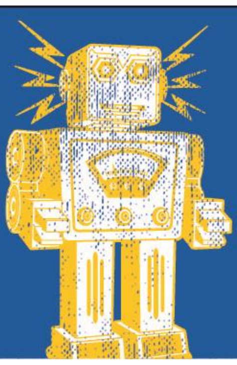
Elektronik



Digital System Design 2

Distance Learning Letter 03: Synchronous Design Methodology

Authors:	Roland Höller
Date:	01/2016
Version:	0.3

1 Introduction

This distant learning letter gives an overview of problems a digital design engineer should be aware of when designing a digital circuit. After presenting the ideal synchronous design paradigm, a set of example circuits are discussed. Where appropriate the VHDL source code of the discussed examples is also given.

1.1Keywords

Synchronous circuits, metastability, flip-flop, register, asynchronous inputs, asynchronous clock domains, synchronization of signals;

1.2Prerequisites

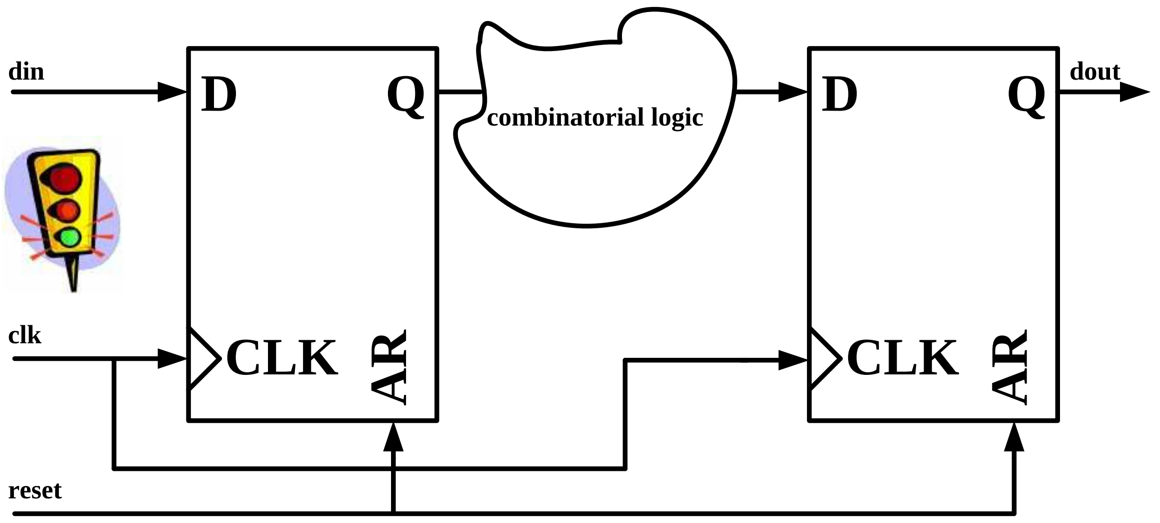
The very basics of electronic, some experience in designing integrated circuits, and an understanding of digital systems are necessary to successfully access the contents of this distant learning letter.

1.3How to proceed

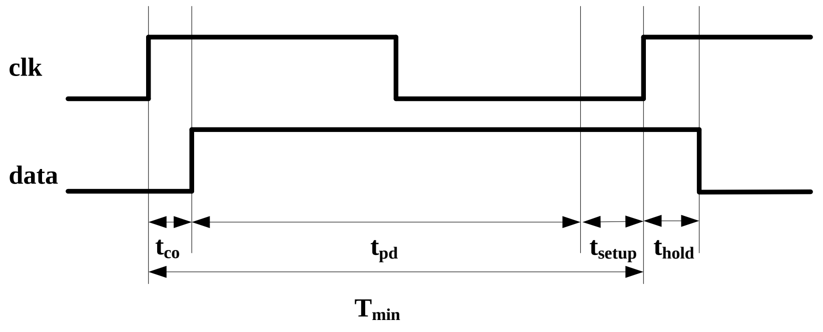
Start at the beginning, read straight through to the end. Try to answer the questions given at the end of this distant learning letter and take advantage of your new and more detailed knowledge during the lab project.

