Computer Architecture

Digital logic: Part 3

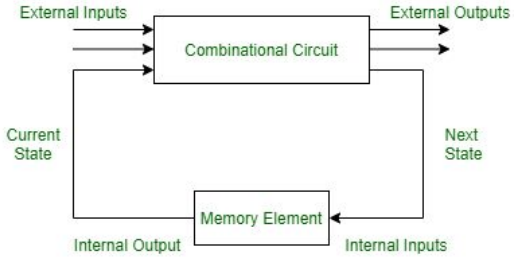
Sequential logic

Two types of logic in digital systems

Logic: In a digital system, it refers to the set of rules and structures that govern how binary values (0s and 1s) are processed to produce a specific outcome. (i.e., the implementation of logic expressions in Boolean algebra.)

Two types of logic:

- **Combinational logic**: Output depends solely on the current inputs, without any memory of past states. It's used for operations that need immediate calculation, like arithmetic and data selection.
- **Sequential logic**: Output depends on both current inputs and past states, providing memory through components like flip-flops and registers. Sequential logic enables systems to remember information and operate in stages, crucial for tasks that need state retention and timing control.



Agenda

- Sequential logic
- Clock signals
- Latches
- Flip-flop
- Register files
- Synchronous sequential circuits

Sequential logic

Sequential logic refers to a type of logic where the output depends not only on the current inputs but also on the history of past inputs.

- This is in contrast to combinational logic, where the output is determined solely by the present inputs.

Correspondingly, sequential circuits are digital circuits that store and use previous state information to determine their next state.

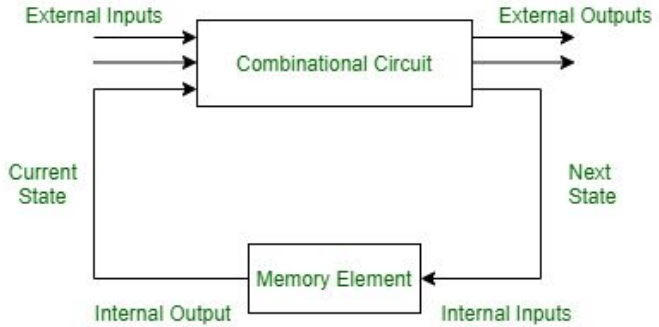
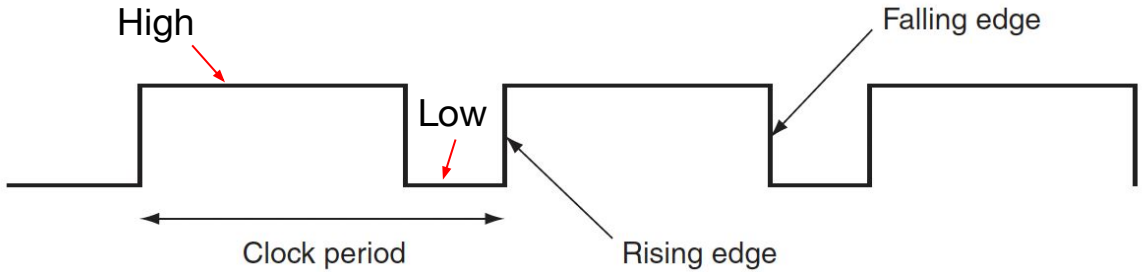


Figure - Sequential Circuit

Clock signals (defining the system time)

To implement sequential logic, we should define the system time first.

- A clock is a free-running signal with a fixed *cycle time*.
- *Clock frequency* is the inverse of the cycle time.
- A clock cycle time (also called clock period) is divided into two portions: **clock high** and **clock low**.
- edge-triggered clocking: all state changes occur on a clock edge (rising edge or falling edge).



