

BBM 231

Logic Design

section instructor:
Ufuk Çelikcan

Lectures

- Mondays: **13:00 – 16:00**
- Attendance is not mandatory in this section but highly advised

BBM 233 Lab

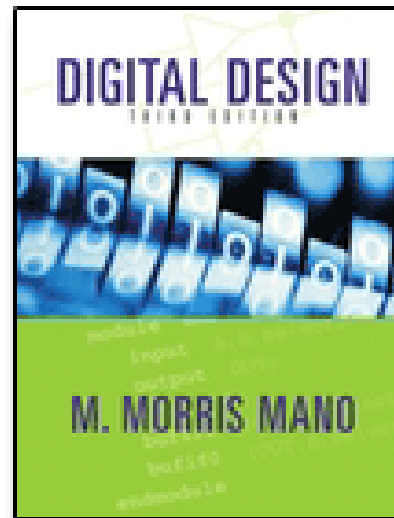
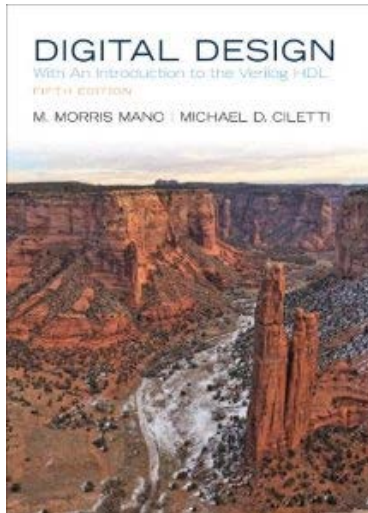
- Different sections
 - Check from the dept. website
- 7 experiments
- Once in two weeks
- It is obligatory to do all the assignments
- See assistants for grading scheme
- Work in groups of 2

Grading

- 2 midterm exams
 - Weight: 25% each
- Final exam
 - Weight: 40%
 - As scheduled by the registration office
- Homeworks:
 - 6 HWs as prep material for exams
 - Total Weight: 10%

Textbook & References

- Textbook
 - M. Morris Mano, **Digital Design: With an Introduction to the Verilog HDL, 5th Edition**, Prentice Hall, 2013.
- Other references
 - Tons of digital design books
 - Lectures from MIT Open Courseware and Stanford



Contact Information

- Ufuk Çelikcan
- e-mail: ufuk.celikcan@gmail.com
- Office hours:
 - send e-mail to schedule meeting

Motivation

- Analysis & design of digital electronic circuits
- Why digital circuits?
 - They are everywhere and generic
 - digital computers, smart phones, data communication, digital recording, digital TV, many others
- Fundamental concepts in the design of digital systems
- Basic tools for the design of digital circuits
- Logic gates (AND, OR, NOT)
 - Boolean algebra



What is a Digital System?

- One characteristic:
 - Ability of manipulating discrete elements of information
- A *set* that has a finite number of elements contains discrete information
- Examples for discrete sets
 - Decimal digits {0, 1, ..., 9}
 - Alphabet {A, B, ..., Y, Z}
 - Binary digits {0, 1}
- One important problem
 - how to represent the elements of discrete sets in physical systems?



