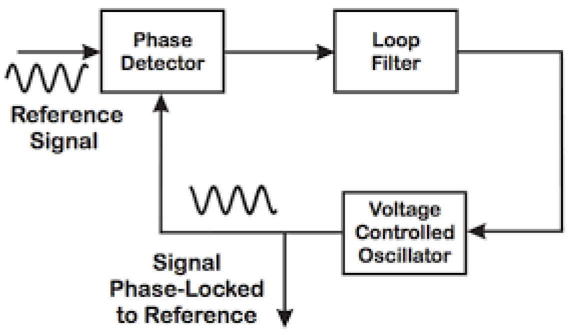

Phase Detector
Loop Filter
Reference Signal
Voltage Controlled Oscillator
Signal Phase-Locked to Reference

Figure 1: Structure of PLLs

0.1 Goal of this work

Electronic clocks are used to provide a time-reference to different components of an electronic system. These clocks need to be synchronized to a common frequency and phase to coordinate the operation and interaction of these components. This is important for the coordination of information processing in complex systems.

The State-of-the-Art concept to the synchronization of electronic clocks is to use a precise reference clock and to distribute its time-signal to low quality electronic clocks, so called phase-locked loops (PLLs). Thereby a time-reference can be spatially distributed over a system. However, as in large the systems the distribution of the reference clocks signal via the hierarchal approach becomes complicated, the so called *GALS* ansatz is commonly used. That means, that systems are operated (g)lobally (a)synchronous and (l)ocally (s)ynchronous.

A novel approach to the synchronization in electronic systems relies on the mutually coupling of PLL clocks. This ansatz with a flat hierarchy and without a governing reference clock allows for self-organized synchronization over networks of PLLs. This works robustly in the presence of large signal transmission delays and clock heterogeneity.

My goal is to implement an all-digital phase-locked loop (ADPLL) in VHDL. I am interested in the dynamics of such elements and want to study their dynamics and function. In a first step I will therefore implement and test a single ADPLL that is entrained by an external reference signal. In the next step I aim at using two ADPLL clocks in a mutually coupled setup. In this case I want to simulate and study self-organized synchronization and the effects of delayed signaling.

1 Phase-locked loops

1.1 The concept of phase-locked loops

What is the purpose of PLLs Phase-locked loops (PLLs) are widely used circuits for frequency synthesis applications. In general in electronic systems there is a main reference clock which is used as reference and entrains PLLs distributed in the system. Thereby a distributed synchronized clock reference can be established.

Where and how are they used (state-of-the-art) They are used in a large range of applications such as radio frequency equipment including radio receivers, test equipment and other items of radio frequency electronics.

1.1.1 Types of phase-locked loops

- Analog PLL (all analog, e.g., multiplier as PD):

For this type of PLLs,analog multipliers are used as a phase detector.They consist of active or passive loop filters and voltage-controlled oscillator (VCO).

- Digital PLL :

They are analog PLLs with a digital phase detector (such as XOR, edge-trigger JK, phase frequency detector). They may have digital divider in the loop. They contain analog loop filter and analog VCO.

- ADPLL (All digital PLLs)

Their phase detector, filter and oscillator are all digital. They contain a numerically controlled oscillator (NCO). In this project, we designed ADPLLs in Xilinx environment.

1.1.2 Comparison between the ADPLLs and PLLs

When ADPLLs and analog PLLs are compared, one of them has no absolute advantage over the another. As an advantage, ADPLLs are low-power-consuming-circuits and they cover less area than the analog PLLs. On the other hand, their frequency range is smaller and their lock time is longer with respect to analog PLLs. Analog PLLs are widely used and show high performance characteristics in terms of jitter and frequency range. However, the high power consumption and large area coverage have always been drawback for this type of PLLs. Moreover, as the technology scales down into deep submicron, the design of these analog circuits becomes very sensitive and increases the design cycle time and time to market. The last decade has shown a lot of interest in replacing the analog PLLs with an All Digital PLL (ADPLL).

1.2 The main components of phase-locked loops

- Phase Detector:
- Loop Filter:
- Controlled Oscillator:

PHASE DETECTOR: It detects the phase difference between the output signal of a PLL and a reference signal. The control voltage for the VCO will be generated according to this phase difference between the two signals. XOR gates can be used for detecting phase difference but those gates are not capable of giving information about which signal is leading and which signal is lagging. Flip-flop phase detectors however, detect the phase and the frequency.

PHASE: The phase is defined as a quantity that increases linearly in time over one period of a periodic process.

