Synchronous sequential logic

ECE_3TC31_TP/INF107

Tarik Graba Ulrich Kühne Guillaume Duc

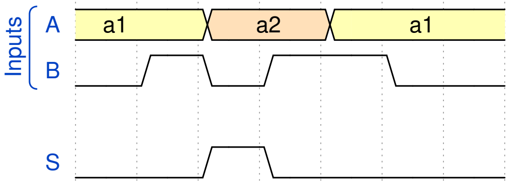
2023-07

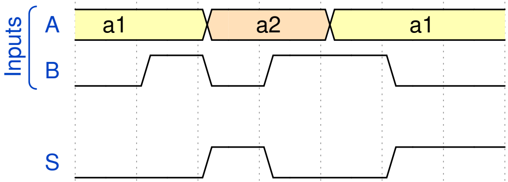
Introduction

Combinational logic

- The output of a function depends only on the present value of the inputs
  - For the same values of the inputs, the output is *always* the same
- It is used to build logic and arithmetic operators

Combinational vs. Sequential logic





Which timing diagram cannot be produced by combinational logic?

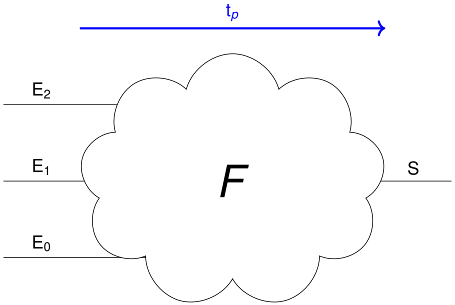
Combinational vs. Sequential logic

With sequential logic, the output of a function depends on:

- the present value of its input signals
- and on the sequence of past inputs

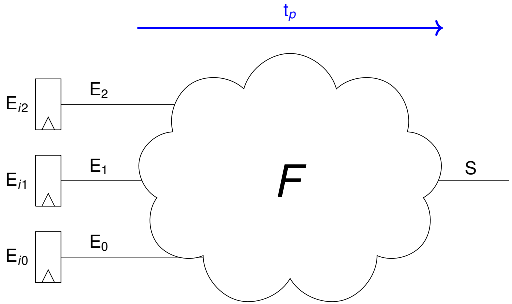
The function has a state (or memory)

How to deal with propagation time



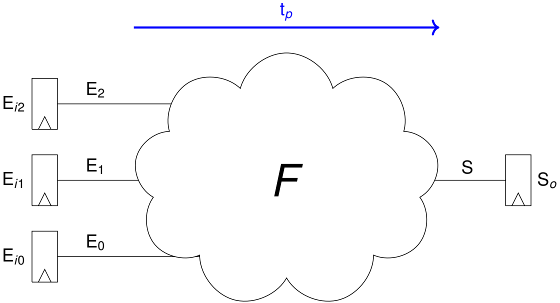
- The propagation time t_p is not null
- During this time
  - **The output (S) is not valid**
  - **The inputs (E_0, E_1, E_2) must not change**
- How to chain computations (i.e. perform $F(E_0, E_1, E_2)$ then $F(E'_0, E'_1, E'_2)$...)?

How to deal with propagation time



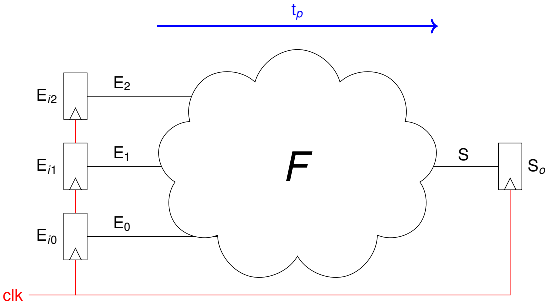
- We maintain the inputs values (E_0, E_1, E_2) stable for at least t_p
- By adding a memory component that
  - samples (updates E_0, E_1, E_2 from the values of its inputs $E_{i_0}, E_{i_1}, E_{i_2}$)
  - memorises (keep the values of E_0, E_1, E_2 stable as long as it is needed)

