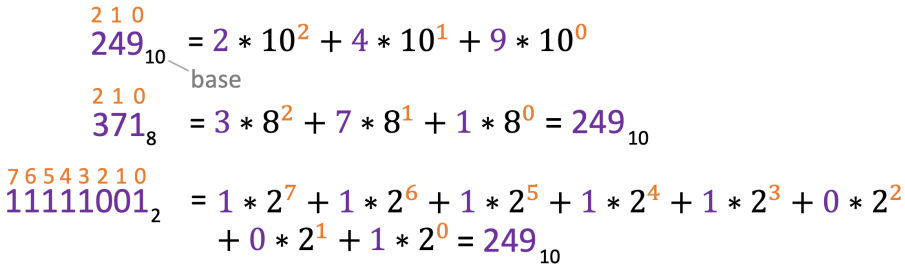

Figure 2.1: Data encoding in base 10, 8, and 2. Figure from EECS270-W24

> **Note 2:** Equation - Max Unsigned Decimal Value with n Digits
>
> $$\text{Max Value} = base^n - 1$$

Common number systems:

- Base-16: **Hexadecimal**
- Base-10: **Decimal**
- Base-8: **Octal**
- Base-2: **Binary**

The number of bits n needed to represent an unsigned decimal number x is given below:

> **Note 3:** Equation - Number of Bits to Represent Unsigned Decimal Number
>
> $$n = ceil(log_2(x + 1))$$
>
> where the ceil() function is a ceiling function that rounds up to the nearest integer.

2.3.1 Conversions

2.3.1.1 Decimal - Binary

To convert from decimal to binary:

- Step 1: Divide the given number repeatedly by 2 until you get 0 as the quotient.
- Step 2: Write the remainders in reverse order.

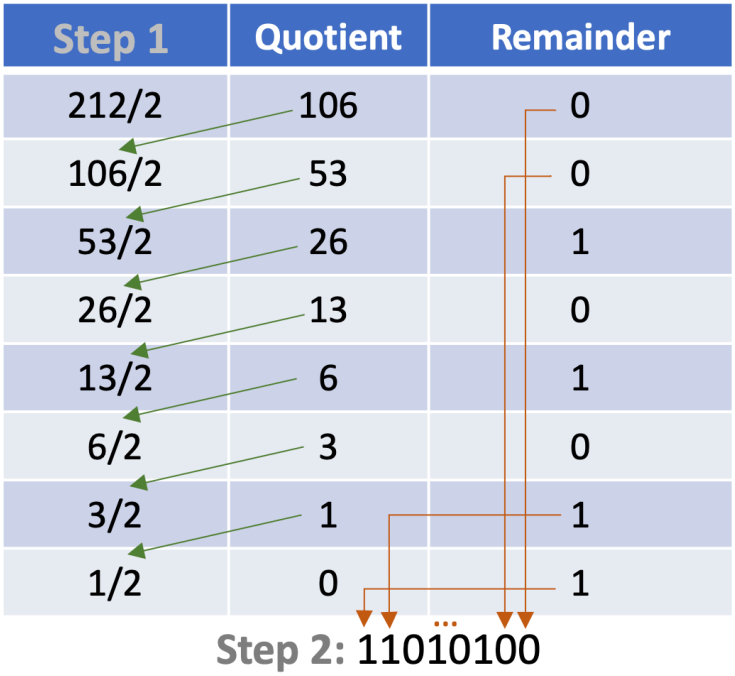

Figure 2.2: Example decimal to binary conversion

2.3.2 Hex - Octal - Binary

Hexadecimal and octal have bases that are powers of 2, which makes conversion much simpler. Since hex is base 16, which is 2^4, we can split each hex digit into 4 bits when converting to binary, conversely group every 4 bits into 1 hex digit. Similarly, an octal digit corresponds to 3 bits.

*TODO: Add hex conversion chart.

2.3.3 ASCII

Text can also be encoded by numbers. ASCII is a common character encoding standard that represents a character in 8 bits.

3 Boolean Algebra

3.1

4 Combinational Logic

> Note 4: Definition - Combinational Logic
>
> **Combinational Logic**: output is a pure function of the present input only.

4.1

4.2 Transistors

4.3 Transistors to Gates

4.4 Transistor Scaling

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