1.4Synchronous Design Paradigm

One of the most important concepts in integrated digital design is that of designing synchronous circuits. This is because asynchronous design problems often cause marginal timing problems that may appear intermittently, or may only appear when the chip vendor changes its semiconductor process. Asynchronous designs that work for years in one process may suddenly fail when the chip is manufactured using a newer or slightly changed process.
Synchronous design simply means that all data is passed through combinatorial logic and flip-flops that are synchronized to a single clock signal. The following figure depicts paradigmatically the principle of synchronous digital design.



All storage elements are D-type flip-flops. All storage elements have a power up reset signal. All flip-flops and registers are connected directly to the system clock. Calculations are done in the combinational logic in between the register outputs and inputs respectively. Thus the timing budget is perfectly known upfront by only specifying the desired clock frequency. The following figure depicts this very simple timing budget of synchronous circuits, which is the basis of every synthesis or static timing analysis tool.



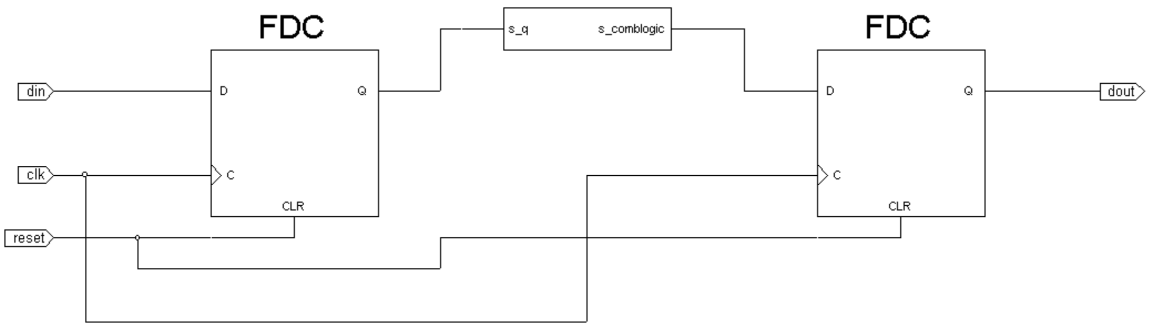
t_{co}	clock to output delay
t_{hold}	data hold time
T_{min}	minimum period
t_{pd}	propagation delay
t_{setup}	data setup time

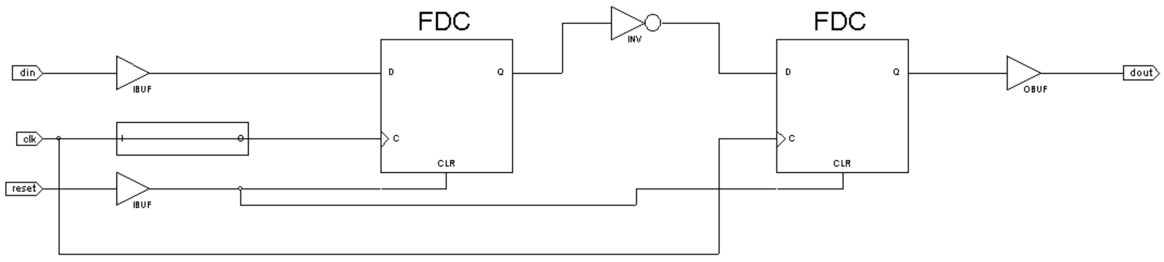
The sample VHDL source code of the above figure is given below. Note that this is only one of several possible descriptions of this simple example.

```
library IEEE;
use IEEE.std_logic_1164.all;

entity exa_ff_ff_sync is

  port (clk    : in  std_logic;
        reset  : in  std_logic;
        din    : in  std_logic;
        dout   : out std_logic);

end exa_ff_ff_sync;
```

```
architecture rtl of exa_ff_ff_sync is

  signal s_comblogic : std_logic;
  signal s_q         : std_logic;

begin  -- rtl

  p_ff1 : process (clk, reset)
  begin
    if reset = '1' then
      s_q <= '0';
    elsif (clk'event and clk = '1') then
      s_q <= din;
    end if;
  end process p_ff1;

  s_comblogic <= not(s_q);  -- Very simple.

  p_ff2 : process (clk, reset)
  begin
    if reset = '1' then
      dout <= '0';
    elsif (clk'event and clk = '1') then
      dout <= s_comblogic;
    end if;
  end process p_ff2;

end rtl;
```

Delay is always controlled by flip-flops, not combinatorial logic in properly designed synchronous digital circuits. No signal that is generated by combinatorial logic is fed back to the same group of combinatorial logic without first going through a synchronizing flip-flop.