How to deal with propagation time



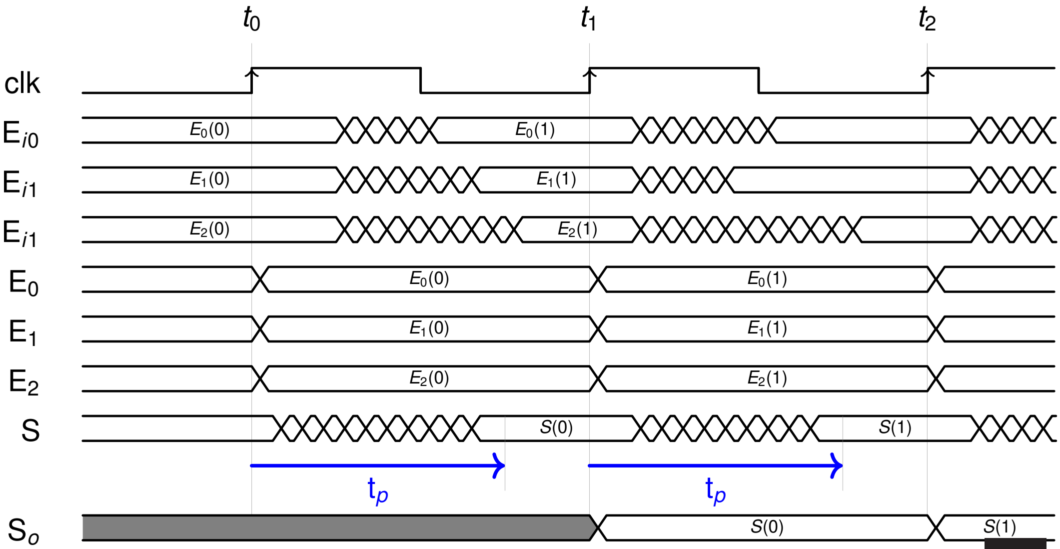
- Once the results is available
  - The output S is sampled
  - The inputs $E_{i_0}, E_{i_1}, E_{i_2}$ can sample new values at the same time
  - The output S_o is memorised and can be used in another computation

How to deal with propagation time



- We use the same signal to **synchronize** sampling in all the flip-flops: **the clock** (clk)

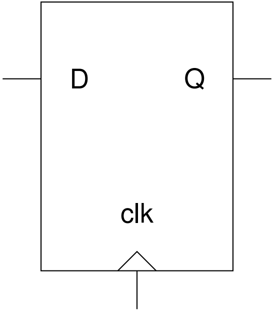
How to deal with propagation time



D flip-flop

D flip-flop

- D flip-flop (dff, register…)
- Input: `D`
- Output: `Q`
- Clock input: `clk`



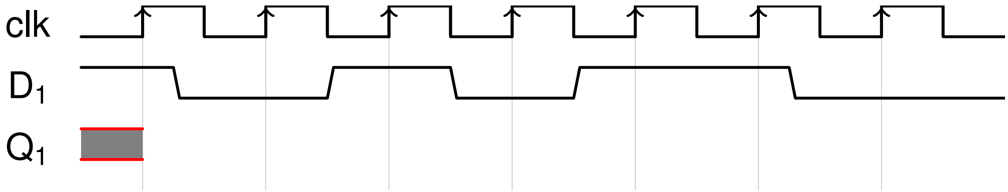
Operation

- When the clock `clk` goes from 0 to 1 (*rising edge*), the value of the input `D` is captured and copied to the output `Q` (*sampling*)
- The rest of the time, the value of the output `Q` does not change (*memorization*)

