Digital Electronics

Dr. I. J. Wassell

Digital Electronics

Introduction

Aims

- To familiarise students with
  - Combinational logic circuits
  - Sequential logic circuits
  - How digital logic gates are built using transistors
  - Simple processor architectures
  - Design and build of digital logic systems

Course Structure

- 12 Lectures
- Hardware Labs
  - 4 Workshops
  - Each Workshop lasts 2.5h
  - In Intel Lab. (SW11), William Gates Building (WGB)
  - Done individually
  - Lab. Sessions begin in week 3
  - COVID may force move to simulation

Objectives

- At the end of the course you should
  - Be able to design and construct simple digital electronic systems
  - Be able to understand and apply Boolean logic and algebra – a core competence in Computer Science
  - Be able to understand and build state machines

Books

- Lots of books on digital electronics, e.g.,
  - D. M. Harris and S. L. Harris, 'Digital Design and Computer Architecture,' Morgan Kaufmann, 2007 (1st Ed.), 2012 (2nd Ed.).
  - R. H. Katz, 'Contemporary Logic Design,' Benjamin/Cummings, 1994.
  - J. P. Hayes, 'Introduction to Digital Logic Design,' Addison-Wesley, 1993.
- Electronics in general (inc. digital)
  - P. Horowitz and W. Hill, 'The Art of Electronics,' CUP, 1989.

Simulation Software

- There are a number of packages available that enable simulation of digital electronic circuits using a graphical interface e.g.,
  - National Instruments (NI) Multisim
  - Yenka Electronics (Technology Package)
- The former is much more powerful (and expensive), but the latter is relatively straightforward to use and is free to use (except between 8.30 and 15.00)
- You may have used Yenka Electronics at school. It is free to download

Other Points

- This course is a prerequisite for
  - Computer Design, ECAD and Architecture Practical Classes (Part IB)
  - Comparative Architectures (Part II)
  - Hardware Security, Advanced Topics in Computer Architecture (MPhil/Part III)
- Keep up with lab work and get it ticked.
- Have a go at supervision questions plus any others your supervisor sets.
- Remember to try questions from past papers

The Bigger Picture

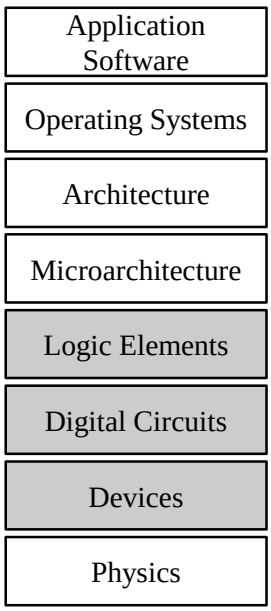
- As you may be aware, probably the most significant application of digital logic is to implement *microprocessors* and microprocessor based computer systems.
- However, digital logic is also employed to build a wide variety of *other* electronic systems that are not microprocessor based.

Managing Complexity

- Modern digital systems e.g., microprocessors, are typically built from millions of transistors.
- It would be impossible for a human to design such a system by for example, writing equations describing the movement of electrons in each transistor and then attempting to solve the equations simultaneously.
- We have to manage complexity in order that we are not swamped in a mass of detail.
- To do this we employ *abstraction*.

Abstraction

- Abstraction, i.e., hiding details when they are not important.
- Indeed a system can be viewed from many different levels of abstraction.
- For example, for an electronic computing system, we can consider levels of abstraction from pure physics (electrons) at the bottom level through to application software (programs) at the top level.
- In this course we will primarily be considering *Devices*, *Digital Circuits* and *Logic Elements* levels of abstraction.



Programs – Application software uses facilities provided by OS to solve a problem for the user

Device drivers – Handles low-level details such as accessing a hard drive or managing memory

Instructions, Registers – e.g., Intel-IA32 defined by a set of instructions and registers

Data paths, Controllers – Combines logic elements to execute instructions defined by the architecture

Adders, Memories, etc. – Complex structures put together from digital circuits

Gates, e.g., AND, NOT – Devices assembled to create 'digital' components

Transistors – well defined I/V characteristics between input/output terminals

Electrons – quantum mechanics, Maxwell's equations

Abstraction

- So the point is that you can browse the web without any regard quantum theory or the organisation of memory in the computer.
- That said, when working at a particular level of abstraction, it is good to know something about the levels of abstraction immediately above and below where you are working, e.g.,
  - A device designer needs to understand the circuits in which it will be used,
  - Code cannot be optimised without understanding the architecture for which it is being written.

Microprocessor

- Defined by its *architecture* and *microarchitecture*
- The *architecture* is defined by its instruction set and registers
- The *microarchitecture* is the specific arrangement of registers, arithmetic logic units (ALUs), controllers, multiplexers, memories and other logic blocks needed to implement a particular architecture.
- Note that a particular architecture may be implemented by many different microarchitectures, each having different trade-offs of performance, complexity and cost.