This book is written for young professionals and graduate students who have prior logic design background and want to learn how to use logic blocks to build complete systems from design specifications. My two-decade-long industry experience has taught me that engineers are “shape-oriented” people and that they tend to learn from charts and diagrams. Therefore, the teaching method I followed in this textbook caters this mind set: a lot of circuit schematics, block diagrams, timing diagrams, and examples supported by minimal text.

The book has eight chapters. The first three chapters give a complete review of the logic design principles since rest of the chapters significantly depend on this review. Chapter 1 concentrates on the combinational logic design. It describes basic logic gates, De Morgan’s theorem, truth tables, and logic minimization. This chapter uses these key concepts in order to design mega cells, namely various types of adders and multipliers. Chapter 2 introduces sequential logic components, namely latches, flip-flops, registers, and counters. It introduces the concept of timing diagrams to explain the functionality of each logic block. The Moore and Mealy-type state machines, counter-decoder-type controllers, and the construction of simple memories are also explained in this chapter. Chapter 2 also illustrates the design process: how to develop architectural logic blocks using timing diagrams, and how to build a controller from a timing diagram to guide data flow. Chapter 3 focuses on the review of asynchronous logic design, which includes state definitions, primitive flow tables, and state minimization. Racing conditions in asynchronous designs, how to detect and correct them are also explained in this chapter. The chapter ends with designing an important asynchronous timing block: the C element (or the Mueller element), and it describes an asynchronous timing methodology that leads to a complete design using timing diagrams.
From Chapter 4 to Chapter 8, computer architecture-related topics are covered. Chapter 4 examines a very essential system element: system bus and communication protocols between system modules. This chapter defines the bus master and the bus slave concepts and examines their bus interfaces. Read and write bus cycles and protocols, bus handover and arbitration are also examined in this chapter. System memories, namely Static Random Access Memory (SRAM), Synchronous Dynamic Random Access Memory (SDRAM), Electrically-Erasable-Programmable-Read-Only-Memory (E²PROM) and Flash memory are examined in Chapter 5. This chapter also shows how to design bus interface for each memory type using timing diagrams and state machines. Chapter 6 is all about the design of a simple Reduced Instruction Set Computer (RISC) for central processing. The chapter starts with introducing a simple assembly instruction set and building individual hardware for each instruction. As other instructions are introduced to the design, techniques are shown how to integrate additional hardware to the existing CPU data-path to be able to execute multiple instructions. Fixed-point and floating-point Arithmetic Logic Units (ALU) are also studied in this chapter. Structural, data and program control hazards, and the required hardware to avoid them are shown. This chapter ends with the operation of various cache architectures, cache read and write protocols, and the functionality of write-through and write-back caches. The design of system peripherals, namely Direct Memory Access (DMA), interrupt controller, system timers, serial interface, display adapter and data controllers are covered in Chapter 7. The design methodology to construct data-paths with timing diagrams in Chapter 2 is closely followed in this chapter in order to design the bus interface for each peripheral. Chapter 8 describes the Field-Programmable Gate array (FPGA), and the fundamentals of data driven processors as special topics.

At the end of the book, there is a small appendix that introduces the Verilog language. Verilog is a widely used Hardware Design Language (HDL) to build and verify logic blocks, mega cells and systems. Interested readers are encouraged to go one step beyond and learn system Verilog to be able to verify large logic blocks.

Dr. Ahmet Bindal
Computer Engineering Department
San Jose State University
San Jose, CA, USA
Fundamentals of Computer Architecture and Design
Bindal, A.
2017, XIV, 533 p. 531 illus., Hardcover
ISBN: 978-3-319-25809-6