Truth table

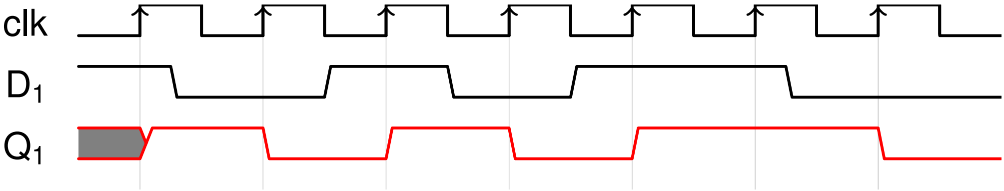
D	clk	Q	Operation
0	$\uparrow$	0	D is copied to Q (*sampling*)
1	$\uparrow$	1	D is copied to Q (*sampling*)
$\times$	0	Q	Q keeps its value (*memorization*)
$\times$	1	Q	Q keeps its value (*memorization*)
$\times$	$\downarrow$	Q	Q keeps its value (*memorization*)

Exercise

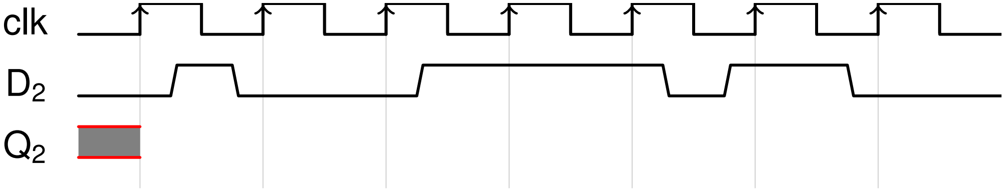


Complete this timing diagram

Correction

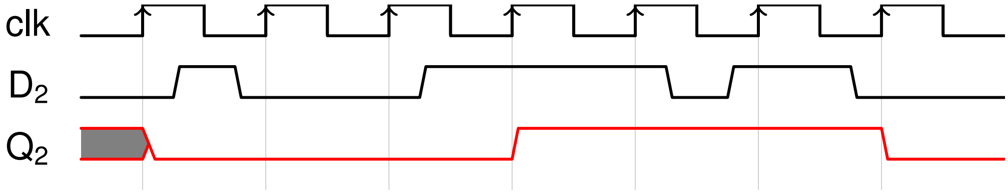


Exercise

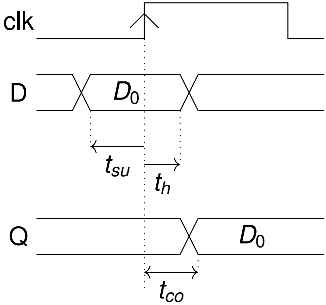


Complete this timing diagram

Correction



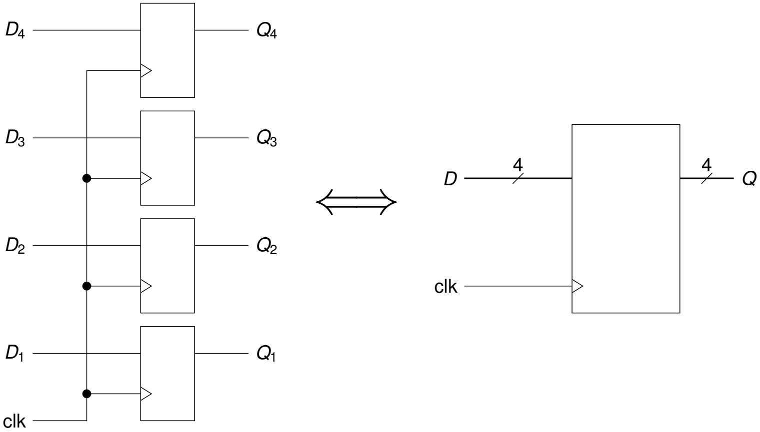
Timing constraints



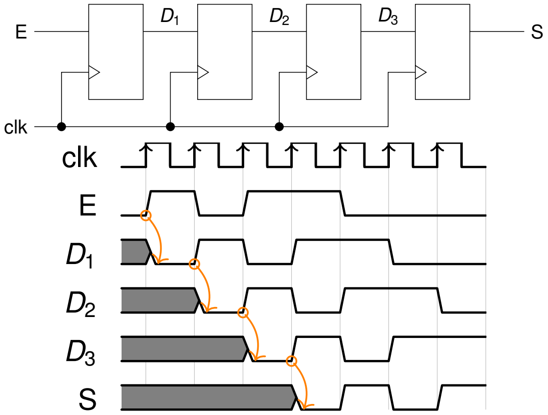
- The input must be stable around the rising edge of the clock
  - The value must be stable t_{su} before the edge (*setup*)
  - The value must be kept stable t_h after the edge (*hold*)
- There is a delay t_{co} (*clock to output*) for the data to be stable at the output

