FPGA tutorial

Lecture 2

04.11.2020

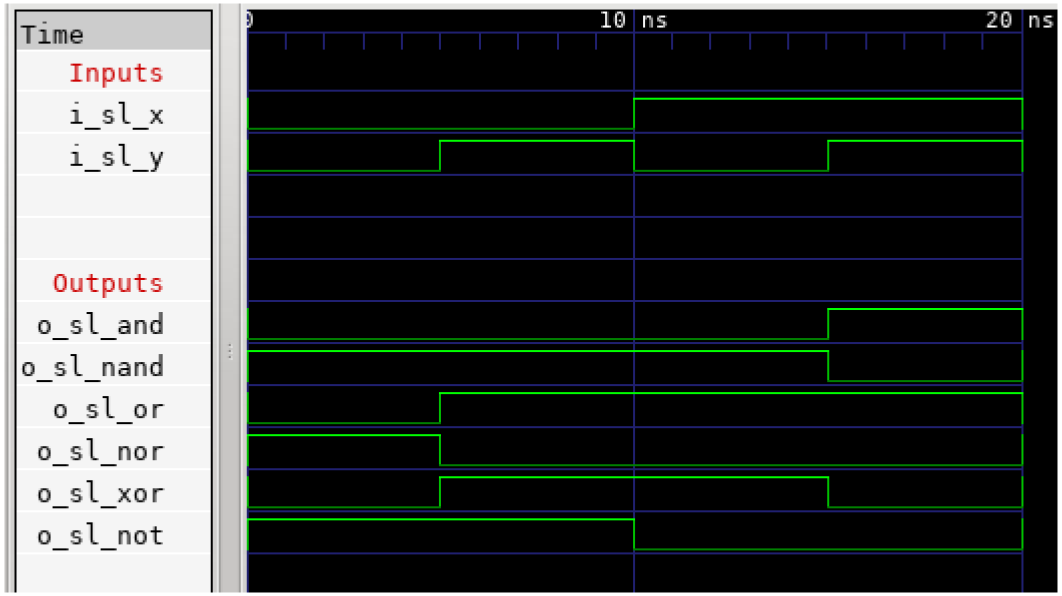
Jochen Steinmann

III. Physikalisches Institut B

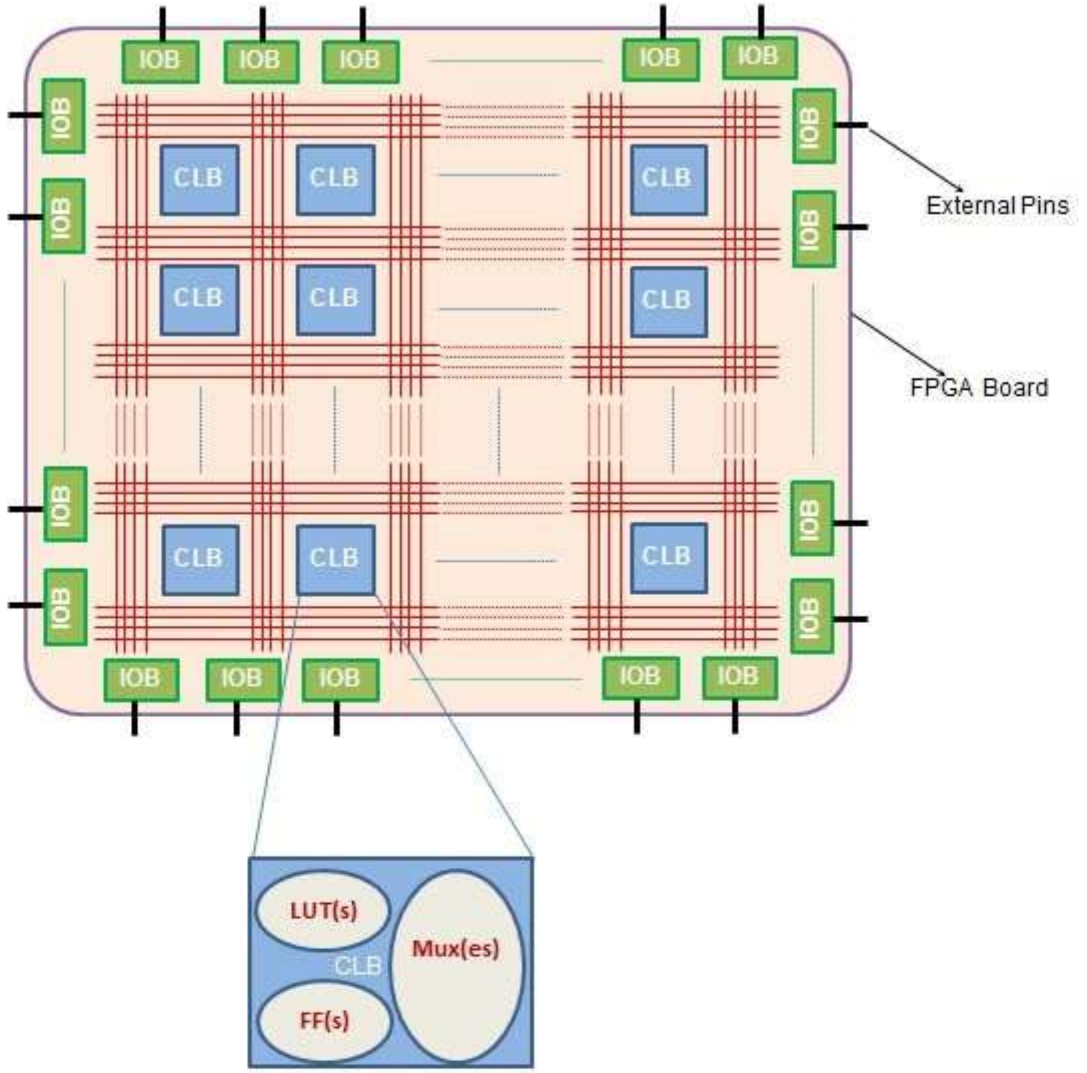
RWTH AACHEN UNIVERSITY

Solution

```
----------------------------------------
-- lecture 01 simple gate
-- just forwarding the signal
----------------------------------------

library ieee;
use ieee.std_logic_1164.all;

----------------------------------------

entity lecture01 is
    port(
        i_sl_x     :  in  std_logic;
        i_sl_y     :  in  std_logic;
        o_sl_and   :  out std_logic;
        o_sl_or    :  out std_logic;
        o_sl_nand  :  out std_logic;
        o_sl_nor   :  out std_logic;
        o_sl_xor   :  out std_logic;
        o_sl_not   :  out std_logic
    );
end lecture01;

----------------------------------------

architecture behaviour of lecture01 is
begin

    process(i_sl_x, i_sl_y)
    begin
        o_sl_and  <= i_sl_x  and i_sl_y;
        o_sl_or   <= i_sl_x   or i_sl_y;
        o_sl_nand <= i_sl_x nand i_sl_y;
        o_sl_nor  <= i_sl_x  nor i_sl_y;
        o_sl_xor  <= i_sl_x  xor i_sl_y;
        o_sl_not  <= not i_sl_x;
    end process;

end behaviour;

----------------------------------------
```



How is this logic realised in the FPGA ?



Components of an FPGA

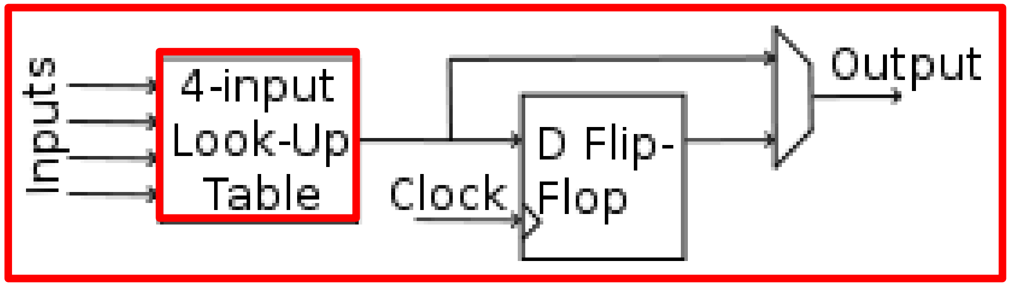
IOB In- / Output Block

- connections to the rest of the world
- configuration of output levels, direction, etc.
- special functions

LUT LookUp-Table

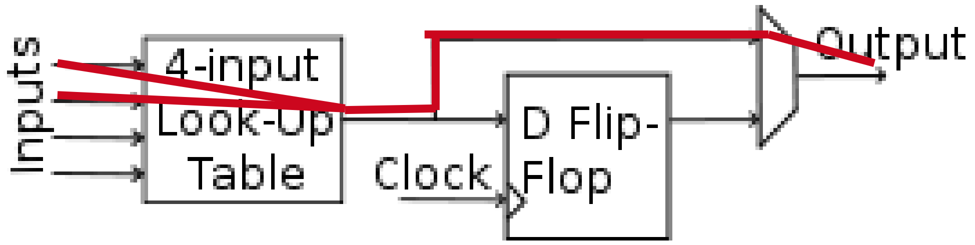
- central device of logic block
- can be programmed with any boolean operation
- logic cell has one additional flip-flop

