HARDWARE IMPLEMENTATION OF TIME FREQUENCY TOOLS FOR POWER QUALITY APPLICATIONS

By
Iván J. Rivera-Lebrón

A thesis submitted in partial fulfillment of the requirements for the degree of

MASTER IN SCIENCE
in
ELECTRICAL ENGINEERING
(Digital Signal Processing)

UNIVERSITY OF PUERTO RICO
MAYAGÜEZ CAMPUS
2007

Approved by:

_____	_____
Efrain O'Neill, PhD Member, Graduate Committee	Date
_____	_____
Rogelio Palomera, PhD Member, Graduate Committee	Date
_____	_____
Domingo Rodríguez, PhD President, Graduate Committee	Date
_____	_____
Omar Colón,PhD Representative of Graduate Studies	Date
_____	_____
Isidoro Couvertier, PhD Chairperson of the Department	Date

ABSTRACT

This work deals with the analysis, design, and implementation of efficient algorithms for time-frequency signal analysis. We propose the design and development of an integrated hardware system for the automated processing of sensor-based signal data information generated in industrial and residential applications. A computational framework for power quality signal analysis has been developed and a hardware system has been designed.

The proposed system is divided in various modules or stages. The first module is named Front End Computing stage (FEC) which contains a Signal Conditioning System (SCS) an Analog to Digital Converter (A/D) and Digital Signal Processor (DSP). A second module named Back end Computing (BEC) is composed by an embedded computer and a user interface. The system implements Time-Frequency algorithms like the Cyclic Short Time Fourier Transform (CSTFT) and the Cyclic Ambiguity Function (CAF).

A third order dynamical system for harmonic analysis is also presented. This dynamical system is an Enhanced Phase Locked Loop (EPLL) which estimates a desired fundamental/harmonic component, tracking its amplitude, frequency and phase parameters.

RESUMEN

Este trabajo presenta el análisis, diseño e implementación de algoritmos eficientes para el análisis de señales tiempo-frecuencia. Nosotros proponemos el diseño y desarrollo de un sistema hardware para el procesamiento automatizado de información generada en aplicaciones residenciales e industriales. Un marco computacional para el análisis de señales de calidad de potencia ha sido desarrollado e implementado.

El sistema propuesto esta dividido en varios módulos o etapas. El primer modulo es el "Front End Computing" (FEC) el cual contiene un sistema de acondicionamiento de las señales (SCS) un convertidor de análogo a digital (A/D) y un procesador digital de señales. El Segundo modulo es el "Back End Computing" (BEC) el cual esta compuesto por un computador y una interfaz para el usuario. El sistema implementa algoritmos de análisis de tiempo-frecuencia tales como la trasformada de Fourier de tiempo corto cíclico y la función de ambigüedad cíclica.

Un sistema dinámico de tercer orden es también implementado. Este sistema dinámico tiene la capacidad de mantener un buen seguimiento de las características de frecuencia fase y amplitud de la señal en estudio.

“The true test of a man character is what

he does when no one is watching”.

-Anonymous

ACKNOWLEDGEMENTS

I want to thanks my advisor Dr. Domingo Rodriguez as well as the other members of my graduated committee, Dr. Rogelio Palomera and Dr. Efraín O'neill for their advice and conversations about this thesis. I also want to thank my fellows at the Automated Information Processing group, Cesar Aceros, Yuji Yunes, Miguel Erazo and Luz Acabá. Thanks to Sahray Gambaro for the elaboration of an appendix of this thesis.

As special thanks to Ana Ramirez for her support and constant conversations about this thesis. I also want to thanks my friend Juddy Gomez for her support and solidarity.

Table of Contents

Figure List

Figures **Page**

1 INTRODUCTION

The term *Power Quality* refers to a wide variety of electromagnetic phenomena that characterize the voltage and current at a given time and location on a power system [Boll06]. Power quality is concerned with deviations of the actual voltage or current waveform from an ideal waveform. These deviations from an ideal voltage or current waveforms are known as power quality disturbances. Power Quality measurements are performed for a number of reasons as presented in [Boll06]:

- Finding the cause of equipment malfunction. Finding the cause of a power quality problem is the first step in mitigating it.
- To get information on the performance of the supply voltage or of some equipment.
- To monitor the power network instead of only the voltage and current quality at the interface with the customer.
- To provide past data at the moment a problem is reported.
- To establish wide-scale monitoring campaigns to define the electromagnetic environment to which end users equipment is subject.
- To analyze events on the power systems that led to an interruption or blackout.

In recent years many Power Quality algorithms have been developed to detect and classify different kind of power quality events or disturbances. The early detection of power quality disturbances can be helpful among others in the prevention of energy faults events, to extend equipment life time, and to prevent monetary losses, for this reason fast computational signal processing methods implemented in hardware/software structures are required.

Using digital signal processing (DSP) tools we can assess power quality issues, among others, in time, frequency and time-frequency domains. A common frequency domain tool is the Discrete Fourier Transform (DFT) which analyzes the spectral content of a given signal. However many times we want to know how this spectral content evolves as time grows up, but the DFT can not offer this information. For this purposes Time-Frequency domain tools have been created, these tools generates two-dimensional signals representations that analyze the spectral content of a signal as a function of time. These kinds of tools are commonly used to obtain a better understanding and characterization of signals behavior. Another DSP tools are the time domain tools, which are commonly used in an adaptive fashion to assess a power quality analysis such as detection and estimation of signals parameters.

This work presents hardware implementations of time-frequency based power quality signal analysis algorithms. The first algorithm is based on the Cyclic Short Time Fourier Transform (CSTFT) and the Cyclic Ambiguity Function (CAF) to analyze transient disturbances. A computational framework based on signal algebra operators is presented to obtain an adequate and effective hardware mapping. The second implementation is based on an Enhanced Phase Locked Loop (EPLL) which is a third order dynamical system used to estimates fundamental and harmonics components, tracking its amplitude, frequency and, phase signal parameters.

This work is divided in six chapters. The first chapter presents an introduction to the work, the previous work related to power quality analysis and, the original contributions accomplished. The second chapter presets the theoretical foundations, an explanation of some DSP concepts that will be needed to understand this work. The third chapter presents

the mathematical framework for Power Quality signals analysis. The explanation of the algorithms contained in this work is presented in this section. Chapter fourth introduces the computational aspects about the hardware implementations of the algorithms. Experimental results are presented in chapter five. Conclusions and future works are presented in chapter six. Other relevant information to this work is presented on the appendix.

1.1 Previous Work

A literary review concerning with Power Quality signals analysis is presented in this section.

Wang and Mamishev present in [Wang04_1] and [Wang04_2] a method to classify voltage and current waveforms events that are related to a variety of Power Quality problems using the Ambiguity plane as a feature extraction tool.

Gu and Bollen in [Gu00] discuss the analysis of voltage disturbance recordings in the time-frequency domain and in the time-scale domain. The Discrete Short Time Fourier Transform is used for the time-frequency domain and two different Wavelets are used in the time-scale domain.

Shin et al. presents in [Shin99] the concept of time-frequency analysis to analyze transient disturbance signals in power system. The authors make harmonic analysis using different kernel functions.

Young-June Shin et al. in [Shin06] introduces some new power quality indices base on Time-Frequency analysis. These indices are computed using a Time-frequency representation of the fundamental component and the disturbance component.

Tao Lin and Alexander Domijan presents in [Lin05] a set of new power quality indices based on a novel Time-Frequency Atom and a recursive algorithm.

S. Lu presents in [Lu05] a Discrete Fourier Transform (DFT) filter bank for power systems harmonic measurements having the ability to measure harmonic contents in time and frequency.

M. Karimi-Ghartemani and A.K. Ziarani presents in [Kari04] and [Kari05] a third order dynamical system capable of extract and measure individual harmonics of a signal with time-varying frequencies.

D.G. Bailey presents in [Bailey05] a harmonic distortion method using spectral warping transformations with two different kernel functions.

L. Durak and O. Arikan present in [Dura03] two fundamentals properties of the short tome Fourier transform for an optimal implementation. The properties presented are the shift and rotation invariance properties.

1.2 Justification

In recent years a lot of work has been made in power quality disturbance monitoring, and many algorithms have been developed using advanced signal processing techniques. Among these techniques Time-frequency Analysis and Adaptive Signal Processing have been among the most useful for signal analysis and information extraction. Time-frequency representations have been demonstrated to be useful among other applications, to detect and classify power line signal disturbances, to help in the synchronization of instrumentation and monitoring equipment, and in the determination of transient disturbance energy flow. On the

other hand Adaptive Signal Processing has been used for disturbance detection and classification, as well as load changes determination in power systems; taking advantage of the algorithms adaptive nature which permits fast computations.

In order to automate the monitoring of power quality signals, effective power quality analysis algorithms are needed and their hardware implementations are critical to obtain reliable results. We envision the creation of an embedded power quality monitor, for this reason two novel power quality algorithms have been selected to be implemented in hardware using a digital signal processor. To build this embedded power quality monitor a computational framework has been created to establish simulation models that can be mapped to hardware.

This research work proposes two hardware implementations of a time-frequency algorithm a third order dynamical system for harmonic analysis of power quality signals.

1.3 Thesis Objectives

This section presents the objectives contained in this Thesis.

- Development of a Computational Framework of each one of the selected algorithms. The idea is to present the selected algorithms in terms of mathematical operators to obtain an understandable hardware mapping.
- Matlab simulation of the selected algorithms:
    1. Time-Frequency Algorithm
    2. Enhanced Phase Locked Loop Algorithm
- Hardware implementation of the selected algorithms.

- Performance Analysis of the selected algorithms.

1.4 Research Methodology

In order to meet the objectives of this work, the following tasks must be accomplished:

- Literary review about power quality analysis with special emphasis on time-frequency tools.
- Selection of potential algorithms for effective hardware implementation.
- Development of a computational framework for each one of the selected algorithms
- Matlab and/or Simulink implementation of the developed computational framework.
- Hardware implementations of the selected algorithms using a Texas Instruments TMS320C6713 digital signal processor.

