7.1 Introduction

(Original section[1])

> “ I always loved that word, Boolean.
> *Claude Shannon, IEEE Spectrum, April 1992. (Shannon's master's thesis showed that the algebra invented by George Boole in the 1800s could represent the workings of electrical switches.)*

This appendix provides a brief discussion of the basics of logic design. It does not replace a course in logic design, nor will it enable you to design significant working logic systems. If you have little or no exposure to logic design, however, this appendix will provide sufficient background to understand all the material in this book. In addition, if you are looking to understand some of the motivation behind how computers are implemented, this material will serve as a useful introduction. If your curiosity is aroused but not sated by this appendix, the references at the end provide several additional sources of information.

COD Sections A.2 (Gates, truth tables, and logic equations) introduces the basic building blocks of logic, namely, *gates*. COD Sections A.3 (Combinational logic) uses these building blocks to construct simple *combinational* logic systems, which contain no memory. If you have had some exposure to logic or digital systems, you will probably be familiar with the material in these first two sections. COD Sections A.5 (Constructing a basic arithmetic logic unit) shows how to use the concepts of COD Sections A.2 (Gates, truth tables, and logic equations) and A.3 (Combinational logic) to design an ALU for the LEGv8 processor. COD Sections A.6 (Faster addition: Carry lookahead) shows how to make a fast adder, and may be safely skipped if you are not interested in this topic. COD Sections A.7 (Clocks) is a short introduction to the topic of clocking, which is necessary to discuss how memory elements work. COD Sections A.8 (Memory elements: Flip-flops, latches, and registers) introduces memory elements, and COD Sections A.9 (Memory elements: SRAMs and DRAMs) extends it to focus on random access memories; it describes both the characteristics that are important to understanding how they are used, as discussed in COD Chapter 4 (The processor), and the background that motivates many of the aspects of memory hierarchy design discussed in COD Chapter 5 (Large and fast: Exploiting memory hierarchy). COD Section A.10 (Finite-state machines) describes the design and use of finite-state machines, which are sequential logic blocks. If you intend to read COD Appendix C (Mapping control to hardware), you should thoroughly understand the material in COD Sections A.2 (Gates, truth tables, and logic equations) through COD Sections A.10 (Finite-state machines). If you intend to read only the material on control in COD Chapter 4 (The processor), you can skim the appendices; however, you should have some familiarity with all the material except COD Sections A.11 (Timing methodologies). COD Sections A.11 (Timing methodologies) is intended for those who want a deeper understanding of clocking methodologies and timing. It explains the basics of how edge-triggered clocking works, introduces another clocking scheme, and briefly describes the problem of synchronizing asynchronous inputs.

Throughout this appendix, where it is appropriate, we also include segments to demonstrate how logic can be represented in Verilog, which we introduce in COD Sections A.4 (Using a hardware description language).

(*1) This section is in original form.

Provide feedback on this section