Lecture 7 — Counters

Table of contents

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

Today's outline

- Poll on last week's bidirectional shift register
- Shift register types — recap from last week
- Counters, and binary counters
- A digital system application built around a counter
- State: current state, next state, and the D inputs
- Counter examples worked live in class

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

- Next time: state machines

From the teaching outline

Slide 2 is a screenshot of the course teaching outline table rather than a content slide. What it shows for now:

- **Week 4, Mon-Tue 17-18 Aug** — this lecture, *Lecture 7: Counters* (17 Aug).
- **Week 4, Thu-Fri 20-21 Aug** — *Lecture 8: Finite State Machines* (20 Aug).
- Learning labs this week: **Lab 6 Shift Registers** (P2, Mon-Tue) and **Lab 7 Counters** (P1, Thu-Fri).
- Week 5: *Lecture 9 ALU and Control Unit* (24 Aug), *Lecture 10 Introduction to AVR and Assembly Language* (27 Aug), with labs 8 (Finite State Machines) and 9 (AVR Assembly Programming 1).

Quiz: shift direction control

> Which of the following statements about the circuit below is true?
>
> A. When A is 1, flip-flop values are shifted to the right B. When A is 0, flip-flop values are shifted to the right C. When A is 0, flip-flop values stay the same D. When FN is 1, flip-flop values are shifted to the left E. When FN is 0, flip-flop values stay the same F. When FN is 0, flip-flop values are shifted to the left

The figure is the mux-per-stage construction from [[2026-08-13-shift-registers|last lecture]], two stages deep. Reading it off:

- **Stage 1**: a 2-to-1 mux with select $S = FN$. Its `1` input is the external signal A; its `0` input is fed back from the **stage 1 flip-flop's own Q output**. The mux output drives that flip-flop's D input.
- **Stage 2**: an identical mux, also selected by FN. Its `1` input is **stage 1's** Q; its `0` input is fed back from **stage 2's own** Q. Its output drives stage 2's D input.
- Both flip-flops share the same CLK.

So writing the two next-state equations, with Q_a = stage 1 and Q_b = stage 2:

$$D_a = \begin{cases} A & FN = 1 \\ Q_a & FN = 0 \end{cases} \qquad D_b = \begin{cases} Q_a & FN = 1 \\ Q_b & FN = 0 \end{cases}$$

- When $FN = 1$, A is loaded into stage 1 and stage 1's value moves into stage 2 — data shifts **left-to-right along the diagram**, one stage per clock.
- When $FN = 0$, each flip-flop's D input is its own Q, so every flip-flop reloads the value it already had — **nothing changes**.

FN here is therefore a shift *enable*, not a direction control (there is no left-shift path in this circuit at all), so every option mentioning "shifted to the left" is out, and A is plain data rather than a control signal, so A/B/C are out too.

Answer: E — when FN is 0, flip-flop values stay the same.

Shift register types — recap

Two recap slides repeat last week's figures: the seven shift-register configurations (serial in/shift right/serial out, serial in/shift left/serial out, parallel in/serial out, serial in/parallel out, parallel in/parallel out, rotate right, rotate left), and the clock-by-clock trace of `0101` being shifted serially into a 4-bit register versus being loaded into it in parallel in a single clock. All of that content lives in shift-registers[4].

Counters

The definition slide, the n-bit binary counter facts (n flip-flops, 2^n states, counts 0 to $2^n - 1$), and the 4-bit binary counting sequence table are all recorded in [counters].

Digital system application

A block-diagram slide (from Floyd's *Digital Fundamentals*) showing where a counter sits in a real system — an automated tablet-bottling line:

- A **keypad** for the number of tablets per bottle feeds an **encoder**, which feeds **Register A**. Register A drives **Decoder A** and a 7-segment display showing the tablets-per-bottle setting, and also a **code converter** producing the binary code for the preset number.
- On the conveyor, a **sensor** emits one pulse per tablet passing the valve. Those pulses are the clock of a **counter**, so the counter holds the binary code for the actual number of tablets in the bottle. A separate pulse **resets the counter to zero** when the next bottle is in place.
- A **comparator** tests the counter value against the preset value. Its output goes HIGH when $A = B$: HIGH closes the valve and advances the conveyor, LOW keeps the valve open.
- The counter value also feeds an **adder**, whose other input is the running total held in **Register B**; the comparator's HIGH also causes the new sum to be stored back into Register B. Register B drives **Decoder B** (total display) and a **MUX/DEMUX** path that sends the accumulated total on to a computer for storage.

[4] courses/csse2010/shift-registers.pdf

The point of the slide is that the counter, adder, register, decoder, comparator, encoder, mux and demux blocks covered so far in the course compose into a complete system — see combinational-logic-blocks[5] for the mux/decoder blocks and 2026-08-04-binary-arithmetic[6] for the adder.

One-bit counter

The slide is titled "One bit counter" and marked *"To be completed in class"*, but the figure itself is drawn: a single D flip-flop, clocked on CLK, with a wire from its $\bar{Q}$ output looped back around to its D input.