$$\frac{1}{360f} = t_{\text{per degree}}$$

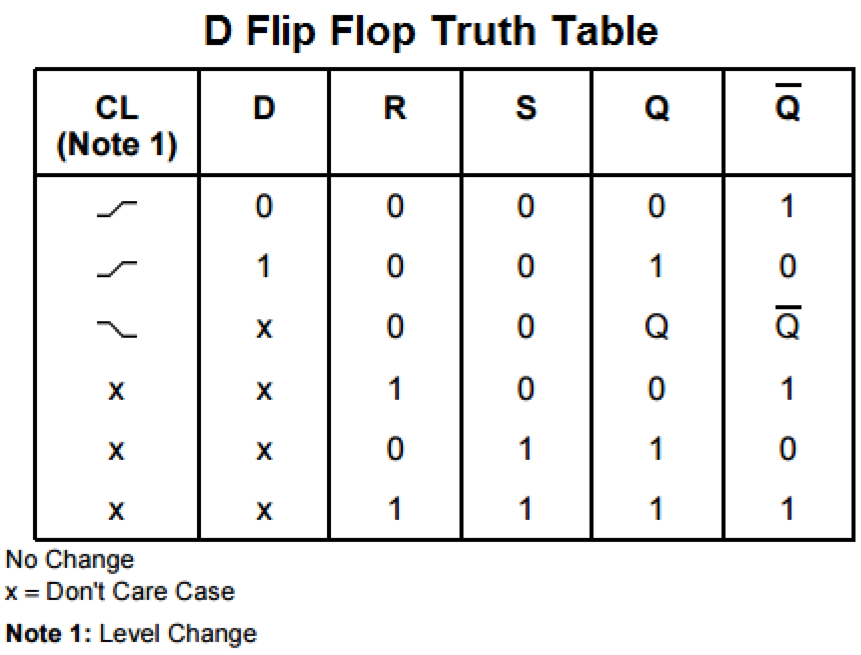
LOOP FILTER: Loop filter generally has two distinct functions.The primary function is to determine loop dynamics, in another way of saying stability. This is how the loop responds to disturbances, such as changes in the reference frequency, changes of the feedback divider, or at startup. Common considerations are the range over how fast the loop achieves lock and damping behavior.The second common consideration is limiting the amount of reference frequency energy (ripple) appearing at the phase detector output that is then applied to the VCO control input.

CONTROLLED OSCILLATOR: All phase-locked loops employ an oscillator element with variable frequency capability. This can be an analog VCO either driven by analog circuitry in the case of an APLL or driven digitally through the use of a digital-to-analog converter as is the case for some DPLL designs. Also there are pure digital oscillators such as a numerically controlled oscillator which are used in ADPLLs.

1.3 All-digital phase-locked loops

We implemented and simulated 2 kinds of phase detector:

- Phase Frequency Detector
- Phase Shifting Detector



D Flip Flop Truth Table

CL (Note 1)	D	R	S	Q	$\overline{Q}$
↑	0	0	0	0	1
↑	1	0	0	1	0
↓	x	0	0	Q	$\overline{Q}$
x	x	1	0	0	1
x	x	0	1	1	0
x	x	1	1	1	1

No Change
x = Don't Care Case

Note 1: Level Change

Figure 2: Truth table of d-flip-flop

PHASE FREQUENCY DETECTOR D-flip-flops are widely used components in digital design and electronics applications. Generally they can be implemented in two different ways: with synchronous or asynchronous reset. These flip-flops have clock and D ports. They are sensitive to rising edge of the clock input. Output value of the d-flip-flop changes only when the rising edge of the clock is detected otherwise flip-flop will keep its state. When there is a rising edge of the clock input, output value(Q) will be assigned by the value of the input signal(D).

This is the phase frequency detector which is also used in the project. It consists of two d-flip-flops. V_{cc} is given as an input of the d-flip-flop, reference and feedback signals are given as clock inputs of the flip-flops. In our design the input port (D) is connected to V_{cc}, i.e., set to high. Whenever the clock input of the flip-flop detects a rising edge, the Q value will be '1'. If it is the rising edge of the reference clock the Up signal will be activated, if it is rising edge of feedback signal the Down signal will be activated. If both outputs become '1' then reset input will be active. As a result of reset, Up and Down signals also will reset to '0'.

DELAY SHIFTING PHASE DETECTOR In this type of phase detector, buffers and d-flip-flops are used. The phase difference between the start and the stop signals will be measured by multiple delay value of a buffers(τ). In one case, stop signal is lagging the start signal. Start signal will be shifted to the right after every buffer transition.Stop signal is given as clock of the flip flops and when there is a rising edge of clock signal, the input value will be transferred to the output port. By this way of implementation, the output values of the flip-flops will be '1' up to the point where start signal is enough shifted to the right. After this point output value of the all flip-flops will become '0'. Resolution of this kind of phase detector design can be controlled by changing the delay values and number of these buffers. If τ is a known value, the range of the phase difference(ΔT) can be calculated with the output values of the flip-flops. For example if the output value of the flip-flops is "11000" , it means that after 2 shifts the output values of flip-flops will be '0'. So the period difference should be like this:

$$2\tau \leq \Delta T < 3\tau$$

In order to have better resolution, buffers with different delay values can be used. In this case, after every transition, phase difference will be decrease by the difference between the delay values of the buffers.

LOOP FILTER: In electrical engineering applications, high frequency part of a system causes instability .In analog case, low pass filters eliminate the high frequency elements. These filters can be obtained