Circuits for sequential logic

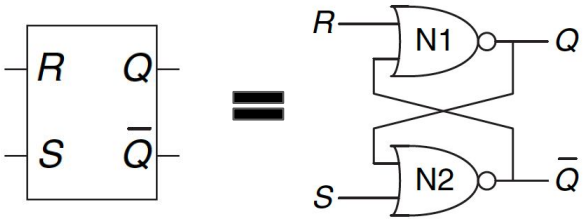
Memory elements

Latch: a fundamental building block in digital circuits used to store a bit of data.

- It can store a single bit of data.
- Two types: SR latch and D latch

Flip-flop: Basic building blocks that store a single bit of data and change state based on the clock signal.

SR latches

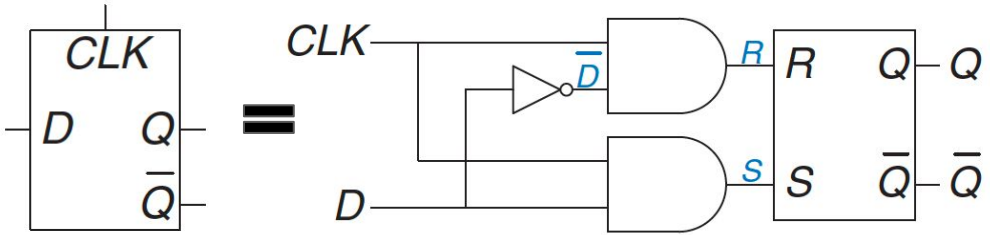


Case	S	R	Q	$\overline{Q}$
IV	0	0	Q_{prev}	$\overline{Q}_{prev}$
I	0	1	0	1
II	1	0	1	0
III	1	1	0	0

SR latch is composed of two cross-coupled NOR gates.

- It has two inputs S and R, and two outputs Q and $\overline{Q}$.
- S and R control the state of the latch:
  - S=0, R=0 ⇒ Q and $\overline{Q}$ do not change.
  - S=0, R=1 ⇒ Q=0, and $\overline{Q}$=1 (reset)
  - S=1, R=0 ⇒ Q=1, and $\overline{Q}$=1 (set)
  - S=1, R=1 ⇒ Q=0, and $\overline{Q}$=0

D latches



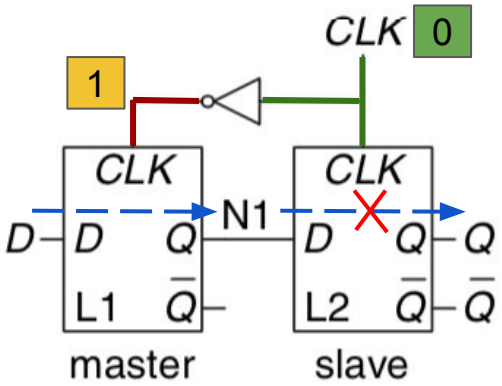
CLK	D	$\overline{D}$	S	R	Q	$\overline{Q}$
0	X	$\overline{X}$	0	0	Q_{prev}	$\overline{Q}_{prev}$
1	0	1	0	1	0	1
1	1	0	1	0	1	0

D latch has a data input D and the clock input CLK.

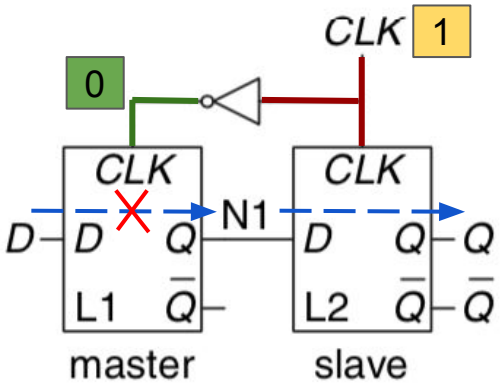
- D controls what the next state should be.
- CLK controls when the state should change.
- CLK=0 ⇒ the outputs do not change (opaque)
- CLK=1 ⇒ the outputs changes corresponding to D (transparent)

D Flip-Flop

A flip-flop is built from two D latches, controlled by a common clock signal (CLK). It copies D to Q on the rising edge of the clock and remembers its state at all other times.



