Frequency Tracking Using Digital Cavities

FREDRIK ZETTERBLOM

BACHELOR´S THESIS

DEPARTMENT OF ELECTRICAL AND INFORMATION TECHNOLOGY |
FACULTY OF ENGINEERING | LTH | LUND UNIVERSITY

Frequency Tracking Using Digital Cavities

Fredrik Zetterblom
mat12fze@student.lu.se

Department of Electrical and Information Technology
Lund University

Supervisors:
Fredrik Edman
Khadga Jung Karki
Liang Liu

Examiner: Erik Larsson

February 13, 2017

© 2017
Printed in Sweden
Tryckeriet i E-huset, Lund

Abstract

Digital cavities are efficient algorithms for comb filters with very low computational costs. They have been used to precisely measure the amplitude and phase of high frequency signals (few giga samples per second) from data acquired by high speed digitizers ($80\,\mathrm{GS\,s^{-1}}$). However, the previous measurements were done offline as the serial processors used in the analysis could not cope with the large data acquisition rate. In this project, we have implemented digital cavities in an FPGA architecture to measure amplitude and phase of signals in real time. Moreover, we have implemented multiple digital cavities that are tuned to slightly different frequencies, that process data in parallel. Based on the interpolation of the responses from each digital cavity, we have also developed algorithms which allows for tracking the change in frequency of the signal. These new advances are expected to be very useful in measuring the drifts in radio and microwave frequencies that are commonly used in FM broadcasting, frequency-shift keying, power systems, laser spectroscopy, synchrotrons, particle accelerators, etc.

Popular Science Summary

New method for detecting frequency shifts in a digitized signal

A new method for detecting frequency shifts in digitized signals has been investigated and developed during a bachelor thesis at Lund University. A real-time version of the method has been designed and implemented on a FPGA-platform with a high-speed digitizer.

Standing waves has long been used in different applications such as in instruments. The strings on a guitar or the air passing through cavities and holes in a flute causing sounds with different frequencies, are typical examples of this. The same principle is used in radio applications where cavities of different shapes and sizes give rise to different resonance frequencies based on constructive or destructive interference. In this bachelor thesis, this principle, and its applications, has been investigated for usage in the digital domain. This has been done by sampling the signal of interest signal with a fast digitizer and then folding it back onto itself with a delay, as shown in Figure 0.1, and thereby mimicking the behaviour of a cavity. If the frequency of the sampled signal corresponds to the length of the digital cavity (number of slots used by the cavity) constructive interference will form (shown in figure 1a). By investigating the values of the resulting signal in the cavity after a pre-determined number of fold backs, it is possible to determine the frequency of the sampled signal based on how well it fits the cavity. By mixing

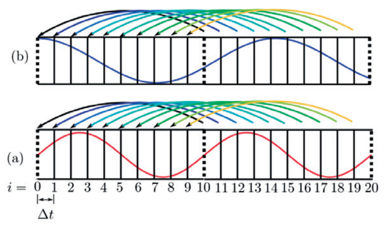


Figure 0.1: A visual representation of a cavity length of $n = 10$. The fold back in (a) forms a constructive interference, (b) forms destructive interference.

the sampled signal with different mix signals, and then feed them in to different digital cavities of the same length and resonance frequency, it is possible, by looking at which of the mixed signals that fits the best in the cavity, to determine the sampled signal with even greater accuracy. This is done by calculating the root mean square value of each cavity after the fold back, resulting in a cavity response. The larger response, the better fit. The frequency estimation is done by interpolating responses, knowing the corresponding resonance frequency for each response, resulting in a polynomial with a specific local max. The local max of the polynomial corresponds to the sampled signals frequency. The concept of using a digital cavity to determine a frequency with very high precision was confirmed by making measurements on signals generated by function generators and a laser that produced pulses at a rate of about 70MHz.

By making further development, this new method could have substantial benefits as a new way of determining the frequency, phase and amplitude of a sampled signal with high precision.

Nomenclature

λ	Wavelength	m
f	Frequency	Hz
L	Length	m
T	Wave period	s
t	Time	t
v	Speed	m/s

Table of Contents

List of Figures

Chapter 1

Introduction

Measuring the frequency of a signal is an important task in many applications such as in frequency-shift keying, FM broadcasting and in power systems[1]. In frequency-shift keying, a faster and more accurate estimations of the signal allows higher throughput of data, while in power systems, frequency estimation and tracking is used as a reference to how the system is performing. The problem can be described as [1]:

$$y_t = \mu + \rho \cos\left(\omega_0(t-\nu) + \phi\right) + \varepsilon_t, \tag{1.1}$$

where μ is the mean value, ρ is the signal amplitude, ω_0 is the frequency of the signal, $\nu = \frac{T-1}{2}$, ϕ is the initial phase and ε_t is some zero-mean random noise with variance σ^2. The problem is to estimate the variables μ, ρ, ω_0, ϕ and σ. Some proven methods to do this are: Maximum Likelihood Estimator of Frequency, Fourier Coefficients and Sample Covariance Methods. Maximum Likelihood estimation minimizes the error between a sampled signal and an estimated reference signal. Fourier techniques iterate through all complex coefficients and find phase and amplitude from the greatest coefficient. Frequency tracking of signals at radio-frequency is also very important in metrology, mainly in the precision measurement of frequency of optical signals. Today, it is common to use frequency combs generated by mode-locked lasers as the rulers for the absolute measurements of light signals[2]. A frequency comb is characterized by the comb spacing (f_r) and the carrier-envelope phase (ϕ_{ce}). Various types of fluctuations in the cavity of a mode-locked laser affect the f_r and ϕ_{ce}. Hence, in order to produce stable frequency combs, both the f_r and ϕ_{ce} have to be stabilized. f_r of a frequency comb is related to repetition rate of the mode-locked laser, and it is typically about 100 MHz. In the contemporary design of a servo-loop to control f_r one uses analog systems to measure f_r. Such a design is fast but error prone due to the electronic noise. Although, digital measurement of f_r can have low noise, it is still rather slow to be used in a real-time feedback loop. However, with the development of fast analog-to-digital converters and efficient algorithms it may be possible to measure f_r digitally such that they can be used to track the changes in radio-frequencies in real-time. This will allow us to implement more accurate real-time feedback loops to stabilize the comb spacing of the frequency combs. In general, frequency tracking of the radio-frequencies by digital techniques can be used in all the systems that require stable source of radio and microwave radiations, such as in particle accelerators, synchrotrons, etc.

The focus of this thesis is to implement digital cavities, which use one of the efficient numerical algorithms to measure frequency of a signal, in a field-programmable gate-array (FPGA) board. The implementation allows us to accurately measure frequency of a signal within few tens of milliseconds, which could be used in a feed-back control loop. The cavities are designed to measure signals with frequencies of tens to few hundreds of MHz.

Chapter 2

Background

A paper published in early 2013 named "Digital Cavities and Their Potential Use" [3] describes the possibility to achieve a high Q-factor for LIA (Locked In Amplifier) applications. Possible applications such as phase-sensitive signal detection, software defined radio and high resolution spectroscopy are presented.

In April 2014, a master thesis regarding this subject was produced at Lunds University by Siyuan Fu and Aohan Jin. In their report, they present different test setups where they analyze signals by using digital cavities. The thesis ascertains the functionality of digital cavities by presenting the result of analyses which process a high precision (64 bits) computer generated signal. The result showed that it is possible to achieve a Q-factor of 10^8 and a precision of 0.1 mHz by using 10^9 wave forms. They proceed by measuring the phase difference between two channels connected to the same source using an $80\,\mathrm{GS\,s^{-1}}$ digitizer where one of the transmission lines was cooled with ice. The result clearly corresponds with the Newton's Law of Cooling [4]. They were also able to detect the change in resistance caused by heating the transmission line between the source and digitizer by analyzing the amplitude variations [4]. However, these tests were done offline using a large amount of captured data (about 4 GBs of sampled data). Real time processing of such large amount of data is not feasible with a general purpose serial processor. Hence, in the thesis they also concluded that the method has to be improved and implemented in a parallelized computing architecture for real time applications such as radar/sonar systems, remote sensing, etc. Moreover, the implementation by Siyuan and Aohan lacked the features of frequency tracking using multiple digital cavities. One of the new concepts that has been developed since the thesis work is the use of interpolation to accurately determine the frequency of a signal. It is possible to significantly reduce the size of the data and yet achieve similar accuracy as compared to using a digital cavity on a full data set. These new concepts of implementing digital cavities in a parallelized computational architecture and interpolation to calculate the frequency of radio signals are described in this thesis.

Chapter 3

Theory

A rigorous description of digital cavity and associated concepts are given in Ref. 3. Here we motivate the functionality of a digital cavity based on the familiar concept of standing waves in mechanical strings.