The above figures show the synthesis results of the VHDL code printed on the previous page. On top is the RTL view, the translation of the VHDL code into the synthesis tool’s internal representation. Below is the technology view, the result of the optimization and technology mapping steps performed by the synthesis tool. **Although the circuitry is described using a high level hardware description language, this does not prevent the designer from having basic implementation and technology knowledge**.

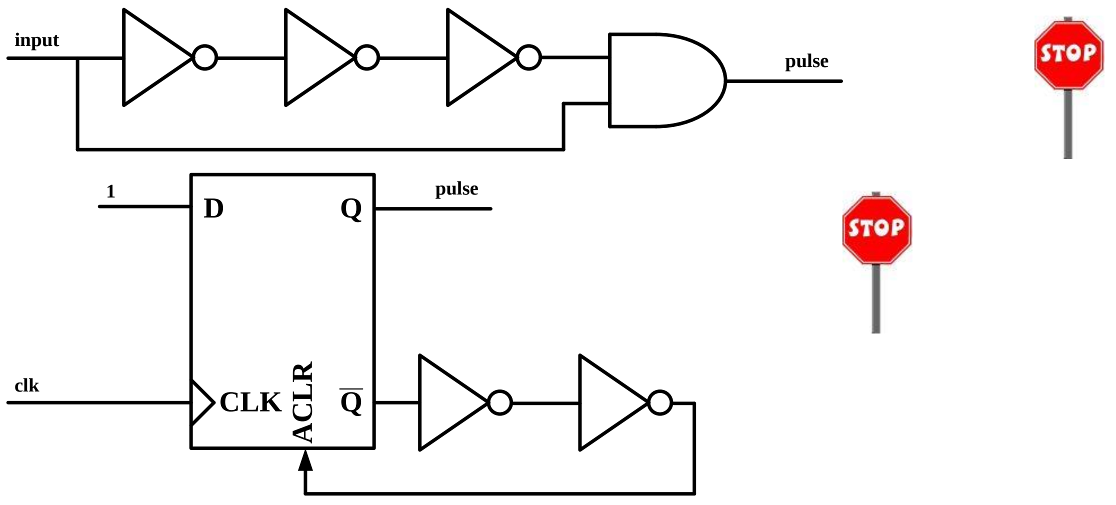
Strictly following synchronous design methodology, clock signals are not gated, thus clocks must go directly to the clock inputs of the flip-flops without going through any combinatorial logic. The following list of general rules simplifies the design process and hence increases the chance to meet the requirements of nowadays projects:

1) prefer synchronous circuits
2) synchronize asynchronous events safely
3) use only one clock signal (one edge of a clock signal) for a synchronous circuit
4) use FIFO buffers to transfer data from one clock domain to another
5) use a master reset signal for circuit initialization during power-up
6) use only D-type flip-flops
7) use pipelined circuits in case of timing problems
8) avoid tri-states on chip, use multiplexers instead

In the remainder of this section these rules will be discussed in more detail using circuit examples and, where appropriate, VHDL source code examples to exemplify the related problems and possible drawbacks.

1.5 Constructing Delays

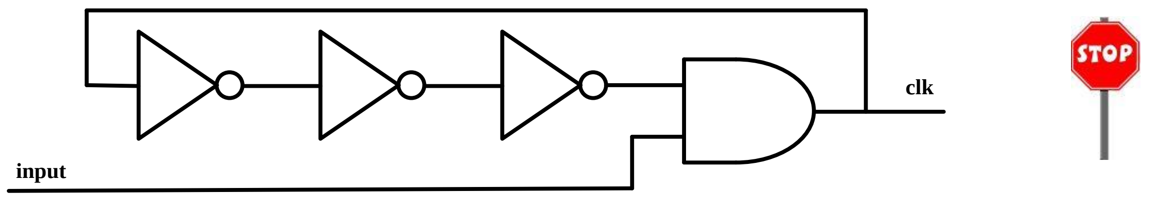
The following two figures depict possible ideas of how to generate pulses in digital circuits. They severely violate, however, the synchronous design paradigm.

