9.1 Introduction

(Original section[1])

The COD ARM non-interactive edition used the term LEGv8 throughout the text of this chapter instead of the term MIPS. The zyBook differs by using the original term MIPS to match the figures and tables.

> "A custom format such as this is slave to the architecture of the hardware and the instruction set it serves. The format must strike a proper compromise between ROM size, ROM-output decoding, circuitry size, and machine execution rate.
> *Jim McKevit, et al., 8086 design report, 1997*

Control typically has two parts: a combinational part that lacks state and a sequential control unit that handles sequencing and the main control in a multicycle design. Combinational control units are often used to handle part of the decode and control process. The ALU control in COD Chapter 4 (The Processor) is such an example. A single-cycle implementation like that in COD Chapter 4 (The Processor) can also use a combinational controller, since it does not require multiple states. COD Section C.2 (Implementing combinational control units) examines the implementation of these two combinational units from the truth tables of COD Chapter 4 (The Processor).

Since sequential control units are larger and often more complex, there are a wider variety of techniques for implementing a sequential control unit. The usefulness of these techniques depends on the complexity of the control, characteristics such as the average number of next states for any given state, and the implementation technology.

The most straightforward way to implement a sequential control function is with a block of logic that takes as inputs the current state and the opcode field of the Instruction register and produces as outputs the datapath control signals and the value of the next state. The initial representation may be either a finite-state diagram or a microprogram. In the latter case, each microinstruction represents a state.

In an implementation using a finite-state controller, the next-state function will be computed with logic. COD Section C.3 (Implementing finite-state machine control) constructs such an implementation both for a ROM and a PLA.

An alternative method of implementation computes the next-state function by using a counter that increments the current state to determine the next state. When the next state doesn't follow sequentially, other logic is used to determine the state. COD Section C.4 (Implementing the next-state function with a sequencer) explores this type of implementation and shows how it can be used to implement finite-state control.

In COD Section C.5 (Translating a microprogram to hardware), we show how a microprogram representation of sequential control is translated to control logic

(*1) This section is in original form.

Provide feedback on this section