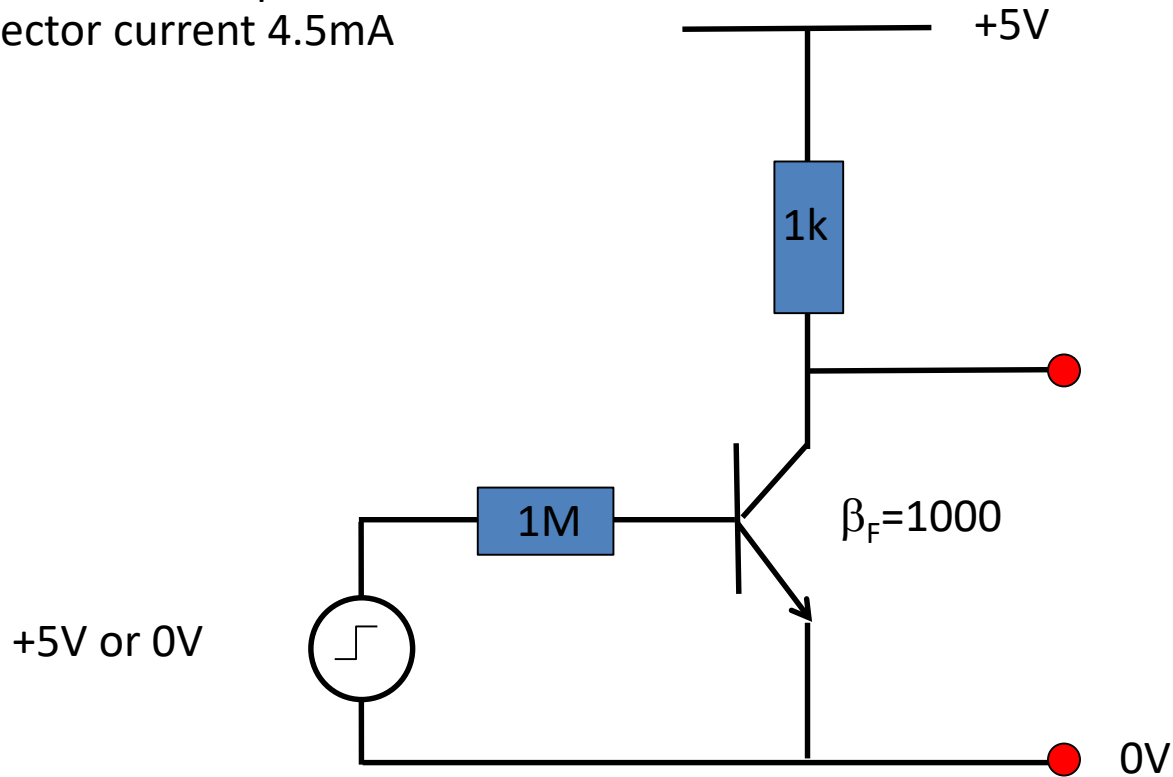
How to Represent?

- In electronics circuits, we have electrical signals
 - voltage
 - current
- Different strengths of a physical signal can be used to represent elements of the discrete set.
- Which discrete set?
- Binary set is the easiest
 - two elements $\{0, 1\}$
 - Just two signal levels: 0 V and 5 V
- This is why we use binary system to represent the information in digital systems.



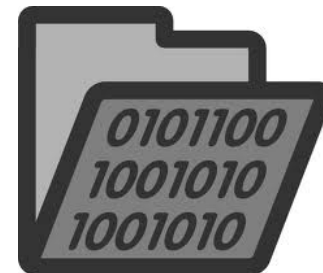
How to Represent?

- In electronics circuits, we have electrical signals
 - voltage
 - current
 - Base current $4.5 \mu\text{A}$
 - Collector current 4.5mA



Binary System

- Binary set {0, 1}
 - The elements of binary set, 0 and 1 are called “binary digits”
 - or shortly “bits”.
- How to represent the elements of other discrete sets
 - Decimal digits {0, 1, ..., 9}
 - Alphabet {A, B, ..., Y, Z}
- Elements of any discrete set can be represented using groups of bits.
 - 9 → 1001
 - A → 1010



How Many Bits?

- What is the formulae for number of bits to represent a discrete set of n elements
- $\{0, 1, 2, 3\}$
 - $00 \rightarrow 0$, $01 \rightarrow 1$, $10 \rightarrow 2$, and $11 \rightarrow 3$
- $\{0, 1, 2, 3, 4, 5, 6, 7\}$
 - $000 \rightarrow 0$, $001 \rightarrow 1$, $010 \rightarrow 2$, and $011 \rightarrow 3$
 - $100 \rightarrow 4$, $101 \rightarrow 5$, $110 \rightarrow 6$, and $111 \rightarrow 7$.
- The formulae, then,
 - #of bits required = $\lceil \log_2 \text{\#of elements(symbols)} \rceil$
 - If $n = 9$, then ? bits are needed

