Durham University

Digital Electronics

Sequential Circuits

Dr. Eleni Akrida
eleni.akrida@durham.ac.uk

Last week

Timing

- Propagation & contamination delay
- Critical & short path
- Glitches

Advanced Adders

- Ripple adder
- Carry-lookahead adder
- 2-level carry-lookahead adder

Overview of today’s lecture

- SR and D-latches
- Flip-flops
- Synchronous sequential circuits
- Pipelining

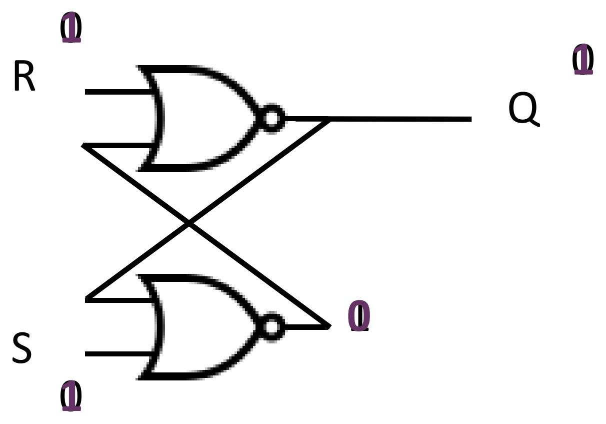
Circuits

- So far we have looked at **combinational circuits:** output depends only on current input (after propagation delay).

- Now we consider **sequential circuits**.
  - Output depends on **history** as well as current input
  - I.e. the circuit has **memory**.
  - Can be modelled as **finite-state machines.**
  - In general hard to analyse.
  - Fundamental components: **latches** and **flip-flops**, each store 1-bit.

- We will consider **synchronous** sequential circuits made from combinational components interleaved with banks of flip-flops containing the state of the circuit.

SR-Latch



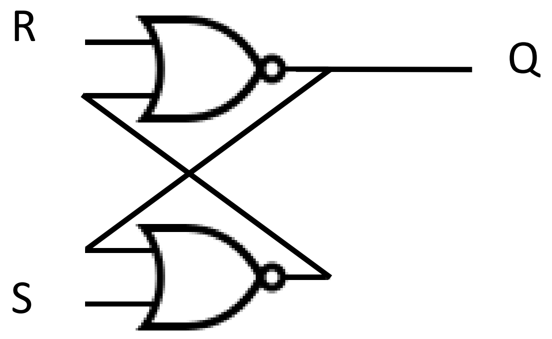
- NOR latch

- Both inputs ‘usually’ set to 0.
- If input S (set) has a pulse of 1, the output becomes 1.
- The output remains 1 even when the pulse is over.
- If input R (reset) has a pulse of 1, the output becomes 0.
- The output remains 0 even when the pulse is over.

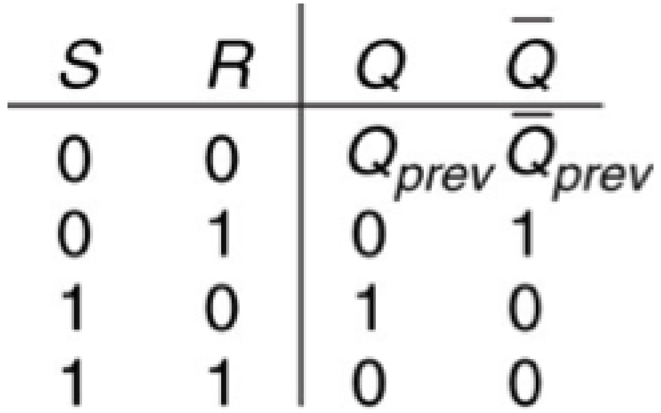
- If the output is already 1, a pulse on S will not change it. If it is already 0, a pulse on R will not change it.
- Latches can be used to store a single bit of memory.
- This circuit has two stable states for a given input – it is called **bistable**.

SR-Latch

- NOR latch

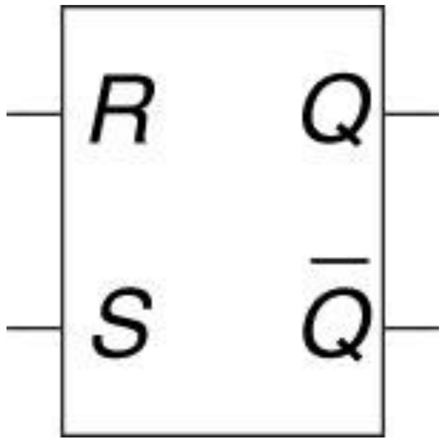


- **Truth table**



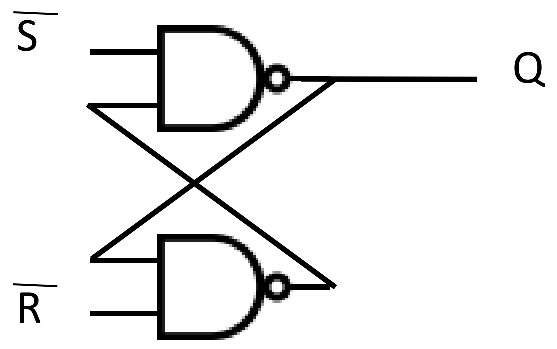
S	R	Q	$\bar{Q}$
0	0	Q_{prev}	$\bar{Q}_{prev}$
0	1	0	1
1	0	1	0
1	1	0	0

Symbol



SR-Latch

- NAND latch

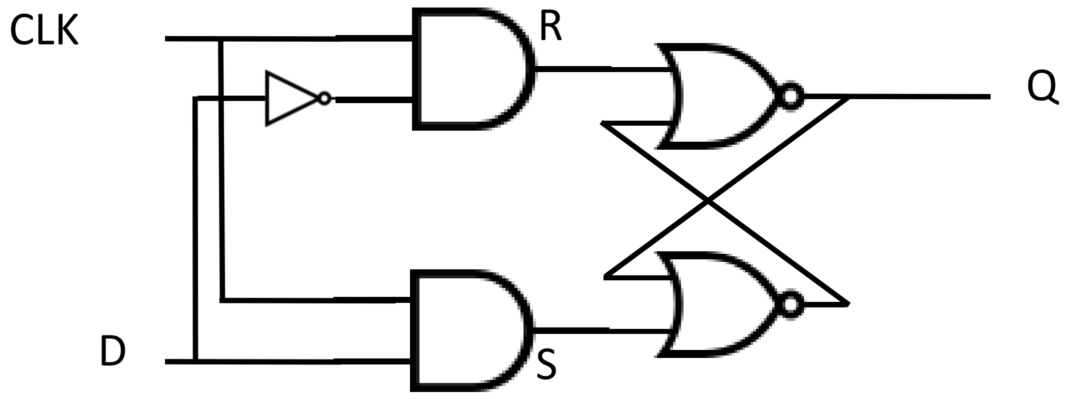


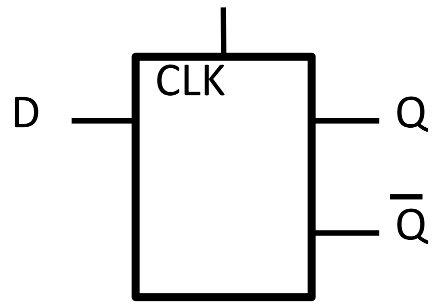
- Latches can also be built from NAND gates.
- In this case the ‘usual’ state for the inputs is **1**, so the inputs are denoted with bars.
- **Exercise**: work out how the NAND latch behaves.