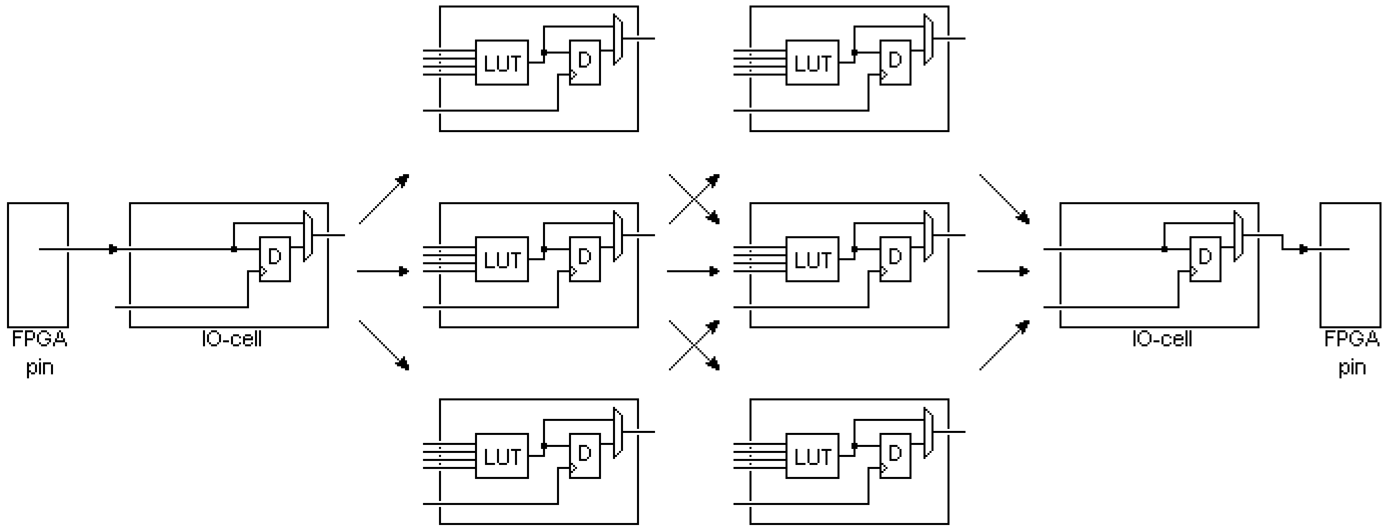
Artix7: 6-input LUT



one logic cell

Implementation of our logic




LUT
D
FPGA pin
IO-cell
FPGA pin
IO-cell

Sterling & Noble
QUARTZ
ALARM CLOCK

Clock – Why?

- Until now only “Glue Logic”
  - simple logic only
  - output action is related to input
  - This “mode” is used for trigger generation:
    → TRIGGER = (A and B) or (A and C)

  - But if we want to introduce timing, e.g. dead-time after a trigger
    - we need to react on something but not the input
      → **CLOCK**

Use of Clock

- We want to let the FPGA doing thing for us
  - Memory storage RAM need a clock to refresh
  - Digital communication to other electronics or PC
- Delays
  - compensate cable length
  - artificial dead times → Trigger Veto
- Measurements of the time between two events:
  - Drift velocity measurement in a TPC
  - Time of Flight measurements to distinguish particles

Clock Requirements

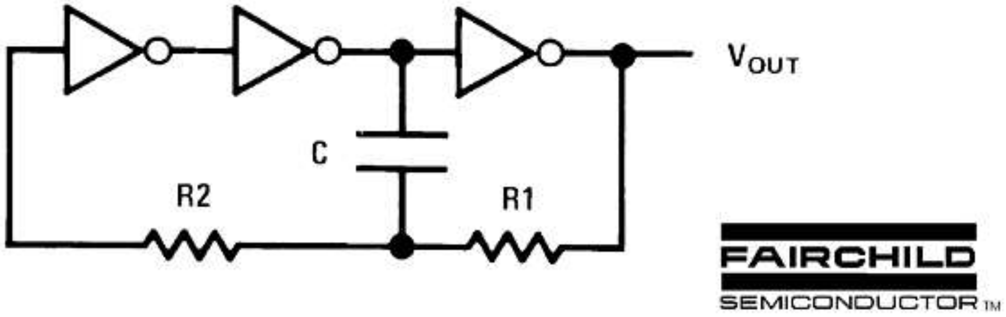
- Requirements:
  - stable frequency
  - stable phase to other clocks
  - stable signal
  - reproducible
    - low temperature dependency
    - low dependency on environment

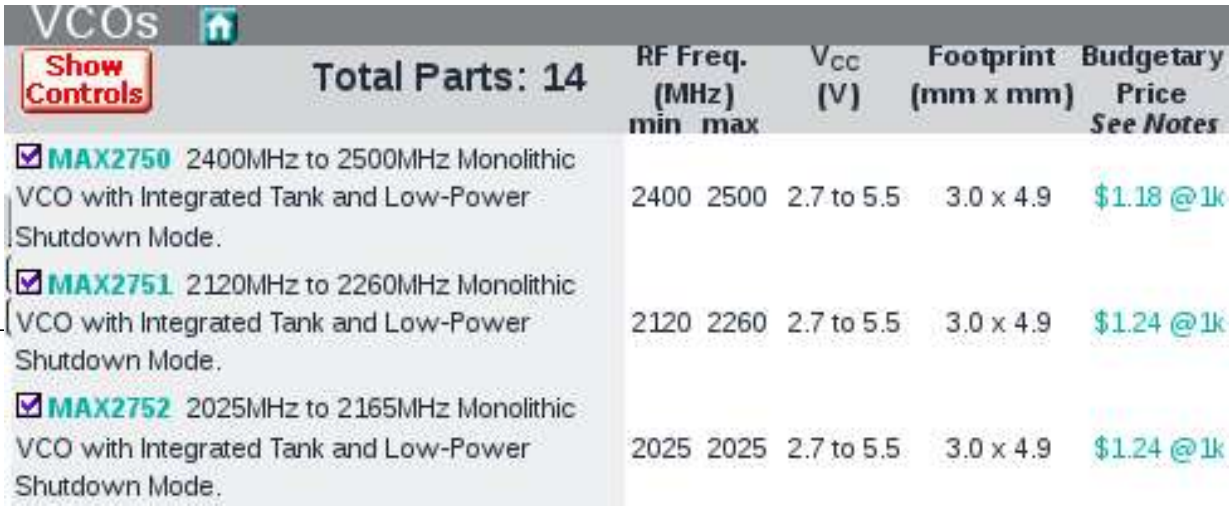
Most FPGAs do not have an internal clock generator!

Need external reference Clock generator

Clock generation

- Driven LC / RC resonator
  - huge tolerances
  - not very stable
  - no high frequencies
- Tuned Voltage Controlled Os
  - can be easily tuned by a voltage

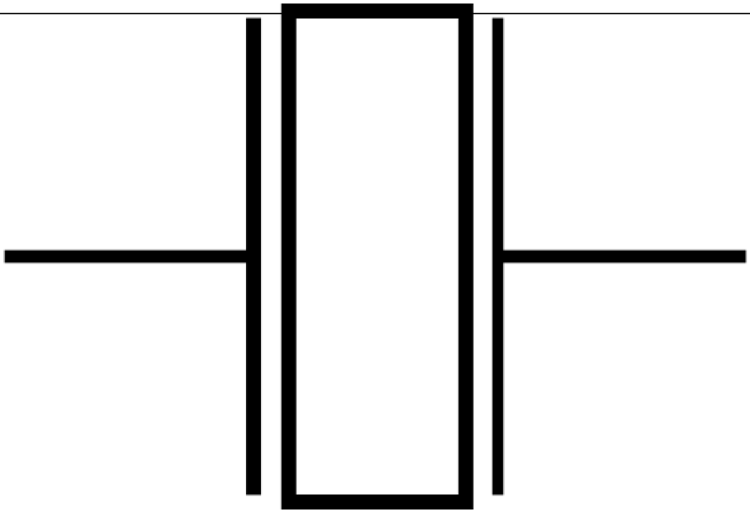




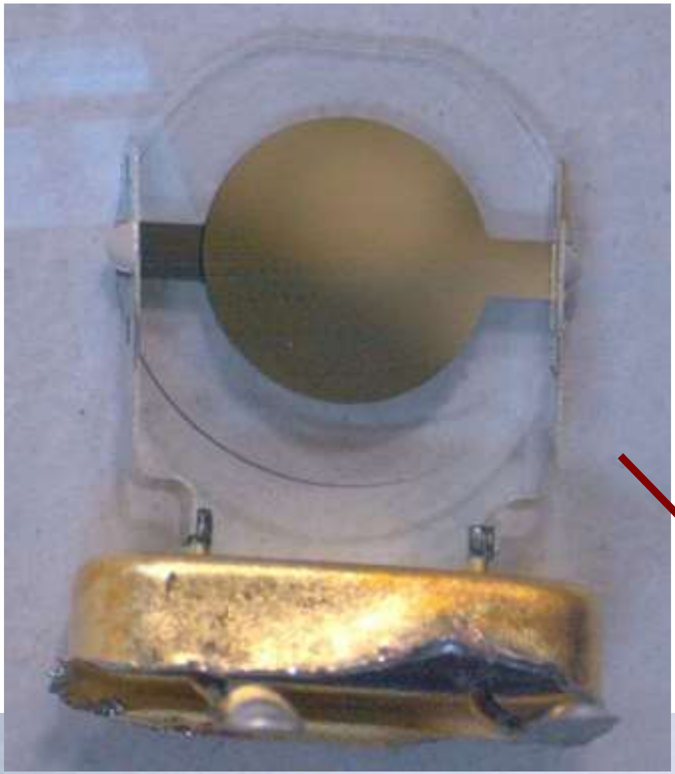
Total Parts: 14

	RF Freq. (MHz) min	max	V_{CC} (V)	Footprint (mm x mm)	Budgetary Price *See Notes*
MAX2750 2400MHz to 2500MHz Monolithic VCO with Integrated Tank and Low-Power Shutdown Mode.	2400	2500	2.7 to 5.5	3.0 x 4.9	$1.18 @1k
MAX2751 2120MHz to 2260MHz Monolithic VCO with Integrated Tank and Low-Power Shutdown Mode.	2120	2260	2.7 to 5.5	3.0 x 4.9	$1.24 @1k
MAX2752 2025MHz to 2165MHz Monolithic VCO with Integrated Tank and Low-Power Shutdown Mode.	2025	2025	2.7 to 5.5	3.0 x 4.9	$1.24 @1k