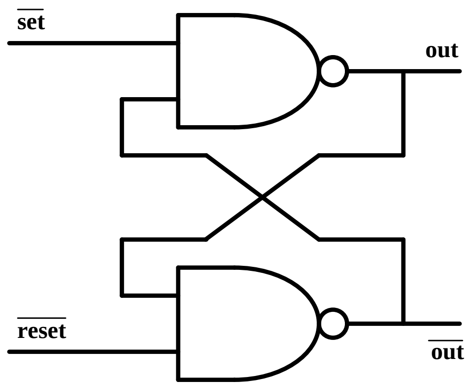


Both examples make use of inverter chains to increase the propagation delay of a certain signal, so that another signal shows a short pulse in the waveform. The actual delay, however, is not under proper control in a standard HDL design flow and will depend on the layout, process, temperature, and voltage. Furthermore it is questionable if the inverters will remain in the netlist after optimization. It is thus preferable to either use special library elements to generate pulses or design a synchronous pulse generator maybe using a multiplied clock.

1.6 Combinational Loops

The following two circuit examples make use of combinational loops for designing an oscillating signal and storage element.

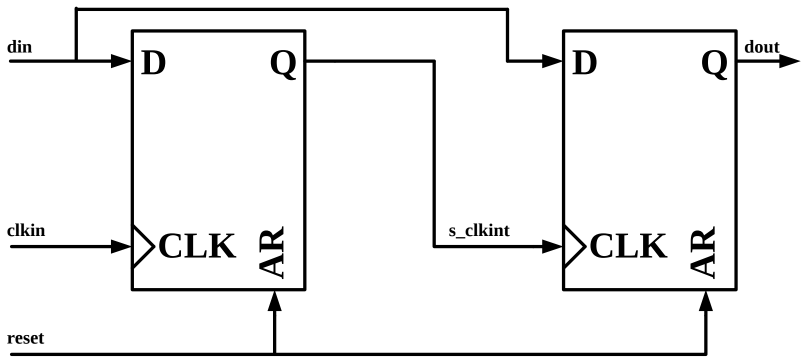




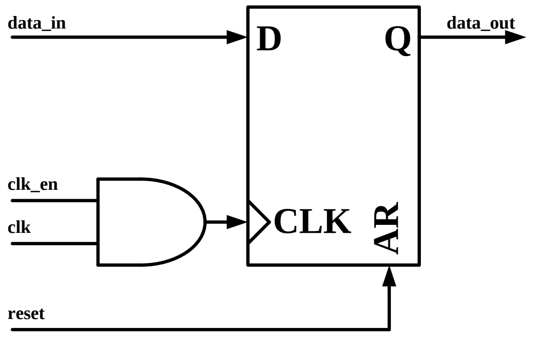
As mentioned in the previous section the delay of those circuits will heavily depend on layout, process, temperature, and voltage and thus must be avoided in all synchronous designs. Furthermore the timing of those circuits will not be checked by the static timing analysis tool. So always use existing library elements or construct the circuit synchronously using a clock signal.

1.7 Clocking

The following circuit examples show how clock signals may be generated thereby sometimes severely violating the synchronous design paradigm.