Consider a string of length L_s that is attached to a vibrator in one end and to a fixed point in the other end. As the vibrator starts to move up and down with a fixed frequency, f_v, several waves are created that travels along the string towards the fixed point at a speed v_s. At the fixed point the waves gets reflected (phase shift by 180°) back towards the vibrator. If the return time, or the time period T_s, of the waves corresponds to the period of the vibrator T_v, the reflected waves will add to the newly produced waves by the vibrator, which then will results in the doubling of the amplitude. This is usually referred to as constructive interference. Wherein the vibrations in the string has reached a resonance frequency f_s (fundamental frequency) with corresponding wavelength λ_s, resulting in a standing wave. The standing wave has a maximum and a minimum at the center, as shown in figure 1 (n=1). If the vibration frequency is increased by a factor of two (mode: $n = 2$) of that fundamental frequency, the string forms two maxima and two minima. Continuing the increase of frequency by an integer factor will result in more maxima and minima. The amplitude is usually limited by damping effects, such as the loss of energy during the reflection and the air drag from the movement. If the traveling time and the period of vibration does not correspond, the vibrations cancel out due to the destructive interference with the reflected vibrations. This lowers the amplitude. [6]

$$v_s = f_s \lambda_s \tag{3.1}$$

$$\lambda_s = 2\frac{L_s}{n}, \quad n = 1, 2, 3, ... \tag{3.2}$$

$$T_s = \frac{1}{f_s}, T_v = \frac{1}{f_v} \tag{3.3}$$

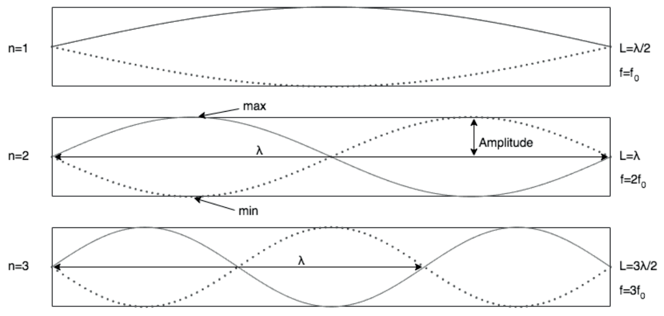


Figure 3.1: The three first modes of a standing wave

The concept of resonance in a string is useful in understanding the functionality of microwave cavities that are commonly used in signal processing. A cavity is used to select a certain frequency and to suppress the others. Unlike the movement of the string in the previous example, it is bouncing back and forth of the electromagnetic waves in the closed space of the cavity that form constructive or destructive interference. The dimensions of the cavity determines the resonance frequency. The resonance frequency for a rectangular cavity is given by [5]:

$$f_{mnl} = \frac{c}{2\sqrt{\mu_r \epsilon_r}} \sqrt{\left(\frac{m}{a}\right)^2 + \left(\frac{n}{b}\right)^2 + \left(\frac{l}{c}\right)^2}, \tag{3.4}$$

where c is the speed of light in vacuum, μ_r is the relative permeability and ϵ_r is the relative permittivity of the cavity filling. m, n and l are the mode numbers. a, b and c are the dimensions of the cavity.

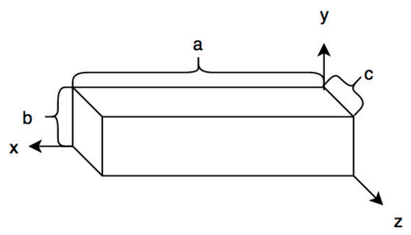


Figure 3.2: A rectangular cavity

Cavities are used as band pass filters, letting a certain frequency pass and blocking the others. This can be used to filter out unwanted signals from the desired ones. The benefit of cavity filters in RF-applications is their high Q-factor. Q-factor is calculated as the resonant frequency divided with the full width at half maximum (see Equ.3.5, the bandwidth Δf, where the power is half of the power at resonance frequency f_{res}). The value gives the bandwidth of a filter at a certain resonance frequency.

$$Q = \frac{f_{res}}{BW} = \frac{f_{res}}{\Delta f} \tag{3.5}$$

In this project we implement the principles of analogue cavity filters in the digital domain. We achieve this by sampling the signal at a certain rate (f_s) and folding back the digitized signal onto itself with a delay given by the length (n) of the cavity. The response y of the cavity is given by [3]:

$$y(j \cdot \Delta t; n) = \sum_{k=0}^{N_c} x \cdot (j \cdot \Delta t + k \cdot T) \quad j = 1, 2, 3, ..., n, \tag{3.6}$$

where x is the sampled signal,and $N_c + 1$ is the number of times the cavity function is applied to the signal, $\Delta t = 1/f_s$, $T = n \cdot \Delta t$ and n is the cavity length. The resonance frequency for the cavity is given by:

$$f_0 = \frac{1}{n \cdot \Delta t} = \frac{f_s}{n} \tag{3.7}$$

The cavity will be resonant for all integer multiples of f_0, including the DC signal. By considering Equ.3.4 in two dimensions (removing the z-dimension), letting $b \to \infty$ (b is not bounded to a finite value), consider the a-distance as a time interval: $a = n \cdot \Delta t$ and removing the constants, we get Equ.3.7.

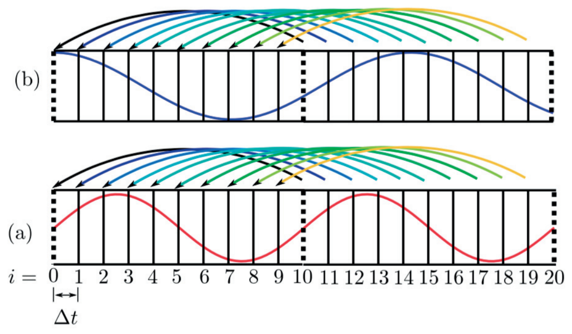


Figure 3.3: A visual representation of a cavity length of $n = 10$. The fold back in (a) forms a constructive interference, (b) forms destructive interference.

In a typical application of the digital cavity, the cavity length and the sampling rate is fixed, which results in a fixed resonance frequency. In order to make the signal resonante in the cavity, the the frequency of the signal has to be up-shifted or down-shifted to the resonance frequency by mixing with an appropriate frequency mix. If we assume a sinusoidal signal and $t \in [0, 2\pi]$, then

$$\begin{aligned} \cos\left(2\pi f_{sample} t\right) \cdot \cos\left(2\pi f_{mix} t\right) = \\ \frac{1}{2}\left(\cos\left(2\pi\left(f_{sample} + f_{mix}\right) t\right) + \cos\left(2\pi\left(f_{sample} - f_{mix}\right) t\right)\right) \end{aligned} \tag{3.8}$$

The frequency mixing produces two signals, one at the down-shifted frequency, $f_{sample} - f_{mix}$, and the other at the up-shifted frequency, $f_{sample} + f_{mix}$. Usually, f_{mix} is chosen such that the up-shifted frequency matches the fundamental resonance frequency, f_0 of the cavity. In this case the down-shifted frequency is guaranteed to be off-resonance with f_0 or its harmonics. In principle, one can also choose f_{mix} such that the down-shifted frequency matches f_0, however this does not ascertain that the up-shifted is not resonant with any of the harmonics of f_0. The frequency by which the signal should be up-shifted is then calculated as $f_0 = f_{sample} + f_{mix} \Leftrightarrow f_{mix} = f_0 - f_{sample}$.

If the mixed signal has a wavelength corresponding to the length of the cavity ($T = n \cdot \Delta t$), then constructive interference leads to the increase in the amplitude with the number of folds (N_c). A signal whose wavelength does not match the length of the cavity interferes destructively, thereby flattening out the response. The output of the digital cavity is a set of numbers x_i, where $i \in \{0, 1, 2, .., n-1\}$. We use the standard deviation given by:

$$s_n = \frac{1}{n}\sqrt{\sum_{i=0}^{n} \left(x_i - \overline{x}\right)^2} \tag{3.9}$$

as a measure of the amplitude of the cavity response, which also can be used to quantify how well the signal fits in the cavity. $\overline{x}$ in Equ.(3.9) is the mean of the different values of x_i, which is given by

$$\overline{x} = \frac{1}{n}\left(x_1 + x_2 + \ldots + x_n\right). \tag{3.10}$$

The value of s_n increases as $|f_0 - (f_{mix} + f_{sample})| \to 0$. The response of the digital cavity in the vicinity of the resonant frequency can be approximated by a sinc^2 function (see Figure 3.6). In order to find the exact frequency of the signal, we vary the frequency, f_{mix}, of the mixing signal and determine the $f_{mix-max}$ for which the cavity response is maximum. The frequency of the signal is then given by $f_0 - f_{mix-max}$. As the algorithms of digital cavity are highly parallelizable, we apply separate cavities for each f_{mix} simultaneously, which allows us to determine the exact frequency of the signal within few milliseconds of data acquistion. In practice, we use three cavities and three different mix signals, which gives us three points, $(f_1, a_1), (f_2, a_2), (f_3, a_3)$, where f_n is the frequency of the up-shifted signal that is fed into the digital cavity and a_n is the corresponding response. We usually choose f_2 close to the resonance frequency of the cavity, f_1 slightly below and f_2 slightly above. We interpolate the three points to a second degree polynomial and estimate the frequency from the maxima of the polynomial. The setup for such a system can be graphically described as in Figure 4.