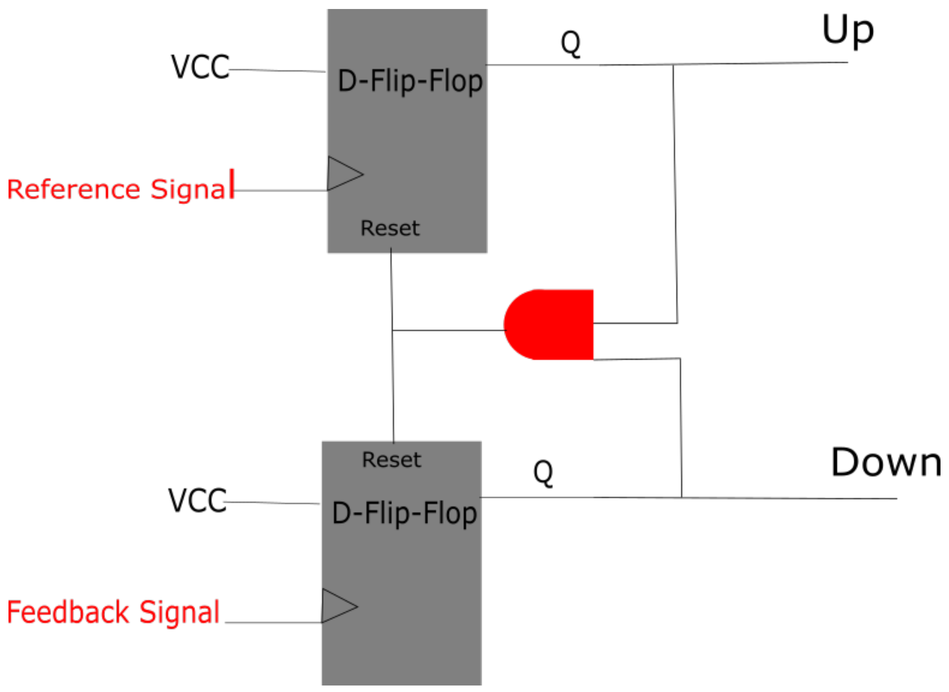
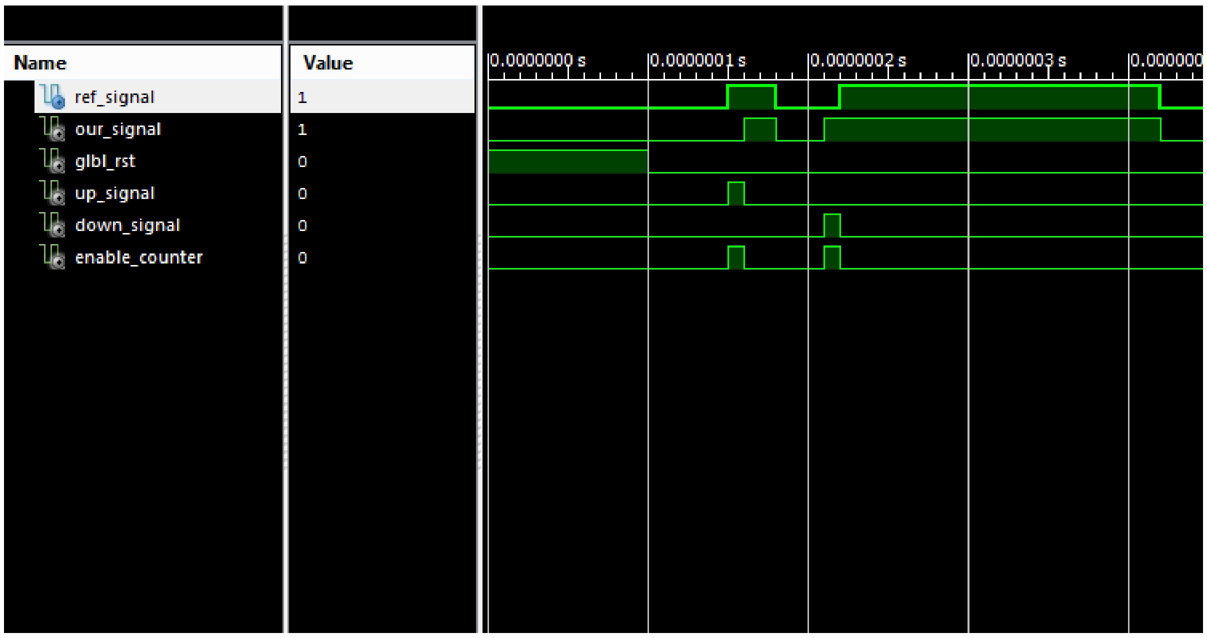


Figure 3: Phase frequency detector (PFD)



Name	Value
ref_signal	1
our_signal	1
glbl_rst	0
up_signal	0
down_signal	0
enable_counter	0