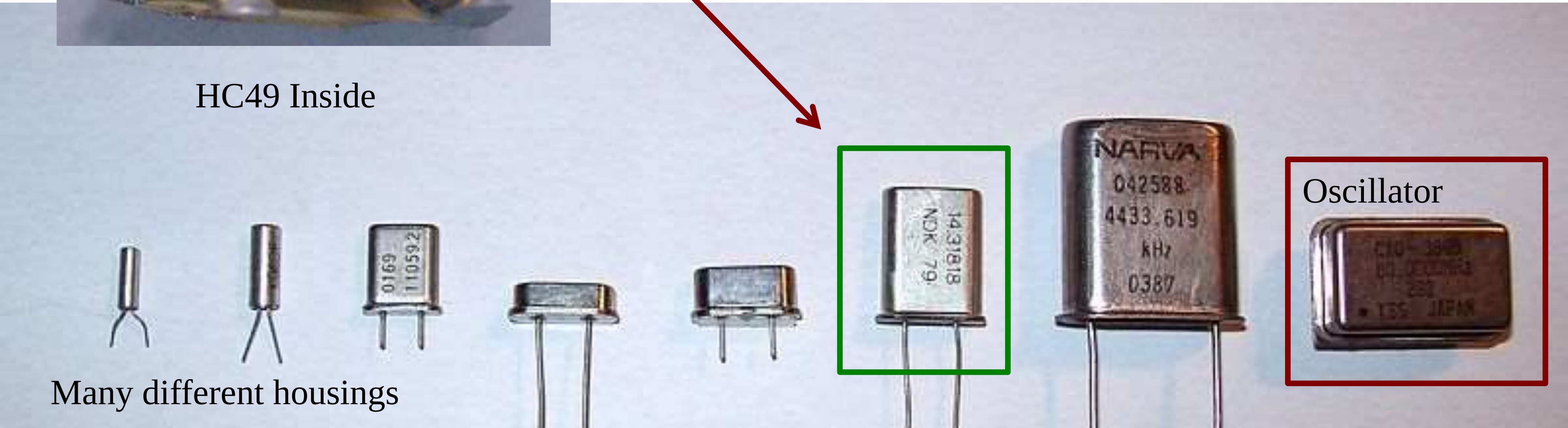
Crystals



Frequencies up to 200 MHz

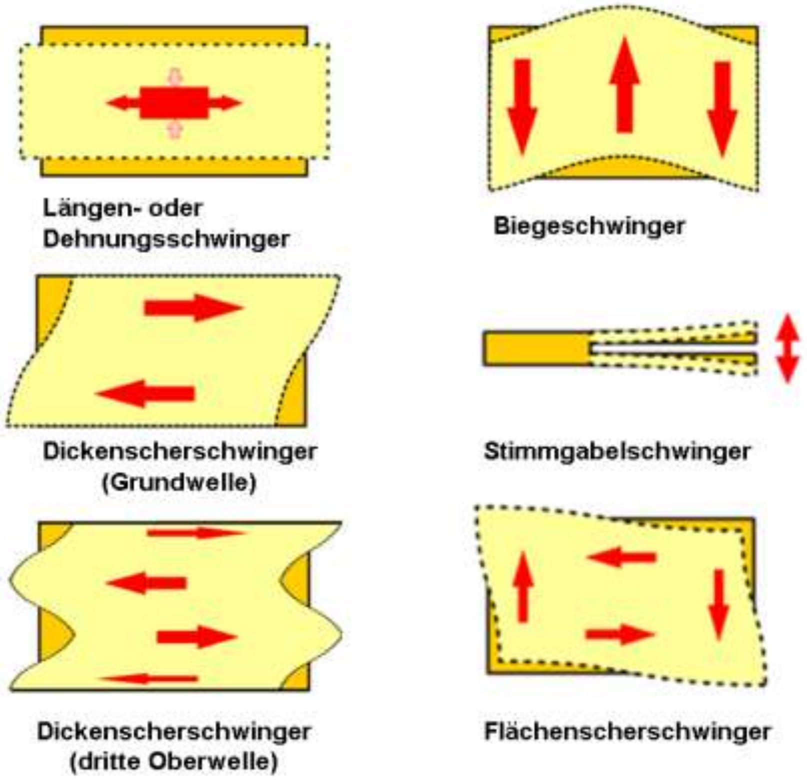


HC49 Inside



Many different housings

Oscillation modes


Längen- oder Dehnungsschwinger
Biegeschwinger
Dickenscherschwinger (Grundwelle)
Stimmgabelschwinger
Dickenscherschwinger (dritte Oberwelle)
Flächenscherschwinger

different Frequencies

Different Properties:
- Temperature
- Stability

Base- and overtone crystals

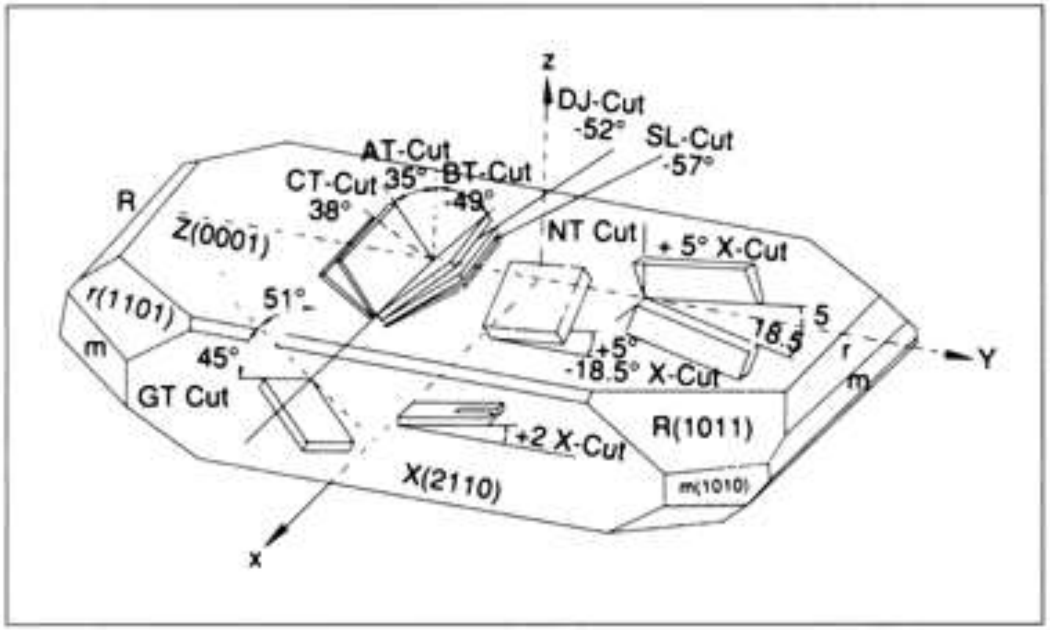


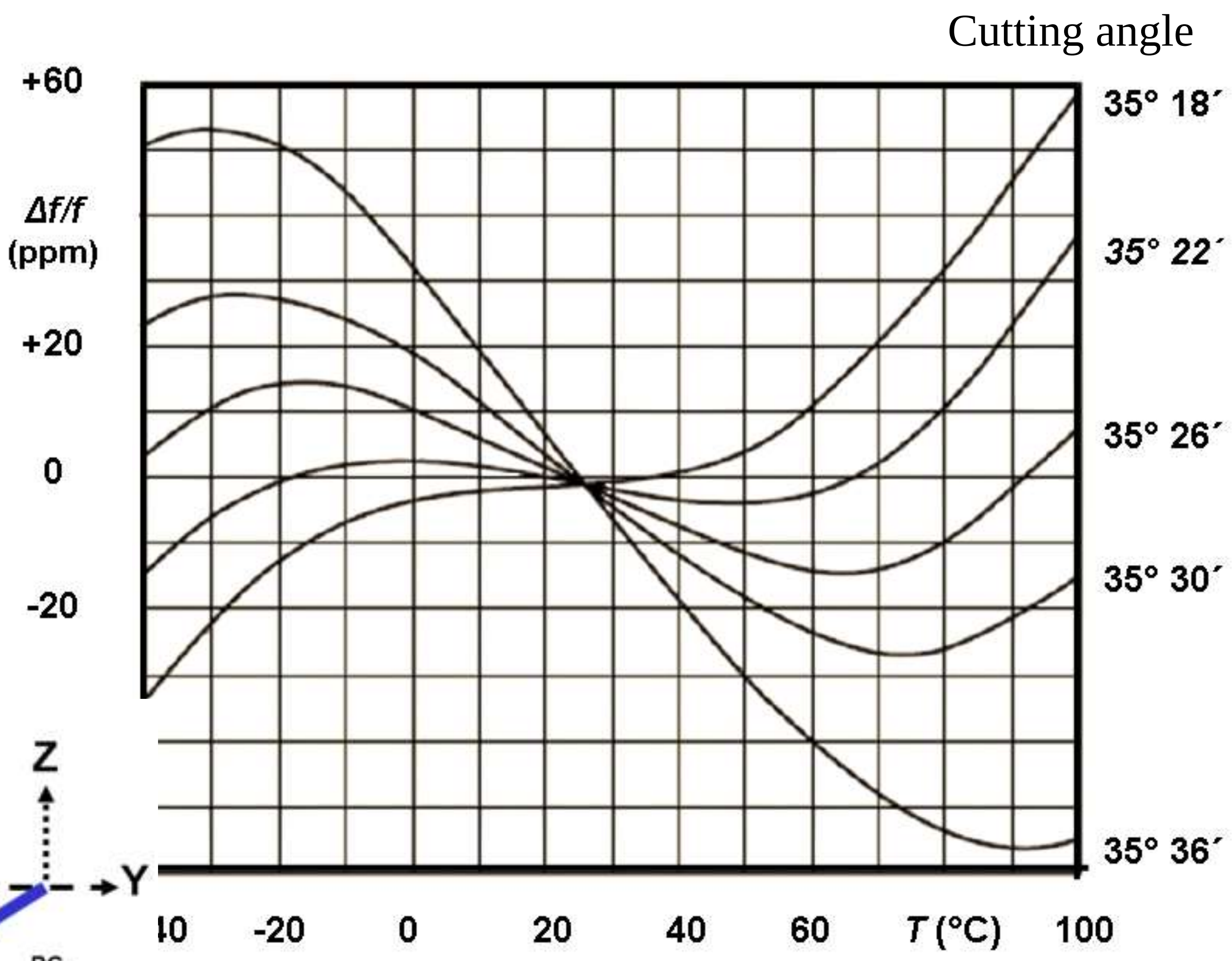
Bild 1: Schnittlage der Quarzplättchen zu den Kristallachsen