1.5 Original Contributions

This section presents the original contributions contained in this work.

- Development of a functional Hardware prototype for the processing of Power Quality signals.
- Development of a special Signal Conditioning System for Transforming Power Quality Signals into audio signals.
- Development of a Cyclic Short Time Fourier Transform algorithm for the treatment of Power Quality signals.
- Development of a Matlab Environment for the modeling of power line disturbances following the IEEE Power Quality Standard.

- Development of a Signal Algebra Framework for the analysis of Power Quality signals.
- Formulation of a System Level Design Approach for the development of Power Quality signal Analysis Algorithms.
- Design and Development of Algorithms Signal analysis using a dynamical system approach.

2 THEORETICAL FOUNDATIONS

This chapter presents the some definitions and operations of digital signal processing, necessary to understand the fundamentals of power quality signal analysis. Descriptions of signal algebra operators necessary for the formulation of time-frequency tools are presented.

2.1 Signal Processing Concepts

Some basic concepts used for the development of this work are presented in this section.

Problem: Any statement that requires a solution.

Algorithm: A well defined procedure to solve a problem in a finite number of steps. A first step in the analysis and of a problem is the identification mathematical tools that may allow the formulation of algorithms.

Information: It is defined as anything that can be sending from one point to another or the quantitative or qualitative measure of interest.

Physical Signal: It is the entity that carries the information. It is the signal under study.

Mathematical Signal: It is defined as a numeric signal or numeric function.

2.1.1 Signal Algebra

This section presents some mathematical concepts and operators used for signal analysis.

Set: A collection of elements that are viewed as a simple entity.

Signal Space: The set A is a signal space or simply a space if each element of A is a signal or a function.

Cartesian Product: It is a new set $A \times B$ where every element of $A \times B$ is an ordered pair (p,q) where p is an element of A and q is an element of B.

$$A \times B = \{(p,q) \mid p \in A, q \in B\}$$

The Cartesian product performs an important role in the formulation of a linear algebra since they are used to describe linear operators.

Relation: ρ is a relation between two sets A and B if it is a subset of the Cartesian group $A \times B$.

$$\begin{aligned} \rho &: A \to B \\ & a_k \mapsto b_n \end{aligned}$$

Thus $(a_k, b_n) \in \rho$ means the relation ρ such that if it goes from the set A to the set B. The first set of a relation in this case A is called the domain of the relation and the second set in this case B is called the co-domain of the relation.

Function: A function f over a two-dimensional Cartesian product, say $C = A \times B$, is a relation where the first entry of each element of the relation appears once and only once.

Space of Finite Complex Signals: Let $l(Z_N)$ be the set of all complex signals of length N defined as follows:

$$\begin{aligned} x &: Z_N \to C \\ & n \mapsto x[n] \end{aligned}$$

The set $l(Z_N)$ is called the space of complex signals of length N . That is $l(Z_N) = \{x: x[n] \in C, n \in Z_N\}$. An arbitrary signal $x \in l(Z_N)$ can be described using the following notation:

$$x = \{x[0], x[1], \ldots, x[N-1]\}$$

Computational Signal Processing: It deals with the algebraic and system theoretical treatment of signals in order to extract information. This treatment of signals is performed through the use of algorithms.

2.1.1.1 Signal Metrics.

Energy of a Signal: The energy associated with a signal x is given by:

$$E_x = \langle x, x \rangle = \sum_{n \in Z_N} x[n]\, x^*[n] = \sum_{n \in Z_N} |x[n]|^2$$

Since the energy associated x is finite $(< \infty)$, the signal x is called a finite energy signal.

Signal Average Power: The average power of a signal is given by:

$$P_x = \frac{1}{N} \sum_{n \in Z_N} |x[n]|^2 = \frac{1}{N} E_x$$

Root Mean Square Value (RMS): The RMS value of a signal is given by:

$$x_{RMS} = \sqrt{\frac{1}{N} \sum_{n \in Z_N} |x[n]|^2} = \sqrt{\frac{1}{N} E_x} = \sqrt{P_x}$$

Signal Average or Mean: The average value or mean of a signal is given by:

$$\mu_x = \frac{1}{N} \sum_{n \in Z_N} x[n]$$

Signal Variance: The signal variance is given by:

$$\sigma_x^{\,2} = \frac{1}{N}\sum_{n\in Z_N} \left| x[n] - \mu_x \right|^2 = P_x - \mu_x^{\,2}$$

2.1.2 Signal Operators.

This section presents some mathematical operators definitions used for the representations of Time-Frequency tools.

Haddamard Product: The Haddamard Product Operator is defined as follows:

$$\bullet_N : \; l(Z_N) \rightarrow \; l(Z_N)$$
$$(x, v) \mapsto \; \bullet_N (x, v) = s$$

Where, $s[n] = \left(\bullet_N (x,v) \right)[n] = \; x[n]\cdot v[n]$, $n \in Z_N$. Normally $x \bullet_N v$ can be used instead of $\bullet_N (x, v)$

Shift Operator: The cyclic Shift operator S_N over the space $l(Z_N)$ with respect to the standard base Δ_N is defined as follows:

$$\Delta_N = \left\{ \delta_{\{k\}} : \delta_{\{k\}}[n] = 1,\, n = k \; ; \; \delta_{\{k\}}[n] = 0,\, n \neq k \right\},\; n, k \in Z_N$$

$$S_N : l(Z_N) \rightarrow l(Z_N)$$

$$\delta_{\{k\}} \quad \mapsto \quad S_N \left\{ \delta_{\{k\}} \right\} = \delta_{\{\langle k+1 \rangle_N\}}$$

Reflection Operator: The reflection operator over the space $l(Z_N)$ is defined as follows:

$$R_N : l(Z_N) \rightarrow l(Z_N)$$

$$x \quad \mapsto \; x^{(-)} = R_N \{x\}$$

Where $R_N\{x\}[n] = x^{(-)}[n] = x\left[\langle -n \rangle_N\right]$

Cyclic Convolution Operator: The cyclic convolution operator is defined as follows:

$$*_N : l(Z_N) \times l(Z_N) \to l(Z_N)$$

$$(x, h) \quad \mapsto \; y = *_N (x, h)$$

Where $y[n] = *_N (x,h)\,[n]$

$$y[n] = \sum_{k \in Z_N} x[k]\, h\left[\langle n-k \rangle_N\right]$$

The $\langle p \rangle_N$ denotes the module arithmetic operation,

$$\langle p \rangle_N = \; remainder \; (P/N)$$

Cyclic Correlation Operation: The cyclic correlation operator is defined as follows:

$$\oplus_N : l(Z_N) \times l(Z_N) \to l(Z_N)$$

$$(x, g) \quad \mapsto \; y = \oplus_N (x, g)$$

Where $y[n] = \oplus_N (x, g)\,[n]$

$$y[n] = \sum_{k \in Z_N} x[k]\, g\left[\langle n+k \rangle_N\right]$$

Discrete Fourier Transform (DFT): The DFT operator acting over the space $l(Z_N)$ is given by the following expression.

$$F_N : l(Z_N) \to l(Z_N)$$

$$x \mapsto F_N\{x\} = X$$

Where,

$$X[k] = F_N\{x\}[k] = \sum_{n \in Z_N} x[n]\, e^{-j\frac{2\pi}{N} k\, n}$$

Inverse Discrete Fourier Transform (IDFT): The IDFT operator acting over the space $l(Z_N)$ is given by the following expression.

$$F_N{}^{-1} : l(Z_N) \to l(Z_N)$$

$$X \mapsto F_N{}^{-1}\{X\} = x$$

Where,

$$x[n] = F_N{}^{-1}\{X\}[n] = \frac{1}{N} \sum_{k \in Z_N} X[k]\; e^{+j\frac{2\pi}{N} k\, n}$$

2.1.3 Description and Representation of Signals

This section presents a brief description of the common signals used in signal processing applications.

Real Signal: is a signal, whose co-domain is a set of real numbers,

$$x : \Re \to \Re$$
$$t \mapsto x(t)$$

Complex Signal: is a signal, whose co-domain is a set of complex numbers,

$$x: \ \Re \ \rightarrow C$$
$$t \mapsto x(t)$$

Analog Signal: is a signal, whose domain is a set of real numbers,

$$x: \ \Re \ \rightarrow C$$
$$t \mapsto x(t)$$

Discrete Signal: is a signal, whose domain is a set of integer numbers,

$$x: \ Z \ \rightarrow C$$
$$n \ \mapsto x[n]$$

Discrete signals can be obtained sampling continuous signals in discrete intervals of time.

Digital Signal: It is a signal whose co-domain is a finite set.

Multidimensional Signal: It is a mathematical signal with more than one independent variable.