Register

- A **register** is a set of flip-flops used in parallel
- Example: a 4-bit register



Shift register

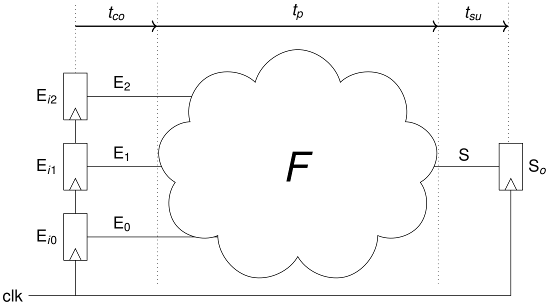


- Works because t_{co} is always greater than t_h
- A shift register can delay a signal by a number of clock cycles

Synchronous Logic Design Rules

- All combinational blocks are surrounded by flip-flops/registers
- All D flips-flops are synchronous
  - They use the same clock signal
  - The rising edge of the clock signal must arrive at the same time
  - No combinational operations on the clock signal
- The clock period must be compatible with the propagation time in combinational logic

Timings in Synchronous Logic



For synchronous sequential block to behave correctly, the following constraint must be satisfied:

$$T_{clk} > t_{co} + t_p + t_{su}$$

If this timing constraint is not respected, the sampled value may be incorrect.

Timings in Synchronous Logic

$$T_{clk} > t_{co} + t_p + t_{su}$$

This constraint must be satisfied for all combinational paths between two flip-flops.

We define t_{crit} as the propagation delay in the longest combinatorial path (critical path).

$$T_{clk} > t_{co} + t_{crit} + t_{su}$$

We can express the maximum working frequency as:

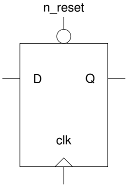
$$F_{max} = \frac{1}{\mathrm{t}_{co} + \mathrm{t}_{crit} + \mathrm{t}_{su}}$$

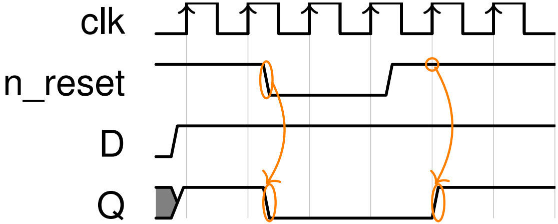
Initialization

- At power-up, the value of the output of a D flip-flop is not predictable (no initial value).
- An external signal must be used to force this value: the **reset** signal (the output is forced to 0).
- Two types of reset signal can be used: asynchronous and synchronous.

Asynchronous reset

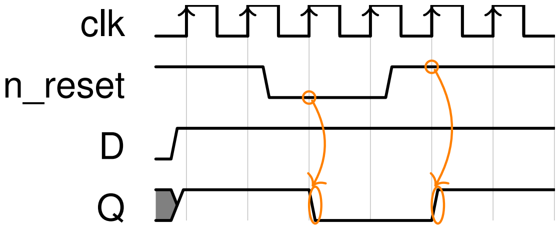
- **Asynchronous** reset: its action is independent of the clock
- It is a special input of D flip-flops
- Can be active on a high-level (positive reset, when the reset is equal to 1) or a low-level (negative reset, when the reset is equal to 0)





Synchronous reset

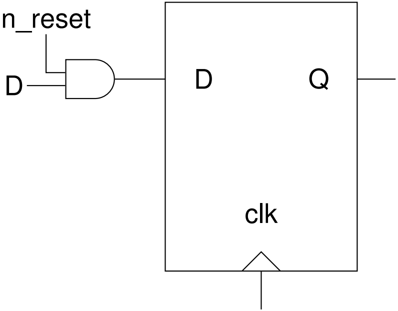
- **Synchronous** reset: it is only effective on rising edges of the clock



Synchronous reset

How to build a D flip-flop with a synchronous reset using a normal D flip-flop and logic gates?