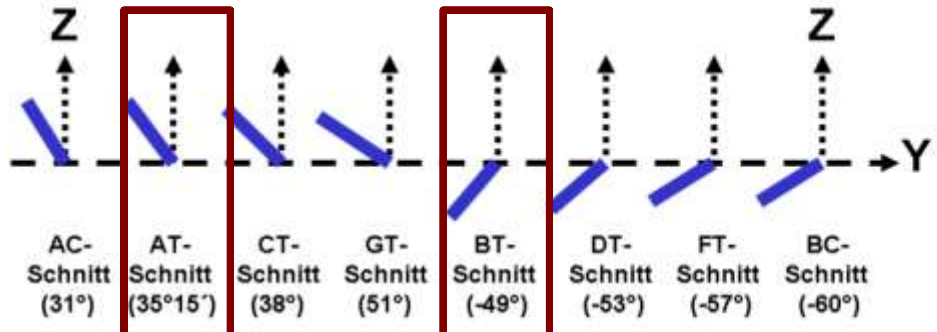
Cuts

Crystals are a good temperature sensor

Entsprechender Schnitt

Bis 573 °C → Phasenübergang



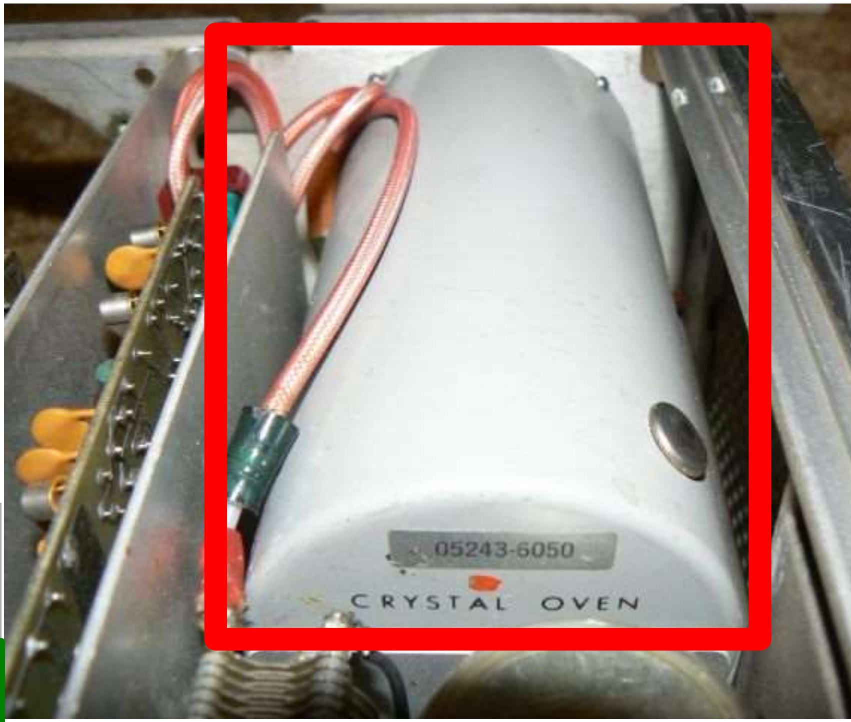


Temperaturesensor ?

- Precise Oscillators
  - keep Temperature constant
  - is used for high precision clock sources

$$\boxed{T = \text{const.} \rightarrow f = \text{const.}}$$

Oszillatortyp	Genauigkeit	Alterung / 10 Jahre
Quarzoszillator	10^{-5} bis 10^{-4}	10 bis 20 ppm
Quarzofen (OCXO) 5 bis 10 MHz 15 bis 100 MHz	 2×10^{-8} 5×10^{-7}	 2×10^{-8} bis 2×10^{-7} 2×10^{-6} bis 11×10^{-9}
Atomuhr (Cs)	10^{-10} bis 10^{-11}	10^{-12} bis 10^{-11}
Global Positioning System (GPS)	4×10^{-8} bis 10^{-11}	10^{-13}

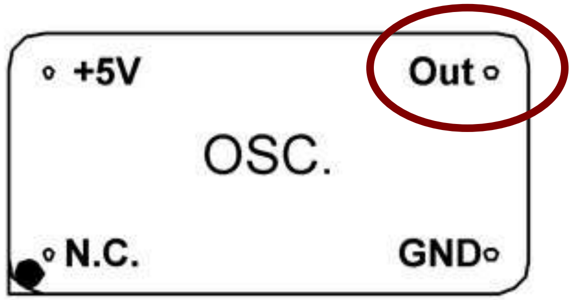


III. Physikalisches Institut B

RWTH AACHEN UNIVERSITY

Different Crystal-Oszillator

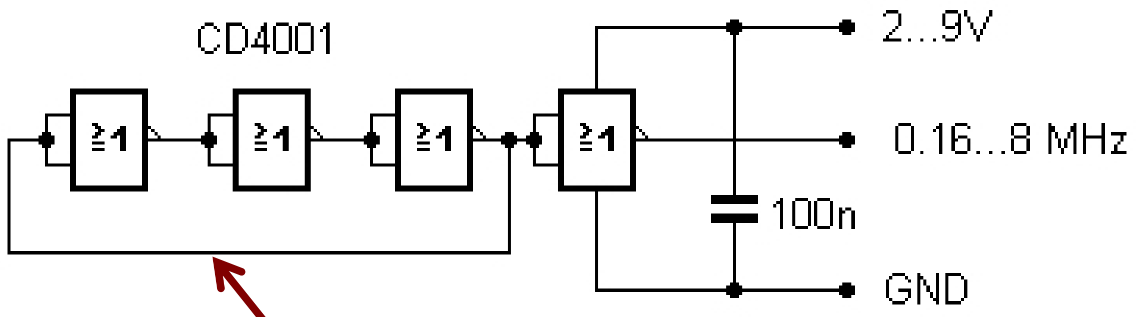
- 5V TTL-signals at the output
  - also 3V3 and other voltages available



- Don't need further parts for clock generation

other clock sources

- resonator made from single gates:



Feedback

$$T = 2n \cdot t_D \leftarrow$$ 2ns

$$f = \frac{1}{T} = \frac{1}{2n \cdot t_D} \longrightarrow$$ 83 MHz

3

Not very stable

Difficult to compute

More speed ...

… higher frequency!

Stable high speed clock...

- Crystals and Oscillators
  are only available up to ~ 200MHz!
- What if we need ~ 1GHz ?

All you need is a Voltage Controlled Oscillator and a slower reference clock!

Frequency Multiplication!

Howto?

- Constraints for Multiplication
  - After n Periods the phase must be the same!

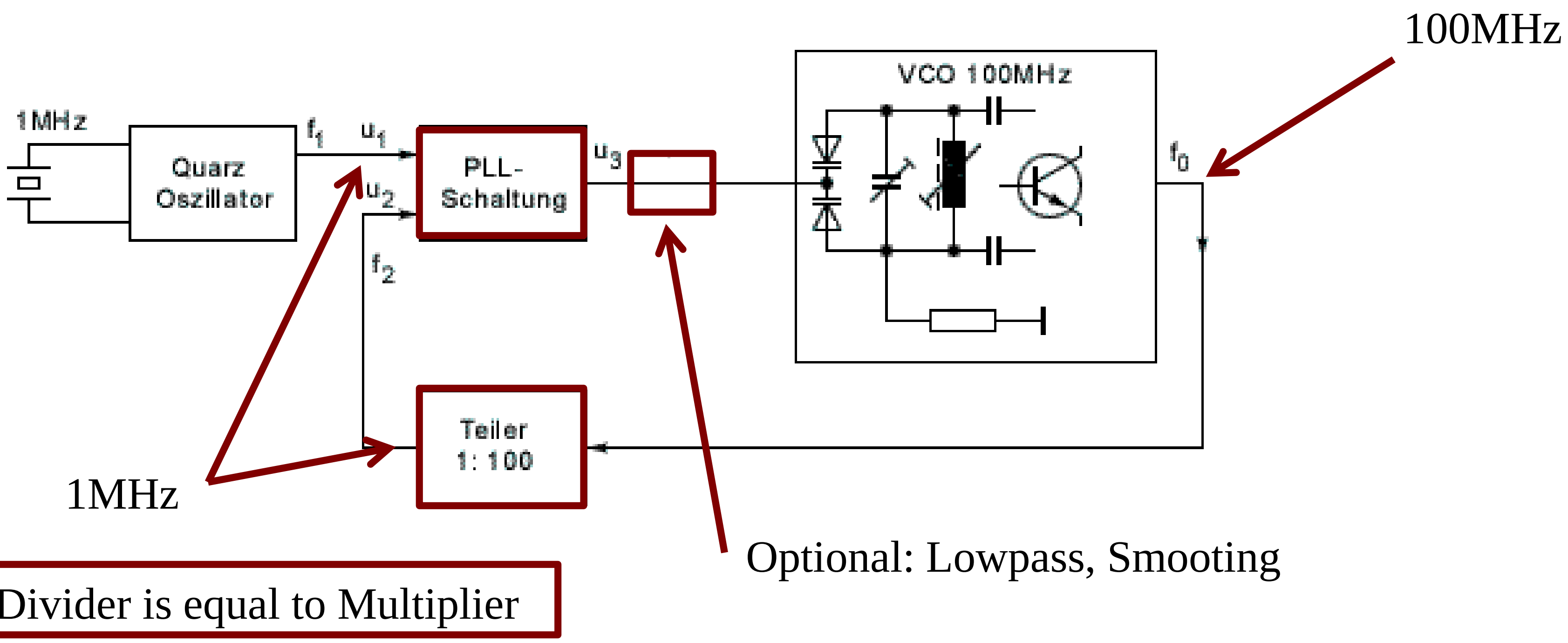
- Possible for integer steps only!
  - for fractional Multiplication a divider has to be used!
    - Example: 100 MHz = 3 / 2 * 66,6 MHz

