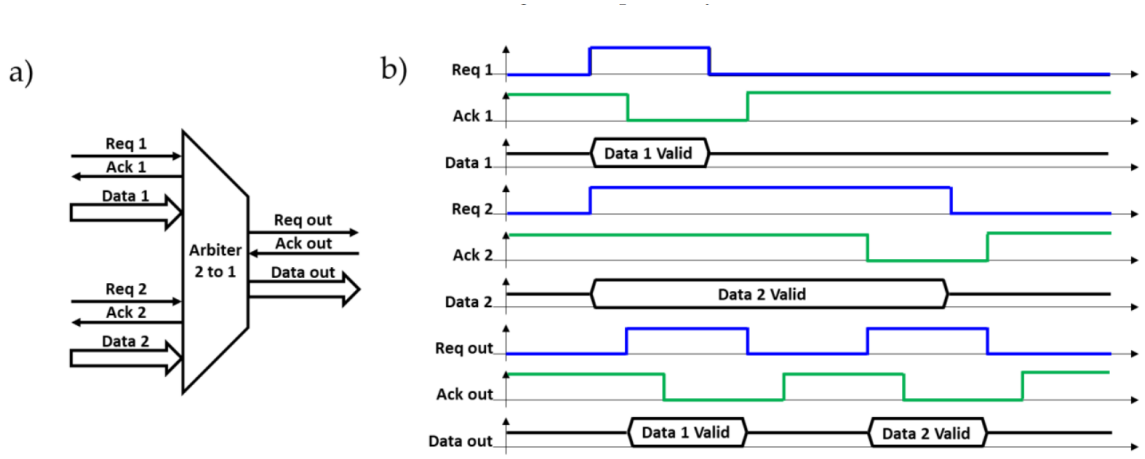
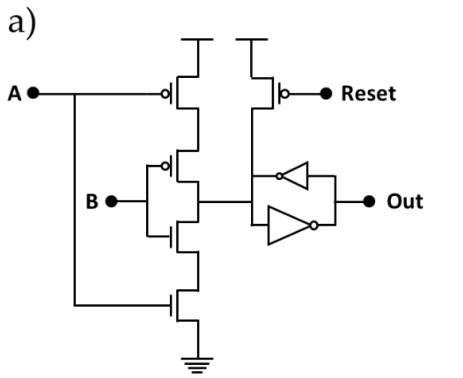
Asynchronous design

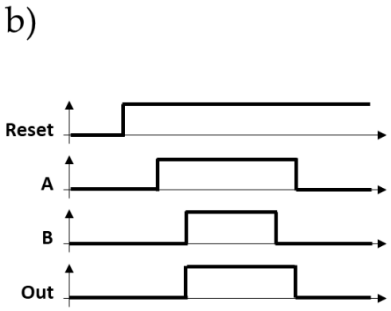
First, an asynchronous controller is build with a request signal to ask for a data to be saved, an acknowledge signal to give a feedback and the data. The request is close to the rising edge on a synchronous design, and the acknowledge is there to reset the request signal (falling edge) as follow with 2 controller input and one output:



[1]
E. Aguénounon *et al.*, « Design and Characterization of an Asynchronous Fixed Priority Tree Arbiter for SPAD Array Readout », *Sensors*, vol. 21, juin 2021, doi: 10.3390/s21123949.

To synchronize the request and acknowledge signals, a special gate is used called Müller gate or C-element and act as a memory or “rendez-vous” function. When A and B is equal the output change to the same state as follow:



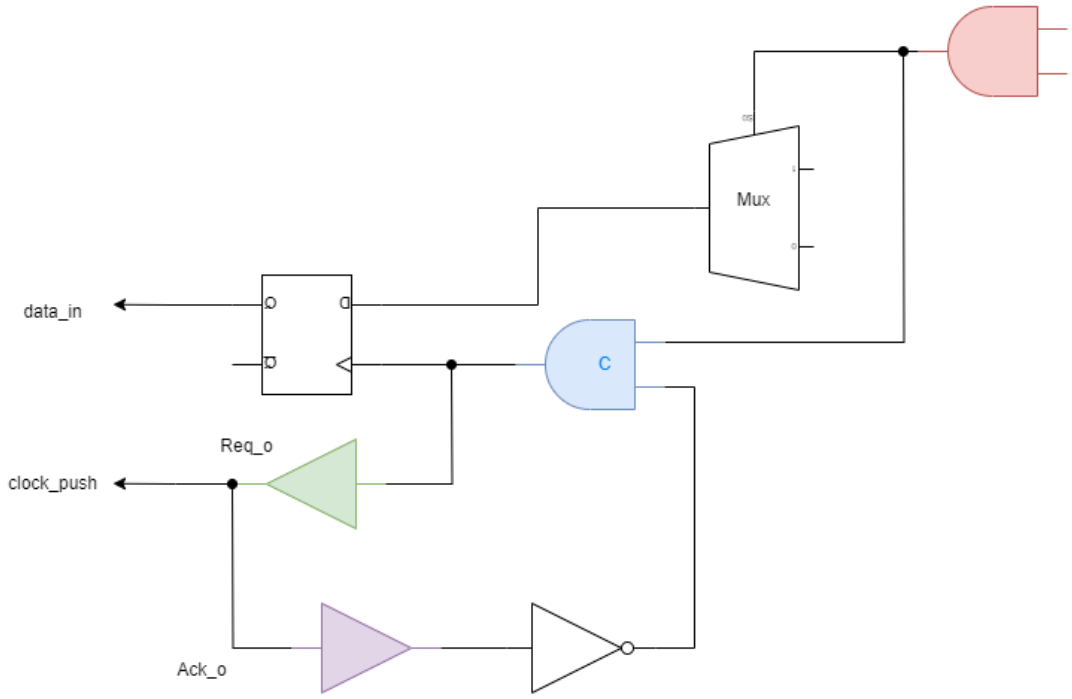


c)