- When CLK = 0,
  - the master latch is transparent and the slave is opaque.
  - D goes through to N1.

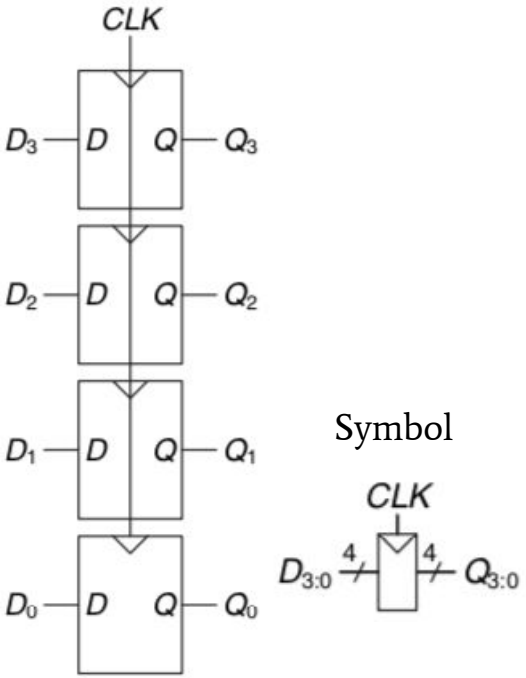


- When CLK = 1,
  - the master goes opaque and the slave becomes transparent.
  - D is cut off from N1
  - N1 goes through to Q.

Registers

A register is a bank of flip-flops that share a common CLK input.

- All bits of the registers are updated at the same time.
- If there is N flip-flops, the register is call N-bit register.
    - e.g., if a register has 32 flip-flops, we call it 32-bit register; the length of its input/output is a 32-bit long binary string.