D-Latch

- In a latch a pulse on set or reset indicates **what** the new state should be, and **when** it should change (now!).

- Designing circuits becomes easier if we can separate **what** and **when**.

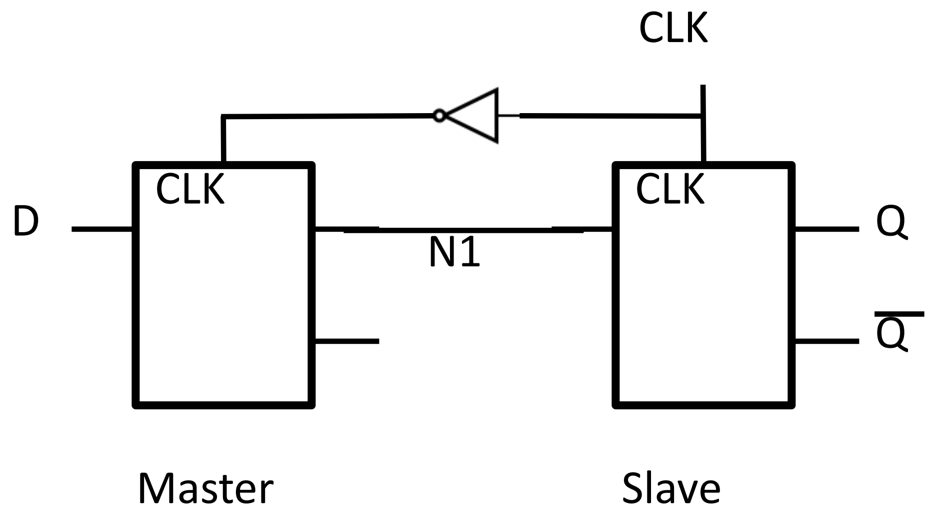




- D – **data** input. Defines **what** the new value should be.
- CLK – **clock** input. Define **when** the new value should arise.

D Flip-flop

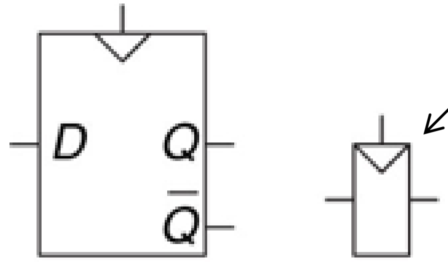
- In the D-latch the output can change **whenever** the clock is high.
- Even better is if we can make it change **only at the moment the clock goes high.**



- **D flip-flop copies D to Q on the rising edge of the clock, and remembers its state at all other times.**

D Flip-flop

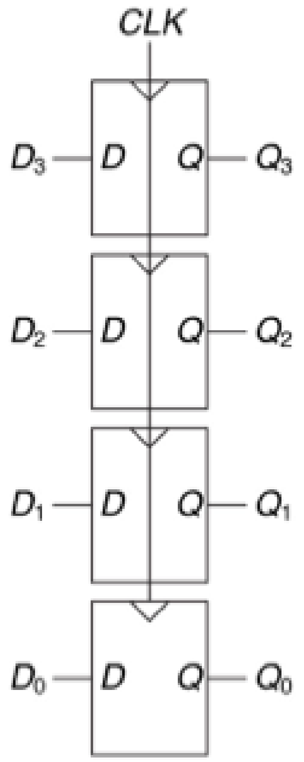
- Full and compact symbols:

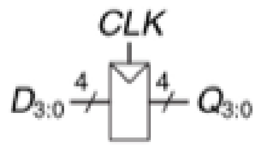


Little triangle denotes 'driven by rising clock edge'

- **Register**:

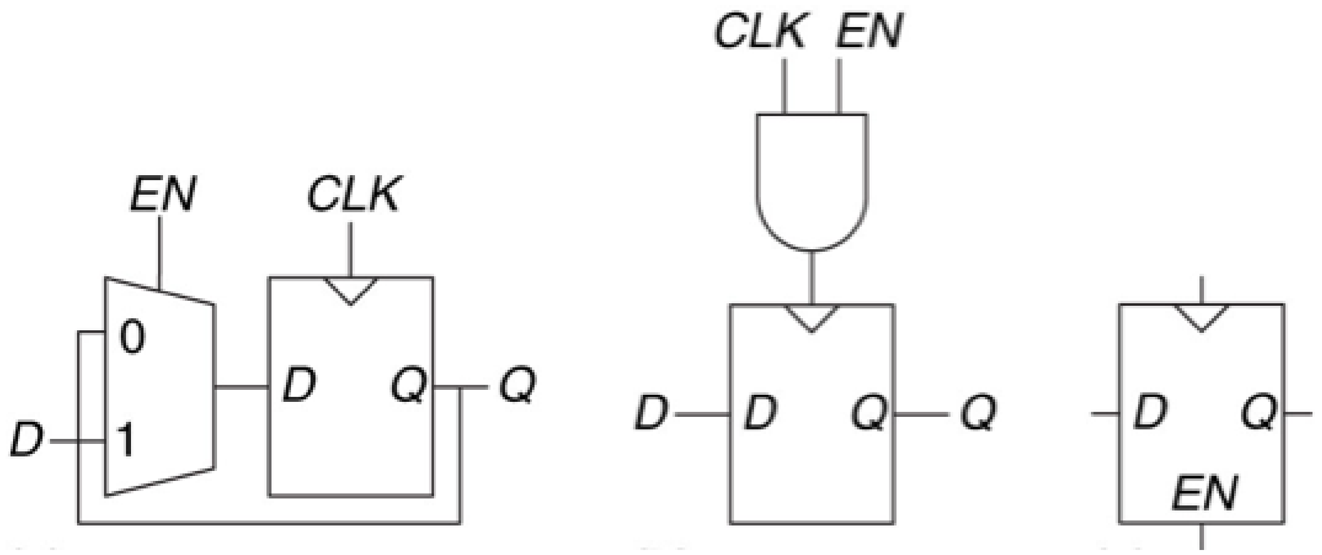
  a bank of flip-flops driven by the same clock





Enabled Flip-flop

- Incorporates an additional input (**enable**) to control whether the data is loaded into the register or not.