Synchronous reset



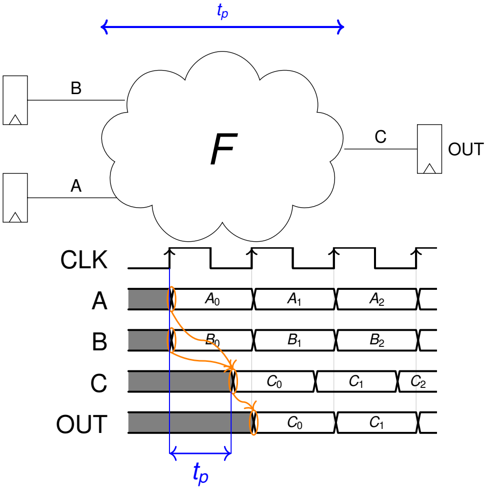
Synchronous sequential logic: summary

- D flip-flops used on inputs and outputs of combinational logic blocks
- One global clock connected directly to all the flip-flops
- The initial state of flip-flops is forced by an global external signal: the reset

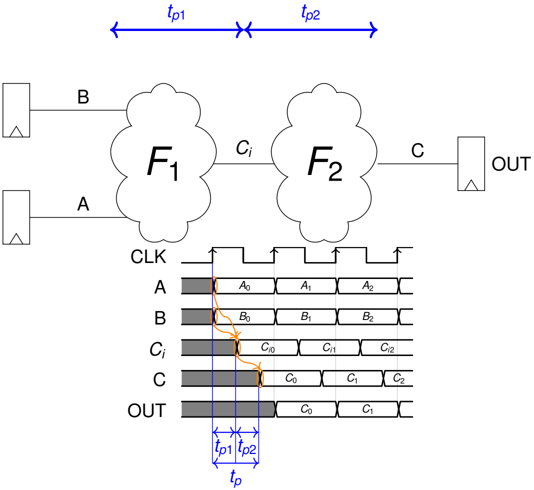
Applications

Pipeline

- F is a combinational function with a propagation delay of t_p
- Constraint: $T_{clk} > t_{co} + t_p + t_{su}$



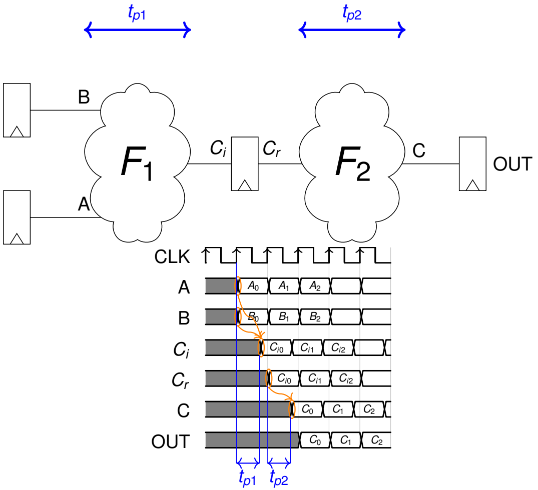
Pipeline



- We decompose F in two combinational functions F_1 and F_2 (propagation delays t_{p1} and t_{p2})
- We suppose that $t_{p1} < t_p$ and $t_{p2} < t_p$
- Constraint: $T_{clk} > t_{co} + t_{p1} + t_{p2} + t_{su}$

Pipeline

- We can introduce a D flip-flop between F_1 and F_2
  - We call this a *register barrier*.
- Constraint: $T_{clk} > t_{co} + t_{p1} + t_{su}$ and $T_{clk} > t_{co} + t_{p2} + t_{su}$
- If $t_{p1} < t_p$ and $t_{p2} < t_p$
  - we can reduce the clock period (increase the clock frequency)



Pipeline

- The pipeline is a method to increase the clock frequency of a circuit
- The size of the circuit is increased (modification of the combinational logic, addition of flip-flops)
- The initial latency is increased