Unit impulse Sequence: An impulse sequence is defined as follows,

$$\delta_k = \left\{ \begin{array}{c} \delta_k[n] : \delta_k[n] = 1, \ n = k, \\ \delta_k[n] = 0, n \neq k; \end{array} \right\}$$

$$\delta_k = [\cdots \quad 0 \quad 1 \quad 0 \quad \cdots]^T$$

kth
Element

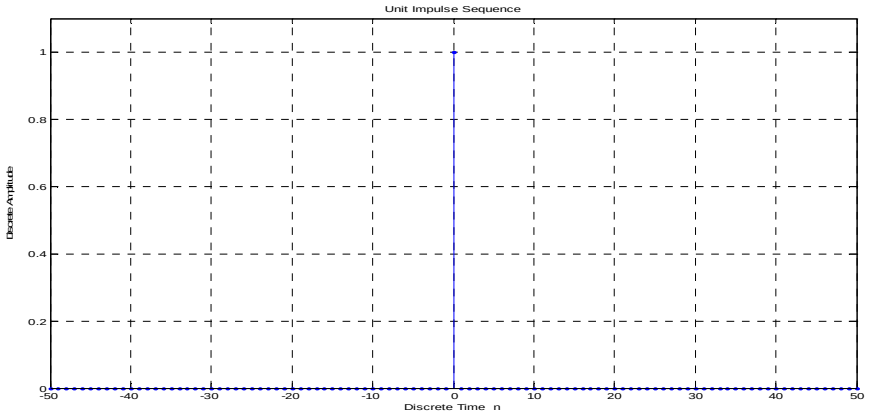

Figure 2.1. Unit impulse sequence

Delayed Unit impulse Sequence: A delayed impulse sequence is defined as follows:

$$\delta_{k+n_0} = \left\{ \begin{array}{r} \delta_{k+n_0}[n] : \delta_{k+n_0}[n] = 1,\ n = k + n_0, \\ \delta_{k+n_0}[n] = 0, n \neq k + n_0 ; \end{array} \right\}$$

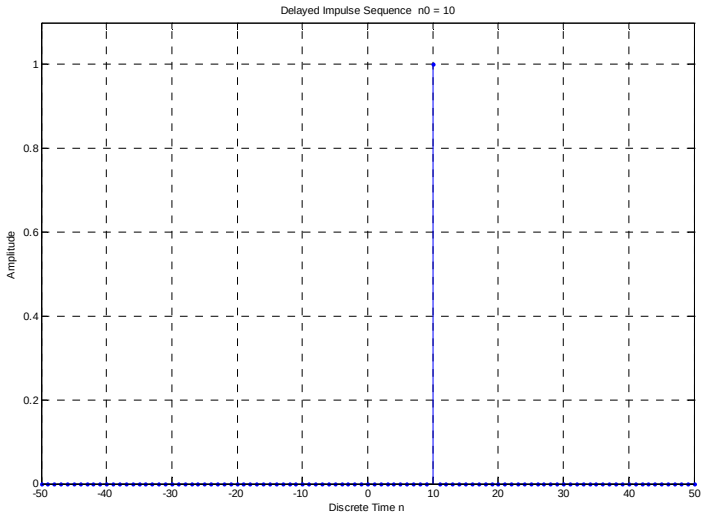

Figure 2.2 Delayed unit impulse sequence

Unit Sample Synthesis: Any arbitrary sequence x can be synthesized as a weighted sum of unit sample sequences, such as:

$$x = \sum_{k=-\infty}^{\infty} x[k]\ \delta_k$$

2.2 Discrete Systems

A discrete-time system is described as an operator $T\{.\}$ that takes a sequence x and transform it into another sequence y. That is

$$y = T\{x\}$$

Discrete systems are broadly classified into liner and nonlinear systems. A discrete system $T\{.\}$ is a liner operator $L\{.\}$ if an only if $L\{.\}$ satisfies the principle of superposition,

$$L\{\alpha x_1 + \beta x_2\} = \alpha\ L\{x_1\} + \beta\ L\{x_2\} \forall\ \alpha, \beta, x_1, x_2$$

Using the definition of the unit sample synthesis then we have:

$$y = L\{x\} = L\left\{\sum_{k=-\infty}^{\infty} x[k]\,\delta_k\right\} = \sum_{k=-\infty}^{\infty} x[k]\ L\{\delta_k\}$$

The response $L\{\delta_k\}$ can be interpreted as the response of a linear system due to a unit sample at time k. It is called an impulse response and is denoted by h_k.

$$y = \sum_{k=-\infty}^{\infty} x[k]\, h_k$$

The computation of the above equation requires the time-varying impulse response h_k which in practice is not very convenient. Therefore time-invariant systems are widely used.

A linear system in which an input-output pair x and y is invariant to a time shift is called a linear time invariant system (LTI). An LTI system can be denoted by the $LTI\{.\}$ operator. Let the x and y be the input-output pair of the LTI system, then the time-varying function h becomes a time-invariant function, and the output is given by :

$$y = LTI\{x\} = \sum_{k=-\infty}^{\infty} x[k]\, h_k$$

The impulse response of an LTI system is given by h, the above equation is called a linear convolution sum and is denoted by:

$$y = x * h$$

Hence an LTI is completely characterized by in the time domain by the impulse response h as shown below.

If x and h are causal finite sequences, with lengths N_x and N_h respectively, then the output y is given by the cyclic convolution sum $y = x *_N h$

$$y = \sum_{k=0}^{N-1} x[k]\, h_k \quad ; \quad N = N_x + N_h - 1$$

That means that each element of the vector y can be computed in the following manner:

$$y[n] = \sum_{k=0}^{N-1} x[k]\, h[\langle n-k \rangle_N]$$

This mathematical expression is composed of N multiplications and $N-1$ additions for each value of n . Typically an alternate computationally less expensive method is used to compute the discrete cyclic convolution using the convolution theorem.

Convolution Theorem: The convolution theorem states that the Discrete Fourier Transform of a convolution between a sequence x and a sequence h equals the point by point multiplication or Haddamard product between their individual Discrete Fourier Transforms:

$$x *_N h \Leftrightarrow X \bullet_N H$$

Where $X = F_N \{ x \}$ and $H = F_N \{ h \}$.

There exist a fast algorithm for the computation of the Forward/Inverse Discrete Fourier Transform F_N and $F_N{}^{-1}$ named the Forward/Inverse Fast Fourier Transforms (FFT/IFFT) algorithms. Figure 2.3 shows the convolution operation using the convolution property.

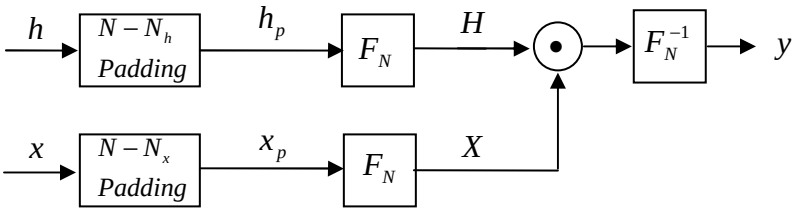

Figure 2.3 Cyclic convolution using the cyclic convolution theorem.

The sequences h and x are padded with $N-N_h$ and $N-N_x$ zeros to obtain two N point sequences h_p and x_p . Then a pair of FFTs is applied to the padded sequences, the

resulting frequency domain sequences H and X is multiplied point by point and finally an IFFT is computed to obtain the corresponding convolution sum y .

The convolution property can be used to compute other operations that can be expressed in terms of convolution sums. This is the case of the cyclic correlation which is used to verify the similarity between two discrete sequences.

$$x \oplus_N h = x *_N h^{(-)} \Leftrightarrow X \bullet_N H^{(-)}$$

As we can se once the cyclic correlation is expressed in terms on the cyclic convolution sum, we can take advantage of the convolution property to make the computation in an efficient way.

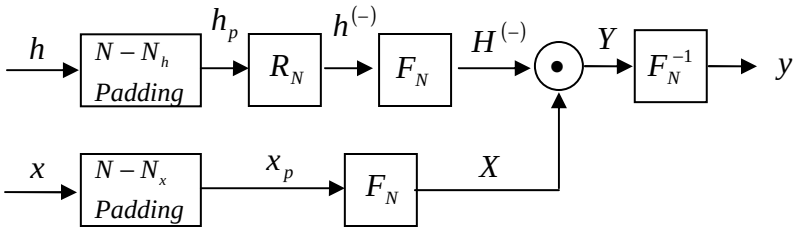

Figure 2.4 Cyclic Correlation using the cyclic convolution theorem.

Is important to note that the cyclic reflection operator appears when we express the cyclic correlation in terms of a cyclic convolution. A particular relationship exists between the Fourier Transform and Reflection operators F_N and R_N:

$$F_N\left\{h^{(-)}\right\} = F_N\left\{R_N\left\{h\right\}\right\} = H^{(-)} = R_N\left\{H\right\} = R_N\left\{F_N\left\{h\right\}\right\}$$

The above equation shows that there exists a commutative relationship between the Discrete Fourier Transform operator F_N and the Reflection operator R_N.

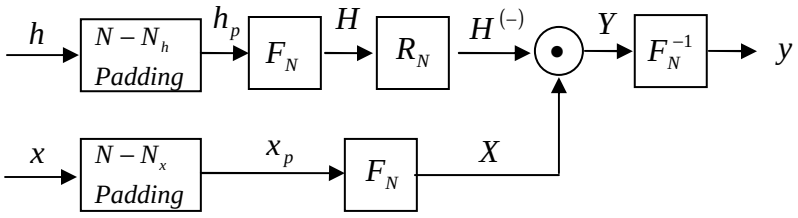

Figure 2.5 Cyclic Correlation using the cyclic convolution theorem and the commutative relationship between F_N and R_N.

3 Mathematical Framework for Time-Frequency Signal Analysis and Power Quality

This chapter presents a mathematical framework about the algorithms implemented in this thesis. Time-Frequency based signal processing techniques are presented first, using an operators signal algebra approach to power line signals. An enhanced phase locked loop model for harmonic analysis is also presented.

3.1 Time-Frequency Analysis

In the previous chapter we saw that we can perform different operations in the spectral domain using the Discrete Fourier Transform; but many times we want to know how the spectral content of a given signal changes as time evolves.

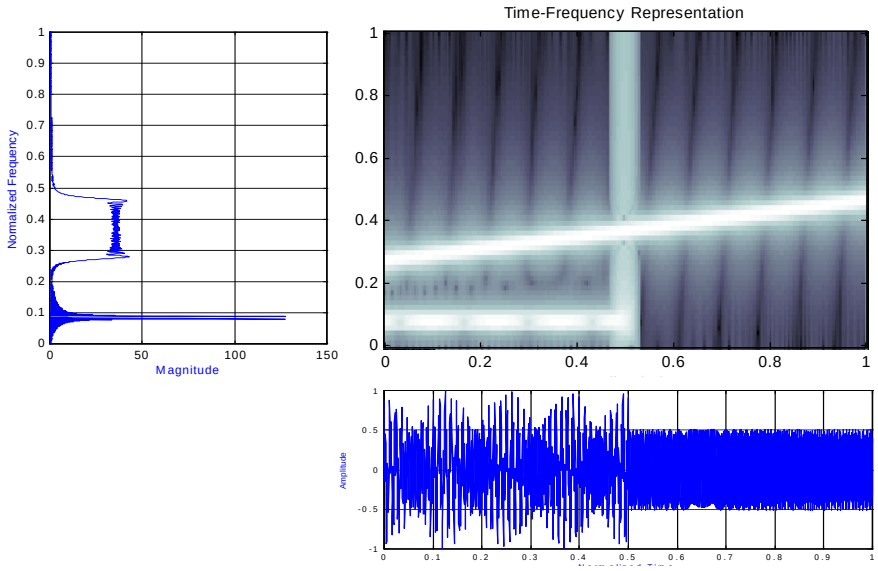

Figure 3.1 Time-Frequency representation example.