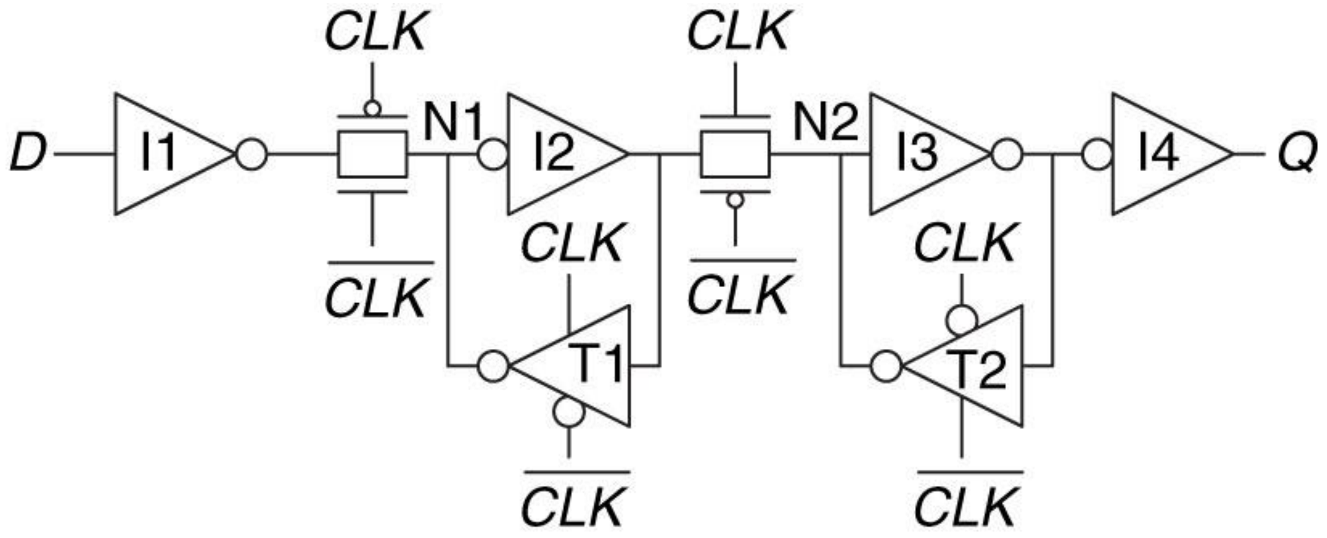


- EN can control the input with a multiplexor, or can control the clock. This is called a **gated clock.**
- Gated clocks can cause **timing errors and glitches** on the clock**.**

Flip-flop Design

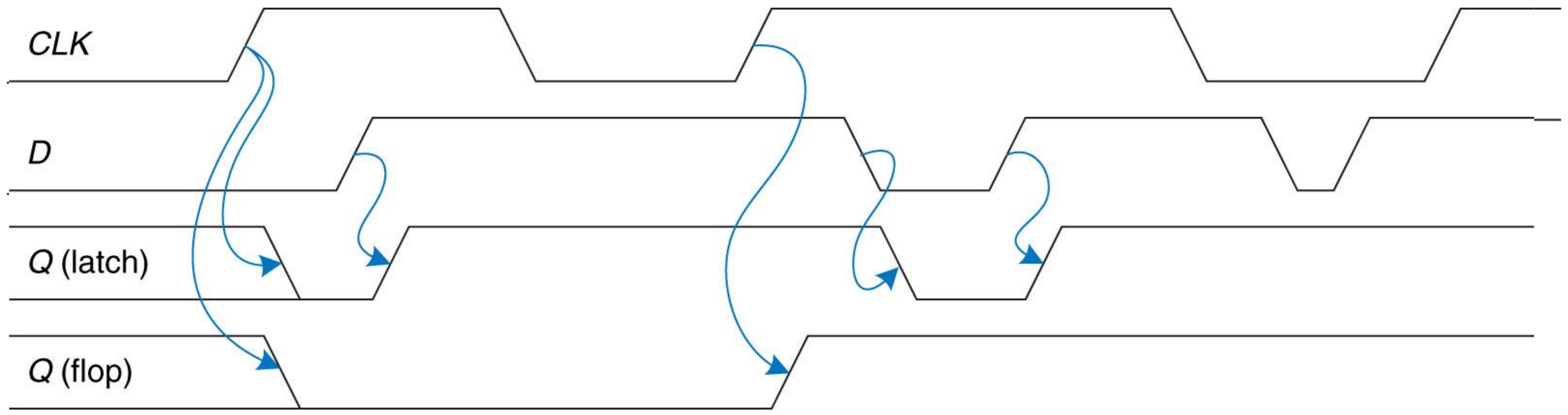
- How many **transistors** are needed to build the D flip-flop?
  - A NAND or NOR gate uses four transistors.
  - A NOT gate uses two transistors.
  - An AND gate is built from a NAND and a NOT, so it uses six transistors.
  - The SR latch uses two NOR gates, or eight transistors.
  - The D latch uses an SR latch, two AND gates, and a NOT gate, or 22 transistors.
  - The D flip-flop uses two D latches and a NOT gate, or 46 transistors.

- Direct design can make more **space-efficient** flip-flops. Here 20 transistors.

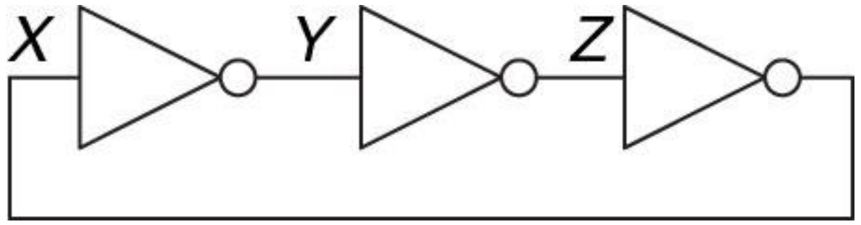


Example

- How would a latch and flip-flop behave in the following setting?