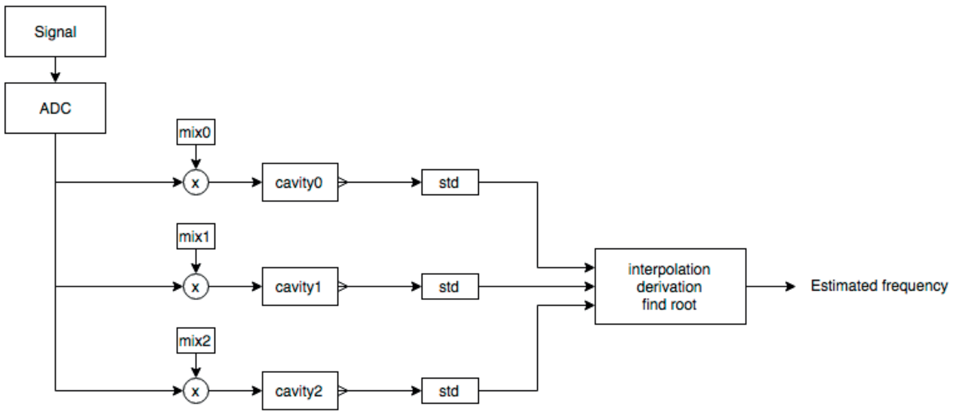


Figure 3.4: Graphic representation of finding the frequency

Some examples of the raw output of the digital cavity (Fig. 3.5) and the amplitude of the response of the digital cavities (Fig.3.6–3.8) are presented below.

To show that the theory in previous part work in practice, four examples will be presented in the following text. All examples was made ”offline” in MatLab.

Example 1

A signal is sampled at a rate of $1.25\,\mathrm{GS\,s^{-1}}$ (Giga Samples per second). The signal frequency is estimated to be $80\,\mathrm{MHz}$. The three mix-signals are set to $79.95\,\mathrm{MHz}$, $80\,\mathrm{MHz}$ and $80.05\,\mathrm{MHz}$. With the use of a cavity of length $n = 10$, the resonance frequency is $f_0 = f_s/n = 1.25\,\mathrm{GS\,s^{-1}}/10 = 125\,\mathrm{MHz}$.

The three mix signals are calculated as follows:

$$
\begin{aligned}
f_{mix_0} &= f_0 - 79.95\,\mathrm{MHz} = 45.05\,\mathrm{MHz}\\
f_{mix_1} &= f_0 - 80.00\,\mathrm{MHz} = 45.00\,\mathrm{MHz}\\
f_{mix_2} &= f_0 - 80.05\,\mathrm{MHz} = 44.95\,\mathrm{MHz}
\end{aligned}
$$

The response from the three cavities when applying the digital cavity algorithm to the signal is plotted in Figure 3.5.

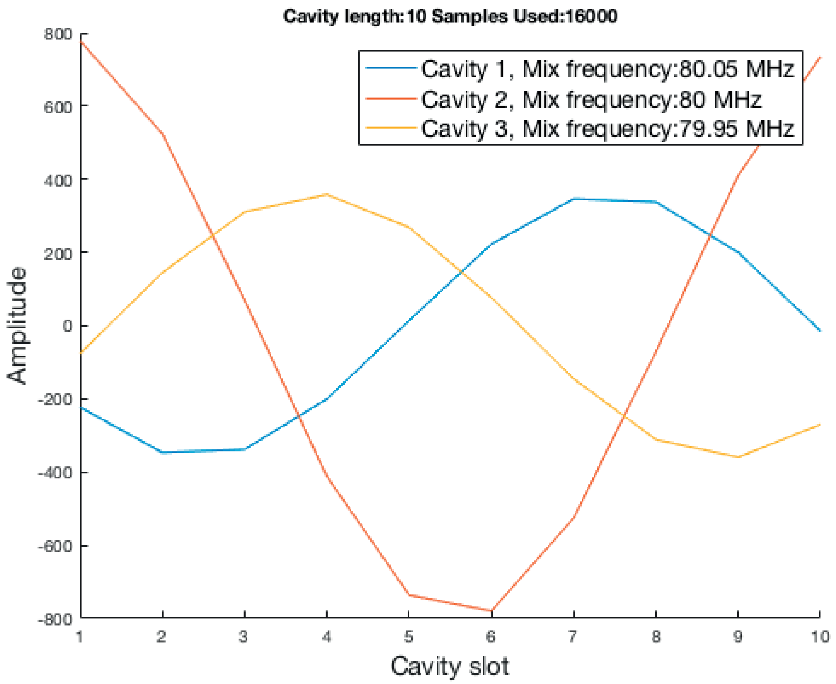


Figure 3.5: The response from three cavities using different mix signals

The square of the amplitude (the responses) are:

$$\begin{aligned} a_0(\text{blue}) &= 268.3959 \\ a_1(\text{read}) &= 595.9486 \\ a_2(\text{yellow}) &= 268.6110 \end{aligned}$$

The response for the second response gives the greatest value. This is expected since the second cavity uses a mix signal that transform the sampled signal to be resonant with the cavity.

Example 2
An 80 MHz signal is multiplied with 400 different mix-signals making cavities resonant for frequencies ranging from 79.8 MHz to 80.2 MHz. The standard deviation of each cavity response is plotted in figure 3.6.

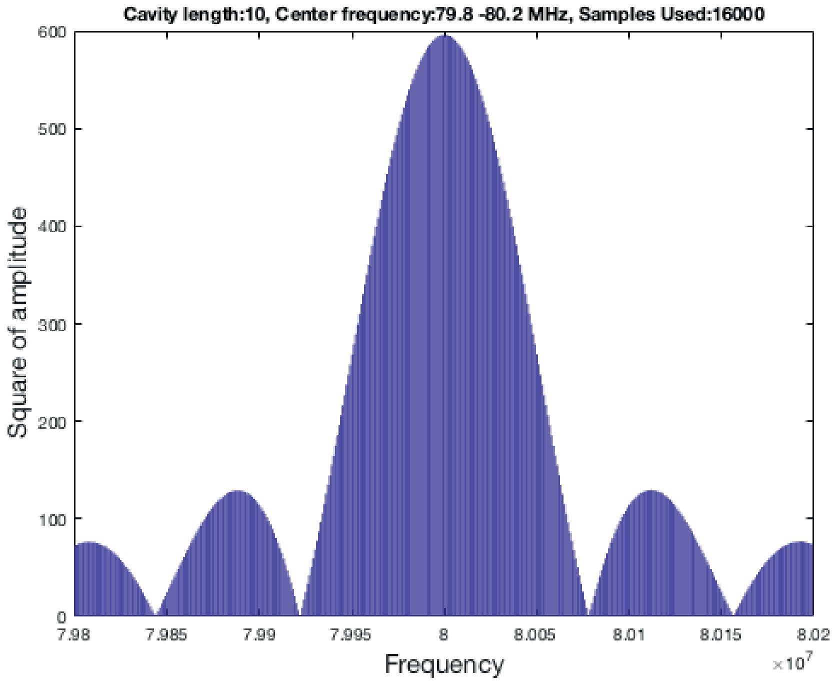


Figure 3.6: The response from 400 cavities using different mix signals

Max amplitude is 600 at 80 MHz, half the amplitude is there for 300 with corresponding frequencies 79.953 MHz and 80.047 MHz. The Q-factor for this specific application is $Q = \frac{80\,\text{MHz}}{(80.047\,\text{MHz}-79.953\,\text{MHz})} \approx 851$.

Next example will show how the frequency of the sampled signal can be estimated by choosing the response of three digital cavities.

Example 3

The frequency of the signal is now shifted to 80.02 MHz, all the other parameters is the same as previous examples. By choosing the three x-values within the center peak, a second degree polynomial fits fairly well to the responses of the cavities as shown on Figure 3.7. The root of the derivative of the polynomial gives a estimation of the signals frequency. By reducing the number of cavities to three, the number of calculations is reduced sufficiently.

The three responses corresponds to: $f_0 = 79.95$ MHz, $f_1 = 80.00$ MHz, $f_2 = 80.05$ MHz. The method gives a good estimation if the signal is shifting within half of the frequency difference between f_1 and f_0 or f_2.

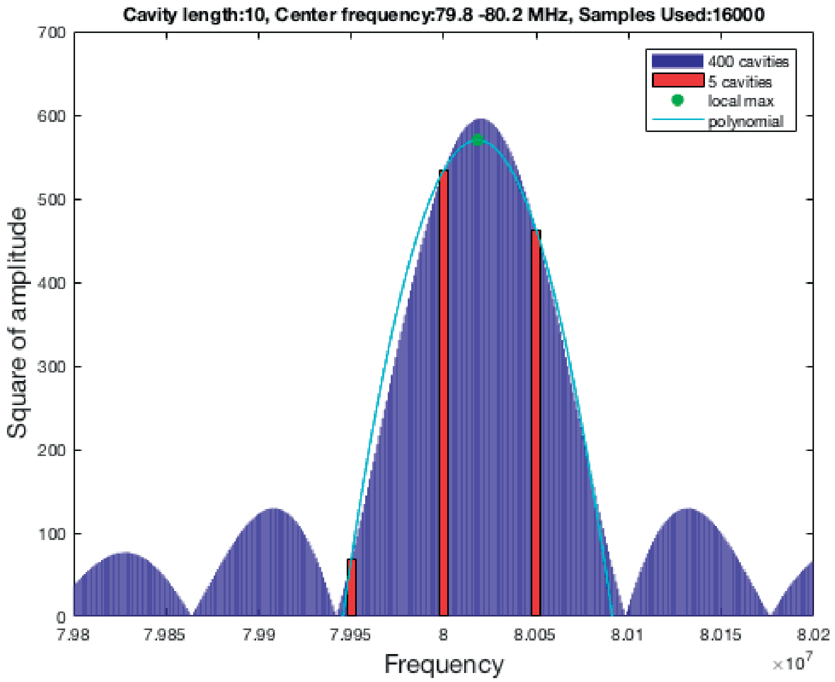


Figure 3.7: Shifted signal, the green dot marks the max of the interpolated polynomial

The three cavities corresponds to following three points:

$$(f_0, a_0) = (79950000, 68.25380)$$
$$(f_1, a_1) = (80000000, 533.8123)$$
$$(f_2, a_2) = (80050000, 461.6987)$$

The polynomial and its derivative's root corresponding to the points is calculated to:

$$\begin{aligned} f(x) = -1.0753 \cdot 10^{-07}x^2 + 17.209x - 688534379 \\ f'(x) = -2.1507 \cdot 10^{-07}x + 17.2094 \\ 0 = f'(x) \Leftrightarrow x \approx 80018294 \approx 80.02\,\text{MHz} \end{aligned}$$

The green dot in Figure 3.7 correspond to local maxima of the polynomial, i.e the estimated frequency.