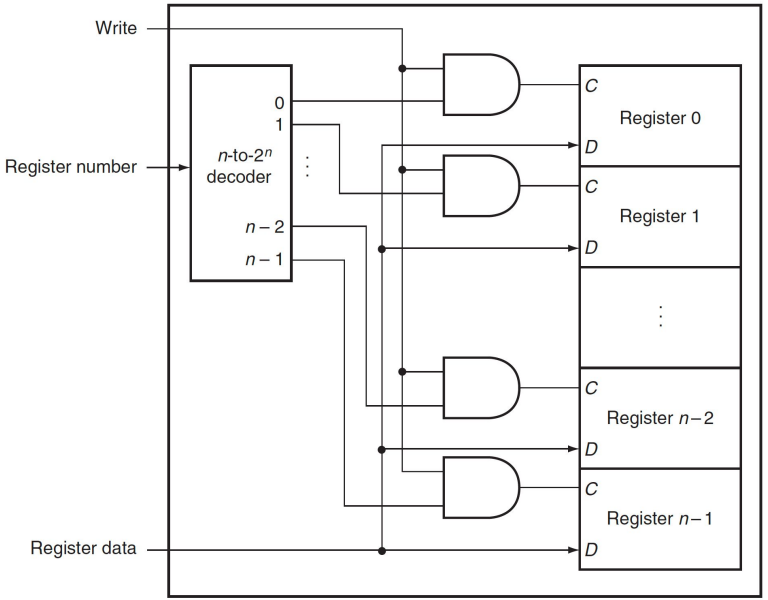


A 4-bit register

Register files

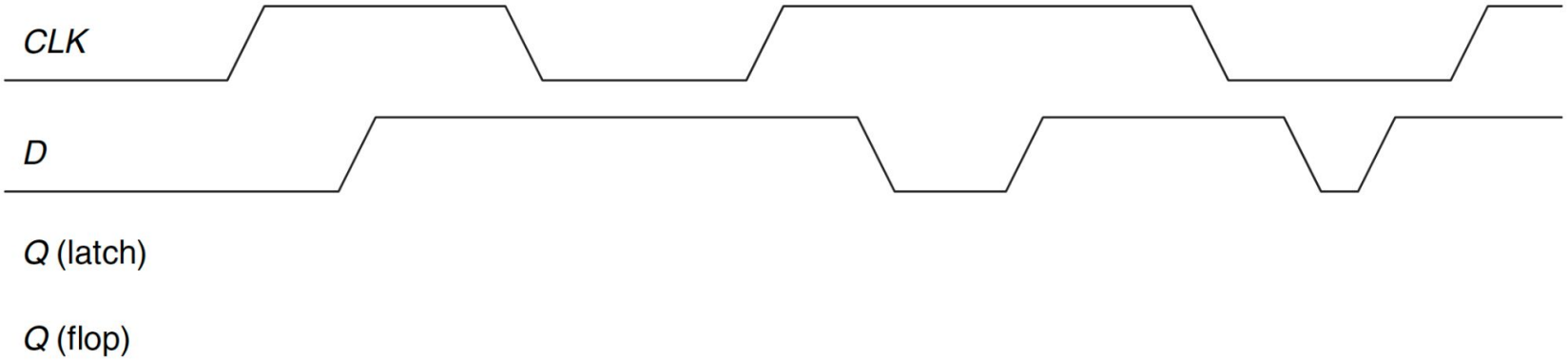
A register file consists of a set of registers that can be read and written by supplying a register number to be accessed.

- A register file can be implemented with a decoder for each read or write port and an array of registers built from D flip-flops. (Note: D flip-flops are not the only circuits for building register files; it can be built from other memory circuits.)



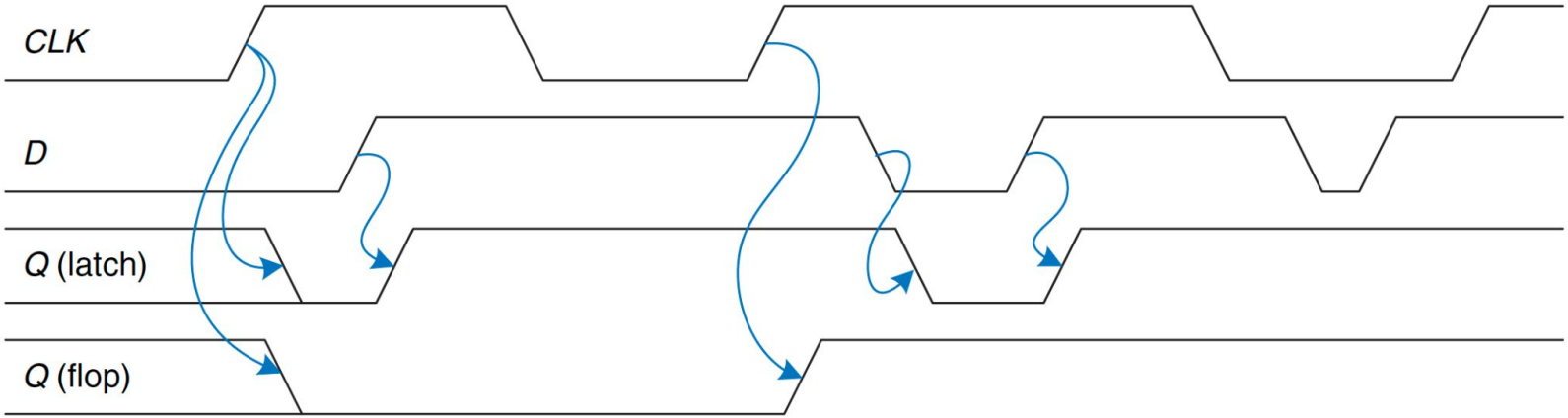
Exercise 1 (H&H Example 3.2)

Ben Bitdiddle applies the D and the CLK inputs shown below to a D latch and to a D flip-flop. Help him determine the output Q of each device.