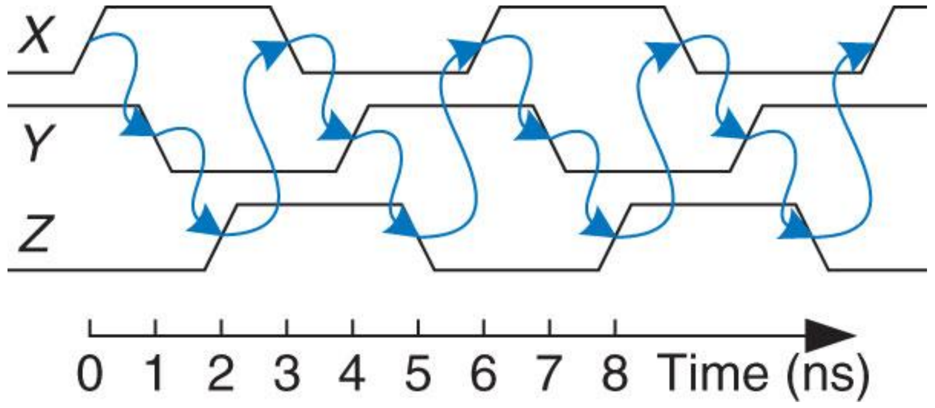


Problem circuits



- The gates have a delay of 1ns.
- How does this circuit behave?
- Suppose X=0, then Y=1 and Z=0, and so X=1!
- X rises to 1, at 1ns Y falls, at 2ns Z rises and at 3ns X falls…

- An **unstable** or **astable** circuit.

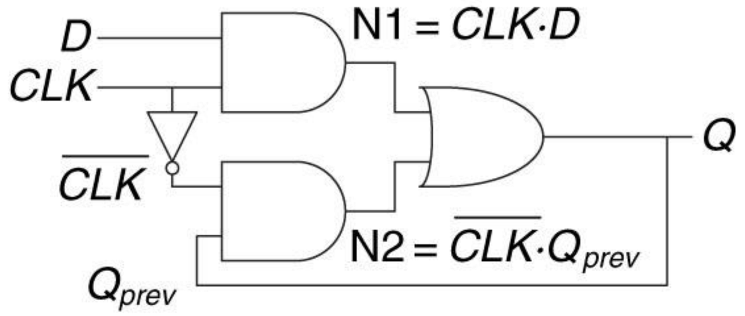


Problem circuits

- An **improved** D latch? Fewer gates, so fewer transistors!

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

$$Q = CLK \cdot D + \overline{CLK} \cdot Q_{prev}$$



Leads to **race conditions** – behaviour depends on which of two routes through the circuits carries signal the fastest.

Logically identical circuits may exhibit different behaviour depending on technicalities of the gate construction, or may exhibit odd behaviour at certain temperatures…

Problem circuits

- The problems are caused by outputs being fed back into inputs: the circuits contain **loops** or **cyclic paths.**
- To avoid this we insert **registers** into cyclic paths:
  - the registers contain the ‘state’ of the circuit
  - they break the paths
  - they only update on a clock edge
  - say they are **synchronised** to the clock
- If the clock is **sufficiently slow**, so that all inputs to all the registers have settled before the next clock edge, then race conditions cannot arise.

Combinational logic

Reminder

Combinational logic rules:

- **Individual gates** are combinational circuits.
- Every circuit **element** must be a combinational circuit.
- Every node is either an **input** to the circuit or connecting **to exactly one output** of a circuit element
- The circuit has **no cyclic paths** – every path through the circuit visits any node at most once.

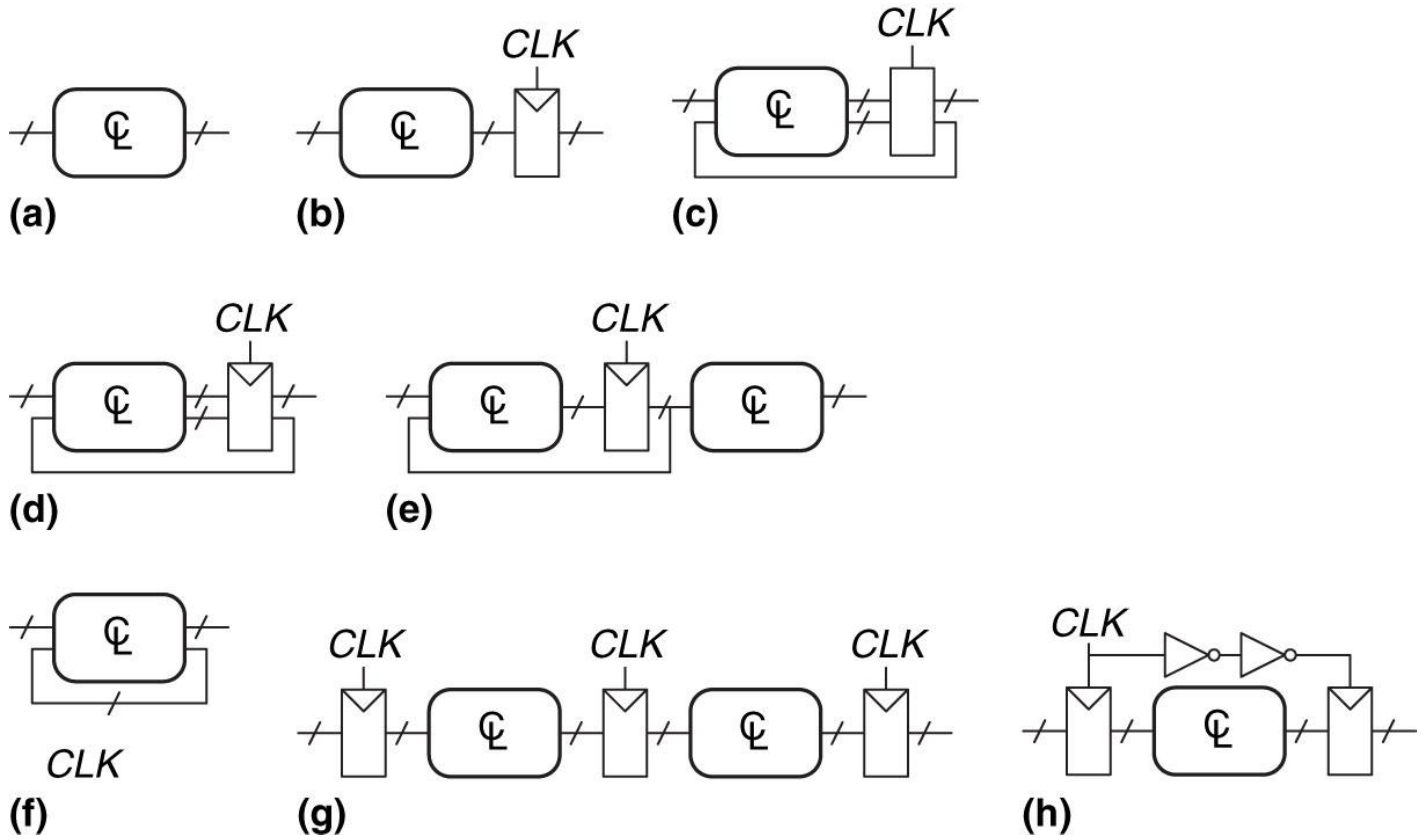
Synchronous circuits

- A **synchronous sequential circuit** consists of interconnected elements such that:
- Every circuit element is either a **combinational circuit** or a **register**
- At least one circuit element is a register
- All registers receive the same **clock signal**
- Every **cyclic path** contains **at least one register**.