This example show that it is possible to estimate an unknown frequency by first making an assumption fairly close to the actual frequency and then make the cavities resonate around this frequency. The estimated frequency from interpolation will have an error that increase with the signals offset from the assumed frequency, showed later in this section. In this case the error is $80020000 - 80018294 = 1706\,\text{Hz}$.

Next example will show how this error can be reduced.

Example 4

In order to get a better estimation of the signals frequency, two additional cavities are implemented. Five digital cavities produce five responses. By interpolating five points (frequency, response), a fourth degree polynomial is produced. The frequency is estimated by finding the local max of the polynomial as shown in the MatLab simulation in Figure 3.8.

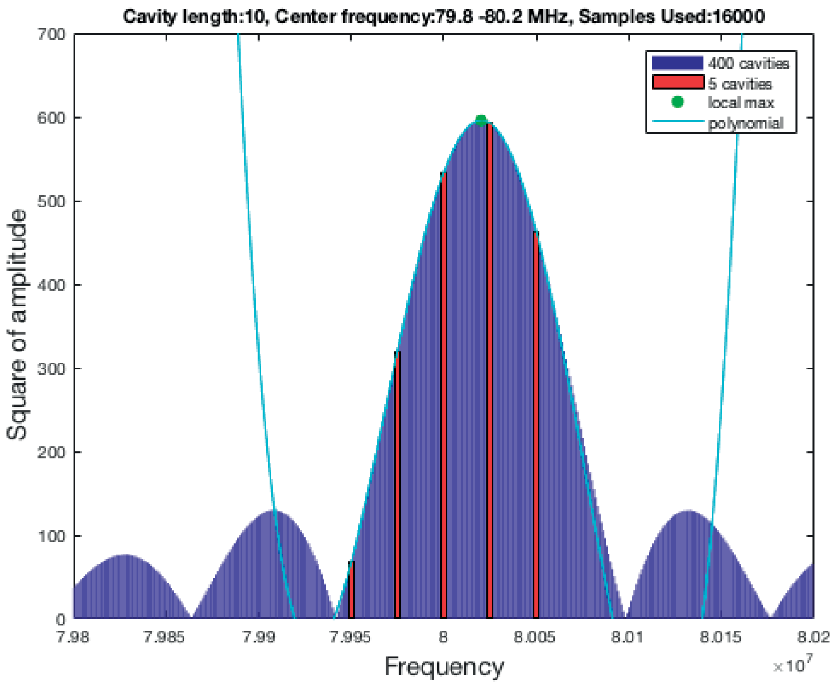


Figure 3.8: Five digital cavities and interpolation. The cavities are centered at 80 MHz, the actual signal is 80.02 MHz.

The result from this simulation gives an error of 237 Hz, which is almost 1/7 of the error in example 3.

Conclusion from examples

Comparing example 3 and 4, using five cavities results in a better estimation. Using more cavities gives a even better estimation, the downside is the requirement of more computing power. When implementing on a FPGA, more computing power corresponds to more use of logic blocks within the chip, which is limited.

Errors

In order to get an understanding of the error caused by interpolation of the responses from five cavities, a simulation was made in MatLab showing the absolute value of the difference between estimated signal and actual signal frequency, Figure . The signal varies from 79.97 MHz to 80.03 MHz, the cavities are fixed to 79.95 MHz, 79.975 MHz, 80 MHz, 80.025 MHz and 80.05 MHz.

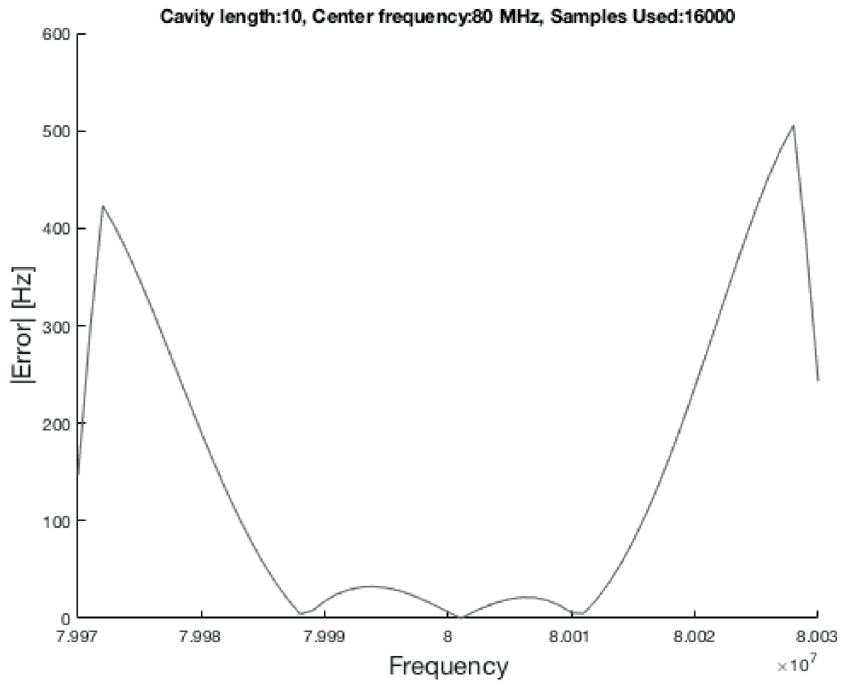


Figure 3.9: Error caused by estimation by interpolation

The error is related to the points used in the interpolation. If the frequency of the sampled signal is close to resonance frequency of the center or one of the closest cavities, the error will be small. If the frequency drifts from the estimated center, the error will increase. This is due to the fact that the points used for interpolation will not form a polynomial corresponding to the peak of the frequency as showed in example 2, Figure 3.6.

Chapter 4

Implementation

4.1 Hardware & Software

In order to capture and process data at high speed the 4DSP's FMC125 and Xilinx's KC705 evaluation board featuring the XC7K325T-2FFG900C FPGA was used.

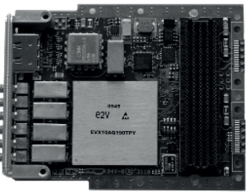


Figure 4.1: 4DSP FMC125

The FMC125 is a Quad-Channel Multi-Mode A/D FMC. The card provides four 8-bit ADC channels that enable simultaneous sampling of 4, 2, or 1 channel with a maximum sample rate of $1.25\,\mathrm{GS\,s^{-1}}$ (4-channel mode), $2.5\,\mathrm{GS\,s^{-1}}$ (2-channel mode), or $5.0\,\mathrm{GS\,s^{-1}}$ (1-channel mode). In this project the default setup of 4-channel mode at $1.25\,\mathrm{GS\,s^{-1}}$ rate was used.

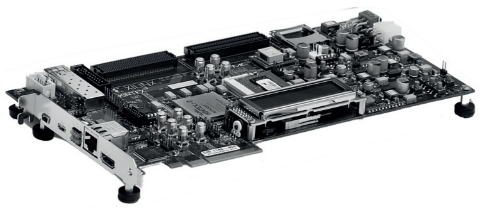


Figure 4.2: Xilinx Kintex-7 FPGA KC705 evaluation board

Xilinx's FPGA programming tool Vivado Design Suite features "Block Design" with pre-designed blocks and the possibility to design custom blocks. To be able to use the FMC125 in the Vivado block design, a block was generated from 4DSP's software StellarIP. The block supports Xilinx AXI-bus standard, which makes the

FMC125 easy to implement to a design in Vivado. A MicroBlaze soft core was used to control the system using C++ code that triggers different functions, calculates a look-up-table (LUT), initiates the FMC125, etc. inside the FPGA.

4.2 MicroBlaze and GPIO

Xilinx Vivado provides a simple implemention of the 32 bit MicroBlaze soft microprocessor core. By using AXI-GPIO blocks (Advanced eXtensible Interface, General-Purpose Input/Output) that connects with the soft core processor, it is possible to control other I/O parts of the system using C++ code. Each AXI-GPIO block is capable of handling two channels configured to be either outputs, inputs or both using 1 to 32 bits. After compiling and creating the bit-file used to program the FPGA, it is possible to export the project and program the soft core using Xilinx SDK.

