9.6 Concluding remarks

(Original section[1])

We began this appendix by looking at how to translate a finite-state diagram to an implementation using a finite-state machine. We then looked at explicit sequencers that use a different technique for realizing the next-state function. Although large microprograms are often targeted at implementations using this explicit next-state approach, we can also implement a microprogram with a finite-state machine. As we saw, both ROM and PLA implementations of the logic functions are possible. The advantages of explicit versus encoded next state and ROM versus PLA implementation are summarized below.

The Big Picture

Independent of whether the control is represented as a finite-state diagram or as a microprogram, translation to a hardware control implementation is similar. Each state or microinstruction asserts a set of control outputs and specifies how to choose the next state.

The next-state function may be implemented by either encoding it in a finite-state machine or using an explicit sequencer. The explicit sequencer is more efficient if the number of states is large and there are many sequences of consecutive states without branching.

The control logic may be implemented with either ROMs or PLAs (or even a mix). PLAs are more efficient unless the control function is very dense. ROMs may be appropriate if the control is stored in a separate memory, as opposed to within the same chip as the datapath.

(*1) This section is in original form.