- A **synchronous sequential circuit** has:
- A discrete set of states $\{S_0,\ldots,S_{k-1}\}$
- A clock input, whose rising edge indicates when a state change occurs
- A functional specification which details the next state and all outputs for each possible current state and set of inputs.

Examples?

- Which are synchronous sequential circuits?



Timing

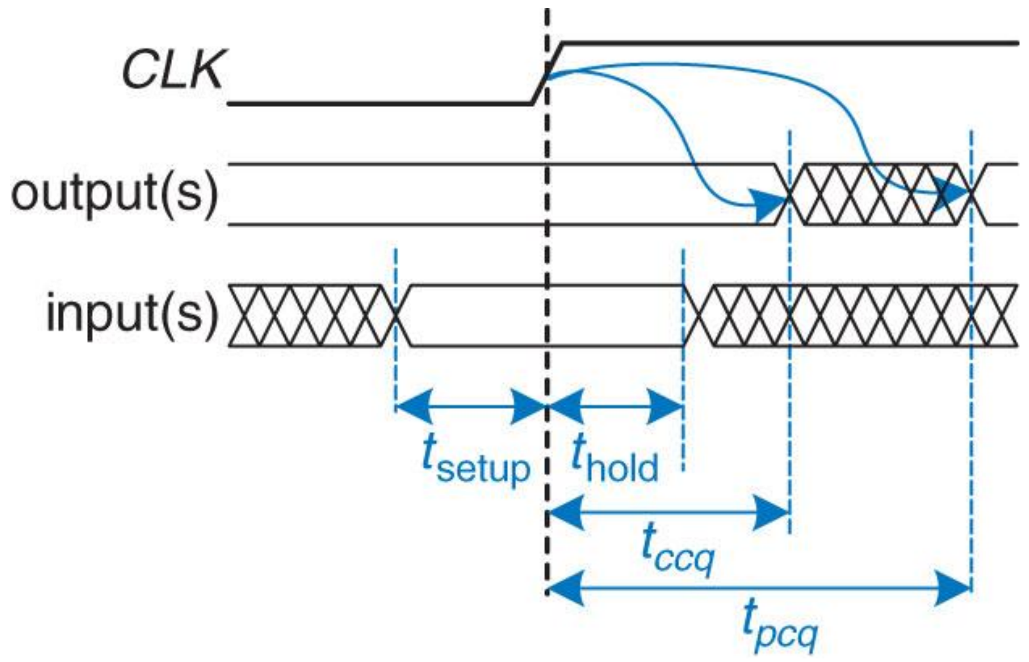
The **dynamic discipline** restricts us to using circuits satisfying timing constraints that allow us to combine components.

t_{setup} – time before rising edge during which inputs must be stable

t_{hold} – time after rising edge during which inputs must be stable

t_{ccq} – time until output starts to change

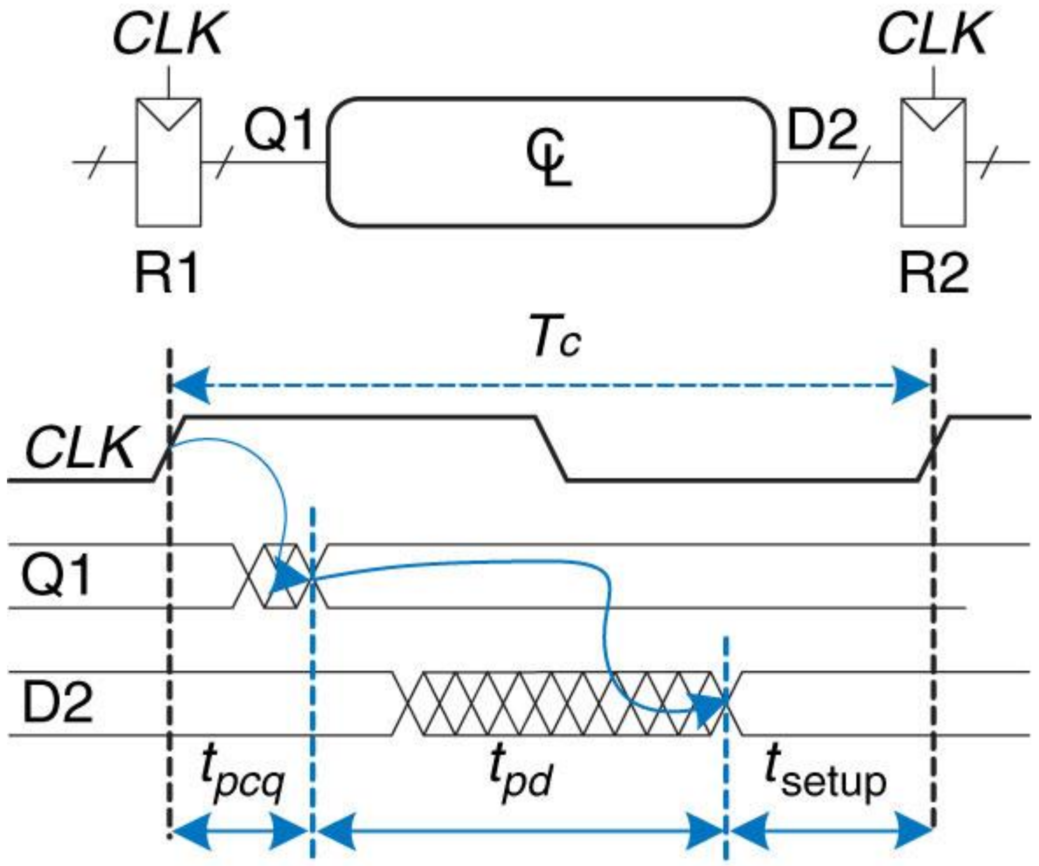
t_{pcq} – time by which output has stabilized



With these constraints we can think of signals as discrete in time as well as logic.