Each GPIO is declared in the code which makes reading or writing custom IO possible:

```
#include "xgpio.h"
#include "xgpio_l.h"

XGpio GPIO_0;
XGpio_Config GPIO_0_conf;

int main(){
  GPIO_0_conf.BaseAddress = XPAR_AXI_GPIO_0_BASEADDR;
  GPIO_0_conf.DeviceId = XPAR_AXI_GPIO_0_DEVICE_ID;
  GPIO_0_conf.InterruptPresent = XPAR_GPIO_0_INTERRUPT_PRESENT;
  GPIO_0_conf.IsDual = XPAR_GPIO_0_IS_DUAL;
  XGpio_CfgInitialize(&GPIO_0, &GPIO_0_conf,
  GPIO_0_conf.BaseAddress);

  int data = 0;
  XGpio_DiscreteWrite(&GPIO_0, 1, 0x00000001);
  //write "1" to GPIO_0, channel 1
  data = XGpio_DiscreteRead(&GPIO_0, 2);
  //read value from GPIO_0, channel 2 and store to variable "data"

  return 0;
 }
```

Using UART (Universal Asynchronous Receiver/Transmitter) to communicate between the MicroBlaze and a PC, data can be processed and presented using MatLab (or any other application preferred) running on the PC.

4.3 Generate LUTs

In order to make the signal resonant with the cavities, it has to be multiplied by the mix signals that are generated by the soft core using "math.h" which includes the cosine-function. The values have to be pre-generated with the same sample rate as the data acquisition rate used in the FMC125, i.e. $1.25\,\mathrm{GS\,s^{-1}}$ in this configuration. The values are stored in separate memory blocks.

The cosine-results are multiplied with 127, which result in signed 8-bit cosine-LUTs. By using pre-generated values, it is possible to read the memories at the same clock rate as the rest of the system (100 MHz). Direct generation of mix-signals using the soft core does not cope with the high rate of data acquisition by the FMC card, hence the LUTs are necessary for the design. The following C++ code generates three LUTs and stores the values at the same address as an incrementing variable using the GPIO connected to the RAMs.

```
        //---Generate LUT
        XGpio_DiscreteWrite(&GPIO_1, 2, 0x00000001); //WEA = 1
        for(int i = 0;i<nbr_of_samples;i++){
                XGpio_DiscreteWrite(&GPIO_0, 1, i); //addr LUT0
                XGpio_DiscreteWrite(&GPIO_2, 1, i); //addr LUT1
                XGpio_DiscreteWrite(&GPIO_3, 1, i); //addr LUT2
                XGpio_DiscreteWrite(&GPIO_0, 2, round(127*cos(m0*i)));
                XGpio_DiscreteWrite(&GPIO_2, 2, round(127*cos(m1*i)));
                XGpio_DiscreteWrite(&GPIO_3, 2, round(127*cos(m2*i)));
        }
        XGpio_DiscreteWrite(&GPIO_1, 2, 0x00000000);//WEA = 0
```

The values $m0$, $m1$, $m2$ in the code are generated for the desired resonance frequency of each cavity. The values are generated in MatLab since MicroBlaze soft core can not handle divisions by large numbers. The values are then multiplied with the incrementing values that ranges from zero to the number of samples used in the configuration.

```
center_f = 80E6;
off=.05E6;
sample_rate = 1.25E9;
cavity_length = 10;
dt=1/sample_rate;
f0 = ((sample_rate / cavity_length) + (center_f-off));
f1 = ((sample_rate / cavity_length) + center_f);
f2 = ((sample_rate / cavity_length) + (center_f+off));
m0 = 2*pi*f0*dt
m1 = 2*pi*f1*dt
m2 = 2*pi*f2*dt
```

A signal (GPIO 1, channel 2) sets write enable to high during the generation, when done its turns low. This signal is inverted and connected to read enable, making it only possible to write to the RAMs during generation of the LUTs.

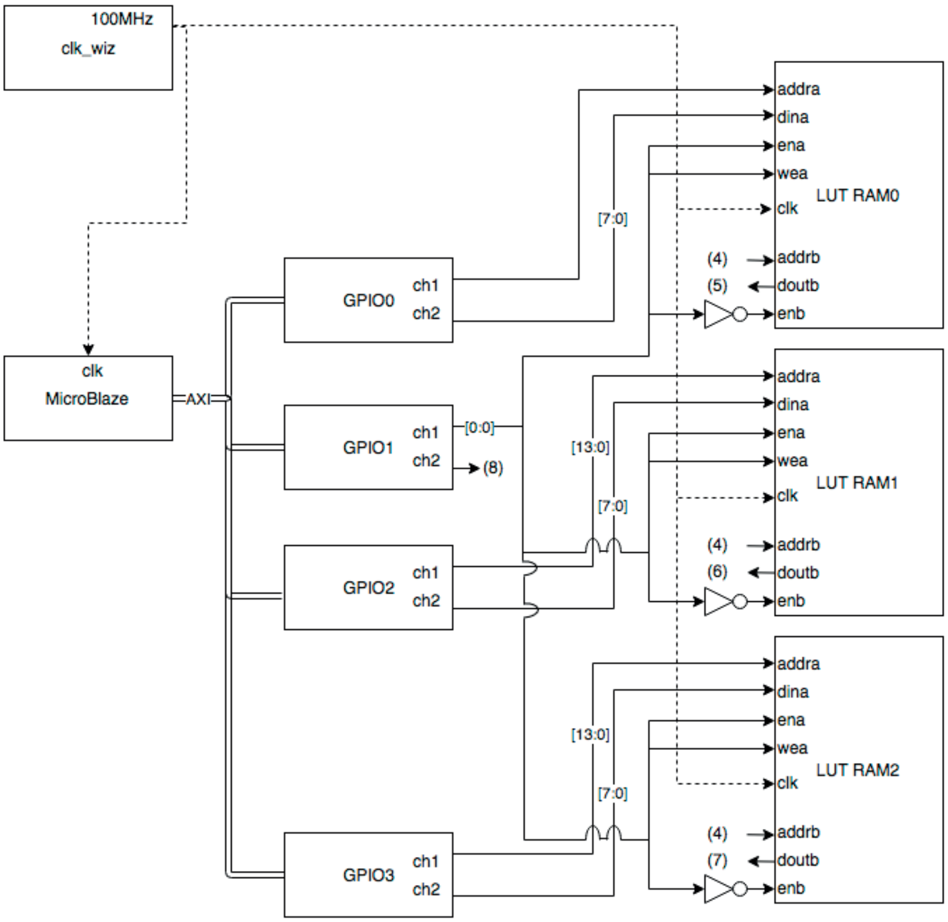
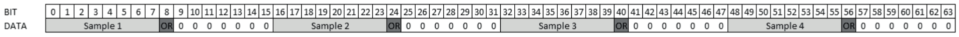


Figure 4.3: Setup of LUTs

4.4 Data acquistion

The FMC125 can be reconfigured to capture 1k, 2k, 4k, 8k or 16k samples. The samples are provided as 64-bit vectors, each vector containing four unsigned 8-bit samples.



BIT	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
DATA	Sample 1								OR	0	0	0	0	0	0	0

BIT	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
DATA	Sample 2								OR	0	0	0	0	0	0	0

BIT	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
DATA	Sample 3								OR	0	0	0	0	0	0	0

BIT	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
DATA	Sample 4								OR	0	0	0	0	0	0	0

Figure 4.4: 64-bit vector from FMC125

In order to read out each sample one at a time, the 64-bit vector is stored in a FIFO memory (First In First Out). The FIFO starts sending data when“rd_en” is active, which is controlled by GPIO 4, channel 1. Data is extracted at 1/4 of the clock frequency (25 MHz) into a custom made multiplexer running at the same clock as the rest of the system (100 MHz), extracting one bit at a time from the vector.

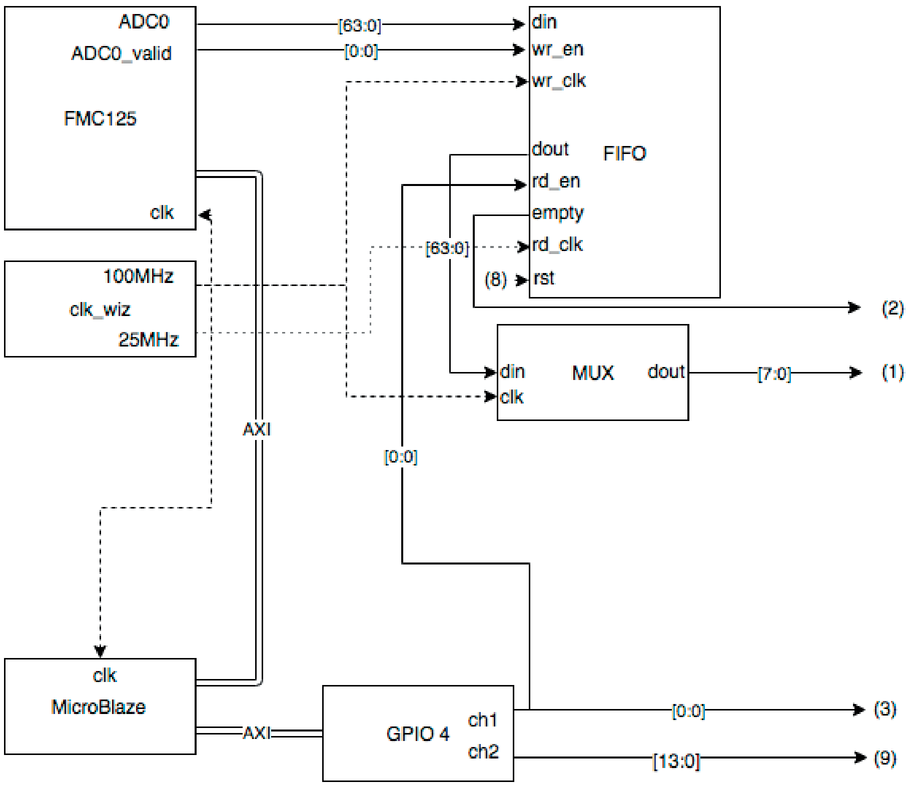


Figure 4.5: Setup of FMC125

4.5 Digital cavity

When the GPIO signal is set to high and if the FIFO has values stored (it is not empty), the cavities get enabled and a counter starts incrementing from 0 to 16384. The counter is connected to the address port of the LUT-RAMs, making the LUT-values available. The LUT-values and the sampled data from the FIFO are multiplied using a multiplication block. The product from each multiplication is sent to each corresponding digital cavity. The cavities perform the fold back loops. The process continues until the number of samples set in the C++ code is reached.