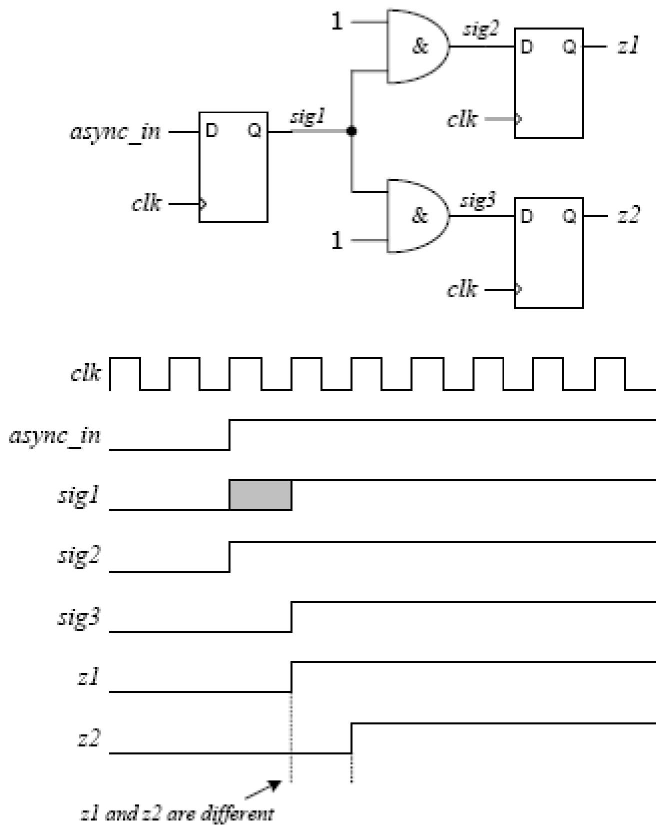
The routing from the Q output of the first flip-flop in the above figure is not perfectly under control, since the routing path may change with every run of the place and route tool. It furthermore introduces additional delay and may thus adversely affect I/O timing.



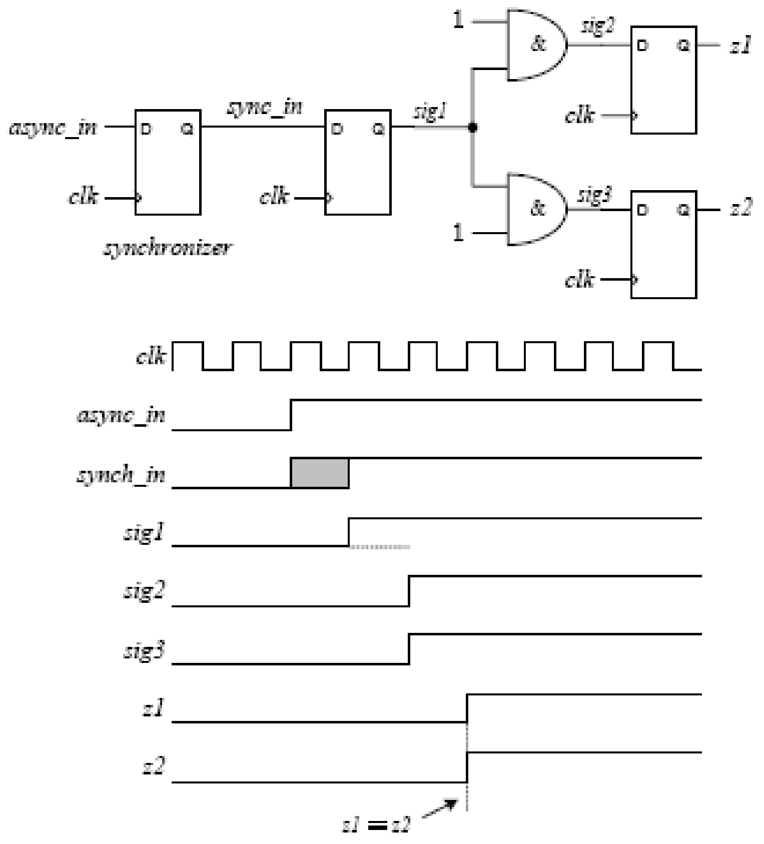
Although gating clocks is a common method in low power design, any combinational logic in the path of the clock signal must be avoided when strictly following the synchronous design paradigm. The logic introduces arbitrary delay, skew, may cause glitches on the clock signal, and thus typically produces warnings if not errors in the implementation tool chain.