Setting time

- The **clock frequency** determines how fast the computer operates.
- Register 2 will not get an answer until the clock ticks a second time.
- 30 years ago 1MHz was a fast computer – now several GHz is common.



Time between ticks (T_c) must be at least $t_{pcq}+t_{pd}+t_{setup}$

Setting time

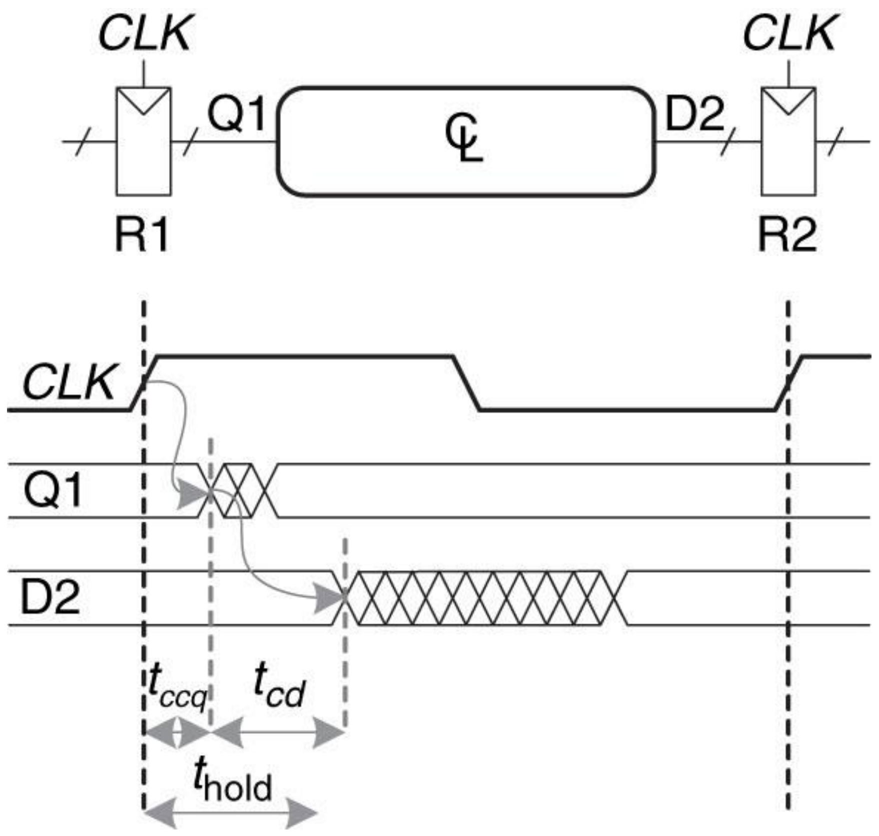
- Rearranging we see that $t_{pd} < T_c - (t_{pcq} + t_{setup})$
- $(t_{pcq} + t_{setup})$ is called the **sequencing overhead.**

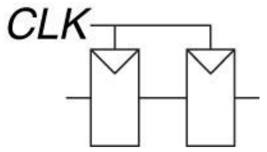
- The clock speed and sequencing overhead are normally fixed, and the designer must work with them.
- Designers must get all elements of combinational logic to work within the bound on t_{pd} in order for the circuit to be reliable.

Setting time

There is also a minimum delay requirement:

- D2 must hold its value for at least t_{hold} time after the rising edge.
- It could change as soon as $t_{ccq} + t_{cd}$.
- Designers have a **min-delay constraint**: $t_{cd} > t_{hold} - t_{ccq}$.
- Often, in order to allow direct connection of flip-flops, $t_{hold} < t_{ccq}$, so the min-delay constraint isn't technically an issue.





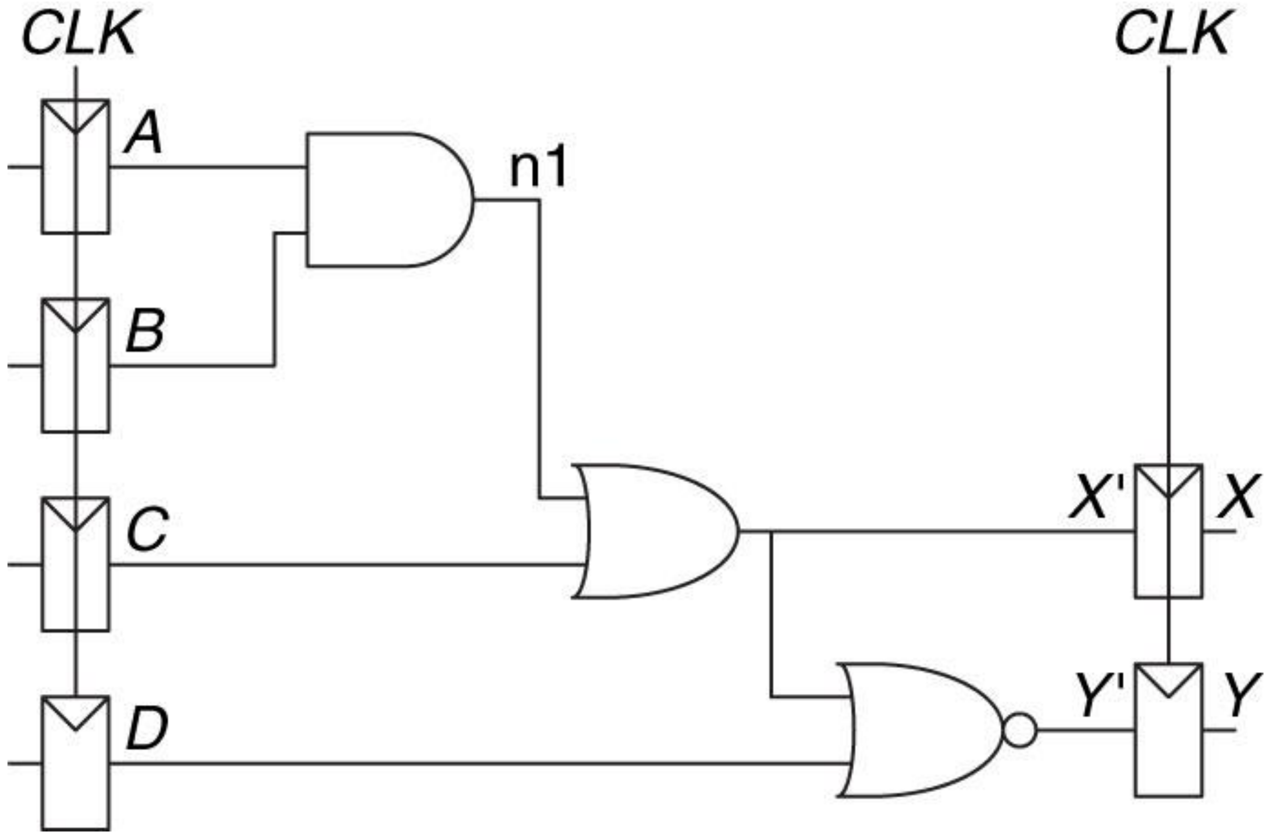
Example

Flip-flops:

- t_{ccq} = 30 ps
- t_{pcq} = 80 ps
- t_{setup} = 50 ps
- t_{hold} = 60 ps

Gates:

- t_{cd} = 25 ps
- t_{pd} = 40 ps



- What is the max clock frequency?
- Are there hold time violations?