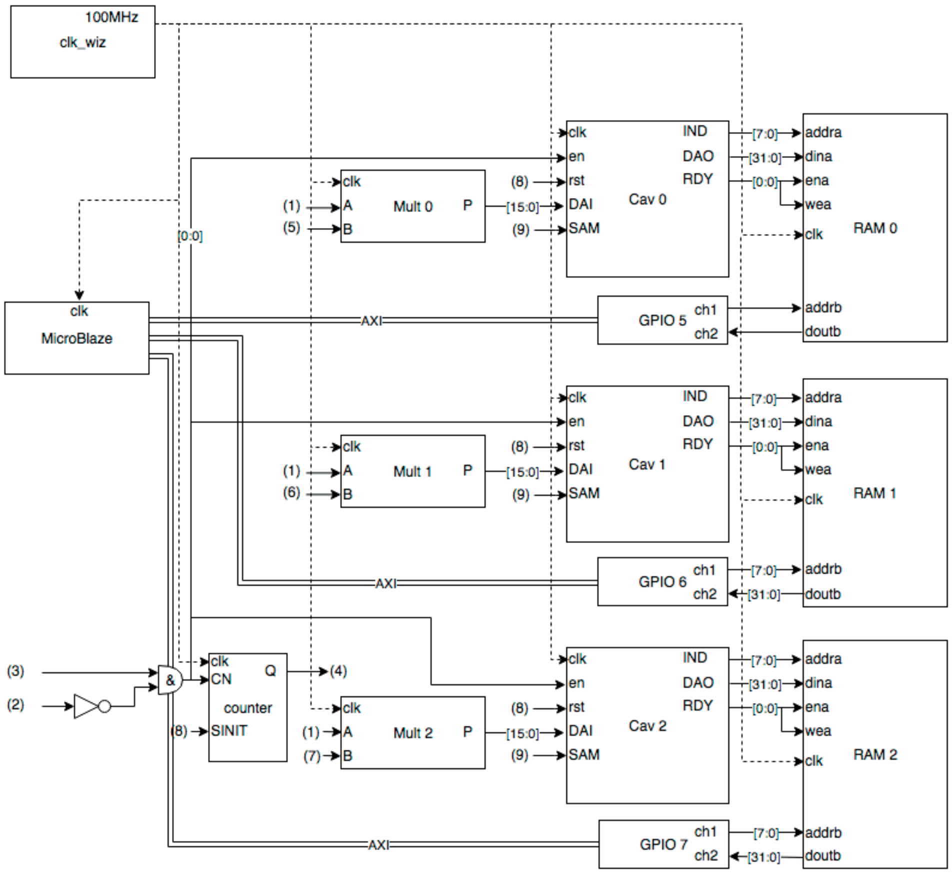


Figure 4.6: Setup of cavities

When finished,“RDY” is set to high and each value from the response is active on “DAO” at the same time as an incrementing index is active on “IND”. Each response value is stored in the RAMs at the same address as the index. The soft core then reads the RAMs and the standard deviation is calculated based on each cavity response.

The cavity block works as a state machine having three “main” states: reset, fold back and output result. There are two vectors used for the fold back, one is used for temporary values and the other stores the actual response.

During each clock cycle, the “DAI” is read and the value is stored as an signed integer. During the fold back-state, another state machine is used to update the response values. For a 10 slot cavity there are 10 states. Each state updates one slot per clock cycle. When the last slot is updated, a counter is set to zero and the fold back starts over from the slot zero. This continues until the counter reaches the value set in the C++ code, making the cavity go to next main state; output. During the output state, the values from each slot is active on the “DAO” together with the slot’s index on “IND”. “RDY” is active during output indicating that the result is available. When the last slot value has been sent out, the cavity is put to reset during the last main state. If “rst” is set high, all the vectors are erased and set to zero. Figure 4.7 shows a visual representation of the digital cavity.

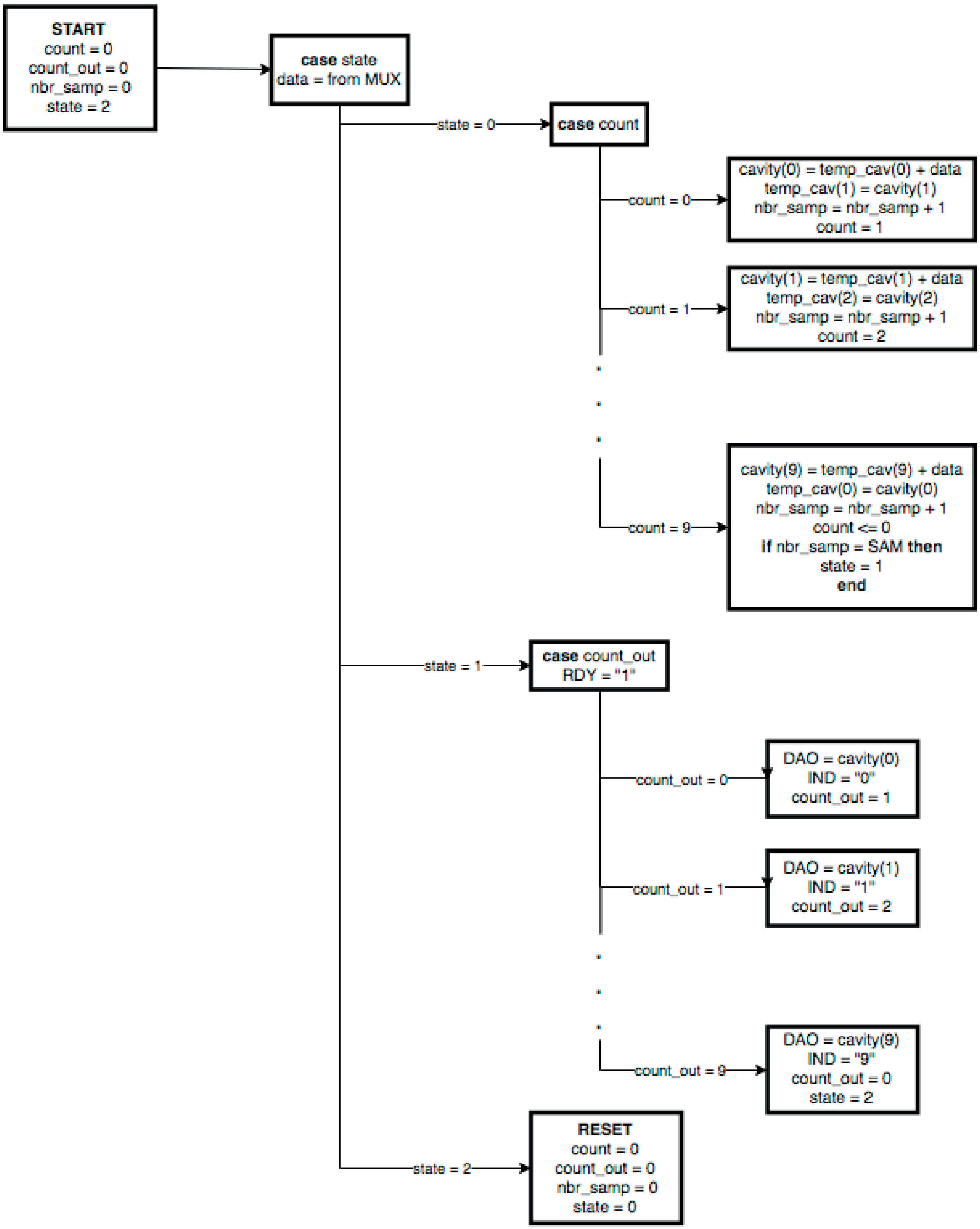


Figure 4.7: Simplified work flow of a digital cavity

4.6 Running the system

The setup is controlled from the MicroBlaze soft core. The code initiates the GPIO blocks and sends instructions to the FMC125's setup register. The same number of LUT values as number of samples used in cavities are calculated and sent to the RAMs. This step takes around two or three seconds to perform (calculating $3 \cdot 16$k values). When done, an infinite while loop is entered. The loop first clears the cavities and the FIFO, then triggers the FMC125 to capture 16k samples. Cavities and the FIFO get enabled and the multiplication and fold back operations are performed by the FPGA logic. The soft core is put to "sleep", waiting for the cavities to process all the samples. Responses from each cavity is then read from RAMs. The standard deviation of each response is calculated and concatenated to a string that is sent via UART.