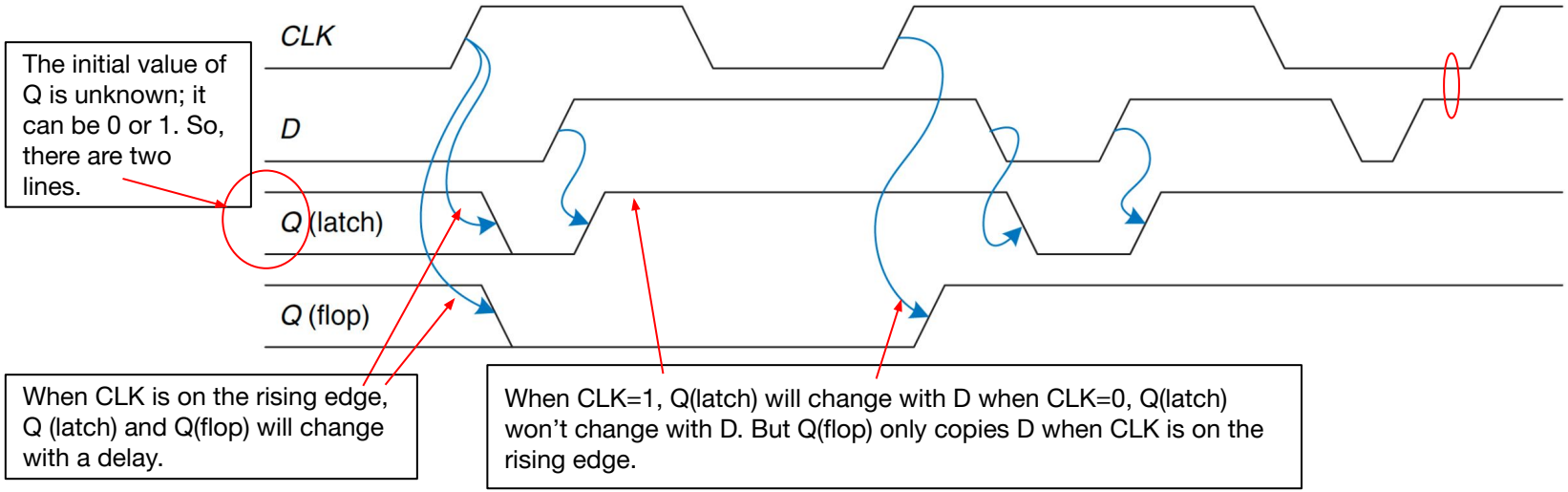
Exercise 1 (H&H Example 3.2)

Ben Bitdiddle applies the D and the CLK inputs shown below to a D latch and to a D flip-flop. Help him determine the output Q of each device.



Exercise 1 (H&H Example 3.2)

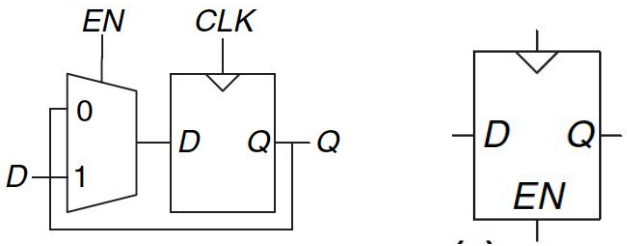
Ben Bitdiddle applies the D and the CLK inputs shown below to a D latch and to a D flip-flop. Help him determine the output Q of each device.



Enable flip-flop

An enable flip-flop adds another input called EN or ENABLE to determine whether data is loaded on the clock edge.

- When EN is True (=1), it behaves like an ordinary D flip-flop.
- When EN is False (=0), it ignores the clock and retains its state.