Figure 4: Simulation of PFD

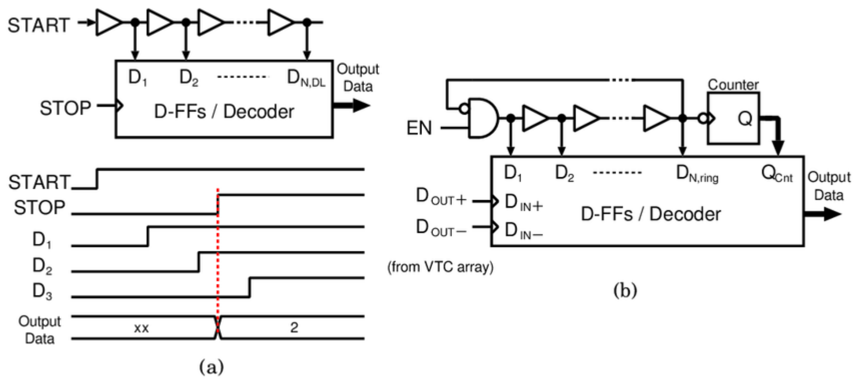
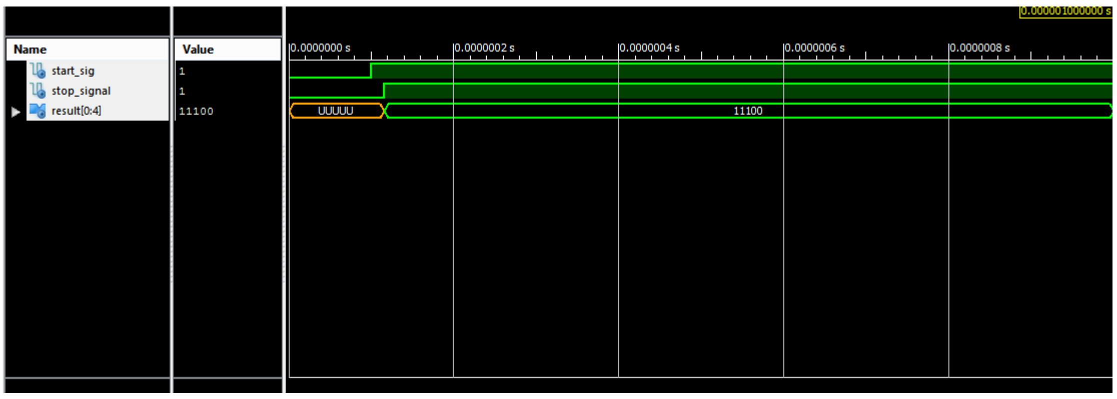


Figure 5: Shifting Phase Detector



```
Name          Value
start_sig     1
stop_signal   1
result[0:4]   11100
```

Figure 6: Simulation for Shifting Phase Detector

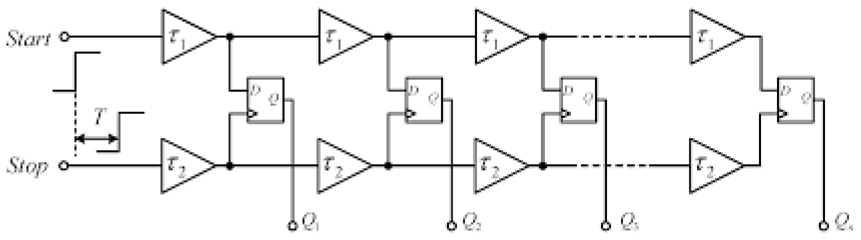


Figure 7: Another design for Shifting Phase Detector

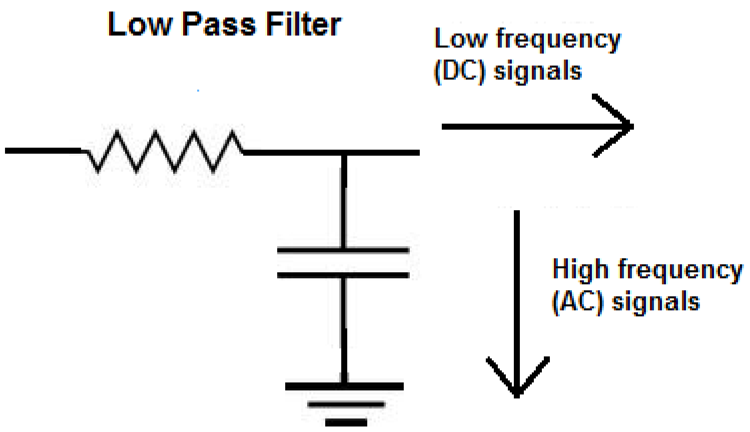


Figure 8: Low Pass Filter

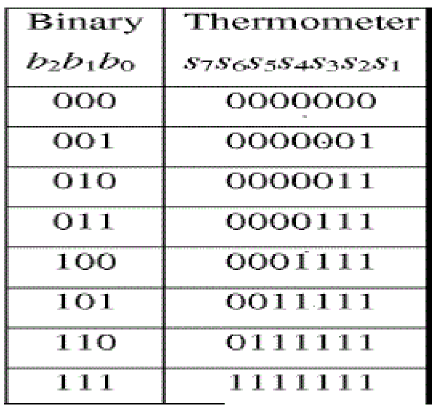
by a capacitor and a resistor.

Counters can be used for filtering in all digital phase locked loop design. In our design there are two counters: up_counter and down_counter. Both of them takes 7 bit logic vector as input.In order to initialize the system, counters are reset. Initially up_counter is set to "0000000" and down_counter is set to "0111111". After that, up_counter counts to up if UP signal is active or down_counter counts to down if DOWN signal is active. In order to count the duration between the rising and the falling edges of a signal, these counters use fast_clock. This fast_clock has a very large frequency value and at every rising edge of fast_clock,the value of the difference signal will be checked. If the value is '1', counting process will continue, if the value is '0' , counting will stop.In order to get rid of sudden fluctuations at the period value of the DCO, modulus can be used. This modulus can be defined by user . Up until counting reaches to the modulus value, up_counter and down_counter will not be updated. Choosing the right value for the frequency of the fast clock and for the modulus is still ambiguous. Whenever this modulus value gets bigger, system reacts slowly in other words system reacts more stable. If fast_clock with a big period is used then this can result in incapability of detecting phase differences.

