Sequential Circuits

Table of contents

Combinational vs sequential 1
State 2
General structure of a sequential circuit 2
Synchronous sequential circuits 3
See also 4

A **sequential circuit** is a circuit whose output depends not only on the current inputs but also on the values it is currently storing. The storage is done with flip-flops — see flip-flops-and-latches[1] for the storage elements themselves.

The output of any logic gate or combinational circuit depends on the **current value of the inputs only**, so if an input changes, the output can change too and **the previous value is lost forever**. A circuit with **memory** can remember a value even when the input changes. First met in 2026-08-10-introduction-to-sequential-circuits[2].

Combinational vs sequential

	Combinational	Sequential
Contents	Logic gates only (**no** flip-flops)	Includes **flip-flops** as well as logic gates
Output determined by	The inputs alone — uniquely	Current inputs **and** current **state**
Repeatability	Same inputs always give the same output	Same inputs can give different outputs, depending on state
When output changes	Whenever an input changes	Only when the clock 'ticks'

[1] courses/csse2010/flip-flops-and-latches.pdf
[2] courses/csse2010/2026-08-10-introduction-to-sequential-circuits.pdf

	Combinational	Sequential
Examples	Adders, multiplexers, decoders, demultiplexers, encoders (see combinational-logic-blocks[3])	Counters, registers (see counters[4], shift-registers[5])

The two example lists are worth memorising as the dividing line: the blocks built earlier in the course are all combinational, and the ones that follow — registers, counters, state machines — are all sequential.

A combinational example from earlier in the course: A into one input of an AND gate, B and C into an OR gate whose output feeds the AND gate's other input, giving $X = A(B + C)$. Nothing in that circuit remembers anything — change A, B or C and the output follows immediately (after propagation delay; see timing-diagrams[6]).

The key contrast: if an input to a combinational circuit changes, the output can change too and the **previous value is lost forever**. A sequential circuit can hold onto a value even after the input that produced it has gone away.

State

- **State** = the value stored in the flip-flops.
- **Output** depends on the inputs *and* the state.
- **Next state** depends on the inputs *and* the (present) state.

"Present state" is what the flip-flops hold right now; "next state" is what they will hold after the next clock edge. The combinational logic computes the next state from the present state and the inputs; the flip-flops only adopt it when the clock edge arrives.

General structure of a sequential circuit

The standard block diagram has two blocks and a feedback loop.

- A **combinational circuit** block (just logic gates). It takes the external **inputs** *plus* the feedback from the flip-flops, and produces the external **outputs** *plus* the values to be loaded into the flip-flops.

[3] courses/csse2010/combinational-logic-blocks.pdf
[4] courses/csse2010/counters.pdf
[5] courses/csse2010/shift-registers.pdf
[6] courses/csse2010/timing-diagrams.pdf

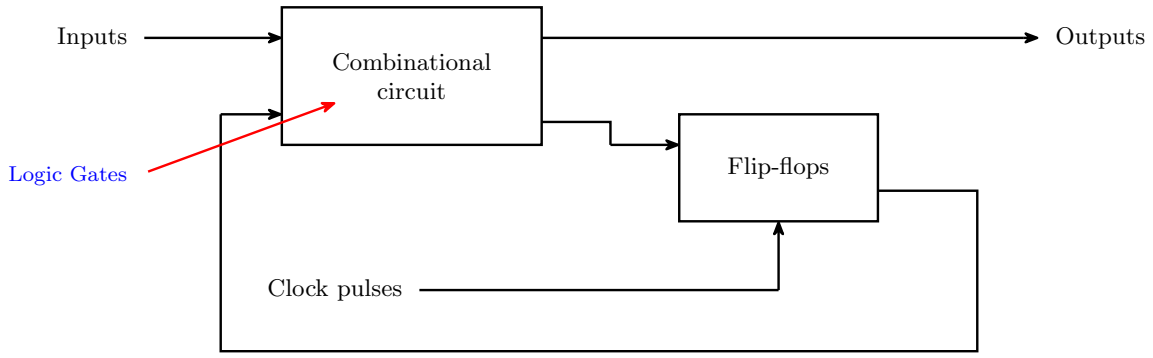

Figure 1: The general sequential circuit (after Mano, Digital Design, 3rd edition). Two things carry the whole idea: the loop running from the flip-flops back into the combinational block, which is what makes the circuit sequential, and the clock stub, which enters the flip-flops and nothing else. The 'Logic Gates' callout marks the combinational block as gates only.