At the center of figure 3.1 we can see a Time-Frequency representation of an arbitrary signal. The bottom-right side of the figure shows the signal waveform as a function of time and, the upper-left shows the spectral content of the signal under study. We can distinguish to types of lines within the time-frequency representation. The first one is a constant line from 0 to 40% of the entire time window, which means that for this period of time there was a constant frequency component like a sinusoidal signal. The second line is a straight line with certain slope from 0 to 100% of the time window, which means that there was a signal component which frequency increase linearly as time grows up, like a chirp signal. A mathematical framework for the efficient computation of time-frequency tools is presented in the next section, using a signal operators approach to power line signals.

3.2 Operator Approach to Power Line Signal Analysis

In order to understand the operators approach to power line signals, we have to define the signals spaces where the power line signals lives. Figure 3.2 shows how the continuous power line signal space $L(R)$ is sampled and windowed to make a mapping to the discrete finite power line signal space $l^2(Z_N)$. The space $l^2(Z_N)$ contains the discrete finite energy power line signals that are being analyzed to extract information.

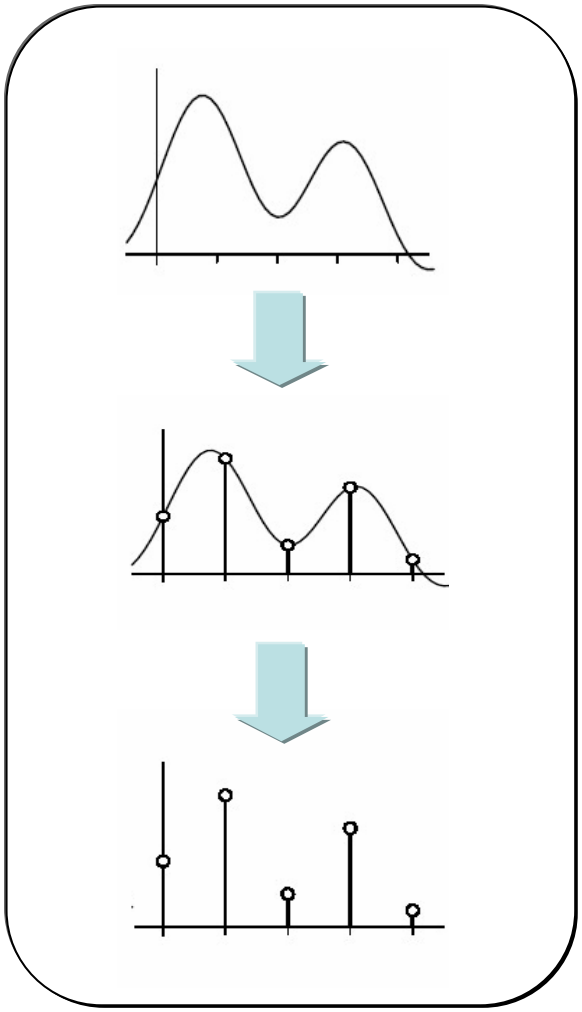

Figure 3.2 Power Line Signal Spaces

Once we have defined the discrete finite power line signal space, we can introduce the operator approach to power line signal analysis. After sampling and windowing we obtain a one-dimensional discrete power line signal, this signal can be processed using one-dimensional signal algebra operators or using time-frequency tools. Since time-frequency

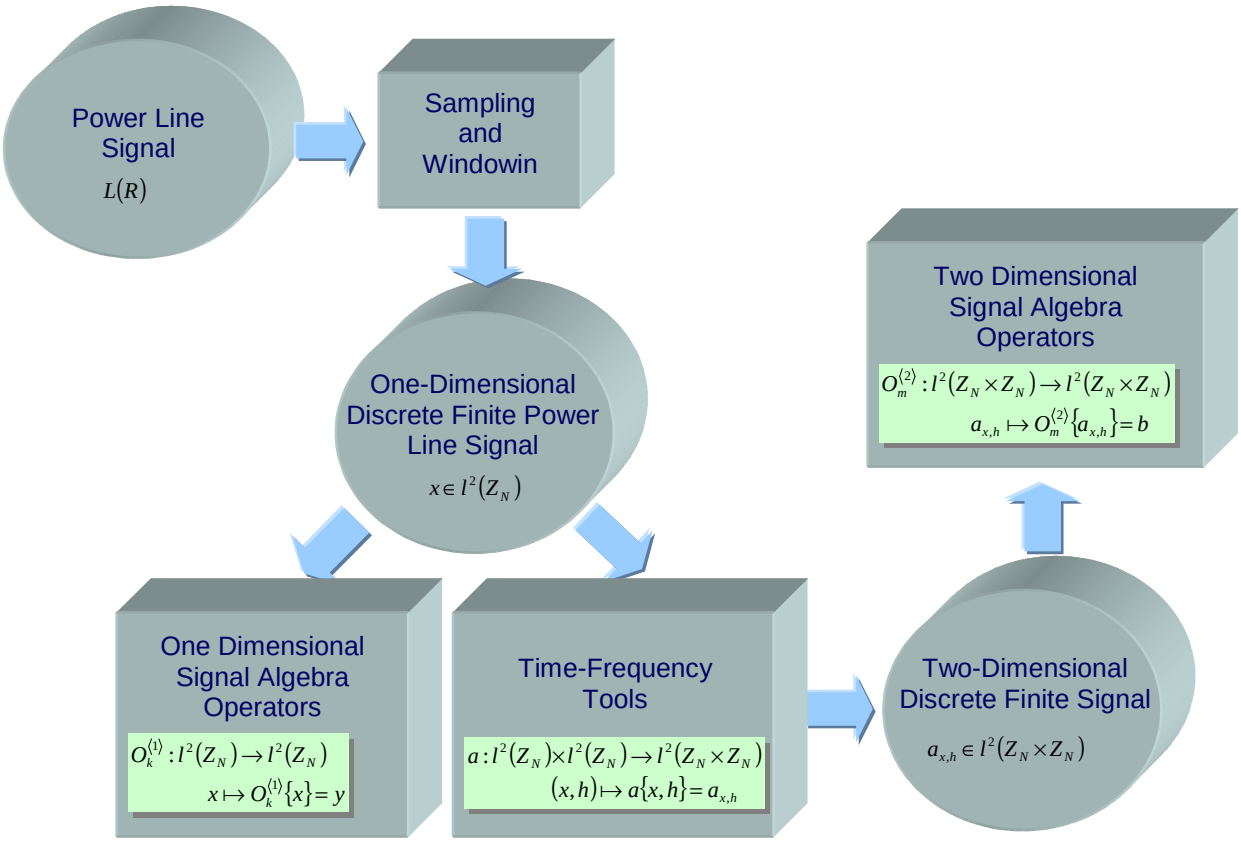

Figure 3.3 Operator Approach to Power Line Signal Analysis

tools maps a one-dimensional signal to a two-dimensional space, two dimensional signal algebra operators can be used for information extraction.

A one dimensional signal algebra operator is defined in the following manner:

$$O_k^{\langle 1 \rangle} = l^2(Z_N) \to l^2(Z_N)$$
$$x \mapsto O_k^{\langle 1 \rangle}\{x\} = y$$

A time-frequency tool is defined in the following manner:

$$a : l^2(Z_N) \times l^2(Z_N) \to l^2(Z_N \times Z_N)$$
$$(x,h) \mapsto a\{x,h\} = a_{x,h}$$

A two dimensional signal algebra operator is defined as:

$$O_m^{\langle 2\rangle} = l^2(Z_N \times Z_N) \to l^2(Z_N \times Z_N)$$
$$a_{x,h} \quad \mapsto O_k^{\langle 2\rangle}\{a_{x,h}\} = b$$

Figure 3.3 shows the operator approach to power line signal analysis.

3.2.1Cyclic Short Time Fourier Transform (CSTFT)

The CSTFT Transform is given by the following equation:

$$y[m,k] = \sum_{n=0}^{N-1} x[n]\, h\left[\langle m-n\rangle_N\right] W_N^{k\,n}\ ;\ W_N = e^{-j\,2\pi/N},\ j = \sqrt{-1}\ ;\ m,k \in Z_N$$

Where $x[n]$ is a N-length signal, $h[n]$ is a N-length windowing function, $\langle , \rangle_N$ is the modulo N operation, m and k denotes discrete time and discrete frequency shifts.

Let $x_k[n] = x[n] \cdot W_N^{kn}$

$$y[m,k] = \sum_{n=0}^{N-1} x_k[n]\, h\left[\langle m-n\rangle_N\right] = y_k[n]$$

Expressed in terms of signal operator we then have that:

$$y_k = x_k *_N h$$

Therefore the CSTFT can be reduced to a cyclic convolution operation between the parameterized signal x_k and the window function h. The convolution property can be used to obtain the following result.

$$y_k = F_N^{-1}\{ X_k \bullet_N H \}$$

Where $X_k = F_N\{x_k\}$ and $H = F_N\{h\}$. We named this as the filter method

Thus,

$$y[m,k] = \begin{bmatrix} y_0[0] & y_1[0] & \cdots & y_{N-1}[0] \\ y_0[1] & y_1[1] & \cdots & y_{N-1}[1] \\ \vdots & \vdots & \ddots & \vdots \\ y_0[N-1] & y_1[N-1] & \cdots & y_{N-1}[N-1] \end{bmatrix}$$

$$= [\, y_0 \quad y_1 \quad \cdots \quad y_{N-1} \,]$$

Figure 3.4 shows the computational framework for the CSTFT.