Ripple_counter sums the values of the up_counter and down_counter. Output of the ripple_counter is the input of the thermometric decoder. Initially when up_counter and down_counter are reset, ripple_counter's value will be "0111111". It is the middle period value which DCO can produce without an external input coming from phase frequency detector. This behavior is similar to analog phase locked loops in a way that even if they are not stimulated with an input,thesy oscillate at their own intrinsic frequency.

THERMOMETRIC DECODER: This decoder takes the ripple counter's 7 bit output as an input and it produces $2^7 = 128$ bit output. These outputs are used for enabling the three-state inverters. This decoder produces as many number of 1's as input's integer value. For example with input 0000111l decoder produces 11111110...00 as an output. In figure 9, input is 3 bit binary number,whereas in our design it is 7 bit binary number. Whenever bit number increases,we get more ability to have different frequency values. 128 different period value is suitable for our design.

DIGITALLY CONTROLLED OSCILLATOR: It takes thermometric decoder's outputs as input. In reality voltage-controlled-oscillators have their own intrinsic frequencies so if there is no input effecting them, they will oscillate with those intrinsic frequency. Since the design is digital rather than analog, oscillator is controlled by some digital code rather than voltage value. This digital code for controlling comes from thermometric decoder. Whenever this code changes oscillator will generate another periodic signal., ring oscillator logic can be used in order to implement an oscillator.Ring oscillator consists of odd number of inverters.If it has even number of inverters it wont be able to generate periodic signals. If inverters have delay (τ) then the oscillator in the figure will generate a periodic signal whose period



Binary $b_2b_1b_0$	Thermometer $s_7s_6s_5s_4s_3s_2s_1$
000	0000000
001	0000001
010	0000011
011	0000111
100	0001111
101	0011111
110	0111111
111	1111111

Fig 5- Binary to thermometric conversion

Figure 9: Thermometric decoder

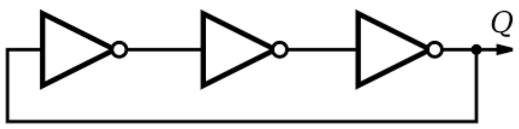


Figure 10: Ring oscillator

is $3 \times \tau \times 2$. By changing the number of inverters or the delay of them, controllable oscillator can be obtained. Since we have 128 enable numbers, we can have maximum that many of inverters.

- By changing the delay of the inverters or by changing the number of inverters, different frequencies can be obtained. In my design I used 7×2×21 inverters. First we implemented a block then by using these block all dco become implemented.

From the figure it can be seen that at every row there are different delay values. At every branch, there are even number of inverters because there should be odd number of inverters in total.

DCO is ring shaped. In the design enable is needed because otherwise Xilinx gives some error messages about initializing. By giving '0' to enable value for initialization, AND gate will ensure that output signal of AND-gate in other words input signal of DCO is initialized to '0' and to stay in safe regime, that value is given during the 1.5 intrinsic period of DCO. Enable values control the whole DCO behaviour for example enable 0 will activate the inverters at the first column and the first row,enable 1 activates the inverters in the first column and the second row and up to whole rows in the first column is activated. Then same thing will be applied for the second column and so on. For every column the the activated branch which has smallest value will be chosen by multiplexer and output of that branch will be transferred into other column as input. Below plots for the period and frequency responses of DCO's can be seen.

As a whole, the designed system looks like as in below:

These are the some scenarios and limit cases given to the system:

- As reference signal, '1' is given and the feedback mechanism is removed from the design. Therefore, up_signal is always on and it gets never '0' because there is no input given to the other flip flop. It will count all the time to up until the saturation value of the up_counter. At saturation value, up_counter still continues to count to up but ripple counter can not continue to count more after its saturation and causes erroneous pulses.

- Same frequency and same phase values are given to our_signal and reference_signal. There is no phase difference between them and DCO operates with intrinsic frequency.In this case, feedback-