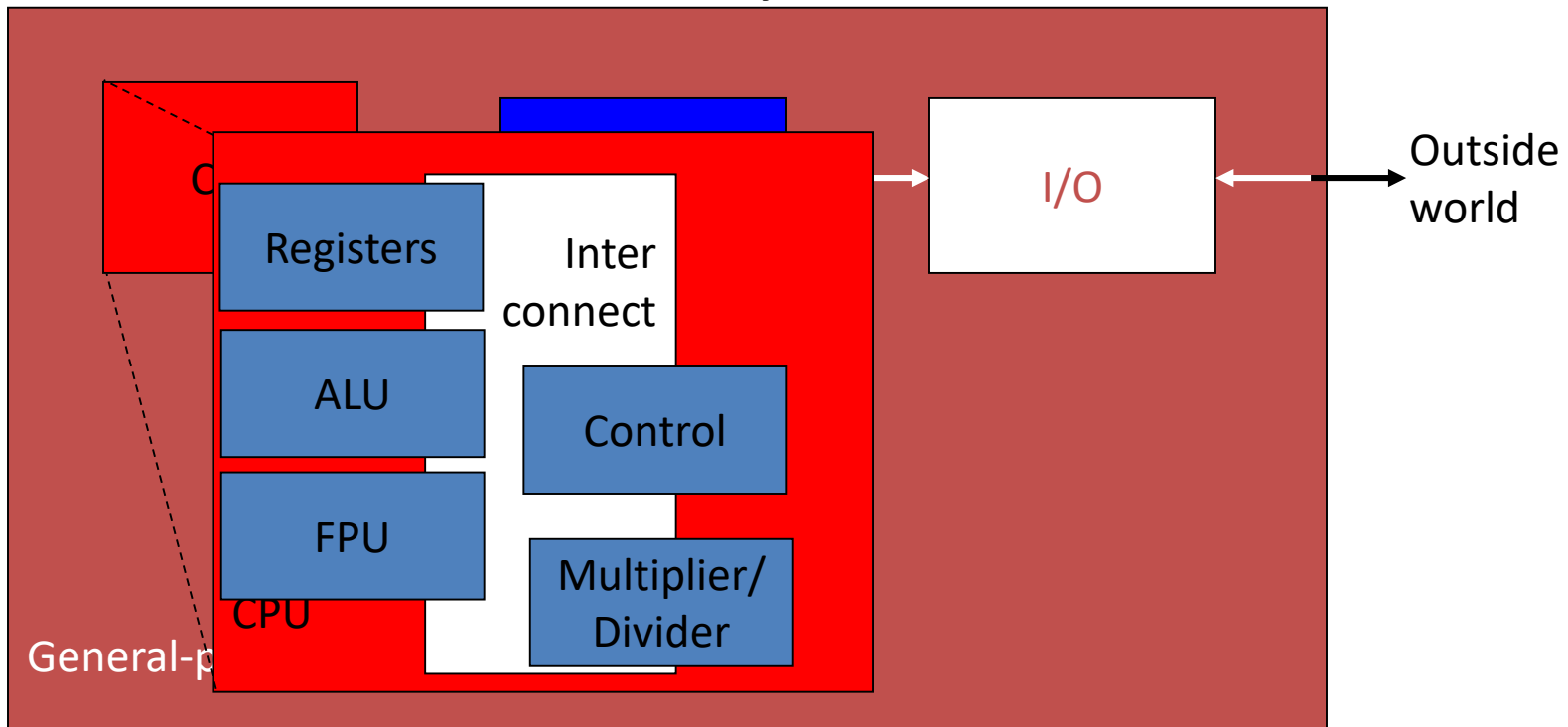
Nature of Information



- Is information of discrete nature?
- Sometimes, but usually not.
 - Anything related to money (e.g. financial computations, accounting etc) involves discrete information
- In nature, information comes in a continuous form
 - temperature, humidity level, air pressure, etc.
- Continuous data must be converted (i.e. quantized) into discrete data
 - lost of some of the information
 - We need ADC (DAC)

General-Purpose Computers

- Best known example for digital systems
- Components
 - CPU, I/O units, Memory unit