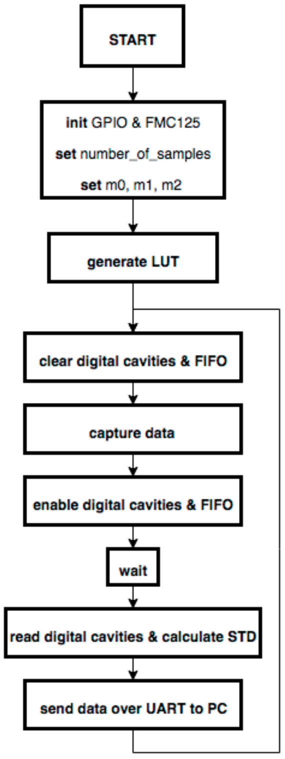

START
init GPIO & FMC125
set number_of_samples
set m0, m1, m2
generate LUT
clear digital cavities & FIFO
capture data
enable digital cavities & FIFO
wait
read digital cavities & calculate STD
send data over UART to PC

Figure 4.8: Work flow of soft core

The final setup in Vivado block design is shown in Figure 4.9.

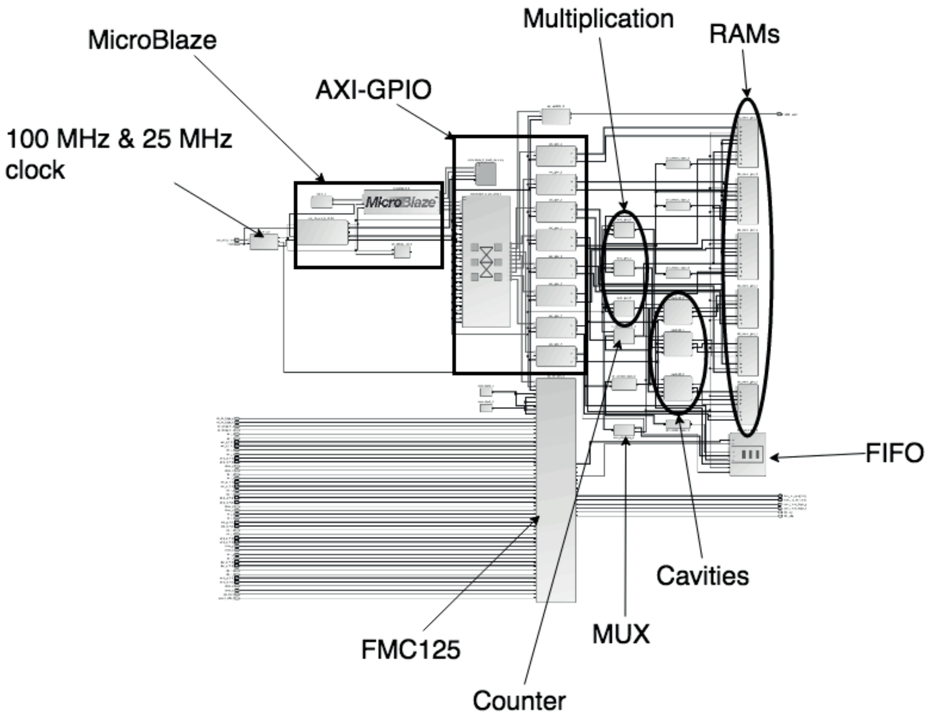


Figure 4.9: Final block design in Vivado

Chapter 5

Measured results

5.1 Frequency measurement using three cavities

A test using the KSG4100 signal generator from Kikusui was made in order to verify the systems functionality. The signal generator was set to output a 80 MHz signal with an amplitude of 250 mV. The three values $m0$, $m1$, $m2$ were generated as described in section 4.3 and inserted to the C++ code running in the MicroBlaze soft core.

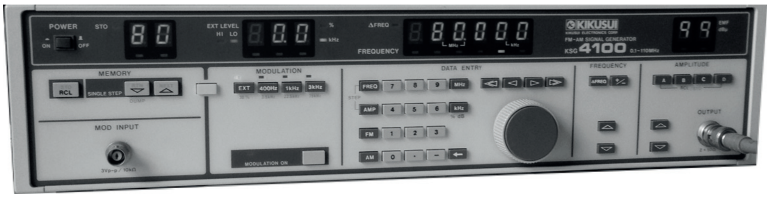


Figure 5.1: KSG4100 generating a $80\,\mathrm{MHz}$ signal

The frequency determined by the digital cavies was sent to a PC and visualized in a MatLab plot. The plotted result is the mean value of the last 100 data points sent from the soft core. The plot changes as new values are calculated from the data received, sweeping from left to right, updating the plot and values.

As shown in Figure 5.2, the result is fluctuating near 80 MHz.

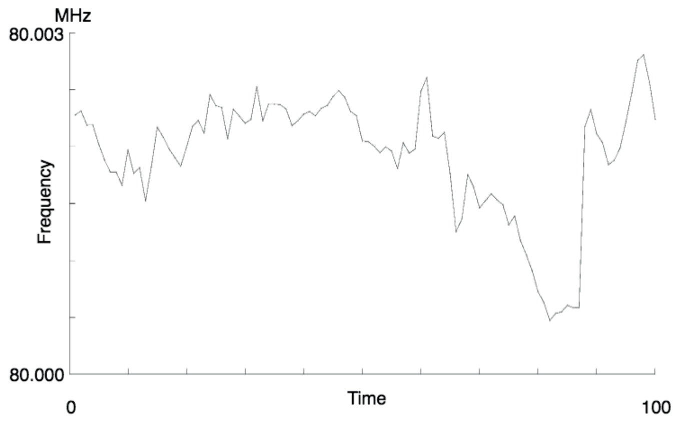


Figure 5.2: Result shown in MatLab

In order to test the device, we changed the frequency of the signal to 80.02 MHz. The change of frequency was clearly seen in MatLab. Values increased from 80 MHz to 80.02 MHz as the update sweeps over the plot. The results are shown in Figure 5.3. The last points in the figure show the frequency stablizing to 80.02 MHz.

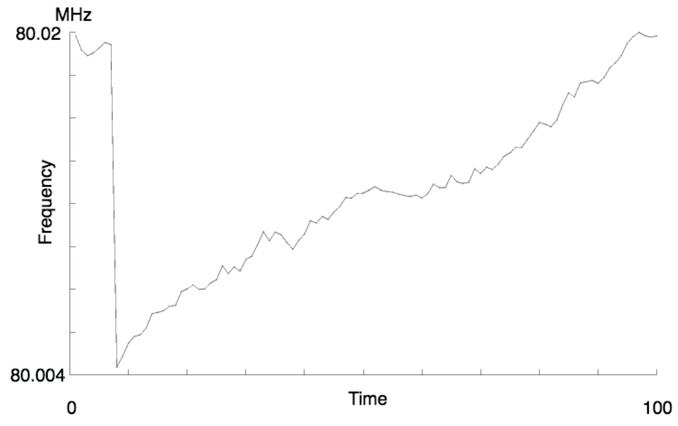


Figure 5.3: New result shown in MatLab

5.2 Frequency measurement using five cavities

The system was set to detect a frequency centered at 50 MHz using five digital cavities with equidistant spacing of 25 kHz. The responses were captured and plotted using MatLab as shown in Figure 5.4.

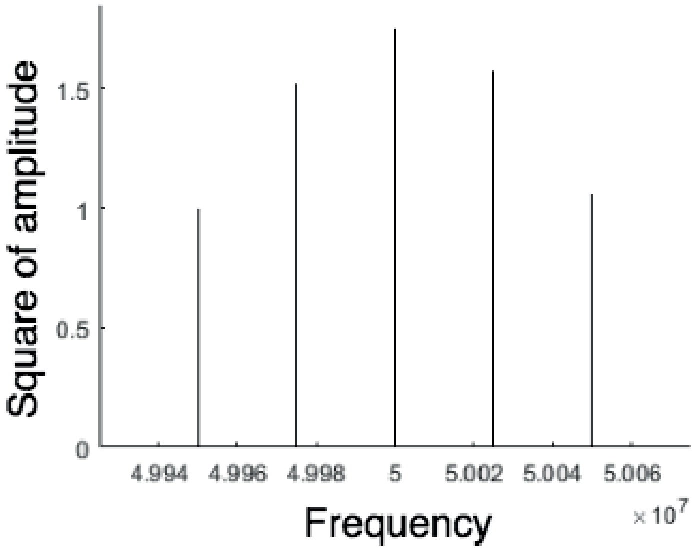


Figure 5.4: Responses from cavities centered at 50 MHz

The result clearly show that the digital cavity resonant with 50 MHz has the greatest amplitude.

5.3 Measurement of a pulsed laser

A femtosecond oscilator, SynergyTM, from Femtolasers was used. The laser produced pulses with a duration of about 10 fs and at a rate of about 70 MHz. The repetition rate varied slightly because of fluctuations in the temperature and air turbulence in the laser cavity. A DET10A/M detector from Thorlabs was used to detect the individual pulses from the laser.

To get an initial value of the pulse rate, a FTT (fast Fourier transform) was done on the captured data extracted from the ADC (see Figure 5.5).