Another approach to compute the CSTFT is shown below:

$$X[m,k] = \sum_{n=0}^{N-1} x[n]\, h[\langle m-n \rangle_N] W_N^{kn};$$

Let $h_m[n] = h[\langle n-m \rangle_N]$ and $x_m[n] = x[n] \cdot h_m[n]$

Thus,

$$X[m,k] = \sum_{n=0}^{N-1} x_m[n] W_N^{kn} = X_m[k]$$

Expressed in terms of mathematical operators:

$$X_m = F_N\{\, x_m \,\}$$

Where $x_m = x \bullet_N h_m$, $h_m = S_N^m\{h^{(-)}\}$ and $h^{(-)} = R_N\{h\}$. This is the transform method.

Thus,

$$X[m,k] = \begin{bmatrix} X_0[0] & X_1[0] & \cdots & X_{N-1}[0] \\ X_0[1] & X_1[1] & \cdots & X_{N-1}[1] \\ \vdots & \vdots & \ddots & \vdots \\ X_0[N-1] & X_1[N-1] & \cdots & X_{N-1}[N-1] \end{bmatrix}$$

$$= \begin{bmatrix} X_0 \\ X_1 \\ \vdots \\ X_{N-1} \end{bmatrix}$$

Figure 3.5 shows this approach.

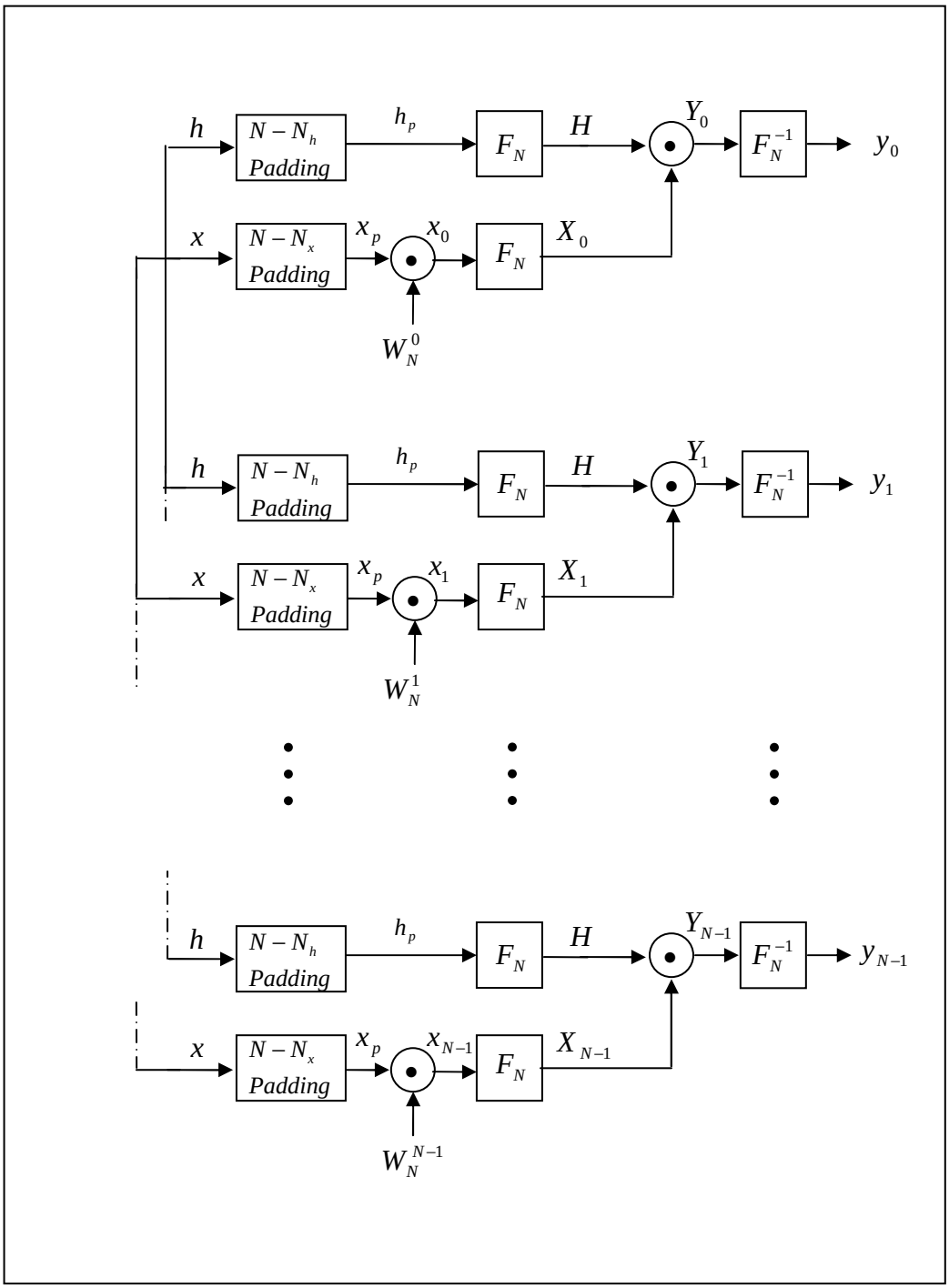

Figure 3.4 CSTFT Filter Method

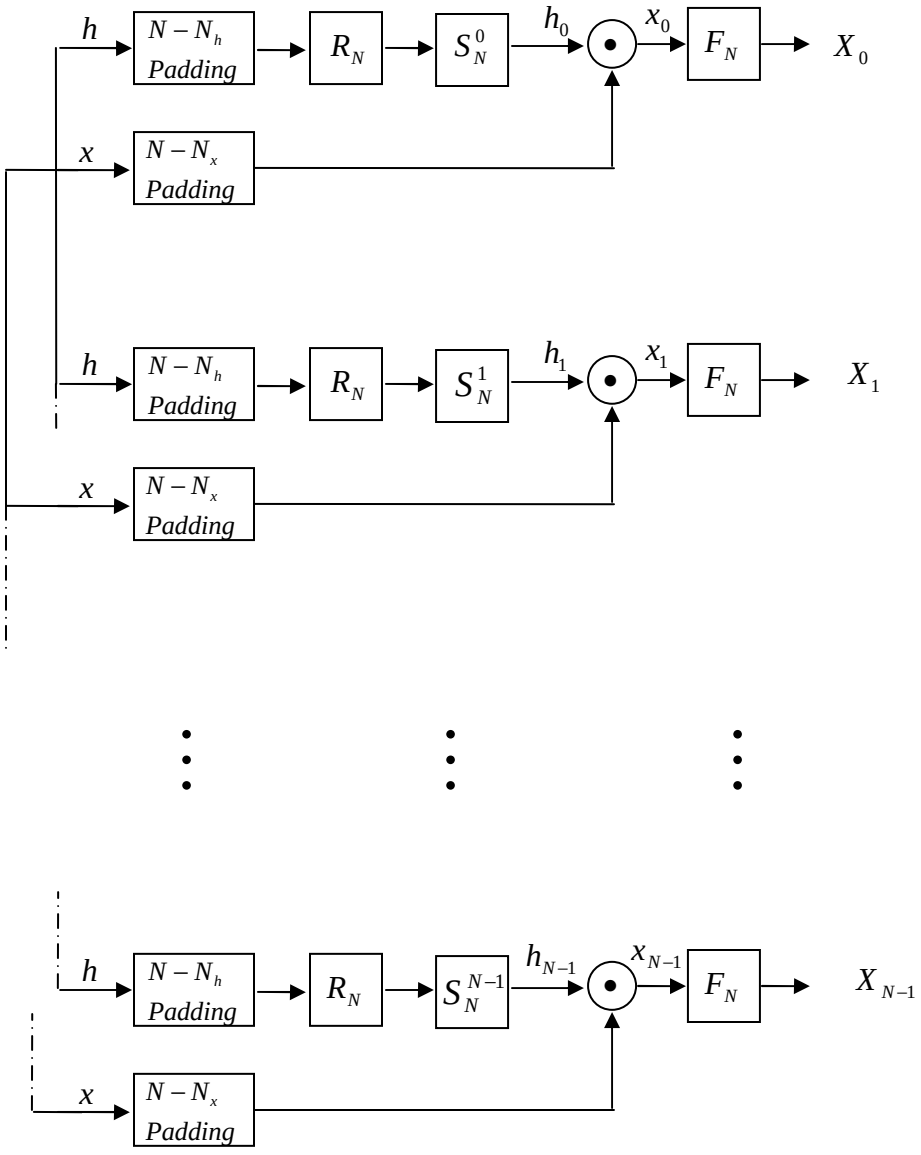

Figure 3.3

Figure 3.5 CSTFT Transform Method

3.2.2Cyclic Ambiguity Function (CAF)

The Discrete Ambiguity Function is given by the following equation:

$$A[m,k]=\sum_{n=0}^{N-1} x[n]\,h\left[\langle m+n\rangle_N\right]W_N^{k\,n}\;;W_N=e^{-j\,2\pi/N},\; j=\sqrt{-1}\;;m,k\in Z_N$$

Where $x[n]$ is a N-length signal, $h[n]$ is a N-length impulse response function, $\langle,\rangle_N$ is the modulo N operation, m and k denotes discrete time and discrete frequency shifts.

Let $x_k[n]=x[n]\cdot W_N^{kn}$,

$$A[m,k]=\sum_{n=0}^{N-1} x_k[n]\,h\left[\langle m+n\rangle_N\right]$$

Expressed in terms of mathematical operators,

$$A_k=x_k\oplus_N h\;=x_k *_N h^{(-)}$$

The Cyclic Ambiguity Function can be reduced to a cyclic correlation operation between the parameterized signal x_k and the impulse response function h. Similar to the CSTFT case we can take advantage of the convolution property to introduce the Discrete Fourier Transform operator.

$$A_k=F_N^{-1}\left\{\;X_k\;\bullet_N\;H^{(-)}\;\right\}$$

Where $X_k=F_N\{x_k\},\; H^{(-)}=F_N\left\{h^{(-)}\right\}$ and $h^{(-)}=R_N\{h\}$, this is the filter method.

Figure 3.6 shows the computational frame work for the CAF.