- A **flip-flops** block (the storage elements). Its data inputs come from the combinational circuit; it also takes **clock pulses** as a separate control input.
- A **feedback path** runs from the flip-flops' outputs back around into the combinational circuit's inputs. That loop is what makes the circuit sequential — the stored state is fed back in and influences the next output and the next state.

The clock pulses go **only** to the flip-flops, not to the combinational logic. Cut the feedback wire and what is left is a plain combinational circuit with a register bolted on the end; it is the loop that makes the difference.

This same block diagram reappears in shift-registers[7], counters[8] and state-machines[9] — those are all just specific choices of what goes in the combinational block.

Synchronous sequential circuits

In a **synchronous** sequential circuit the storage elements can only change at **discrete instants of time**, set by a common clock signal.

- Assume a clock signal — a regularly repeating train of pulses alternating between 0 and 1.
- The outputs of the storage elements change **only on the edges** of that control signal.
- Contrast with logic gates, whose outputs change whenever their inputs change.

[7] courses/csse2010/shift-registers.pdf
[8] courses/csse2010/counters.pdf
[9] courses/csse2010/state-machines.pdf

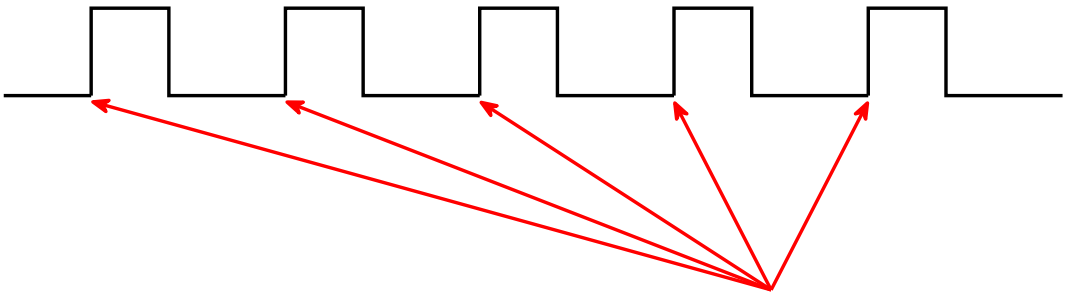

Figure 2: A clock signal. The five arrows all point at rising edges: these are the only instants at which any storage element in a synchronous circuit is allowed to change. Everywhere else on the axis the flip-flops are inert, however much their inputs move — which is exactly the opposite of a logic gate, whose output changes whenever its input does.

Because every flip-flop shares the same clock, the whole circuit's state changes in one step at each clock edge, which is what makes such circuits tractable to design and analyse.

It also means the clock period sets the pace of the design: everything the combinational block has to compute must settle within one tick. This follows from the propagation delays of timing-diagrams[10].

See also

- 2026-08-10-introduction-to-sequential-circuits[11] — Lecture 5
- state-machines[12] — the same circuits described by their states and transitions, which is how they are actually designed.
- flip-flops-and-latches[13] · counters[14] · shift-registers[15]
- csse2010-diagram-notation[16] — the block-diagram conventions the figures above follow

[10] courses/csse2010/timing-diagrams.pdf
[11] courses/csse2010/2026-08-10-introduction-to-sequential-circuits.pdf
[12] courses/csse2010/state-machines.pdf
[13] courses/csse2010/flip-flops-and-latches.pdf
[14] courses/csse2010/counters.pdf
[15] courses/csse2010/shift-registers.pdf
[16] courses/csse2010/csse2010-diagram-notation.pdf