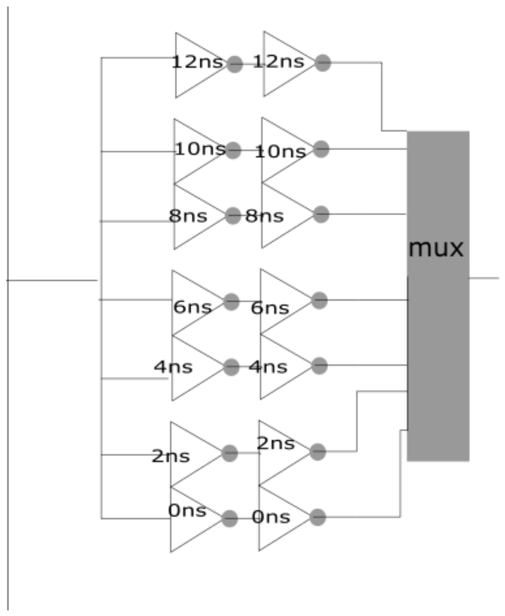


Figure 11: One column for dco

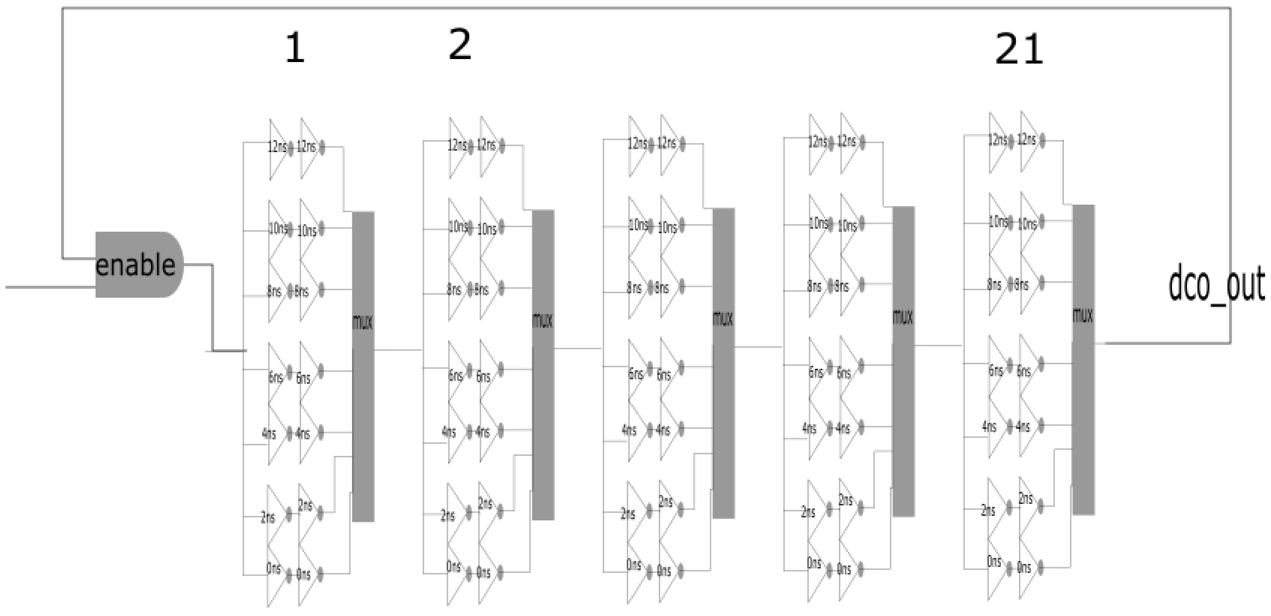


Figure 12: DCO

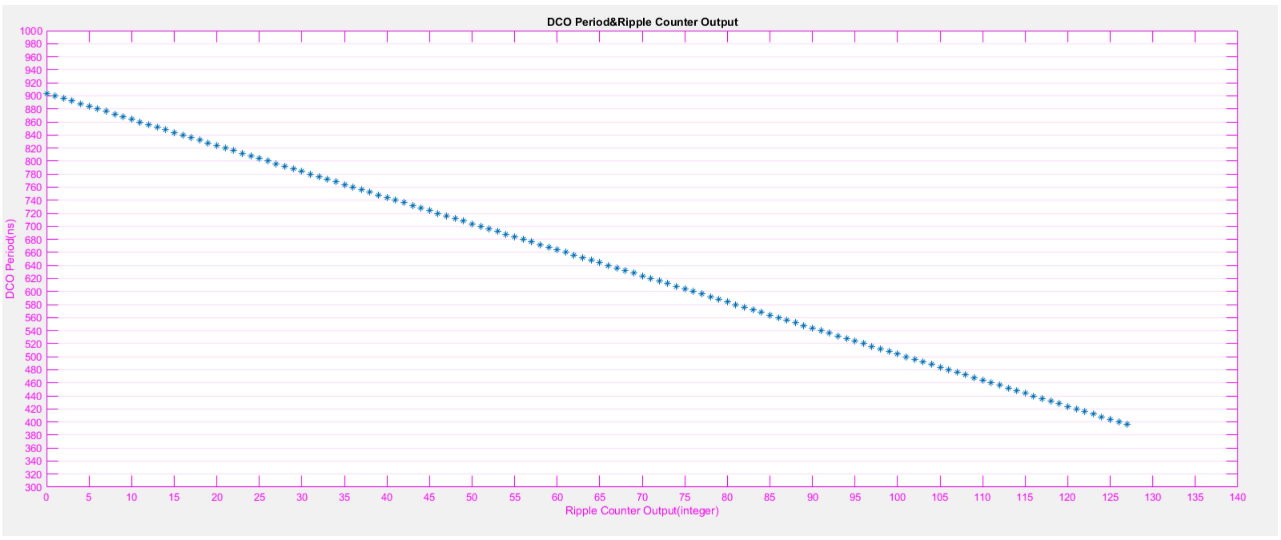