Derived from that figure: since D is the next state and $D = \bar{Q}$, the flip-flop's value inverts on every clock edge — the counter counts

$$0 \to 1 \to 0 \to 1 \to \ldots$$

which is exactly the $n = 1$ binary counter ($2^1 = 2$ states, counting 0 to $2^1 - 1 = 1$). This is the standard *toggle* configuration.

State

The slide that makes counter design mechanical, illustrated with two D flip-flops on a common CLK whose D_1 and D_0 inputs are both drawn as "?":

- the values stored in the flip-flops are the **current state** of the circuit;
- the D inputs are the **next state**;
- the D inputs are some (combinational) **function of the current state and the inputs**.

The design method that follows from this is written up as a recipe in [counters]; the underlying state terminology is in sequential-circuits[7].

Counter example (blank slide)

Slide 11 is a "Counter Example" marked *"To be completed in class"*. It gives an empty present-state / next-state table with columns $Q_1, Q_0 \mid D_1, D_0$ and rows for all four states $00, 01, 10, 11$, alongside a figure of two D flip-flops on a common CLK with **nothing connected** to their D inputs and no target sequence stated anywhere.

[5] courses/csse2010/combinational-logic-blocks.pdf
[6] courses/csse2010/2026-08-04-binary-arithmetic.pdf
[7] courses/csse2010/sequential-circuits.pdf

Because no count sequence is given, the intended answer is *not* recoverable from the deck — this one needs the lecture recording or a classmate's notes. Structurally it is the same exercise as the worked example below.

Quiz: what sequence does this counter count through?

> What sequence does the counter below count through? (Assume Q_1Q_0 starts at 00.)
>
> A. 00 → 11 → 10 → 01 → 00 B. 00 → 10 → 11 → 01 → 00 C. 00 → 11 → 01 → 10 → 00 D. 00 → 01 → 11 → 10 → 00 E. 00 → 01 → 10 → 11 → 00 F. 00 → 10 → 01 → 11 → 00

Reading the figure: two D flip-flops on a common CLK.

- The **left** flip-flop's $\bar{Q}$ output is wired back to its own D input, and its Q output is Q_1. So it is the toggle from the one-bit counter slide: $D_1 = \bar{Q}_1$.
- The **right** flip-flop's Q output is Q_0. Its D input is driven by a **two-input XNOR gate** (OR body with the extra input-side arc, and an inversion bubble on the output). The gate's inputs are Q_1 (tapped off the left flip-flop's Q) and Q_0 (fed back from the right flip-flop's own Q). So $D_0 = \overline{Q_1 \oplus Q_0}$, i.e. $D_0 = 1$ exactly when $Q_1 = Q_0$.

Tabulating:

Q_1	Q_0	$D_1 = \bar{Q}_1$	$D_0 = Q_1 \odot Q_0$	next state
0	0	1	1	11
1	1	0	1	01
0	1	1	0	10
1	0	0	0	00

Starting from 00: 00 → 11 → 01 → 10 → 00 → ...

Answer: C.

Sanity check on the gate reading — if it were a plain NOR rather than an XNOR, 00 → 11 → 00 would be a two-state cycle, which is not among the options; the XNOR reading is the one that produces a listed 4-state sequence.

Key points

The summary slide's four key points (next state is a function of the previous state and possibly inputs; the count sequence need not be binary; these circuits are synchronous with all flip-flops on the same clock; asynchronous counters exist but aren't covered) are recorded in [counters].

Example worked in class

The last content slide poses:

> 2-bit counter that counts $00 \rightarrow 10 \rightarrow 01 \rightarrow 00 \rightarrow$

and is otherwise blank, marked *"(to be worked through in class)"*. The slide after it is entirely blank — presumably the space the working was meant to fill.

Unlike the earlier blank, this one *is* recoverable, because the sequence is stated. The full present-state/next-state table and the derivation of $D_1 = \bar{Q}_1\bar{Q}_0$ and $D_0 = Q_1$ (with state 11 treated as a don't-care, and the resulting counter turning out to be self-correcting) are written up as the worked example in [counters].

Next time

More sequential circuits — **state machines**. The closing slide reuses the general synchronous sequential circuit block diagram (inputs and fed-back state into a combinational circuit, whose outputs are both the circuit outputs and the flip-flop inputs; flip-flops driven by clock pulses, their outputs fed back), see sequential-circuits[8], with the annotation:

> Note that a counter may or may not have external inputs.

The final slide of the deck is empty apart from the CSSE2010 banner.

Reminders

- **Lab 7 (Counters)** runs in the P1 sessions this week (Thu-Fri) — the pre-lab schematic for the 3-bit synchronous counter must be drawn beforehand. See week4-lab-lab-7-exercises[9], and device-pinouts[10] for the flip-flop packages.
- Lab 6 (Shift Registers) runs in the P2 sessions (Mon-Tue) this week.
- The teaching outline slide also flags the centrally scheduled 90-minute in-semester theory exam on **Saturday 5 September**, and the Digital Logic **lab exam during the scheduled lab sessions in week 7**.

[8] courses/csse2010/sequential-circuits.pdf
[9] courses/csse2010/week4-lab-lab-7-exercises.pdf
[10] courses/csse2010/device-pinouts.pdf