1.8 Metastability

One of the great buzzwords, and often misunderstood concepts, of synchronous design is metastability. Metastability refers to a condition which arises when an asynchronous signal is clocked into a synchronous flip-flop. While digital designers would prefer a completely synchronous world, the unfortunate fact is that signals coming into a digital logic will depend on a user pushing a button or an interrupt from a processor, or will be generated by a clock which is different from the one used by the digital logic. In these cases, the asynchronous signal must be synchronized to the clock so that it can be used by the digital circuitry. The designer must be careful how to do this in order to avoid metastability problems as shown in the next figure.
If the async_in signal goes high around the same time as the clock, we have an unavoidable race condition. The output of the flip-flop can actually go to an undefined voltage level that is somewhere between a logic 0 and logic 1. This is because an internal transistor did not have enough time to fully charge to the correct level. This meta-level may remain until the transistor voltage leaks off or decays, or until the next clock cycle. During the clock cycle, the gates that are connected to the output of the flip-flop may interpret this level differently. In the figure, the upper gate sees the level as logic 1 whereas the lower gate sees it as logic 0. In normal operation, the outputs Z1 and z2 should always be the same value. In this case, they are not and this could send the logic into an unexpected state from which it may never return. This metastability can permanently lock up your digital design.



The solution to this metastability problem is shown in the next figure. By placing a synchronizer flip-flop in front of the logic, the synchronized input will be sampled by only one device, the second flip-flop, and be interpreted only as logic 0 or 1. The upper and lower gates will both sample the same



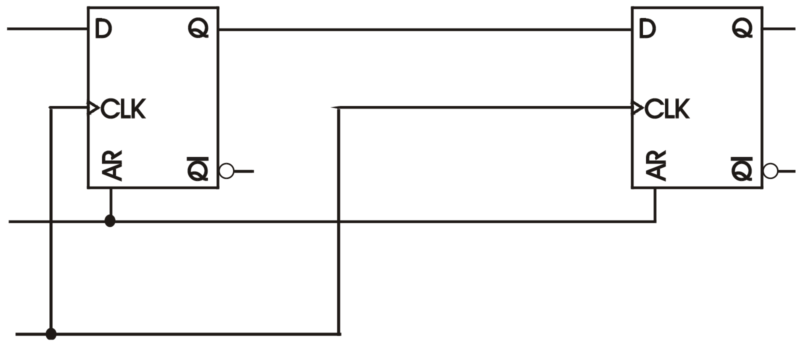
logic level, and the metastability problem is avoided. Or is it? The word solution is in quotation marks for a very good reason. There is a very small but non-zero probability that the output of the synchronizer flip-flop will not decay to a valid logic level within one clock that the output of the synchronizer flip-flop will not decay to a valid logic level within one clock period. In this case, the next flip-flop will sample an indeterminate value, and there is again a possibility that the output of that flip-flop will be indeterminate. At higher frequencies, this possibility is greater. Unfortunately, there is no certain solution to this problem. Some ASIC vendors provide special synchronizer flip-flops whose output transistors decay very quickly. Also, inserting more synchronizer flip-flops reduces the probability of metastability but it will never reduce it to zero.
The correct action involves discussing metastability problems with the ASIC vendor, and including enough synchronizing flip-flops to reduce the probability so that it is unlikely to occur within the lifetime of the product. Notice that each synchronizer flip-flop may delay the logic level change on the input by one clock cycle before it is recognized by the internal circuitry of the ASIC. Given that the external signal is asynchronous, by definition this is not a problem since the exact time that it is asserted will not be deterministic. If this delay is a problem, then most likely the entire system will need to be synchronized to a single clock.

1.9 Asynchronous Interfaces

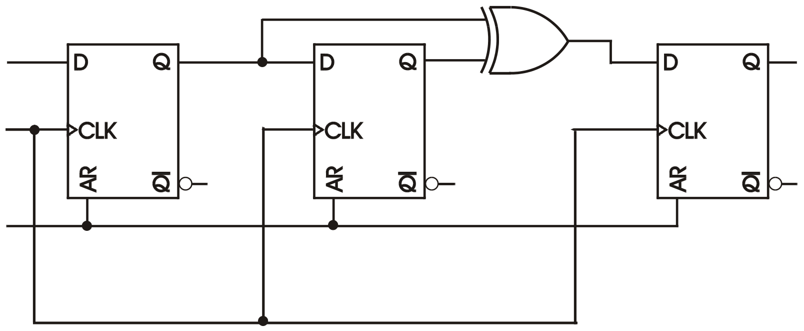
Asynchronous interfaces exist in practically every integrated circuit design and thus have to be properly addressed during conception and specification as well as implementation of the design.