Output **Input**

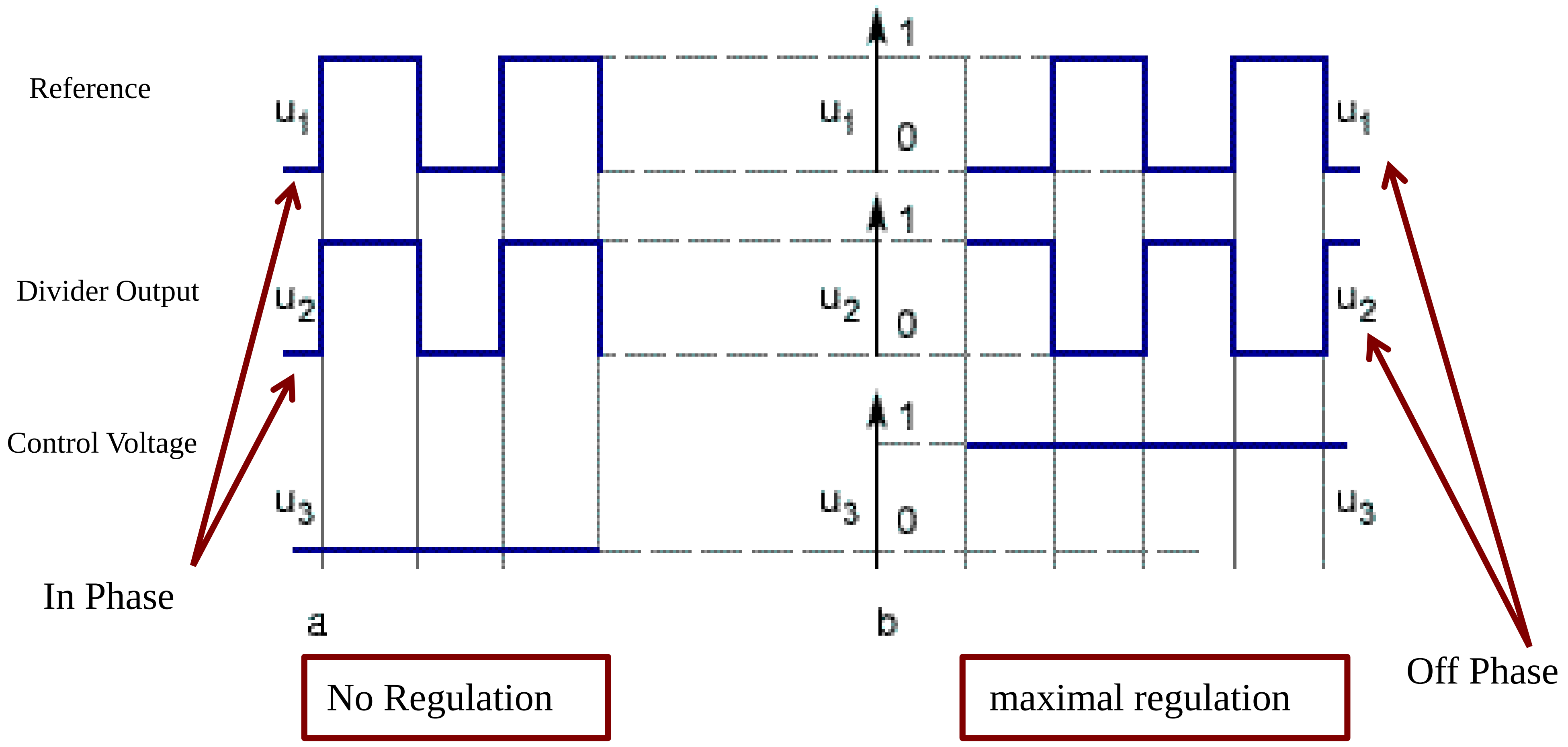
multiply by 3 and divide by 2

Phase Locked Loop (PLL)

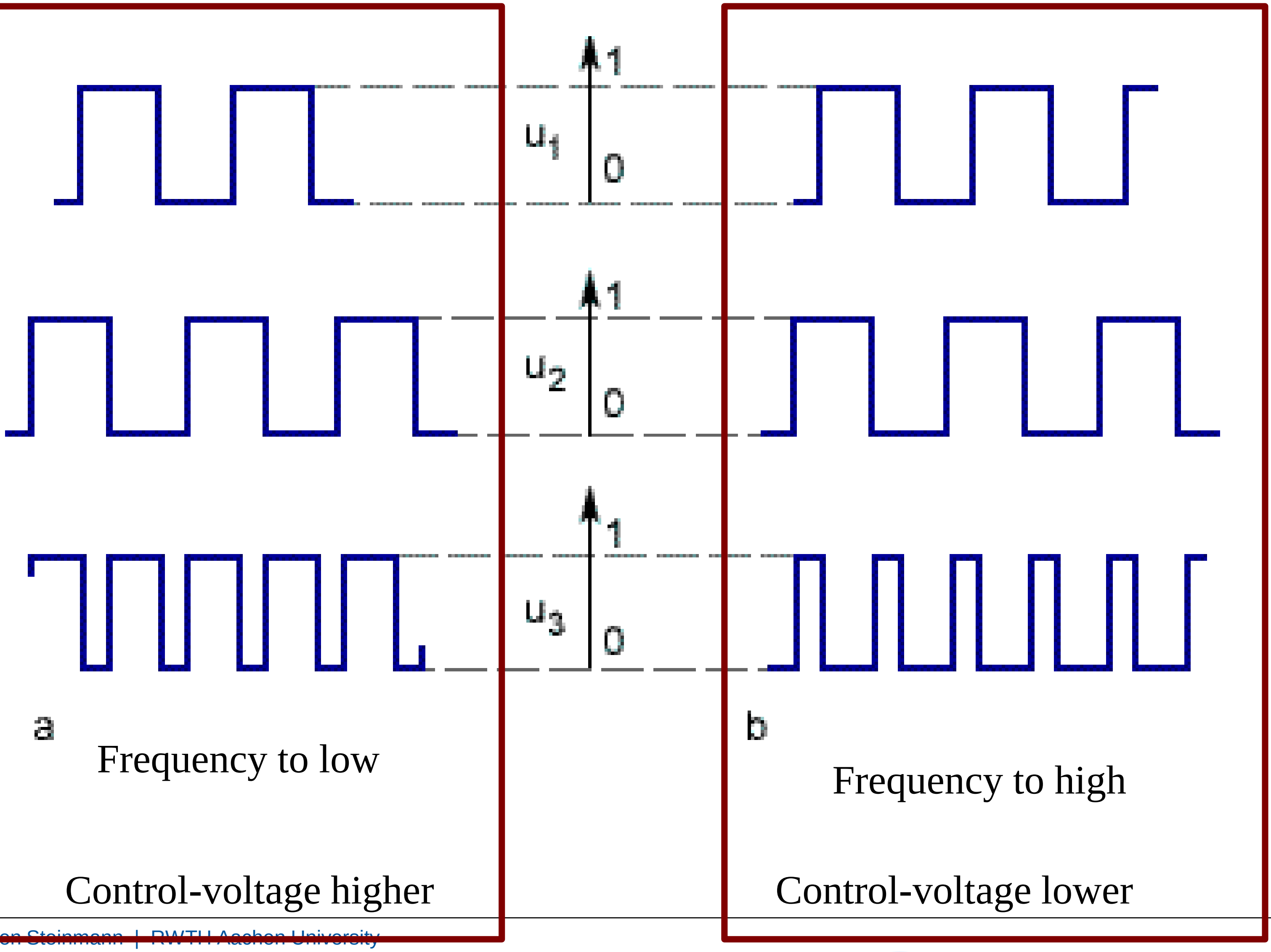
- Comparison of the phase with a reference oscillator:
  - Output up to GHz possible



Function I


Reference
Divider Output
Control Voltage
In Phase
u1
u2
u3
1
0
a
b
No Regulation
maximal regulation
Off Phase