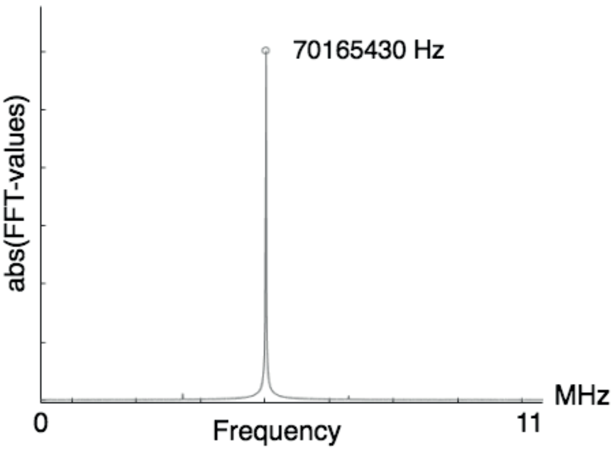


Figure 5.5: FTT of captured data

The digital cavities were configured to center at 70 165 430 MHz. The frequency of the signal was estimated from the polynomial interpolation of the responses.

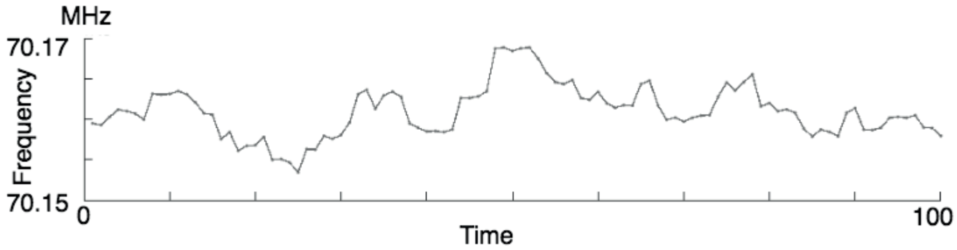


Figure 5.6: 100 points calculated from interpolation of cavity responses

Seen in Figure 5.6, the tracking of the variation of pulse rate without any averaging seems to vary around 70.16 MHz.

Chapter 6

Conclusions

Methods used in this project show that it is possible to estimate the frequency of a signal using digital cavities in real time.

The Q factor is related to how many samples that are used in the fold back, more samples gives a higher Q. In example two the Q factor is calculated to 851, which is fairly low in terms of filters. This is not that bad for this application, if the signal is drifting a lot, having a low Q-factor allows for more spread cavities and a wider detection range.

By using an FPGA, it is possible to create tailored solutions that is able to calculate results much faster than using a program running on a PC. The multiplications and fold back operations in the digital cavities used in this project are all processed in parallel.

By using the standard configuration of the ADC (FMC125) it is only possible to capture a limited amount of samples each time the ADC is triggered. The manufactures, 4DSP, claim that it is possible to do continuous sampling by making a reconfiguration of the ADC using their provided tools. The system setup described in this project will miss an amount of data during evaluation of the responses from the digital cavities, the signal is not sampled during this time. Continuous sampling would benefit to accuracy of the result, all the data would be accounted for rather than the current setup where only parts of the signal is used for frequency estimation. If continuous sampling were to be used, the system clock would have to run at the same rate as the sampling frequency. That is not possible for the FPGA used in this project.

In Figure 3.6, the response from multiple cavities is shown. The responses look more like a modification of a $\text{sinc}^2(x)$ function. During this project no further efforts was made to investigate the best interpolation option. Polynomial interpolation using five points gave a fairly good estimation of the frequency. Using higher order polynomial interpolation with more points equally divided around a center would increase accuracy. Similarly, using a narrower frequency range when choosing the points increases the accuracy of the estimation, however, this decreases the possible detection range. The FPGA's amount of logic sets a limit to how many cavities can be implemented in the design.

Figure 5.2 shows fluctuations in the estimated frequency. There are many things in the setup that could cause this. For example, the signal generator has not been calibrated since it was bought according to the LTH lab staff. The FMC125 ADC clock is affected by temperature that affects the sampling accuracy.

During the final weeks of this project, measurement of two channels and calculation of phase difference was implemented. The phase difference was calculated using the following equations:

$$\varnothing = \tan^{-1}\left(\frac{x_1 \cdot \sin\left(\frac{2\pi}{n}\right)}{x_2 - x_1 \cdot \cos\left(\frac{2\pi}{n}\right),}\right) \tag{6.1}$$

where x_i is the value of slot i in the digital cavity after folding, n is the cavity length. The phase difference in degrees between the two channels is given by:

$$\left|\varnothing_{channel1} - \varnothing_{channel2}\right| \cdot \frac{360}{2\pi} \tag{6.2}$$

The accuracy of real time phase estimation was not fully tested.

Chapter 7

Possible improvements

Calculating the frequency by fitting a sinc^2 function to the responses of the cavity could increase the accuracy, however this method could be slower.

If faster method of generating LUTs could be implemented it would be possible to "lock" on to the peak of the cavity responses and track the frequency more accurately.

A faster FPGA (allowing higher clock frequency) would make it possible to design a faster system.

The use of more data/samples would increase the Q value, making the result even more accurate, but at the same time reduce the range of possible detection of frequency.

More effective ways of implementing the digital cavity algorithm is possible. A sliding window that accepts a continuous stream of samples would increase speed of the result, rather than as done in this project where samples are captured and then calculations are made. By using a sliding window, a result would always be available.

A possible implementation would be that each slot in the cavity contains a certain amount of samples and the sum of all values in the slot are continuously calculated and sent to the user, as shown in Figure 7.1.

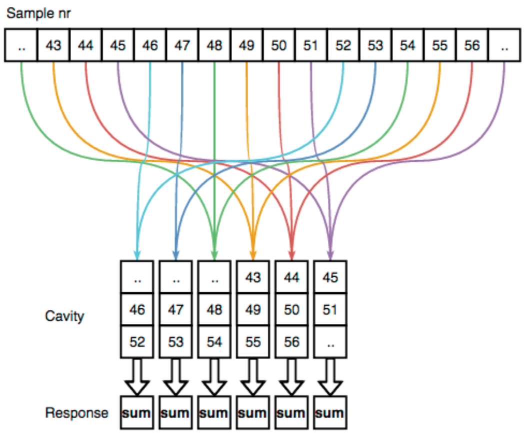


Figure 7.1: Visualization of a digital cavity using sliding window

Yet other approach might be to use only sums. Each sample is added to a sum in the digital cavity. The sample is then delayed in a FIFO and then subtracted from the the same sum as it was added. The delay is determined by the amount of samples desired for the fold back. This would also make it possible to always have a result available. A suggested implementation is visual represented in Figure 7.2.

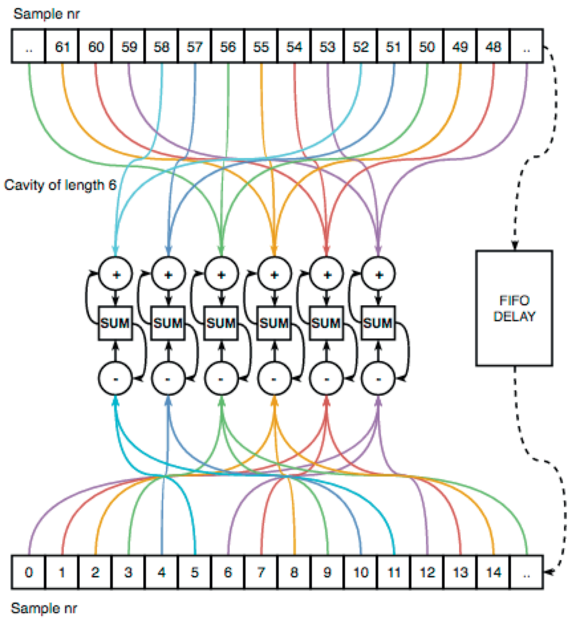


Figure 7.2: Visualization of a digital cavity using sums

Chapter 8

References

[1] P.J. Kootsooks
A Review of the Frequency Estimation and Tracking Problems
`http://espace.library.uq.edu.au/view/UQ:10626/comparison-t.pdf`
[21 Feb 1999]

[2] Y. Jun and S. Cundiff
Femtosecond Optical Frequency Comb: Principle, Operation and Applications
(New York: Springer) 2005.

[3] K. Karki, M. Torbjornsson, J. R. Widom, A. H. Marcus and T. Pullerits
Digital Cavities and Their Potential Applications, J. Instrum. **8**, T05005 (2013).

[4] A. Jin, S. Fu, A. Sakurai, L. Liu, F. Edman, T. Pullerits, V. Öwall and K. J. Karki
Note: High precision measurements using high frequency gigahertz signals, Rev. Sci. Instrum. **85**, 126102 (2014).

[5] D. M. Pozar
Microwave Engineering, 4th Edition
`ISBN-13: 978-0470631553, ISBN-10: 0470631554`

[6] P. A. Tipler and G. P. Mosca
Physics For Scientists And Engineers Mechanics, Oscillations And Waves, Thermodynamics, 6th Edition
`ISBN-13: 978-1429201322`

Printed by Tryckeriet i E-huset, Lund 2017

LUND
UNIVERSITY

Series of Bachelor’s theses
Department of Electrical and Information Technology
LU/LTH-EIT 2017-558

http://www.eit.lth.se