As in the case of the CSTFT the CAF can be expressed in other way,

Let $x_m[n]=x[n]\cdot h\ [\langle n+m\rangle_N]$, Thus

$$A[m,k]=\sum_{n=0}^{N-1}x_m[n]W_N^{kn}=A_m[k]$$

Expressed in terms of mathematical operators, then we have:

$$A_m=F_N\{x_m\}$$

Where $x_m=x\bullet_N h_m$, $h_m=S_N^{N-m}\{h\}$, which is the transform method.

Figure 3.7 shows this implementation.

3.2.3Discrete Wigner Distribution (DWD)

The Discrete Wigner Distribution is the two-dimensional Discrete Fourier Transform of the Discrete Ambiguity Function and is given by the following equation.

$$D_W[m,k]=\sum_{\tau=0}^{N-1}\sum_{v=0}^{N-1}A[\tau,v]W_N^{vm}\ W_N^{k\tau};W_N=e^{-j2\pi/N},\ m,k\in Z_N$$

Thus we can use the computational model of the Discrete Ambiguity function to compute the Discrete Wigner Distribution, but adding a two-dimensional Discrete Fourier Transform as a final stage of the computation.

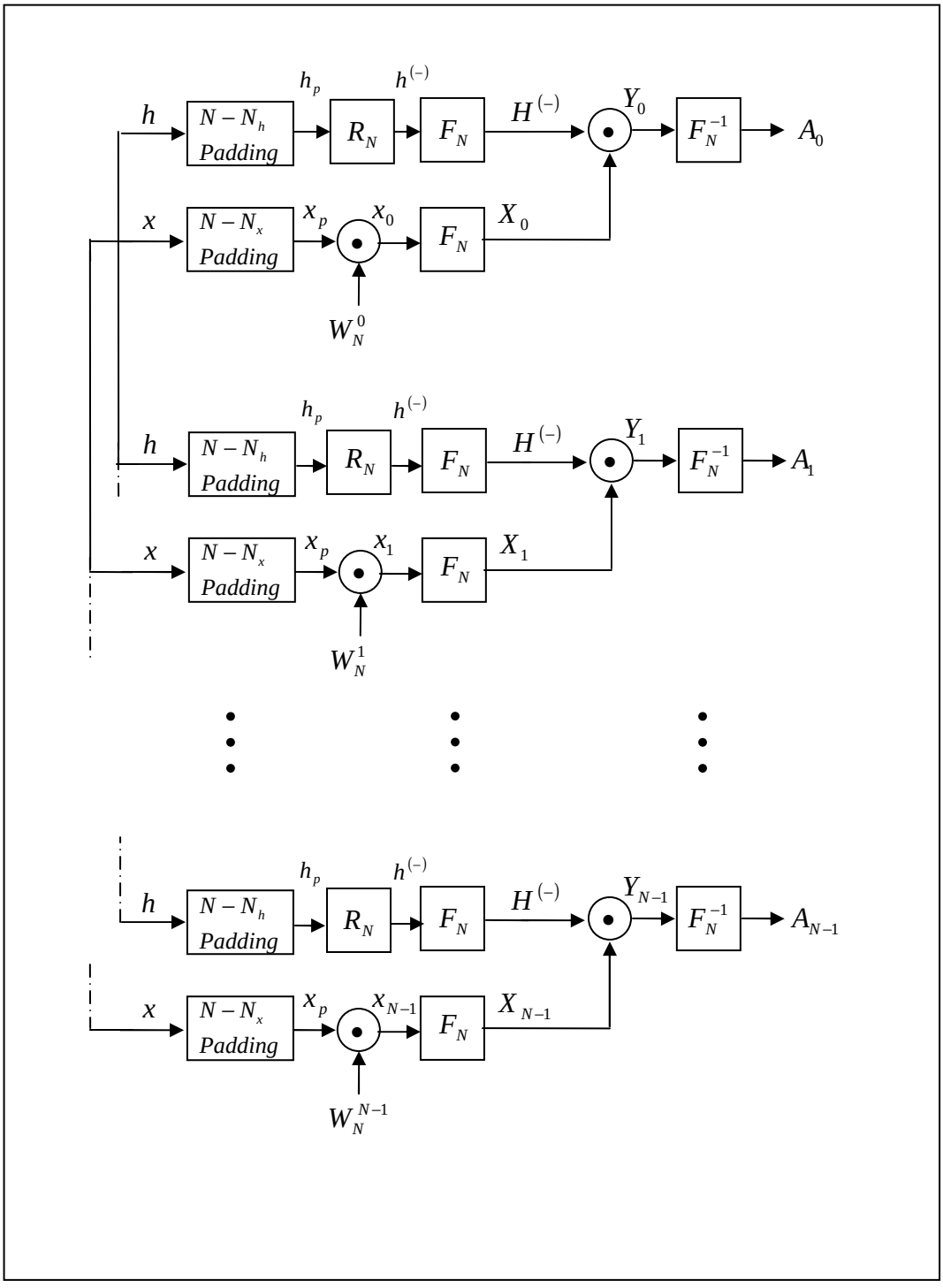

Figure 3.6 CAF Filter Method

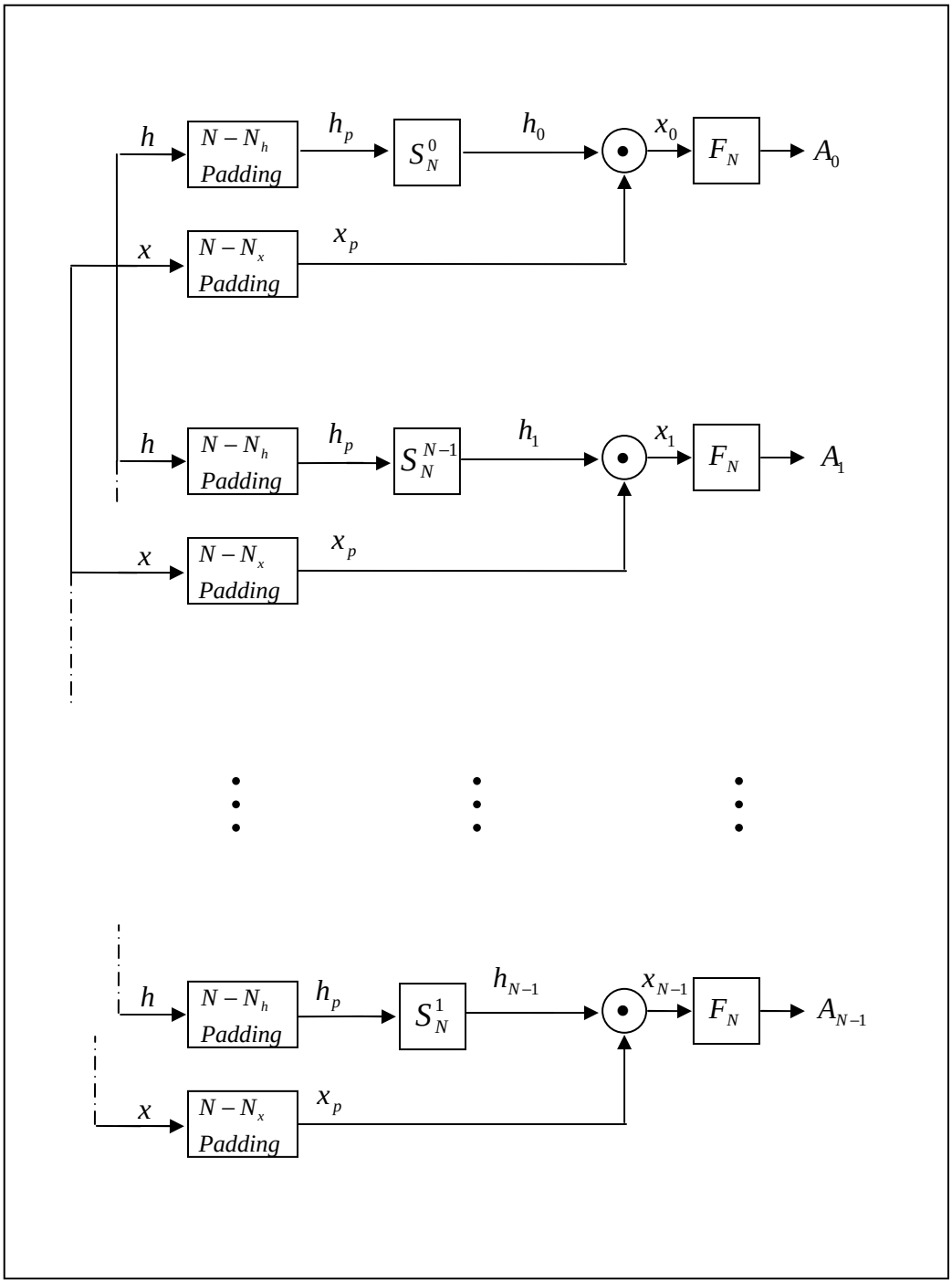

Figure 3.7 CAF Transform Method

3.3 Enhanced Phase Locked Loop Model

This section presents the mathematical formulation of an enhanced phase locked loop EPLL. This EPLL is a third order dynamical system, used to estimate the amplitude, frequency and, phase of a desired fundamental/harmonic component of a given signal.

3.3.1 Time Varying Harmonics

The Discrete Fourier Transform (DFT) has been widely used as the standard method of harmonic analysis. However it assumes a specific frequency spectrum of the input signal, which consists of a dc component, a fundamental component with a given frequency together with a number of its harmonics. Therefore wherever a deviation from this assumption occurs the DFT loses its capability to provide an accurate decomposition of the signal in the frequency domain. This is the case of some power quality signals, like the transients occurring in power systems, inter-harmonics, and voltage flickers for example. Another DFT issue is it sensitivity with respect to variations in the fundamental frequency, sampling rate, and noise. The DFT presumes a known value of the fundamental frequency, sampling rate, and observation window length. Any deviation occurring in these parameters can result in erroneous information.