Function II

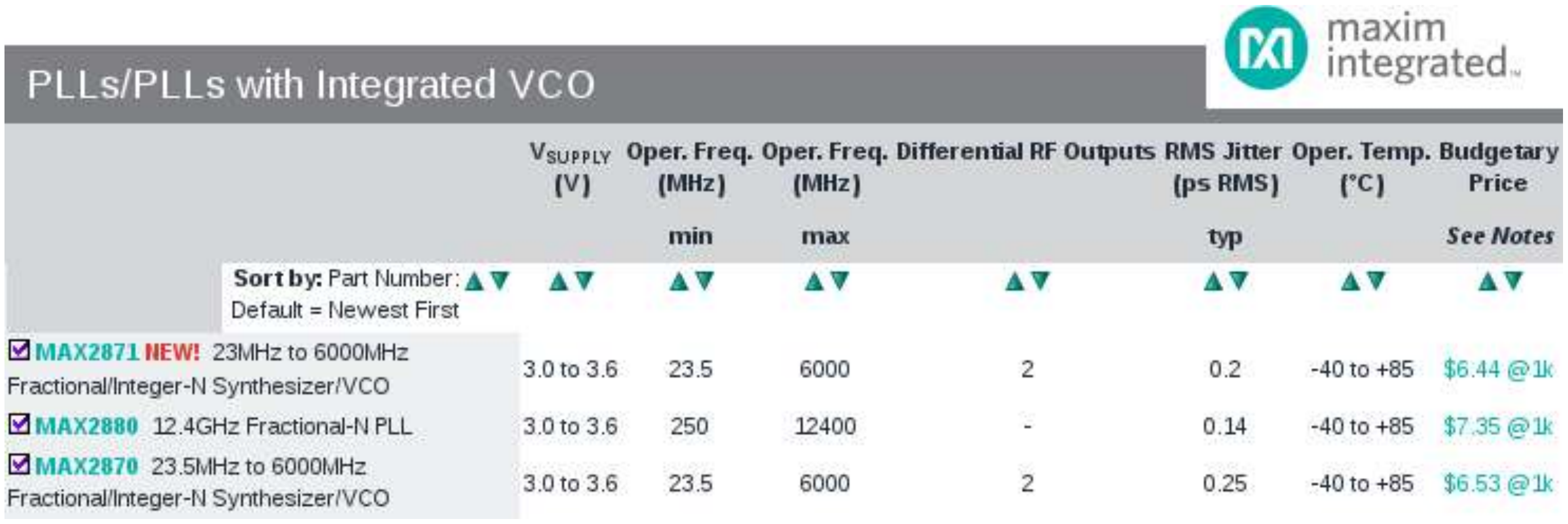

1
u_1
0
1
u_2
0
1
u_3
0
a
Frequency to low
Control-voltage higher
b
Frequency to high
Control-voltage lower

Function III & FPGA

- PLL is stable operating, when the output voltage is stable. (u3 = 0)
- There are so called Lock-In detectors available, which are able to detect this case.

- FPGAs have build in PLL blocks, which can be used to create a stable clock.
  - mainly used to derived secondary (slower) clocks

External PLLs



PLLs/PLLs with Integrated VCO

maxim integrated

	V_{SUPPLY} (V)	Oper. Freq. (MHz)	Oper. Freq. (MHz)	Differential RF Outputs	RMS Jitter (ps RMS)	Oper. Temp. (°C)	Budgetary Price
		min	max		typ		See Notes
Sort by: Part Number: ▲▼ Default = Newest First	▲▼	▲▼	▲▼	▲▼	▲▼	▲▼	▲▼
☑ MAX2871 NEW! 23MHz to 6000MHz Fractional/Integer-N Synthesizer/VCO	3.0 to 3.6	23.5	6000	2	0.2	-40 to +85	$6.44 @1k
☑ MAX2880 12.4GHz Fractional-N PLL	3.0 to 3.6	250	12400	-	0.14	-40 to +85	$7.35 @1k
☑ MAX2870 23.5MHz to 6000MHz Fractional/Integer-N Synthesizer/VCO	3.0 to 3.6	23.5	6000	2	0.25	-40 to +85	$6.53 @1k

tunable over a wide frequency range
up to 12.4 GHz

Clock in Experiments

- Stability is observed by a slower “reference Clock” and a counter
  - Possible reference clock sources
    - GPS → many GPS modules supply a 1 PPS pulse
    - Atomic Clocks → DCF77
    - Rubidium Clock Generators which can also supply any Master Clock signal

GPS as Reference

- Often used in experiments
  - OPERA
  - T2K
- Precise clock output by
  - Rubidium Reference Clock → 10MHz
- Huge accuracy and precision

Helpful, when timestamps should be matched over large distances.

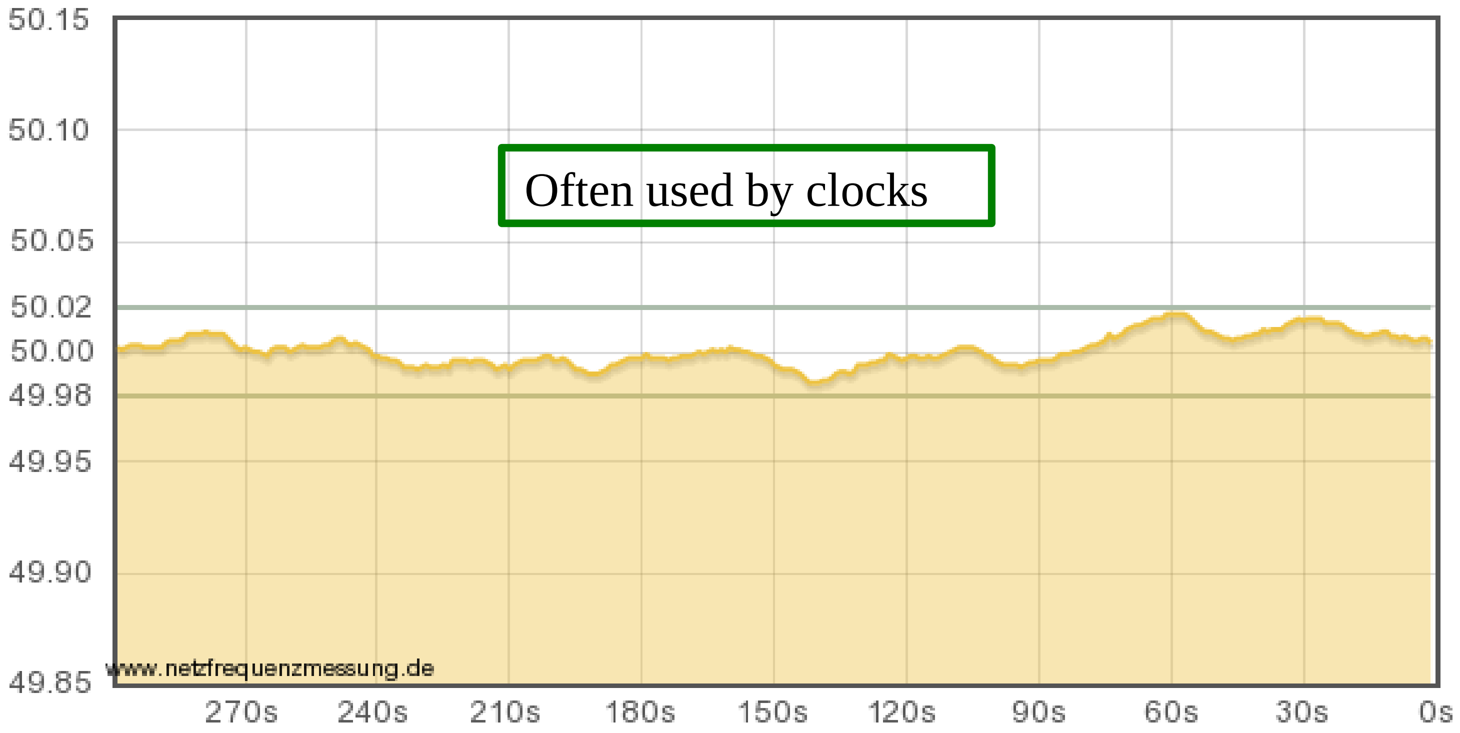
50Hz reference

- Frequency of the main power should have 50Hz
- The power companies try to fix this value +- 20mHz

Alternative:
DCF 77

One pulse
per second

Time of Day



Often used by clocks

Clock inside FPGA

- Routing of clock inside the FPGA is a challenge
- Requirements:
  - small clock delay
    - at every point inside the FPGA
  - phase must match signal
    - most modules latch their data on rising or falling edge
- There is a special high speed net inside the FPGA, which is only used for clock distribution

FPGA Clocking Challenge

Virtex™-7 2000T
– 2.000.000 Logic Cells
– 6.800.000.000 Transistors
(Switching Freq > 3.8GHz)
– DSP48E1 Slices with 741MHz global clock
– 12.5Gb/s Serial Transceivers
– 1066MHz Fmax Analog PLLs

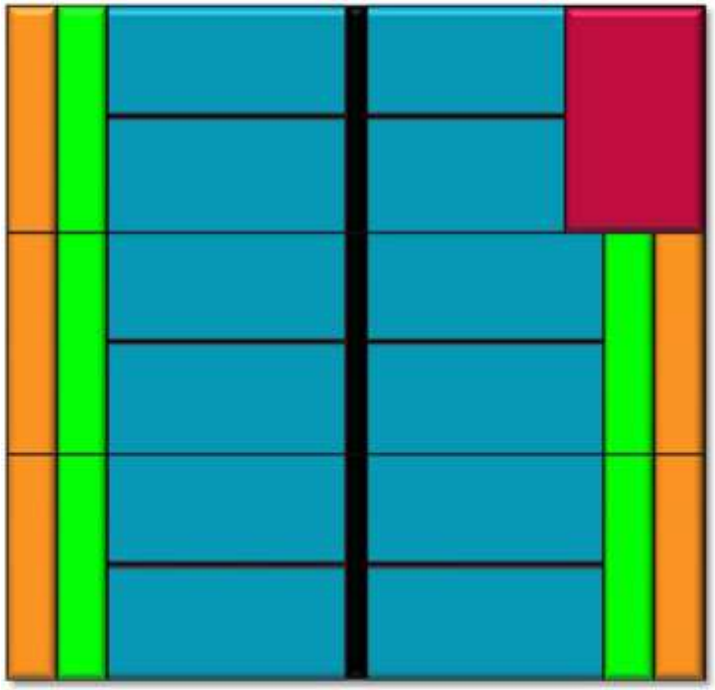
The internal PLL can not cover all requirements.
This fact has to be considered and calculated before the PCB design is done.