Figure 13: Period Response Of DCO

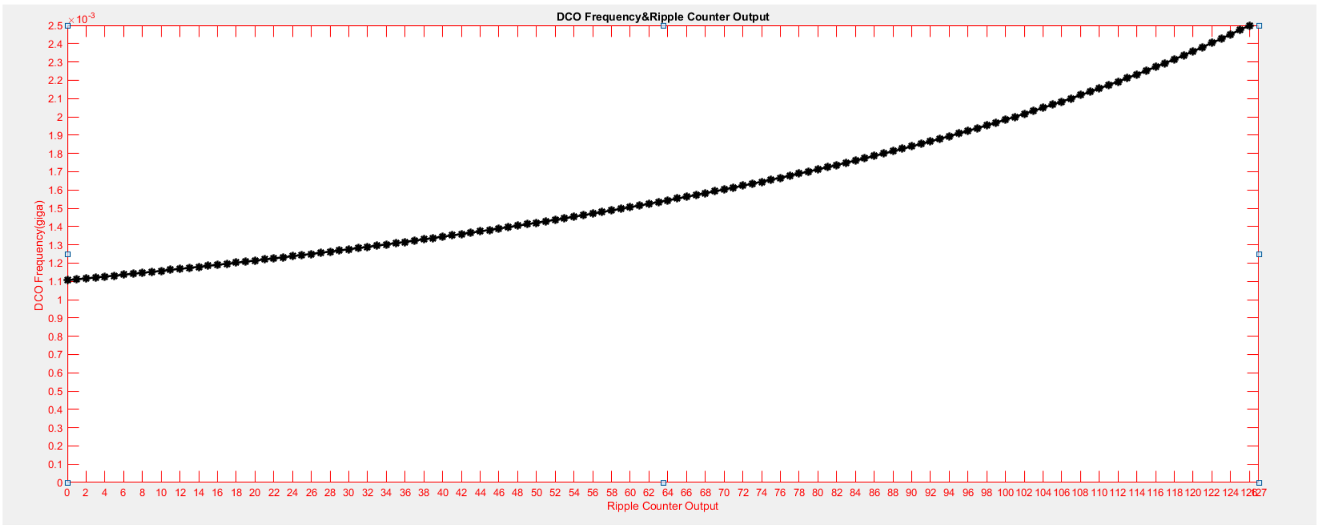


Figure 14: Frequency Response of DCO

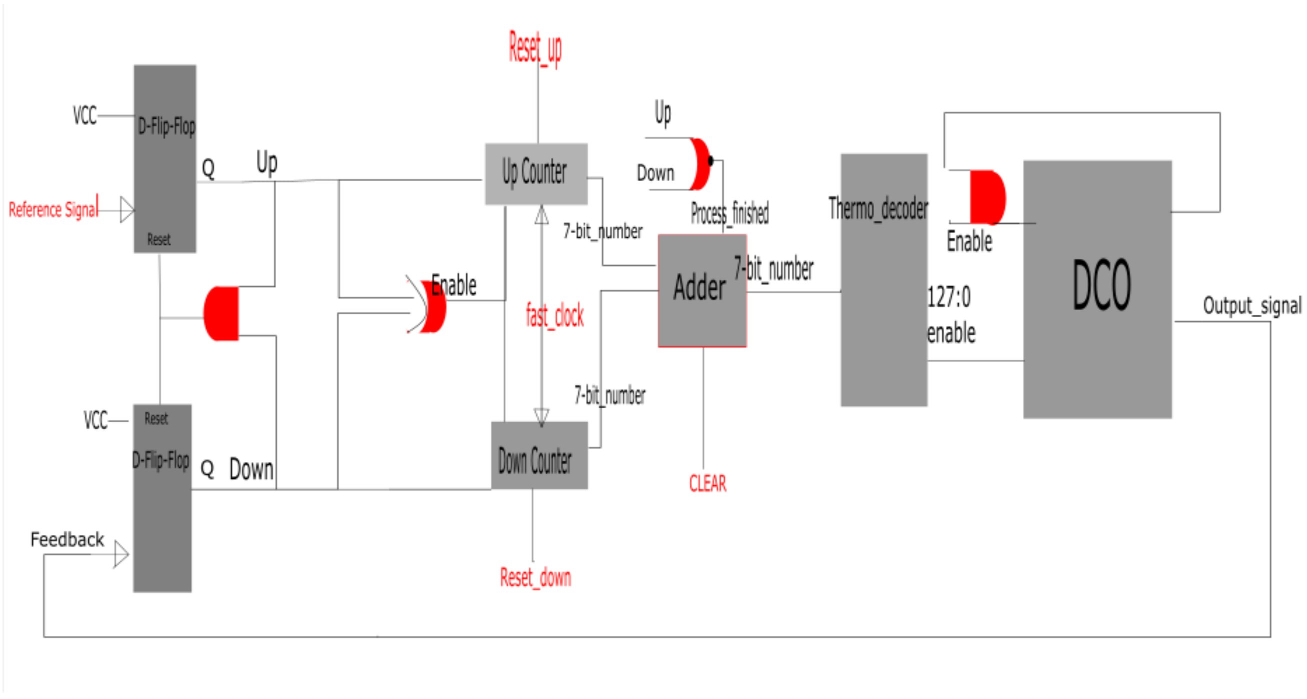
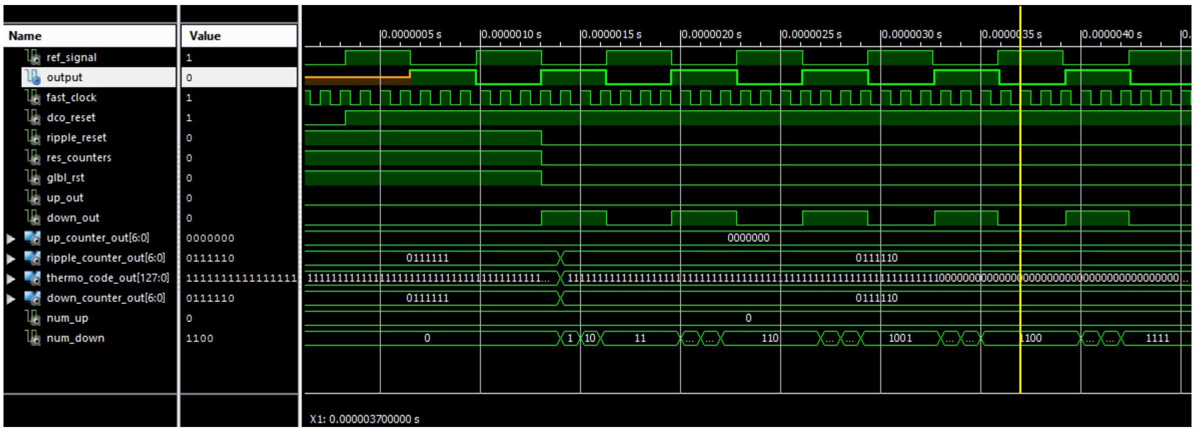