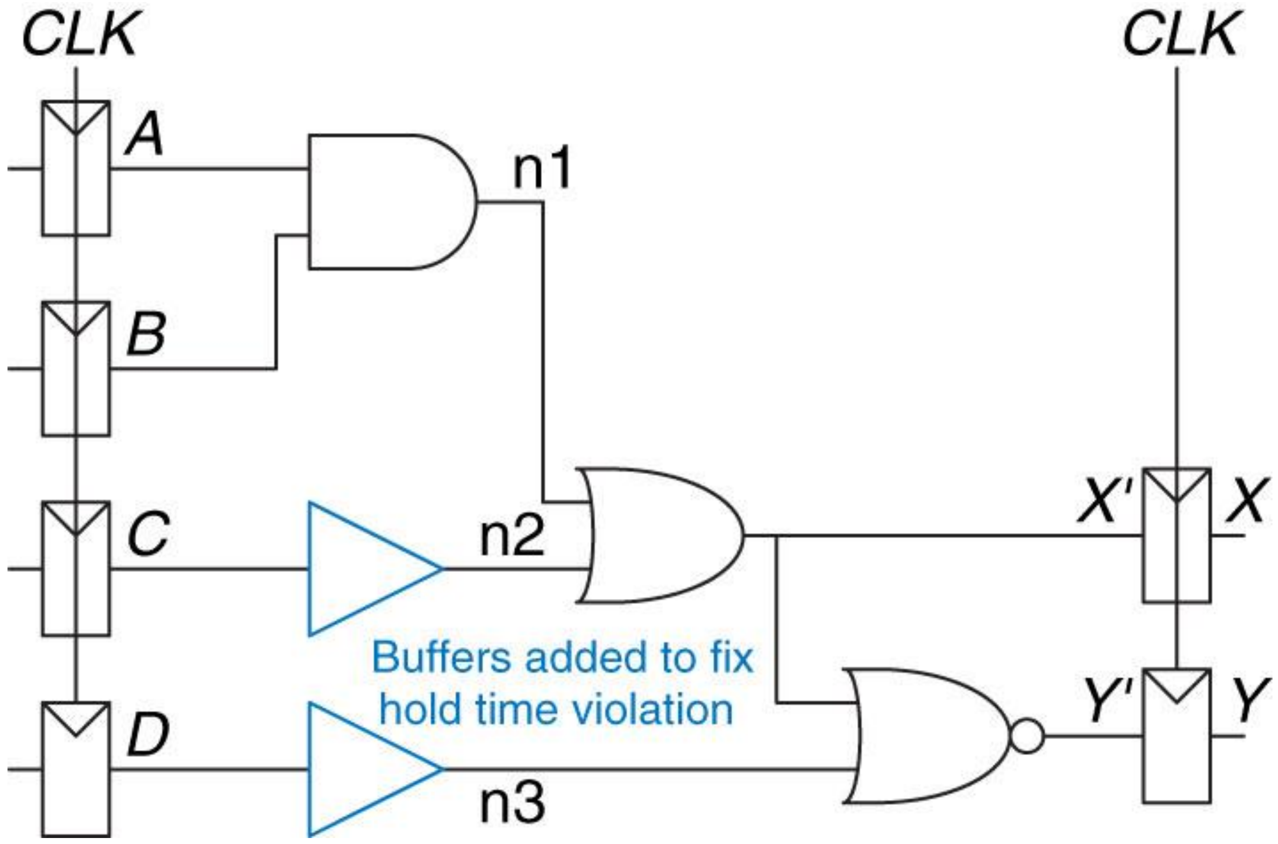
Fixing the hold time violation

Flip-flops:

- t_{ccq} = 30 ps
- t_{pcq} = 80 ps
- t_{setup} = 50 ps
- t_{hold} = 60 ps

Gates:

- t_{cd} = 25 ps
- t_{pd} = 40 ps



- What is the max clock frequency now?
- Are there hold time violations?

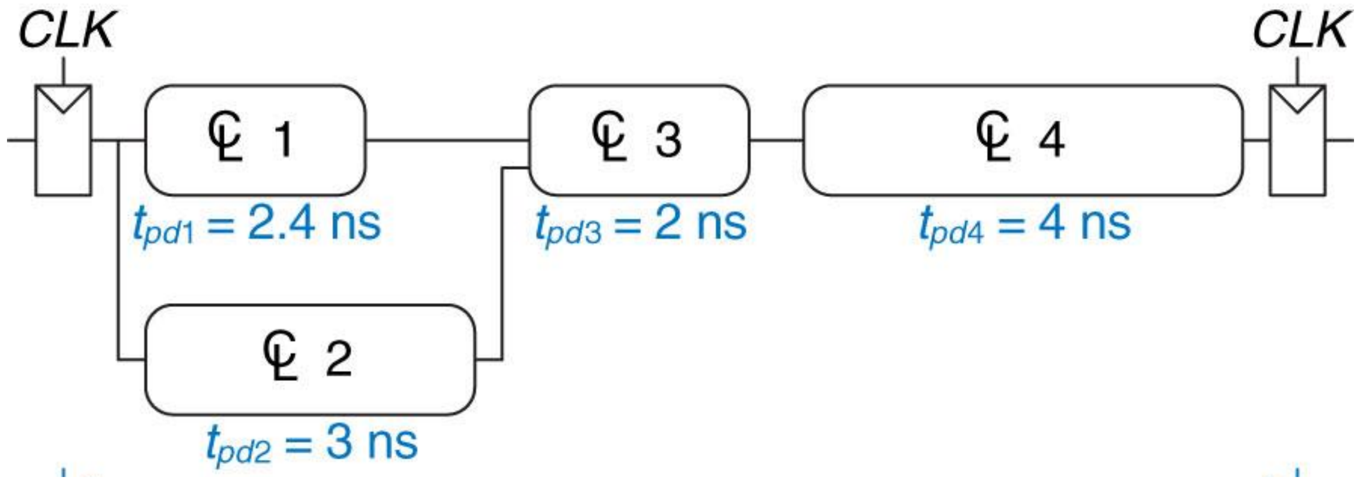
Pipelining

(T_{pcq} = 0.3ns, T_{setup}=0.2ns)

What is the minimum time between clock ticks allowed for this circuit?

