Lab 09: Shift Registers

Arturo Salinas-Aguayo
CSE 2301: Principles and Practice of Digital Logic Design
Dr. Mohammad Khan, Section 003L-1248
Electrical and Computer Engineering Department

College of Engineering, University of Connecticut
Coded in LaTeX

Theory

0.1 Main Functional Purpose of a Register

Definition 1. A **register** is a crucial component in digital systems, acting as a small, high-speed storage unit within a processor or digital circuit. In isolation, a register's primary job is to temporarily hold data bits, usually as an array of flip-flops, which can be accessed or manipulated synchronously with a clock signal.

Registers allow data to be stored, loaded, and transferred within a system, supporting operations such as:

- **Data Storage**: Holds binary information temporarily for immediate use by the processor.
- **Data Manipulation**: In specialized registers, data can be manipulated (e.g., shift registers, where bits can be shifted left or right).
- **Synchronized Operations**: Registers work in sync with a clock signal to ensure that data is processed in an orderly fashion, avoiding timing issues.

0.2 Example of When to Use Serial Inputs

Definition 2. Serial inputs are used when data needs to be transferred one bit at a time, often to save pin count or when minimizing wiring is essential.

Example 1. Examples of when to use serial inputs include:

- **Communication Protocols**: Interfaces like SPI or I^2C use serial data lines to transfer data between devices such as sensors and microcontrollers.
- **Microcontroller Communication**: Reading data from a serial peripheral, like a temperature sensor over a single data line, to save GPIO pins.

0.3 Example of When to Use Parallel Inputs

Definition 3. Parallel inputs are used when multiple bits need to be processed simultaneously, which allows for faster data transfer compared to serial inputs.

Example 2. Examples of when to use parallel inputs include:

- **Microcontroller GPIO Initialization**: Setting an entire 8-bit GPIO port's value at once with an 8-bit parallel input.
- **Loading Data into Registers**: Loading a data register with an entire byte (or more) from a bus for quick manipulation.

0.4 Effect of Using a Clock Without a Debouncer

Definition 4. A **debouncer** is a circuit or algorithm that removes noise or mechanical bounce from a signal, ensuring stable and predictable behavior.

Without a debouncer, the clock signal could suffer from noise or spurious pulses caused by mechanical contacts bouncing. This bouncing can lead to:

- **False Triggers**: Noise can cause the clock line to trigger unintended edges, causing registers to latch incorrect data.
- **Data Corruption**: Unstable signals could make a register latch or shift data unexpectedly, leading to unpredictable outcomes.
- **System Instability**: The entire circuit could malfunction if synchronization relies on an erratic clock signal.

0.5 Difference Between SN7454 and SN7474

Definition 5. The **SN7454** and **SN7474** are distinct integrated circuits with different functionalities:

- **SN7454 (Dual 4-bit Binary Adder)**: Designed to perform arithmetic operations, specifically the addition of two 4-bit binary numbers.
- **SN7474 (Dual D-type Flip-Flop)**: Composed of two D-type flip-flops, which are edge-triggered devices used to store a single bit of data each.

The key differences between the SN7454 and SN7474 are:

- **Functionality**: The SN7454 handles arithmetic tasks, while the SN7474 is meant for data storage and synchronization.
- **Application**: The SN7454 is used in computational circuits requiring arithmetic, whereas the SN7474 is found in sequential logic for data synchronization, timing control, or edge-triggered operations.

Shift Register Analysis

Problem Description

For this example, we'll use the last two digits of my zip code: **40**. Each digit is converted into Binary Coded Decimal (BCD) and then concatenated. Here are the steps:

- The digit **4** in BCD is: 0100
- The digit **0** in BCD is: 0000

After concatenating these BCD values, we get: 0100 0000.

We will input this 8-bit number into an imaginary shift register starting from the Least Significant Bit (LSB) to the Most Significant Bit (MSB). The shift register starts cleared (0000 0000) and each bit is shifted right with every clock pulse.

Shift Register Table

Step	Input	Registers	Decimal
0	N/A	00000000	0
1	0	00000000	0
2	0	10000000	128
3	0	01000000	64
4	0	00100000	32
5	1	00010000	16
6	0	00001000	8
7	0	00000100	4
8	0	00000010	2

Table 1: Shift Register Steps for Input 01000000

Final Result

The final 8-bit binary value in the shift register is 0100 0000, which corresponds to **64** in decimal.