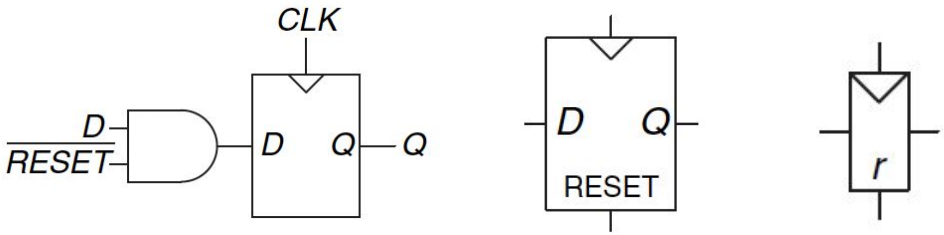


Enable flip-flop schematics (left) and its symbol (right): It allows us to load a new value into a flip-flop only some of the time, rather than on every clock edge.

Resettable flip-flop

A resettable flip-flop adds another input called RESET.

- When RESET is False, it behaves like an ordinary D flip-flop.
- When RESET is True, it ignores D and resets the output to 0.



The schematic (left) and symbols (middle and right) of Resettable flip-flop. It allows us to force a known state (i.e., 0) into all the flip-flops in a system when we first turn it on.

Sequential circuits

Sequential circuits: the output cannot be determined by simply looking at the inputs.

Some problematic circuits: unstable circuits

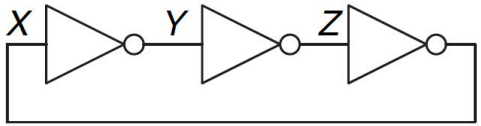
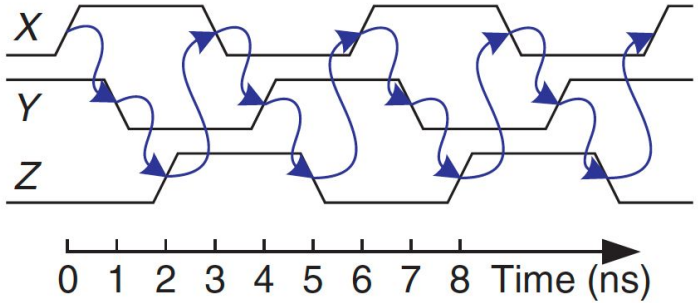


Figure 3.16 Three-inverter loop

(Ring oscillator)

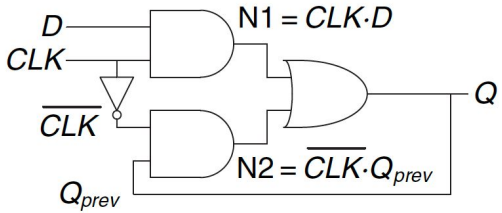


Suppose X is initially 0; then Y=1, Z=0, and hence X=1, which leads to each variable oscillates between 0 and 1.

Sequential circuits

Some problematic circuits: race conditions

CLK	D	Q_{prev}	Q
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1



The circuit is designed to behave as a D latch, but it has a problem:

- When CLK=1, D=1, we have Q=1, then CLK falls, Q should keep Q=1.
- But if the delay through inverter from CLK to $\overline{CLK}$ is longer than the delays of the AND gate and OR gate, N1 and Q may both fall and result in Q=0 before $\overline{CLK}$ rise. In such a case, N2 will never be rise, and Q=0.

Synchronous sequential circuits

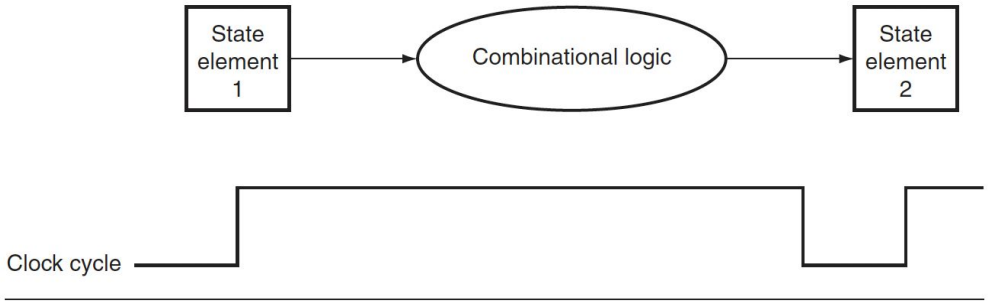
The problems (unstable/races) in the previous two sequential circuits is because they have cyclic paths, in which outputs are fed directly to inputs.