Harmonic analysis requires means for accurate measurement of individual and total harmonic content of a signal. Harmonic measurement of a fixed-frequency signal is a straight forward task and can be efficiently addressed with conventional DFT based methods.

However these methods suffer loses in accuracy under time varying conditions, due to the mentioned shortcomings. The voltage and current harmonics can be time-varying due to:

- Continuous changes in the system configuration and load conditions
- Rapid proliferation of distributed resources

Time-varying harmonics poses difficulties with respect to their proper analysis. Any frequency deviation from the nominal value of 50/60 Hz can substantially degrade the performance of the measurement devices which operate based on the assumption of constant frequency.

This section presents a harmonic extraction and measurement method which employs an enhanced phase-locked loop (EPLL) as the main building block. The mechanism of the EPLL is such that it extracts the amplitude and phase of the sinusoidal component of its input signal for which its internal operating point is preset while adaptively follows time variations in the characteristics of the signal. Two main features of this harmonic estimator are:

- Accurate performance in non stationary frequency-varying environments
- Immunity to noise.

3.3.2 Time Varying Harmonics Measurements Method

Let $u(t)$ denote a signal comprising a fundamental component of nominal frequency ω_1, other frequency components and noise.

$$u(t) = A_1 \sin\phi_1 + \sum_{k=2}^{N} A_k \sin\phi_k + n(t)$$

Where $\phi_k = \omega_k t + \theta_k$ is the total phase of the kth component and $n(t)$ denotes the noise imposed on the signal and, A_k is the amplitude of the kth harmonic. Usually ω_1 is fixed, and $\omega_k = k\,\omega_1$, but that is not necessarily true in all cases. In general, ω_k may vary in time in an ε_k - neighborhood of $k\,\omega_1$ where ε_k is determined by the application. A component of $u(t)$ that has a frequency $k\,\omega_1$ or has a frequency which is in the ε_k neighborhood of $k\,\omega_1$ is defined as the kth harmonic of the fundamental frequency.

The problem of harmonic measurement is to find a way to estimates individual harmonic components as fast and accurate as possible. This scheme should not be sensitive to noise and more importantly not to the time variations of frequencies. In general to track the amplitude A , frequency ω and phase ϕ of the desired harmonic component $y(t) = A(t)\sin\phi$, a set of three differential equations was derived in [Kari05].

$$\frac{d}{dt}A(t) = \mu_1 e(t)\sin\phi$$
$$\frac{d}{dt}\omega(t) = \mu_2 e(t)\cos\phi$$
$$\frac{d}{dt}\phi(t) = \omega(t) + \mu_3 \frac{d}{dt}\omega(t)$$

Where μ_1, μ_2, and μ_3 control the amplitude ,frequency , and phase tracking properties respectively, and $e(t) = u(t) - y(t)$ is the error signal, representing the difference between the signal under study and the desired fundamental or harmonic component. A block diagram of the core unit is presented in Fig 3.6 in which ω_0 is the nominal value of ω_1. Three integrations three multiplications, three gains, two additions, and two trigonometric

functions are the arithmetic operations needed to implement a 3rd order dynamic system to track the amplitude, frequency and phase of the fundamental component $y(t)$ of a given signal $u(t)$.

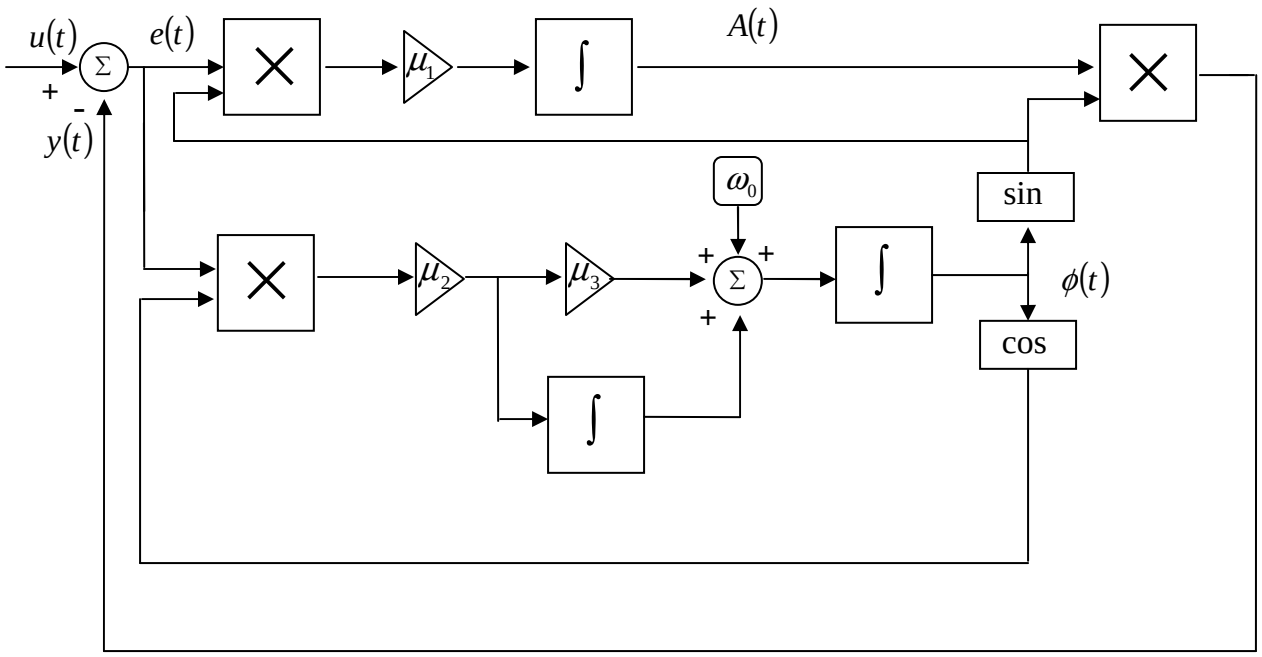

Figure 3.8 EPLL core unit.

The core unit extracts the fundamental component, estimating its amplitude, phase and frequency and accommodating variations in these parameters. The fundamental and harmonic components are sinusoidal components embedded in the signal with

frequencies f_1, f_2,…, f_N. A cascaded configuration consisting of N core units can extract the fundamental and the harmonics of the signal. The remainder which is the error signal of the nth core unit is ideally the sum of all harmonics of order N+1 and higher as shown in Figure 3.7.

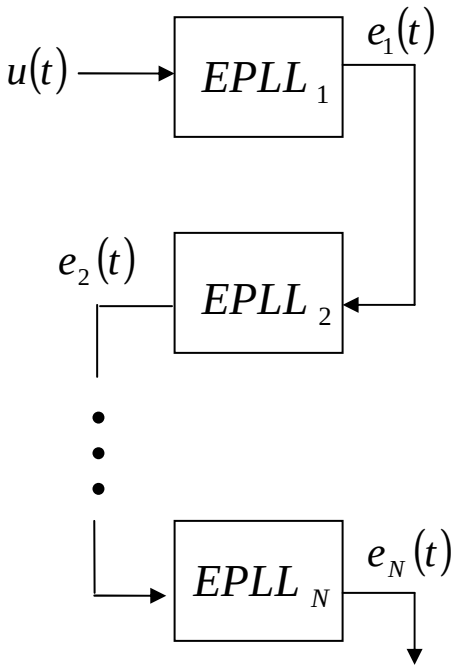

Figure 3.9 EPLL cascade configuration.

In the structure of figure 3.7, the core units are not bound to exclusively find the sinusoidal component whose frequencies are predetermined by their internal settings. For example in case the third unit is set to extract the third harmonic , but the signal does not have a third harmonic component its output moves toward and converges to the next harmonic

component The third unit may provide the fourth harmonic while the fourth unit is also trying to extract the fourth component. To retain consistency and avoid competition/repetition of units is necessary to define not overlapping territories. The solution is to define a range of $(k\ f_0 - (f_0/2), k\ f + (f_0/2))$ as the territory of the kth unit where f_0 is the fundamental frequency 50/60 Hz.

4 Computational Aspects for Power Quality Signal Analysis

This chapter presents the computational aspects about the hardware implementations of the algorithms contained in this work. This section proposes the design and development of an integrated system for the automated processing of sensor-based signal data information generated in industrial and residential applications. The developed system can perform time-frequency analysis of power line signals. The proposed system is named Signal Imaging and Monitoring for Power Line Environment (SIMPLE). Figure 4.1 shows a block diagram of the proposed prototype.