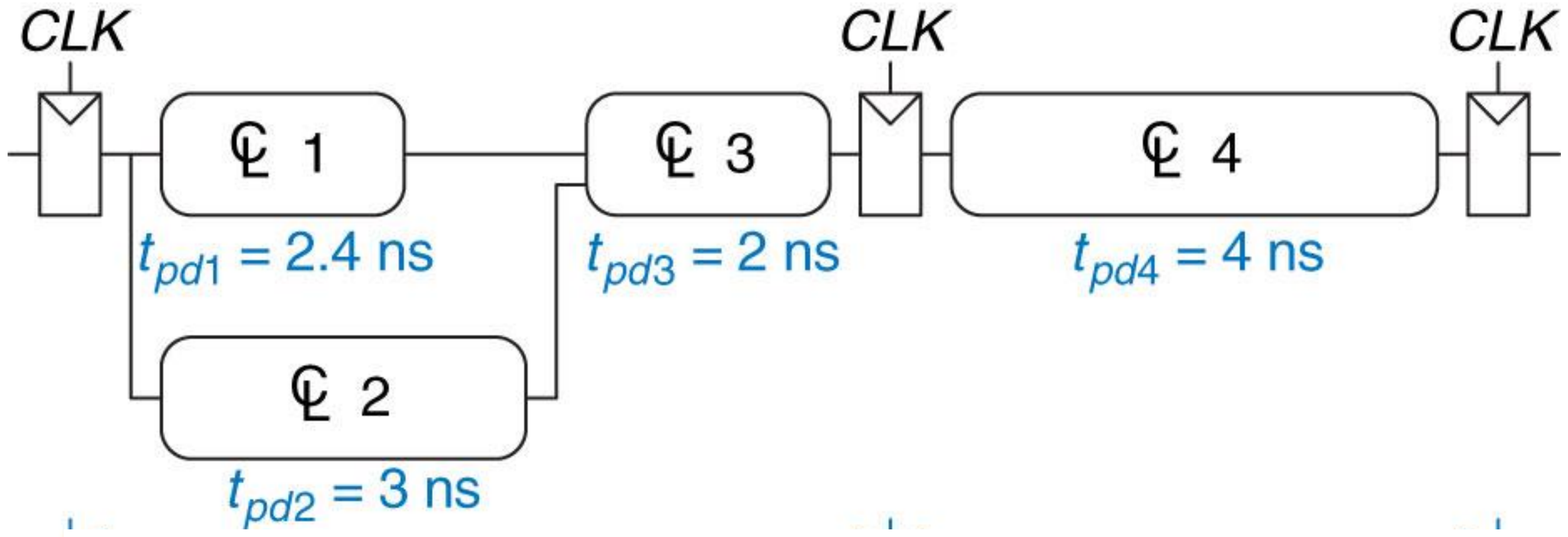
What is the **latency**, i.e. time it takes for a single block of data to get from the beginning to the end?

What is the **throughput,** i.e. rate at which we get data through?



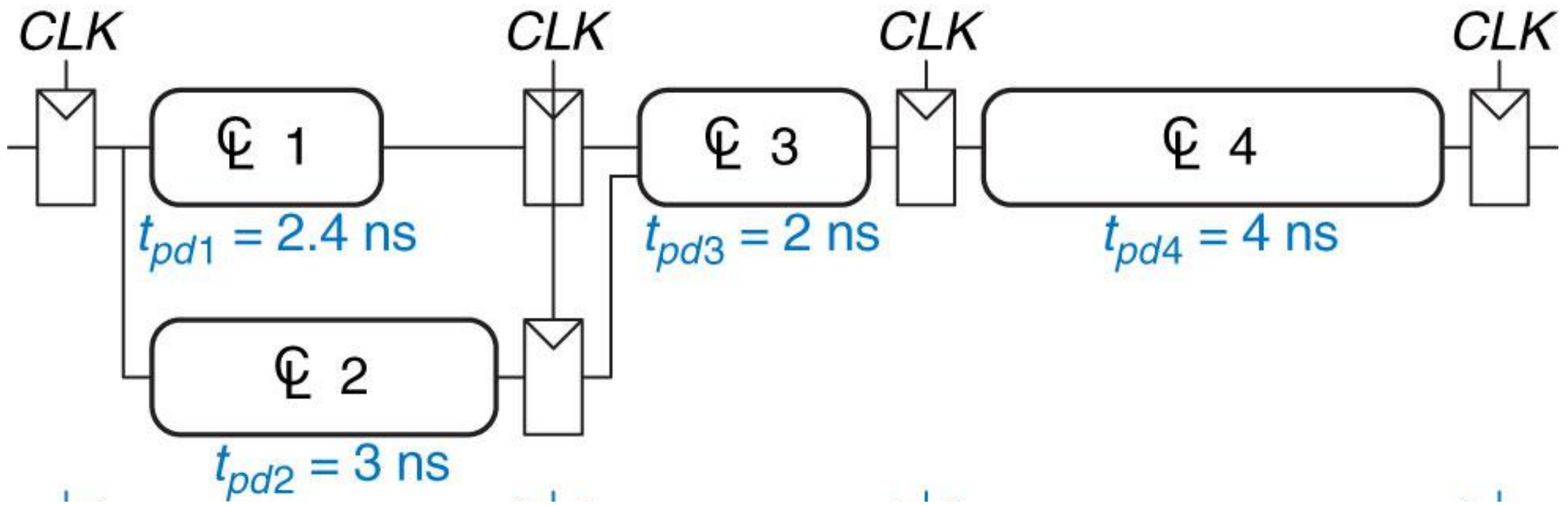
Pipelining

Insert an additional register in the circuit. (T_{pcq} = 0.3ns, T_{setup}=0.2ns)



Pipelining

Insert another register in the circuit. (T_{pcq} = 0.3ns, T_{setup}=0.2ns)


CLK
CLK
CLK
CLK
CL 1
CL 3
CL 4
CL 2
t_{pd1} = 2.4 ns
t_{pd3} = 2 ns
t_{pd4} = 4 ns
t_{pd2} = 3 ns