To avoid these problems, designers can break the cyclic path by inserting registers somewhere in the path. So, the circuit is transformed into a collection of combinational logic and registers.

- The registers contain the state of the system, which changes only at the clock edge. So, the state is synchronized to the clock.
- The clock cycle must have a long enough period that allows the inputs to all registers to be settled before next clock edge.

Synchronous sequential circuits

The figure shows the relationship among the state elements and the combinational logic blocks in a synchronous, sequential logic design.



- The state elements, whose outputs change **only after** the clock edge, provide valid inputs to the combinational logic block.
- To ensure that the values written into the state elements on the active clock edge are valid, the **clock must have a long enough period** so that all the signals in the combinational logic block stabilize, then the clock edge samples those values for storage in the state elements. This constraint sets a **lower bound** on the length of the clock period, which must be long enough for all state element inputs to be valid.

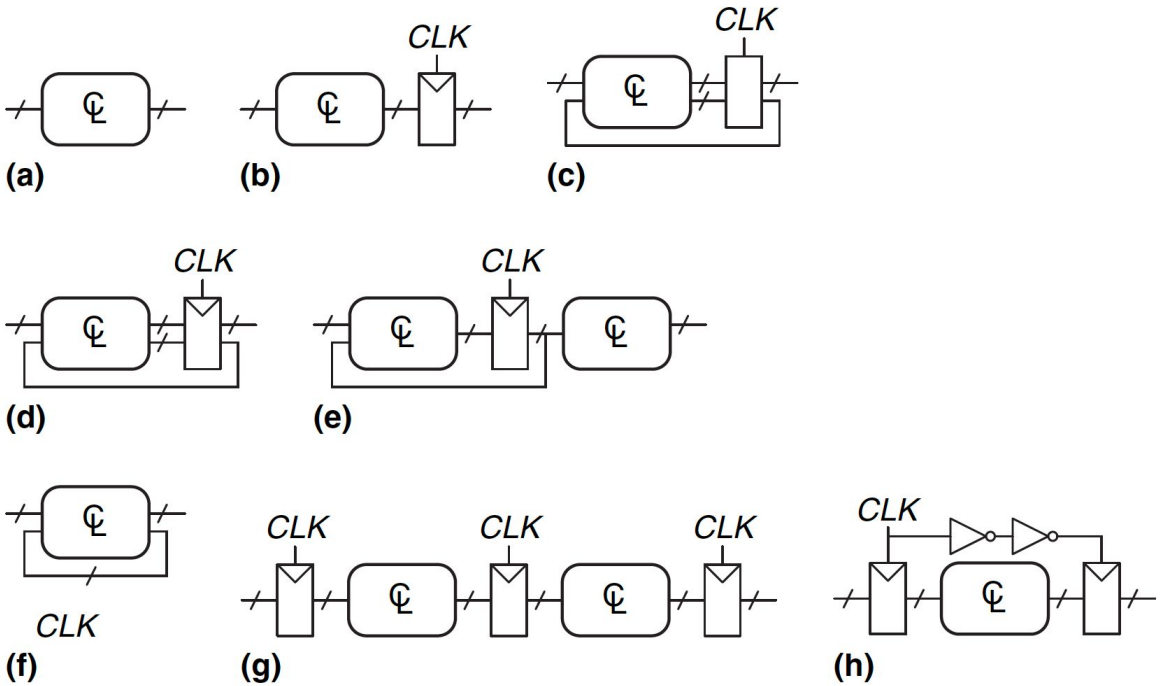
Synchronous sequential circuits

A circuit is sequential synchronous when:

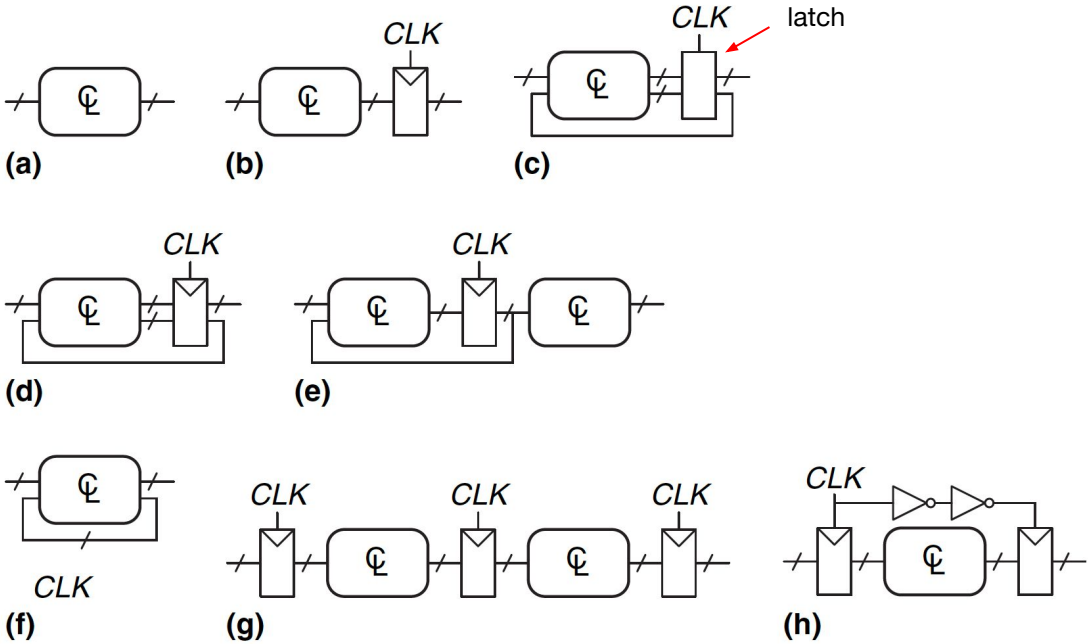
- every circuit element is either a register or a combinational circuit
- at least one circuit element is a register
- all registers receive the same clock signal
- every cyclic path contains at least one register.

Sequential circuits that are not synchronous are called asynchronous.

Exercise: which circuits below are sequential synchronous?



Exercise: which circuits below are sequential synchronous?



(a) Combinational
(b) Synchronous
(c) Neither CL nor Sync
(d) Synchronous
(e) Synchronous
(f) Neither CL nor Sync
(g) Synchronous
(h) Strictly, not synchronous, the second register receives a CLK delayed by the inverters.

Asynchronous sequential circuits

These circuits do not use a clock signal but uses the pulses of the inputs.

- These circuits are faster than synchronous sequential circuits because the state transitions happen immediately when inputs change, without a clock signal.
- We use asynchronous sequential circuits when speed of operation is important and independent of internal clock pulse.
    - In modern vehicles, **airbag deployment systems** or **crash sensors** rely on asynchronous circuits to immediately detect an event (e.g., collision) and trigger the appropriate response without waiting for a clock pulse.
- These designs are more complex and sensitive to input changes; their output is uncertain.

Exercise 2 (H&H exercise 3.1)

Given the input waveforms shown in the figure below, sketch the output Q, of an SR latch.

Case	S	R	Q	$\bar{Q}$
IV	0	0	Q_{prev}	$\bar{Q}_{prev}$
I	0	1	0	1
II	1	0	1	0
III	1	1	0	0