10 Sequential Circuit Design

10.1 Introduction

Chapter 9 addressed *combinational* circuits in which the output is a function of the current inputs. This chapter discusses *sequential* circuits in which the output depends on previous as well as current inputs; such circuits are said to have *state*. Finite state machines and pipelines are two important examples of sequential circuits.

Sequential circuits are usually designed with flip-flops or latches, which are sometimes called *memory elements*, that hold data called *tokens*. The purpose of these elements is not really memory; instead, it is to enforce sequence, to distinguish the *current* token from the *previous* or *next* token. Therefore, we will call them *sequencing elements* [Harris01a]. Without sequencing elements, the next token might catch up with the previous token, garbling both. Sequencing elements delay tokens that arrive too early, preventing them from catching up with previous tokens. Unfortunately, they inevitably add some delay to tokens that are already critical, decreasing the performance of the system. This extra delay is called *sequencing overhead*.

This chapter considers sequencing for both static and dynamic circuits. *Static circuits* refer to gates that have no clock input, such as complementary CMOS, pseudo-nMOS, or pass transistor logic. *Dynamic circuits* refer to gates that have a clock input, especially domino logic. To complicate terminology, sequencing elements themselves can be either static or dynamic. A sequencing element with *static storage* employs some sort of feedback to retain its output value indefinitely. An element with *dynamic storage* generally maintains its value as charge on a capacitor that will leak away if not refreshed for a long period of time. The choices of static or dynamic for gates and for sequencing elements can be independent.

Sections 10.2–10.4 explore sequencing elements for static circuits, particularly flip-flops, 2-phase transparent latches, and pulsed latches. Section 10.5 delves into a variety of ways to sequence dynamic circuits. A periodic clock is commonly used to indicate the timing of a sequence. Section 10.6 describes how external signals can be synchronized to the clock and analyzes the risks of synchronizer failure. Wave pipelining is discussed in Section 10.7. Clock generation and distribution will be examined further in Section 13.4.

The choice of sequencing strategy is intimately tied to the design flow that is being used by an organization. Thus, it is important before departing on a design direction to ensure that all phases of design capture, synthesis, and verification can be accommodated. This includes such aspects as cell libraries (are the latch or flip-flop circuits and models available?); tools such as timing analyzers (can timing closure be achieved easily?); and automatic test generation (can self-test elements be inserted easily?).