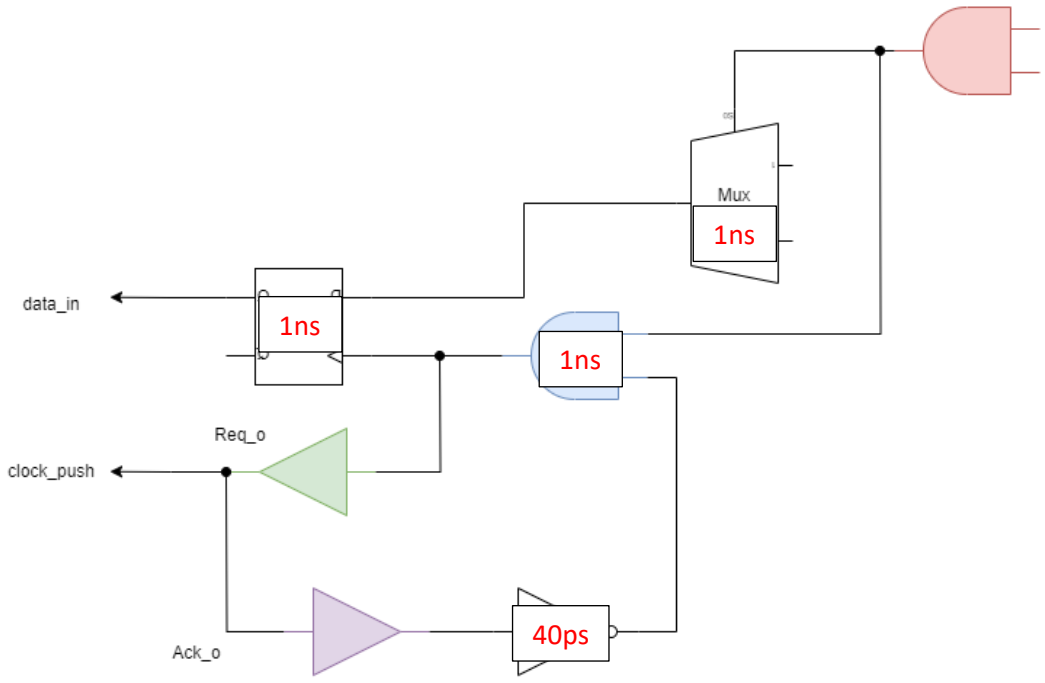
Reset	A	B	Out
1	0	0	0
1	0	1	Out^{-1}
1	1	0	Out^{-1}
1	1	1	1
0	X	X	0

[1]
E. Aguénounon *et al.*, « Design and Characterization of an Asynchronous Fixed Priority Tree Arbiter for SPAD Array Readout », *Sensors*, vol. 21, juin 2021, doi: 10.3390/s21123949.

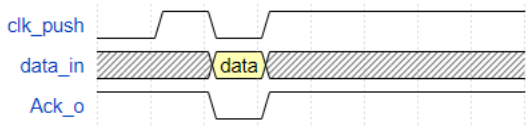
The last controller close to the FIFO is build as follow (C element in blue):



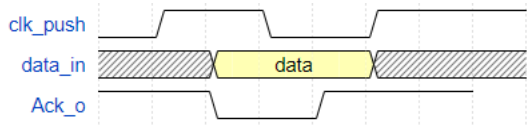
The red and gate is the starting common point of the request for both upper controllers. It controls the data from the upper or downer controller before and activate the C element. The timings are presented bellow:



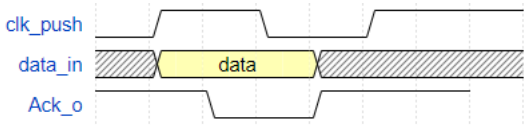
Without delay cells, the data has a delay of 1ns to the clock_push signal. The UP time of the clock is the Tup = Tblue + Tgreen and the DOWN time is Tdown = Tpurple + 40ps + Tblue + Tgreen. Adding delay cells will degrade the asynchronous readout speed (green or purple). The green delay will compensate the delay on the data but is counted 2 time as it is also in the Tdown. The best way is to add delay with the purple and a data delay of 1ns. The chronogram is the following:



No delays (data valid for 1ns), best case



With 1ns purple delay (data valid for 3ns)



With 1ns green delay (data valid for 3ns)

So the constraints on the input clock are:

- Possible 500MHz
- Min_pulse_width at 1ns ⇔ duty cycle of 60/40% max
- Data only valid for 1ns (can be changed)
- Transitions time of 200ps
- Uncertainty of 200ps
- Max corner values ⇔ timing changed for TYP and MIN corners to lower values.

MAX: max_c, max_rc, skew2_c, skew2_rc

TYP: typ

MIN: min_c, min_rc, skew1_c, skew1_rc

An other solution is to add a C element in the FIFO to synchronize the fifo ready and the request ?

Constraints on the pop_clock:

- 25 MHz
- Transitions time of 200ps