Figure 15: All Digital PLLs

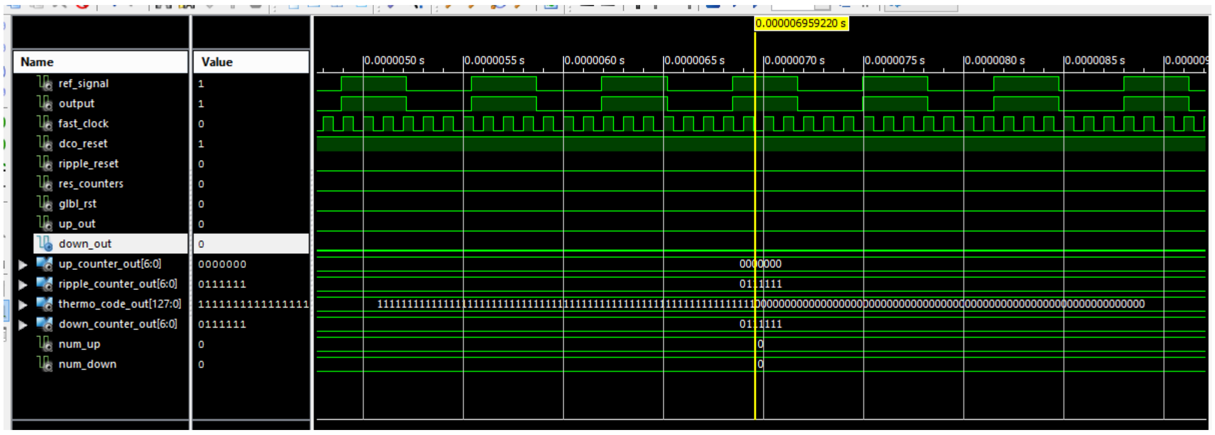
loop is removed again to test the tansition from the PFD to oscillator. Since PFD does not create a new input value, ripple counter value "0111111" will stay the same.



Name	Value
ref_signal	1
output	0
fast_clock	1
dco_reset	1
ripple_reset	0
res_counters	0
glbl_rst	0
up_out	0
down_out	0
up_counter_out[6:0]	0000000
ripple_counter_out[6:0]	0111110
thermo_code_out[127:0]	1111111111111111
down_counter_out[6:0]	0111110
num_up	0
num_down	1100

X1: 0.000003700000 s

Figure 16: antiphase input



Name	Value
ref_signal	1
output	1
fast_clock	0
dco_reset	1
ripple_reset	0
res_counters	0
glbl_rst	0
up_out	0
down_out	0
up_counter_out[6:0]	0000000
ripple_counter_out[6:0]	0111111
thermo_code_out[127:0]	1111111111111111
down_counter_out[6:0]	0111111
num_up	0
num_down	0

Figure 17: Equal phases and frequencies

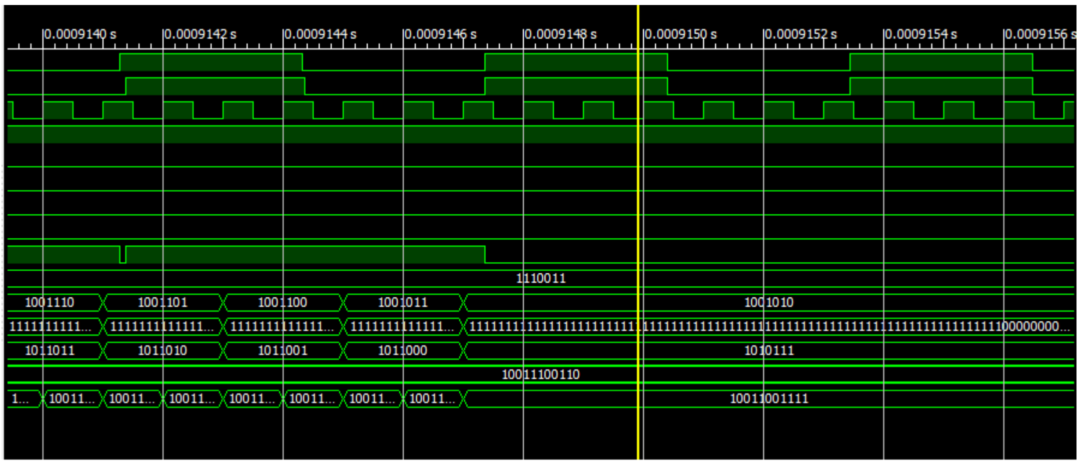


Figure 18: locked example