Shift Registers

Table of contents

Registers and shift registers are the simplest useful [[sequential-circuits|sequential circuits]] — groups of [[flip-flops-and-latches|D flip-flops]] sharing a common clock.

Registers

- A **register** is a group of flip-flops. An n**-bit register** consists of n flip-flops and is capable of storing n bits.
- A register is a sequential circuit **without any combinational logic** — it is just the storage elements and their shared control lines.
- Registers are used to store binary information (data/instructions) **inside a processor**.

Structure of the 4-bit register example (Mano, *Digital Design*, 3rd ed.):

Signal	Connection
$I_0 \dots I_3$	the four parallel data inputs, one to each flip-flop's D input
$A_0 \dots A_3$	the four parallel outputs, taken from each flip-flop's Q
`Clock`	a single common line driven to the C (clock) input of **all four** flip-flops

Signal	Connection
`Clear`	a single common line driven to the R (reset) input of all four flip-flops, drawn **active-low** (bubble on the input)

So on each rising clock edge all four flip-flops simultaneously capture their I inputs; asserting `Clear` forces all four outputs to 0 asynchronously (see flip-flops-and-latches[1] for asynchronous SET/CLR).

Shift registers

> A **shift register** is a register which is capable of shifting its binary information in one or both directions.

The 4-bit shift register example is built as a **chain**: the serial input SI feeds the D input of the first flip-flop; each flip-flop's Q output feeds the next flip-flop's D input; the last flip-flop's Q is the serial output SO. All four flip-flops share a common CLK line.

On each rising clock edge every stored bit moves one place along the chain, the bit currently on SI enters the first stage, and the bit that was in the last stage leaves at SO.

Worked through by hand, with stages labelled Q_0 (nearest SI) through Q_3 (= SO), starting from all zeros and shifting in the bit sequence 1, 0, 1, 1:

Clock edge	SI	Q_0	Q_1	Q_2	Q_3 (= SO)
(initial)	—	0	0	0	0
1	1	1	0	0	0
2	0	0	1	0	0
3	1	1	0	1	0
4	1	1	1	0	1

After 4 clock edges the 4 serially-supplied bits sit in the 4 flip-flops, and one further edge would push the first of them out of SO.

[1] courses/csse2010/flip-flops-and-latches.pdf

Serial parallel conversion

Shift registers can be used to do **serial-to-parallel conversion, and vice versa**. This is the reason they show up everywhere data has to travel over a single wire but be *used* a word at a time.

- **Serial in, parallel out**: drive the bits one per clock edge into SI; after n edges, the n flip-flop Q outputs — read as a group — are the parallel word. This is exactly the table above: clock 4 bits in, then read $Q_0Q_1Q_2Q_3$ in parallel.
- **Parallel in, serial out**: load the n bits into the flip-flops in one clock edge (a parallel load — see below), then clock n more times, reading one bit per edge off SO.

> **Note**
>
> The lecture slide for this figure is blank and marked *"figure to be completed in class"*. The description above is derived from the standard construction: the parallel outputs are simply the Q pins of the flip-flops already present in the 4-bit shift register, brought out as a group.

Parallel load vs serial shift

A plain shift register can only be filled one bit per clock. To also allow **parallel load** — all n bits written at once from n parallel inputs — each flip-flop's D input is fed from a **2-to-1 multiplexer** (see combinational-logic-blocks[2]) instead of directly from the previous stage:

Mux select	Mux data input chosen	Effect on that stage
"shift"	the previous stage's Q (or SI for the first stage)	serial shift by one place
"load"	that stage's parallel data input I_i	parallel load of the whole word in one clock edge

The select line is common to all the muxes, so on each clock edge the register either shifts by one place or loads the whole parallel word.

> **Note**
>
> The lecture slide for this figure is also blank and marked *"figure to be completed in class"*, so the exact drawing done in the lecture is not recoverable from the deck. The mux-per-flip-flop construction above is derived, and is confirmed by the following slide,

[2] courses/csse2010/combinational-logic-blocks.pdf

which refers back to "the same multiplexer concept".

The mux-based bidirectional shift element

The lecture gives a single building block for a shift register that shifts in **either** direction:

- a **2-to-1 multiplexer** whose output drives the D input of one D flip-flop;
- the mux's select line is a control signal called **DIRN**;
- **DIRN = 0** selects mux input **0** → **left shift**;
- **DIRN = 1** selects mux input **1** → **right shift**;
- the flip-flop's Q is that stage's output, and the flip-flop is clocked from the common clock.

Chaining n of these elements and wiring, for each stage, mux input 0 to the neighbour on the left-shift side and mux input 1 to the neighbour on the right-shift side gives an n-bit **bidirectional shift register**: one shared DIRN line decides which way the whole register shifts on the next clock edge.

For a 3-bit register with bits $Q_2Q_1Q_0$ (Q_2 most significant), taking *left shift* to mean bits move towards the more-significant end:

Stage	Mux input 0 (DIRN=0, left shift)	Mux input 1 (DIRN=1, right shift)
Q_2	Q_1	SI_R (serial input for right shifts)
Q_1	Q_0	Q_2
Q_0	SI_L (serial input for left shifts)	Q_1

One clock edge applied to a 3-bit register holding $Q_2Q_1Q_0 = 110$, with both serial inputs held at 0:

DIRN	Operation	Before ($Q_2Q_1Q_0$)	After ($Q_2Q_1Q_0$)
0	left shift	110	100
1	right shift	110	011

The physical layout drawn in class is not in the slides; the derivation above is the standard construction. See 2026-08-13-shift-registers[3] for the exercise as it was set.

[3] courses/csse2010/2026-08-13-shift-registers.pdf

Universal shift register

A **universal shift register** is the combination of all of the above capabilities in one device: it can hold its value, shift left, shift right, and be parallel-loaded, with the operation selected by control inputs.

> ⚠ Warning
>
> The "Universal Shift Register" slide in the lecture deck is **entirely blank** — the title only. The one-line description above is the standard definition; everything specific (the control encoding, the mux width, the figure) was covered in class and is not in the slides.

Wide (multi-bit) shift registers

A shift register does not have to shift one *bit* at a time — it can shift a whole multi-bit word per clock edge. The lecture's example is an **8-bit wide, 4-stage queue**:

- four **registers** in a row, each 8 bits wide (inputs labelled A through H, outputs labelled Q_1 through Q_8);
- the 8-bit `Input` bus feeds the first register's 8 data inputs;
- each register's 8 Q outputs feed the next register's 8 data inputs, as a bus;
- the last register's outputs are the 8-bit `Output`;
- all four registers share a **common clock** line, and each has an `ENB` (enable) input.

Functionally this is the same chain as the 1-bit shift register, but each "stage" holds a byte rather than a bit — so it behaves as a 4-deep queue of bytes: a byte presented at `Input` appears at `Output` four clock edges later.

Related

- flip-flops-and-latches[4] — the D flip-flop these are built from
- sequential-circuits[5] — state, present/next state, synchronous clocking
- combinational-logic-blocks[6] — the multiplexer used for load/direction selection
- counters[7] — the other classic thing built from a row of D flip-flops
- device-pinouts[8] — 74HCT175 quad D flip-flop and 74HCT157 quad 2-input multiplexer pinouts

[4] courses/csse2010/flip-flops-and-latches.pdf
[5] courses/csse2010/sequential-circuits.pdf
[6] courses/csse2010/combinational-logic-blocks.pdf
[7] courses/csse2010/counters.pdf
[8] courses/csse2010/device-pinouts.pdf