Applications:

PCIe GEN3 Jitter < 1ps
SDR Jitter < 300 fs
DDR3 Memory Jitter < 150ps

Largest Problem: Jitter

Clock-Distribution in FPGA



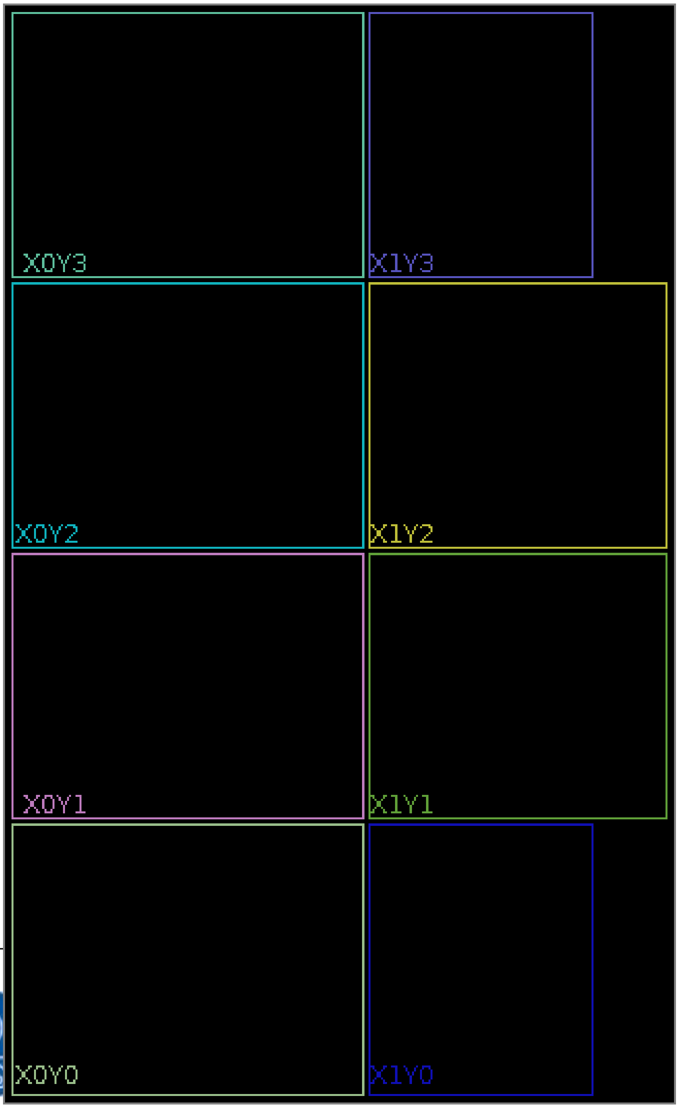
I/O Columns

CMT Columns — Clock Management Tile

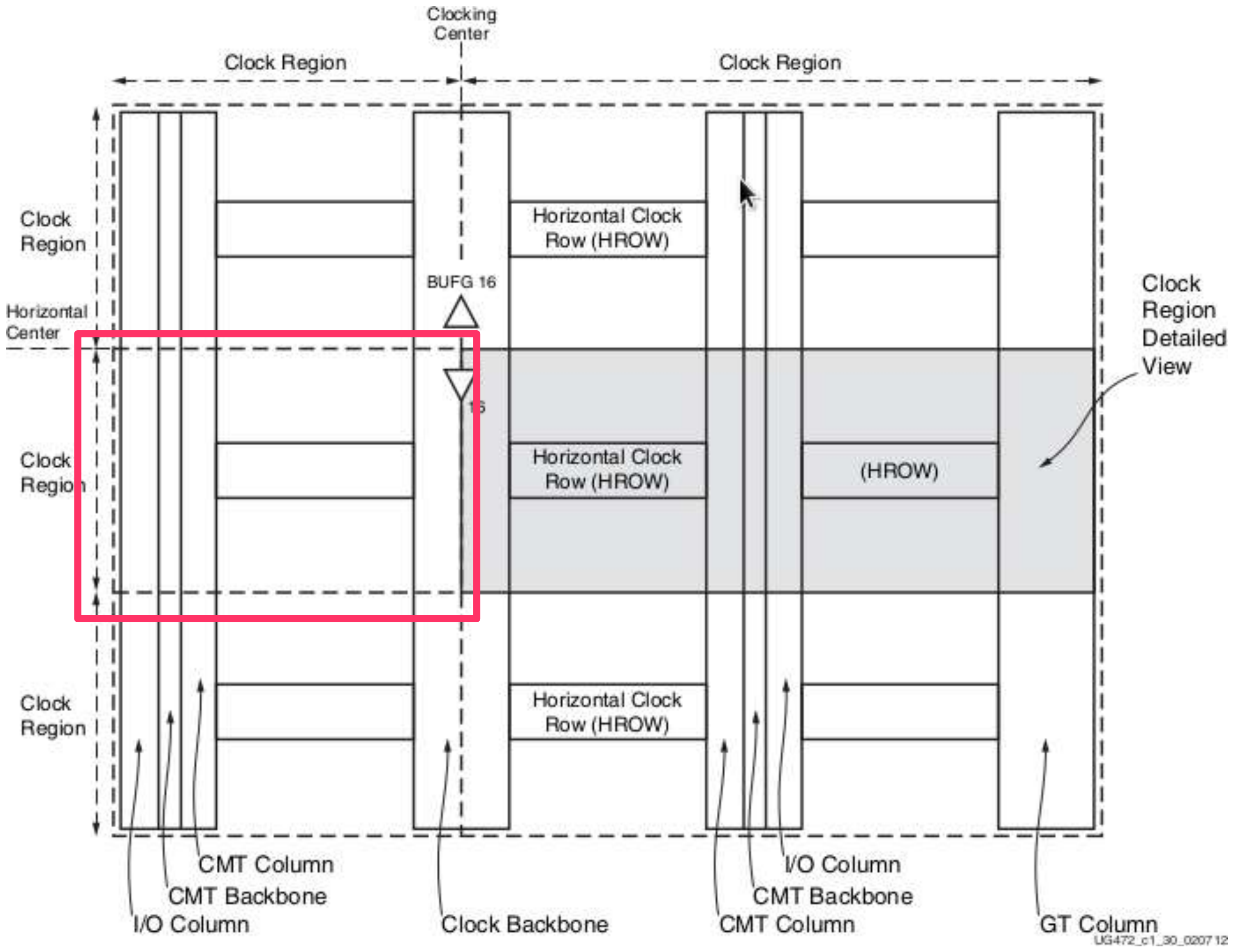
Clock Routing

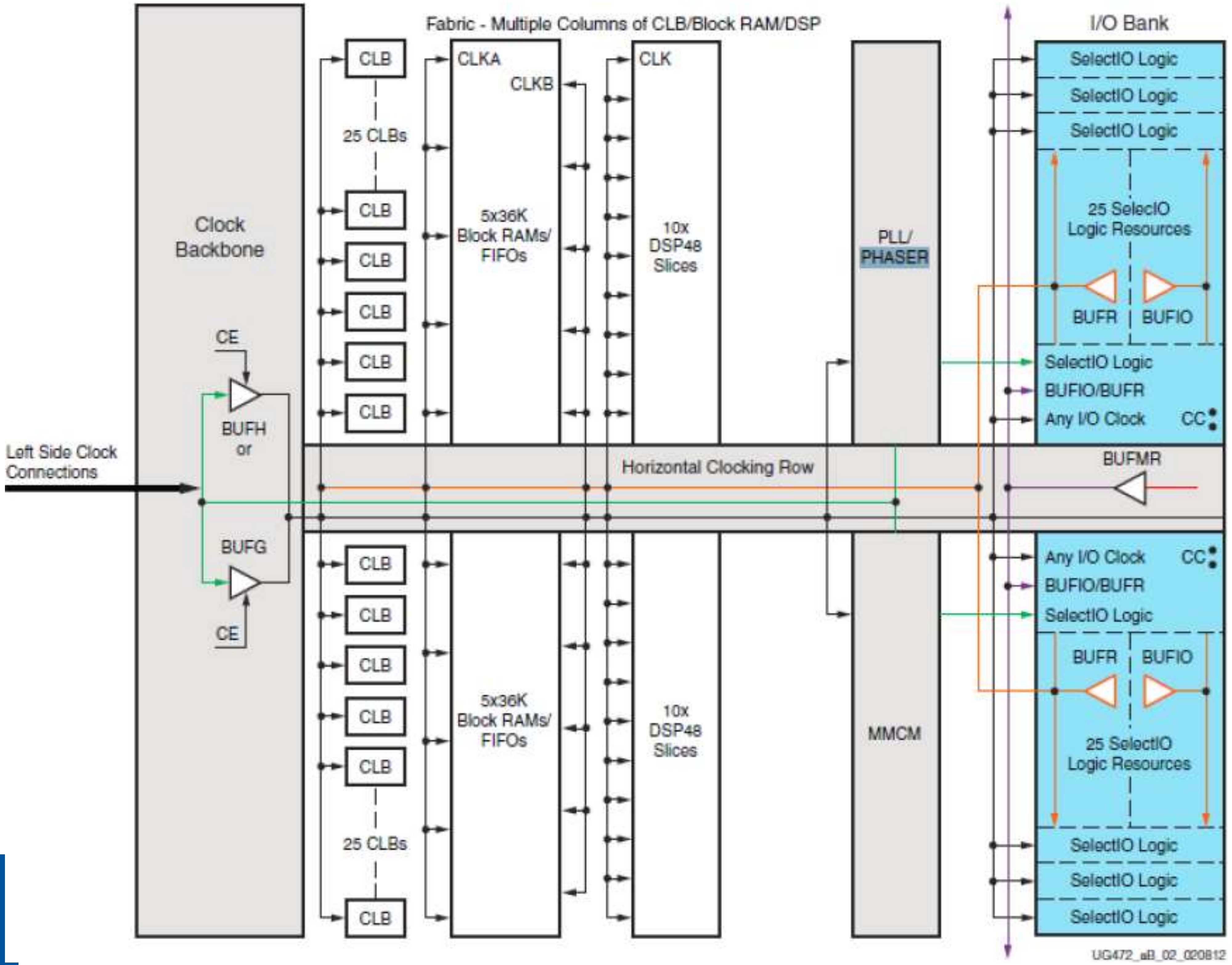
CLB, BRAM, DSP Columns

GT Columns — Gigabit Serial Transceiver



more detailed view ...




