Sequential Logic Design

Otto J. Kunkel

October 21, 2024

1 Sequential Logic Design

This section covers the fundamental concepts of sequential logic design, including latches, flip-flops, synchronous logic, finite state machines, and memory arrays.

1.1 Introduction

Sequential logic differs from combinational logic as it depends on both current and past input values. Sequential circuits maintain state using memory elements like latches and flip-flops. These circuits can be analyzed and designed as synchronous sequential systems that consist of combinational logic and memory elements, making them simpler to manage.

1.2 Latches and Flip-Flops

Latches are the fundamental building blocks of memory in sequential circuits. A bistable element, such as a cross-coupled inverter pair, can store a single bit of information with two stable states (0 or 1). The output nodes, Q and $\bar{Q}$, indicate the state of the circuit.

1.2.1 SR Latch

The SR latch, composed of two cross-coupled NOR gates, stores one bit and has two inputs: Set (S) and Reset (R). Depending on the inputs, the latch can set Q to 1, reset Q to 0, or maintain its previous state. The SR latch has an invalid condition when both S and R are high.

1.2.2 D Latch

The D latch simplifies the SR latch by ensuring that S and R cannot be asserted simultaneously. It has a data input (D) and a clock input (CLK). When CLK is high, the latch is transparent, meaning Q follows D. When CLK is low, Q holds its previous value.

1.2.3 D Flip-Flop

A D flip-flop is created by cascading two D latches controlled by complementary clock signals. It captures the value of D on the rising edge of the clock and maintains that value until the next clock edge. This makes it edge-triggered, unlike the level-sensitive D latch.

1.3 Synchronous Logic Design

Synchronous sequential circuits consist of combinational logic and flip-flops, with all state changes synchronized by a clock signal. The design process involves defining state variables, creating a state transition diagram, and deriving the next-state and output logic.

1.4 Finite State Machines (FSMs)

Finite State Machines are a method to design and analyze sequential circuits. They consist of a finite number of states, transitions between states based on inputs, and outputs for each state. FSMs can be classified as Moore or Mealy machines, depending on whether their outputs depend only on the current state (Moore) or on both the current state and inputs (Mealy).

1.5 Timing of Sequential Logic

The timing of sequential circuits is determined by the propagation delay of combinational logic, the setup and hold times of flip-flops, and the clock period. Proper timing ensures that data is correctly captured by flip-flops, avoiding issues like metastability.

1.6 Parallelism

Parallelism is often used to increase the speed of sequential circuits by replicating hardware to perform multiple operations simultaneously. This can be seen in pipelined designs, where multiple stages of computation are executed concurrently.

1.7 Summary

Sequential logic design introduces memory elements, such as latches and flip-flops, which enable circuits to remember past input values. Synchronous design methodologies simplify the analysis and implementation of sequential systems. FSMs provide a structured way to represent and implement these systems, while careful timing analysis ensures their correct operation.

1.8 Exercises

To reinforce these concepts, exercises involve designing and analyzing latches, flip-flops, and FSMs, as well as exploring the timing and performance improvements through parallelism.

1.9 Interview Questions

- What is the difference between a latch and a flip-flop?
- How does a D flip-flop differ from a D latch?
- What are the main components of a finite state machine?
- Explain the importance of setup and hold times in sequential circuits.