Contents

- **1 Digital Systems and Binary Numbers 1**
 - 1.1 Digital Systems *1*
 - 1.2 Binary Numbers *3*
 - 1.3 Number-Base Conversions *6*
 - 1.4 Octal and Hexadecimal Numbers *8*
 - 1.5 Complements of Numbers *10*
 - 1.6 Signed Binary Numbers *14*
 - 1.7 Binary Codes *18*
 - 1.8 Binary Storage and Registers *27*
 - 1.9 Binary Logic *30*

Contents

- **2 Boolean Algebra and Logic Gates 38**
 - 2.1 Introduction 38
 - 2.2 Basic Definitions 38
 - 2.3 Axiomatic Definition of Boolean Algebra 40
 - 2.4 Basic Theorems and Properties of Boolean Algebra 43
 - 2.5 Boolean Functions 46
 - 2.6 Canonical and Standard Forms 51
 - 2.7 Other Logic Operations 58
 - 2.8 Digital Logic Gates 60
 - 2.9 Integrated Circuits 66

Contents

- **3 Gate-Level Minimization 73**
 - 3.1 Introduction 73
 - 3.2 The Map Method 73
 - 3.3 Four-Variable K-Map 80
 - 3.4 Product-of-Sums Simplification 84
 - 3.5 Don't-Care Conditions 88
 - 3.6 NAND and NOR Implementation 90
 - 3.7 Other Two-Level Implementations 97
 - 3.8 Exclusive-OR Function 103
 - 3.9 Hardware Description Language 108

Contents

- **4 Combinational Logic 125**
 - 4.1 Introduction 125
 - 4.2 Combinational Circuits 125
 - 4.3 Analysis Procedure 126
 - 4.4 Design Procedure 129
 - 4.5 Binary Adder–Subtractor 133
 - 4.6 Decimal Adder 144
 - 4.7 Binary Multiplier 146
 - 4.8 Magnitude Comparator 148
 - 4.9 Decoders 150
 - 4.10 Encoders 155
 - 4.11 Multiplexers 158
 - 4.12 HDL Models of Combinational Circuits 164

Contents

- **5 Synchronous Sequential Logic 190**
 - 5.1 Introduction *190*
 - 5.2 Sequential Circuits *190*
 - 5.3 Storage Elements: Latches *193*
 - 5.4 Storage Elements: Flip-Flops *196*
 - 5.5 Analysis of Clocked Sequential Circuits *204*
 - 5.6 Synthesizable HDL Models of Sequential Circuits *217*
 - 5.7 State Reduction and Assignment *231*
 - 5.8 Design Procedure *236*

Contents

- **6 Registers and Counters 255**
 - 6.1 Registers 255
 - 6.2 Shift Registers 258
 - 6.3 Ripple Counters 266
 - 6.4 Synchronous Counters 271
 - 6.5 Other Counters 278
 - 6.6 HDL for Registers and Counters 283

Contents

- **7 Memory and Programmable Logic 299**
 - 7.1 Introduction 299
 - 7.2 Random-Access Memory 300
 - 7.3 Memory Decoding 307
 - 7.4 Error Detection and Correction 312
 - 7.5 Read-Only Memory 315
 - 7.6 Programmable Logic Array 321
 - 7.7 Programmable Array Logic 325
 - 7.8 Sequential Programmable Devices 329

Contents (If time permits)

- **8 Design at the Register Transfer Level 351**
 - 8.1 Introduction 351
 - 8.2 Register Transfer Level Notation 351
 - 8.3 Register Transfer Level in HDL 354
 - 8.4 Algorithmic State Machines (ASMs) 363
 - 8.5 Design Example (ASMD Chart) 371
 - 8.6 HDL Description of Design Example 381
 - 8.7 Sequential Binary Multiplier 391
 - 8.8 Control Logic 396
 - 8.9 HDL Description of Binary Multiplier 402
 - 8.10 Design with Multiplexers 411
 - 8.11 Race-Free Design (Software Race Conditions) 422
 - 8.12 Latch-Free Design (Why Waste Silicon?) 425
 - 8.13 Other Language Features 426