1.9.1 Single Wire

Asynchronous signals coming in as single wires or single control lines can be synchronized using a simple synchronizing circuit, as depicted in the following figure.

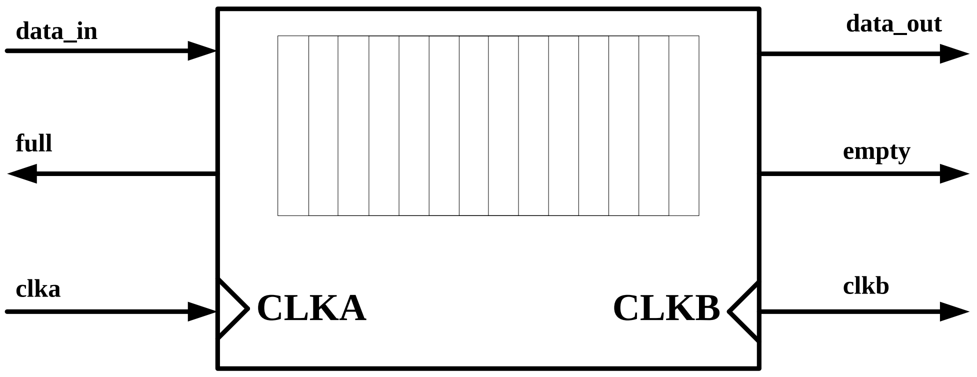


If the input of this circuit example forces the first fillip-flop into a metastable state, the flip-flop has one entire clock cycle to fall back to either logic 1 or logic 0. The subsequent circuitry is thus not affected. For high clock frequencies even three synchronizing flip-flops in series should be used. Edges on asynchronous inputs can also be detected synchronously as depicted in the follwing figure.



1.9.2 Busses

Busses in integrated circuit design always transport some form of data and thus the method applied for the synchronization of a single wire signal cannot be applied. The use of dual ported on-chip RAM blocks to build dual clock FIFO buffers is hence the method of choice to cope with asynchronous data interfaces. The price for the proper transfer of data from one clock domain to the other comes at the price of additional delay on the data path. The following figure shows the principles.



2 Abbreviations

FIFO	First In First Out
MTBF	Mean Time Between Failure
RAM	Random Access Memory
VHDL	VHSIC Hardware Description Language
VHSIC	Very High Speed Integrated Circuit

3 Literature

In the library of the Technikum Wien there are several books on the topic of programmable logic devices in English and German language.
A very complete book on the topic of FPGA design including chapters on the topic of this distant learning letter is:
The Design Warrior's Guide to FPGAs; Devices, Tools, and Fows. Elsevier, USA, 2004. ISBN – 0-7506-7604-3.

4 Directions and Questions

4.1 Directions

Take advantage of your new and more detailed knowledge about the problem of metastability and synchronous design during the lab project.

4.2 Questions

1) What is a race condition in the context of the design of synchronous digital circuits?
2) What is a glitch in the context of the design of synchronous digital circuits?
3) What means the term bus contention in the context of the design of synchronous digital circuits?
4) What means the term metastability in the context of the design of synchronous digital circuits?
5) When and how must a design engineer deal with the problem of metastability?
6) Is there a general solution for the problem of metastability?
7) Illustrate an approach to reduce the problem of metastability for a single asynchronous input?
8) How can the mean time between failure (MTBF) be calculated for a series of two subsequent flip-flops?
9) What do the terms fan-in and fan-out mean in the context of the design of synchronous digital circuits?
10) What does the term propagation delay mean in the context of the design of synchronous digital circuits?
11) What does the term clock skew mean in the context of the design of synchronous digital circuits?
12) What does the term clock jitter mean in the context of the design of synchronous digital circuits?

5 Document Version History

Version 0.1, 06.03.2006	Created document
Version 0.2, 14.03.2006	Removed duplicate text in Section 3.
Version 0.3, 13.01.2016	Update.

If you happen to find any errors, typos, or inconsistencies, please report them via email to hoeller@technikum-wien.at. Thanks!