Fabric - Multiple Columns of CLB/Block RAM/DSP
I/O Bank
Clock Backbone
CLB
25 CLBs
CLKA
CLKB
5x36K Block RAMs/ FIFOs
CLK
10x DSP48 Slices
PLL/ PHASER
SelectIO Logic
25 SelectIO Logic Resources
BUFR
BUFIO
BUFIO/BUFR
Any I/O Clock
CC
CE
BUFH or
BUFG
Left Side Clock Connections
Horizontal Clocking Row
BUFMR
MMCM
UG472_aB_02_020812

Clock routing in FPGA

- very complex topic
  - we want to concentrate on the function and logic inside!

For those, who are interested in detail:

http://www.xilinx.com/support/documentation/user_guides/ug472_7Series_Clocking.pdf

Clock Domain Crossing

- Sometimes it is necessary to transfer data between different clock domains.
  - special requirements on the design
  - Use of storage elements:
    - FlipFlops
    - FiFos (sometimes also LiFo, FiLo)
    - RAM, ...

Gated Clock

- If the clock is switched of for non used areas / modules, power can be saved.
- Clock switching must be done via special clock gates.
- **Never use a single AND-gate to switch off the clock.**
- If you need to use a gated clock, always have a look at the RTL and check for right implementation.

Why is clock gating problematic?

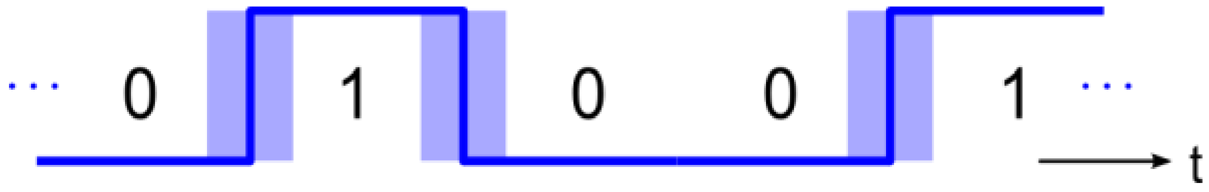
- When a simple clock gate is switched on
  - it is not done synchronous to the main clock
  - Very short pulses might appear → Glitch
- Runtime delays inside the gate
  - a phase difference between input and output

Some more …

- If high speed frequencies are used, one has to think a bit about the design and how it should be realised.
- Data should be transferred on a defined edge of the clock (in our case rising edge)

Digital effects I

- Jitter
  - Variation of a constant clock in time domain



- Glitch
  - Change of output due to timing issues

Q = A and B and **NOT** C

causes additional run_time

FPGAs will synthese this simple equation without a glitch
because all three signals are send into one LUT

VHDL

new elements

III. Physikalisches Institut B

RWTH AACHEN UNIVERSITY

Working with multiple bits at the same time

std_logic_vector

- Up to now, we just could handle a single bit by using std_logic

- std_logic_vector is a vector of std_logic

```
i_slv_test : in std_logic_vector (7 downto 0)
```

Highest bit

Lowest bit

Elements can be accessed by:

i_slv_test(4)	– returns single element
i_slv_test(4 downto 1)	– returns elements 4,3,2,1

How to assign values to std_logic_vector

all 0

all 1

undefined

```
signal Slv1 : std_logic_vector(7 downto 0);
signal Slv2 : std_logic_vector(7 downto 0) := (others => '0');
signal Slv3 : std_logic_vector(7 downto 0) := (others => '1');
signal Slv4 : std_logic_vector(7 downto 0) := x"AA";
signal Slv5 : std_logic_vector(7 downto 0) := "10101010";
signal Slv6 : std_logic_vector(7 downto 0) := "00000001";
```

Hexadecimal value

Binary values

```
Slv1      <=      slv2(7 downto 4) & slv3(3 downto 0);              -- concatenation
          results in “00001111“ upper 4 bits from slv2 and lower 4 bits from slv3
```

Signals in VHDL

- VHDL cannot read back outputs
- Means, if we e.g. want to build a shift register or a counter, we need a internal signal, which handles the data

- When we have done the modification, we assign the value to the output

```
architecture behavior of lecture02 is
    signal shift_reg : std_logic_vector(7 downto 0) := "00000001";
                -- define signal, because we cannot readback outputs
begin

    process(i_sl_CLK)
    begin
        if rising_edge(i_sl_CLK) then

                -- do something with shift_reg

                o_slv_shift <= shift_reg;
        end if;
    end process;
end behaviour;
```

Handling the clock in VHDL

- The clock should be the only signal on the sensitivity list!
- Actions typically happen on the rising edge (transition from 0 to 1)

```
process(i_sl_CLK)
    begin
        if rising_edge(i_sl_CLK) then
```

Handling the clock in VHDL II

- Implementing a enable

```
architecture behaviour of lecture02 is
    signal shift_reg : std_logic_vector(7 downto 0) := "00000001";
                -- define signal, because we cannot readback outputs
begin

    process(i_sl_CLK)
    begin
        if rising_edge(i_sl_CLK) then
            if (i_sl_en = '1') then

            end if;
            o_slv_shift <= shift_reg;
        end if;
    end process;

end behaviour;
```

Calculation in VHDL

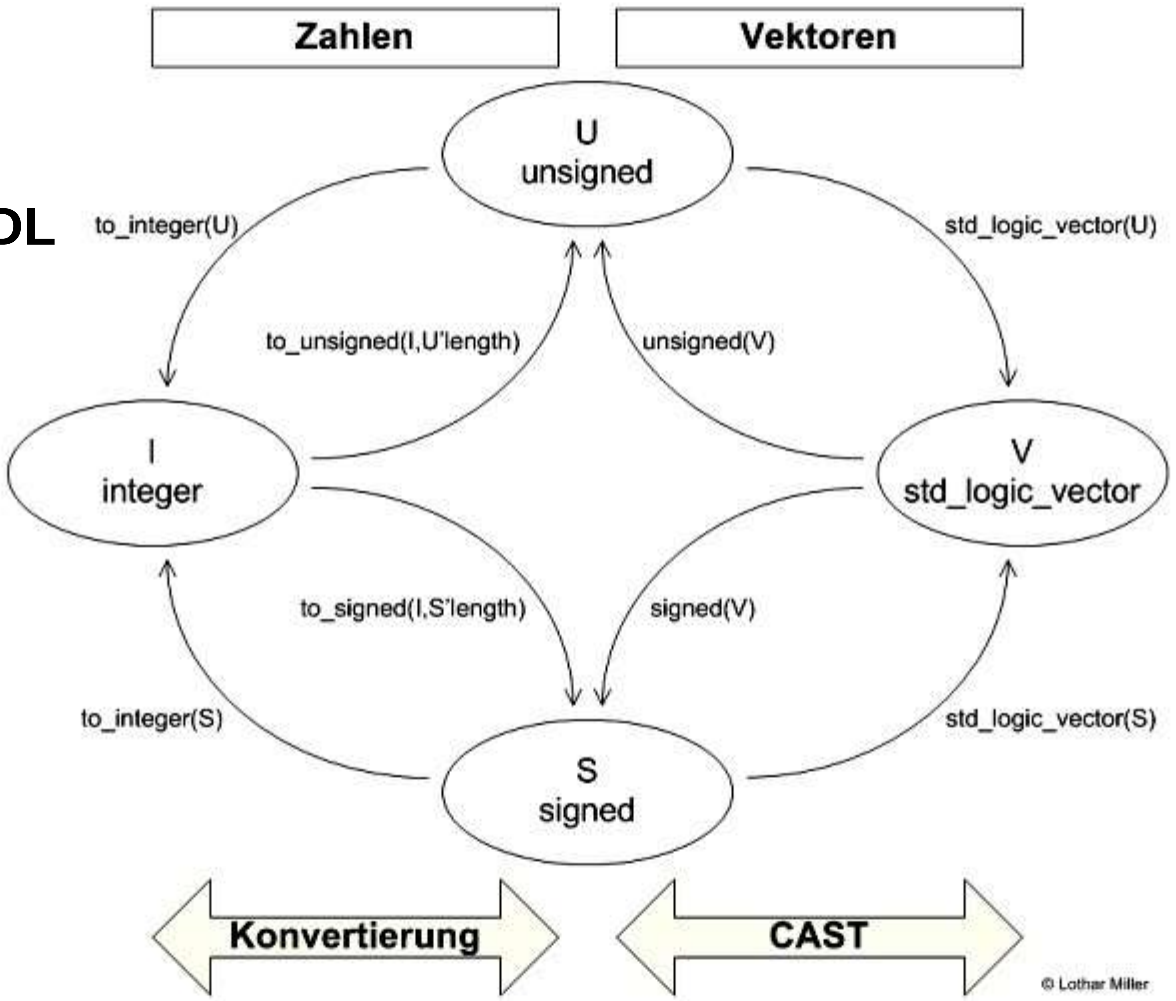
- Not possible using
  std_logic_vector

Have to convert / cast to unsigned

Recipe to do calculations in VHDL

1. Cast std_logic_vector to unsigned
2. Do calculation
3. Cast result back to std_logic_vector

```
library IEEE;
use IEEE.STD_LOGIC_1164.all;
use IEEE.NUMERIC_STD.all;
```



Exercises for Today

- Lecture 2a
  - A 8 bit shift register, which can shift left and right and can be enabled / disabled
- Lecture 2b
  - A 8 bit counter, which can count up and down and can be enabled / disabled