Lecture 6 — Sequential Circuits 1: Shift Registers

Table of contents

See shift-registers[1] for the reference material this lecture introduces, and sequential-circuits[2] and flip-flops-and-latches[3] for the recap slides it opens with.

Today's outline

- Admin
- Sequential circuits
- Shift registers

Recap slides

The first three content slides are repeats from [[2026-08-10-introduction-to-sequential-circuits|Lecture 5]] and their content lives in the concept notes:

[1] courses/csse2010/shift-registers.pdf
[2] courses/csse2010/sequential-circuits.pdf
[3] courses/csse2010/flip-flops-and-latches.pdf

- **"Reminder: Memory element — D Flip Flop"** — D input, Q output, CLK control input; Q copies (and remembers) D on the **rising edge** of CLK. Only D flip-flops are used in this course. Optional **asynchronous SET and CLR** inputs set/clear Q outside clock edges and are typically **active-low**. D flip-flops are what you build sequential circuits from — e.g. counters[4]. Full detail in flip-flops-and-latches[5].
- **"Combinational vs. Sequential Circuits"** — combinational = logic gates only, output uniquely determined by the inputs (the slide's example is $A(B + C)$, see logic-gates[6]); sequential = includes flip-flops, output determined by current inputs *and* current **state**, and can change when the clock ticks. See sequential-circuits[7].
- **"Sequential Circuits" / "Synchronous Sequential Circuit"** — state = values in the flip-flops, present state vs next state, and the rule that in a synchronous sequential circuit **all sequential elements share a common clock signal**. See sequential-circuits[8].

Registers

A **register** is a group of flip-flops: an n-bit register is n flip-flops storing n bits. Two points the slide makes explicitly:

- a register is a sequential circuit **without any combinational logic** — unlike the general sequential-circuit block diagram from the recap slides, which has a combinational block in the feedback path;
- registers store binary information (data/instructions) **inside a processor**.

The worked example is a 4-bit register with parallel inputs $I_0..I_3$, parallel outputs $A_0..A_3$, and common `Clock` and (active-low) `Clear` lines. Structure and behaviour in shift-registers[9].

Shift registers

A **shift register** is a register capable of shifting its binary information in one or both directions. The lecture's example is the 4-bit chain (serial input SI $\rightarrow$ four D flip-flops in series $\rightarrow$ serial output SO, all on a common CLK). Construction and a clock-by-clock trace are in shift-registers[10].

Two applications follow, both of which the slides leave as blank figures marked *"figure to be completed in class"*:

[4] courses/csse2010/counters.pdf
[5] courses/csse2010/flip-flops-and-latches.pdf
[6] courses/csse2010/logic-gates.pdf
[7] courses/csse2010/sequential-circuits.pdf
[8] courses/csse2010/sequential-circuits.pdf
[9] courses/csse2010/shift-registers.pdf
[10] courses/csse2010/shift-registers.pdf

- **Serial parallel conversion** — the slide states that shift registers can do serial-to-parallel conversion and vice-versa, then shows four unconnected D flip-flops on a common `clk` with the figure left to be completed live.
- **Parallel load and serial shift** — likewise four unconnected D flip-flops on a common `clk`, with the whole figure left blank. The next slide's phrase "using the same multiplexer concept" tells us the completed figure put a **multiplexer in front of each flip-flop's D input** to choose between the shift source and the parallel input — see combinational-logic-blocks[11] for the mux, and shift-registers[12] for the derived construction.

Neither completed figure is recoverable from the deck; only the derived standard constructions are recorded.

Exercise: bidirectional shift register

> Using the same multiplexer concept, draw a **3-bit shift register which allows data to be shifted in either direction**.
>
> Hint: consider this element, where **DIRN will be 0 for left shift, 1 for right shift**.

The hint element given on the slide is a **2-to-1 multiplexer feeding one D flip-flop**: the mux has data inputs labelled `0` and `1`, its select input is `DIRN`, its output goes to the flip-flop's D input, and the flip-flop's Q is the stage output.

The answer slide is blank — the worked construction was drawn live and is not in the deck. Derived from the hint element and the standard construction, the answer is: instantiate three copies of the element, share one `DIRN` line and one clock across all three, and for each stage wire

- **mux input 0** (selected when DIRN = 0, left shift) to the neighbouring stage on the left-shift *source* side, and
- **mux input 1** (selected when DIRN = 1, right shift) to the neighbouring stage on the right-shift *source* side,

with an external serial input supplying whichever end of the register has no neighbour in that direction. Writing the bits as $Q_2Q_1Q_0$ with Q_2 most significant, and taking "left shift" to move bits towards the more-significant end:

Stage	Mux input 0 (DIRN=0, left)	Mux input 1 (DIRN=1, right)
Q_2	Q_1	SI_R
Q_1	Q_0	Q_2

[11] courses/csse2010/combinational-logic-blocks.pdf
[12] courses/csse2010/shift-registers.pdf

Stage	Mux input 0 (DIRN=0, left)	Mux input 1 (DIRN=1, right)
Q_0	SI_L	Q_1

Sanity check on $Q_2Q_1Q_0 = 110$ with both serial inputs at 0: one edge with DIRN = 0 gives 100; one edge with DIRN = 1 gives 011.

The essential insight the exercise is testing: **a bidirectional shift register is just a unidirectional one with a mux per flip-flop**, and the direction control is a single line shared by every mux, so the whole register commits to one direction per clock edge.

Universal shift register

The slide titled "Universal Shift Register" is **completely blank** — title only. Everything on it was done live. The concept (hold / shift left / shift right / parallel load in one device, selected by control inputs) is recorded in shift-registers[13].

Wide shift registers

Shift registers can shift **multiple bits at a time**. The lecture's example is a **4-stage, 8-bit queue**: four 8-bit registers in a row (inputs $A..H$, outputs $Q_1..Q_8$, each with an `ENB` enable), each register's outputs feeding the next register's inputs as an 8-bit bus, all on a common clock. A byte at `Input` emerges at `Output` four clock edges later. See shift-registers[14].

Lab 06 preparation task

Stated on the slides as (verbatim):

> **2-digit lock/unlock circuit**: User inputs two decimal digits (4 bits each) **AB** in serial and the circuit should match the two input digits with a code (say **CD**) and unlock if the input matches with the code (i.e. **AB = CD**).

Both Lab 06 preparation-task slides are marked *"to be discussed in class"* and the second is blank, so **no worked solution is in the deck**. The shape of the solution is signposted by the lecture, though: serial digit entry into a shift register is exactly the serial-to-parallel conversion above — clock the 8 serially-entered bits into an 8-bit shift register, then compare the parallel output against the stored code CD with combinational logic (see combinational-logic-blocks[15]) and drive the unlock output.

[13] courses/csse2010/shift-registers.pdf
[14] courses/csse2010/shift-registers.pdf
[15] courses/csse2010/combinational-logic-blocks.pdf

Reminders

- **Labs 6 and 7** (next week, week 4) have **preparation tasks** which should be attempted **before** coming to the labs.
- Attempt the weekly exercise and quizzes.