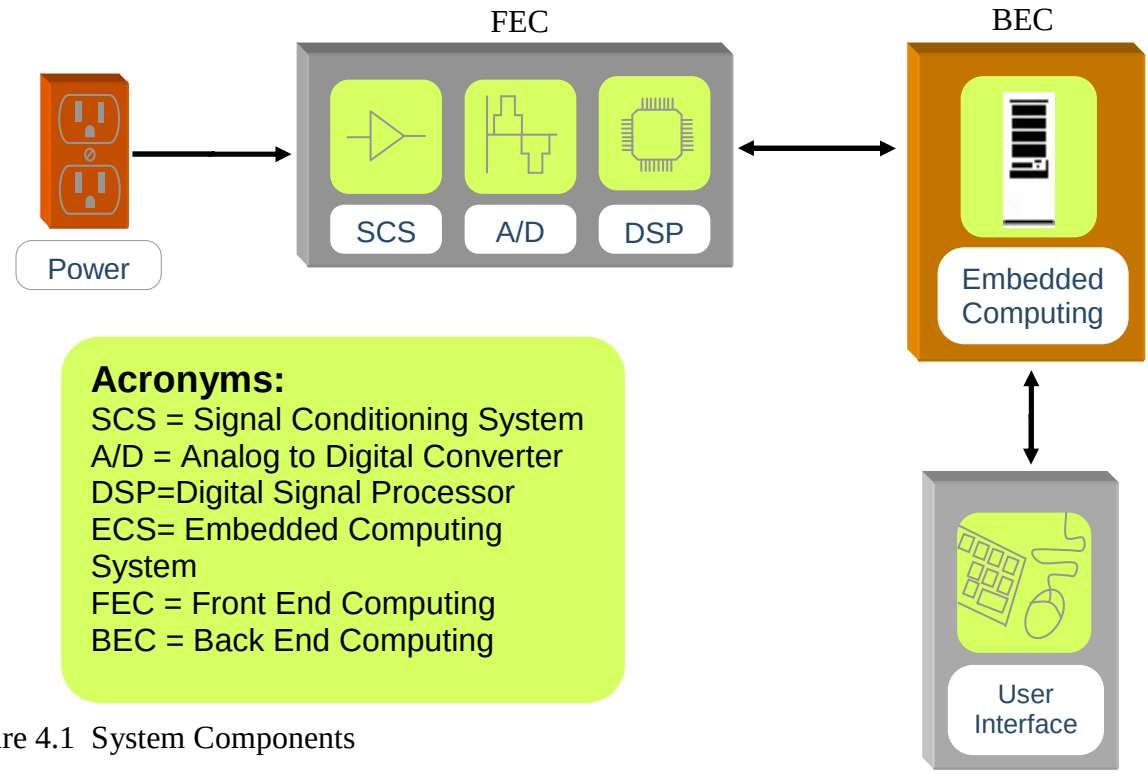

Figure 4.1 System Components

The proposed system is composed of two main design blocks: the front end computing (FEC) and the back end computing (BEC). The front end computing is as well composed of other building blocks. The first one is the signal conditioning system (SCS), which prepares the data to be sampled, that is to say, the incoming signal amplitude is conditioned to obtain an appropriate amplitude level for further processing. The second building block is the analog to digital converter (A/D) which is in charge of the sampling process, to obtain the data in a digital form. The last FEC building block is the most important, the digital signal processor (DSP). The DSP is the FEC's brain; it is in charge of algorithms implementations as well as of some basic signal metrics computations. The BEC is composed of the embedded computing stage and the user interface. These two stages store and display the data to the users.

4.1 Front End Computing (FEC)

As was mentioned at the previous section the FEC is composed of three building bocks. This section presents their functionality and characteristics. The SCS make a mapping between a high level voltage to a low level voltage. For this work SCS circuit was build using discrete components.

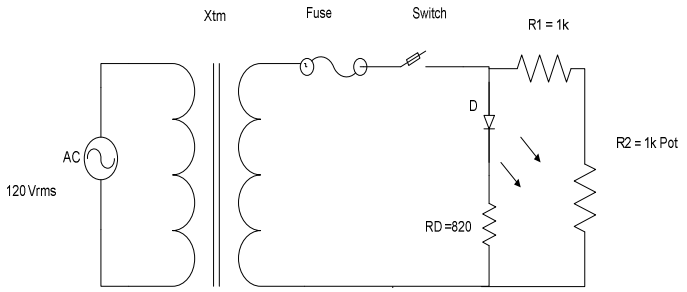

Figure 4.2 Signal Conditioning System (SCS) schematic

Figure 4.2 shows the implemented SCS circuit. The idea is to build a circuit that attenuates the incoming 120 Vrms line voltage without degrade the signals shape. This is achieved using a step-down transformer. Since the transformer secondary has to feed the DSP audio port that is between -3V and +3V; a voltage divider is used in this case to obtain an appropriate voltage level.

A Texas Instruments TMS320C6713 DSK was used as the DSP. This device is a development board which contains an analog to digital converter.

Figure 4.3 Signal Conditioning System.

Thus the A/D stage is embedded within the DSP stage. The TMS320C6713 is based on the Very Long Instruction Word (VLIW) architecture which is very well suited for numerically intensive algorithms. Some TMS320C617 features are:

- Embedded JTAG support via USB
- High Quality 24 bits stereo codec
- 512K words of Flash and 8 MB SDRAM
- Expansion port connector for plug-in modules
- 225 MHz, delivering up to 1800 MIPS and 1350 MFLOPS

Figure 4.4 Digital Signal Processor (DSP)

4.2 Back End Computing (BEC)

The back end computing and the user interface can be implemented in a personal or embedded computer. Figure 4.5 shows the entire system. In this case the system is acquiring a signal from the power line and applying the CSTFT.

A system level design approach has been used as the design methodology. This approach permits a connection between the mathematical formulations of the desired power line signal metrics and the hardware implementation using DSPs.

A Real-Time Data Exchange is used to communicate Matlab with the TM320C6723 DSP development environment named Code Composer Studio. Figure 4.6 presents the system level approach.

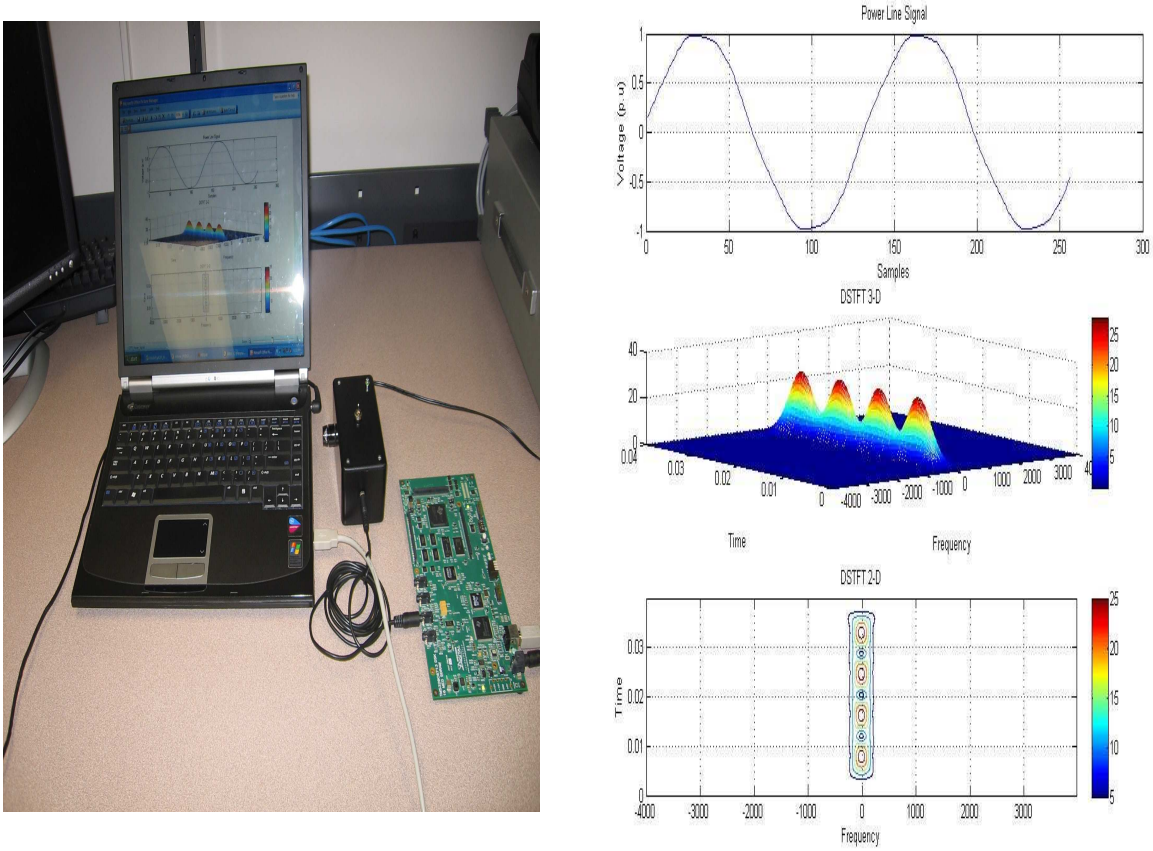

Figure 4.5 Signal Imaging and Monitoring Power Line Environment

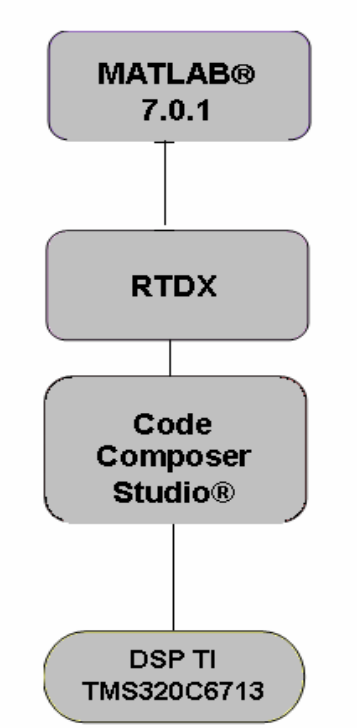

Figure 4.6 Matlab/DSP Communication Protocol

4.3 Cyclic Short Time Fourier Transform

This section presents the computational aspects of the CSTFT, using the computational framework presented in chapter 3:

$$X_m = F_N\{x_m\} = F_N\{x \bullet_N h_m\} = F_N\left\{x \bullet_N S_N^m\left\{h^{(-)}\right\}\right\} = F_N\left\{x \bullet_N S_N^m\{R_N\{h\}\}\right\}$$

Since we always know the filter h, we can pre-compute $h_m = S_N^m\{R_N\{h\}\}$ for all values of m without acquiring the signal x. Therefore the matrix $[H]$ can be obtained in the

following manner:

$$[H]=\begin{bmatrix} h_0 \\ h_1 \\ \vdots \\ h_{N-1} \end{bmatrix}=\begin{bmatrix} S_N^0\{h^{(-)}\} \\ S_N^1\{h^{(-)}\} \\ \vdots \\ S_N^{N-1}\{h^{(-)}\} \end{bmatrix}=\begin{bmatrix} h[0] & h[N-1] & h[N-2] & \cdots & h[2] & h[1] \\ h[1] & h[0] & h[N-1] & \cdots & h[3] & h[2] \\ h[2] & h[1] & h[0] & \cdots & h[4] & h[3] \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ h[N-2] & h[N-3] & h[N-4] & \cdots & h[0] & h[N-1] \\ h[N-1] & h[N-2] & h[N-3] & \cdots & h[1] & h[0] \end{bmatrix}$$

This kind of matrix is known as a circulant matrix. Where, each row and each column is a down shifted version of the previous.

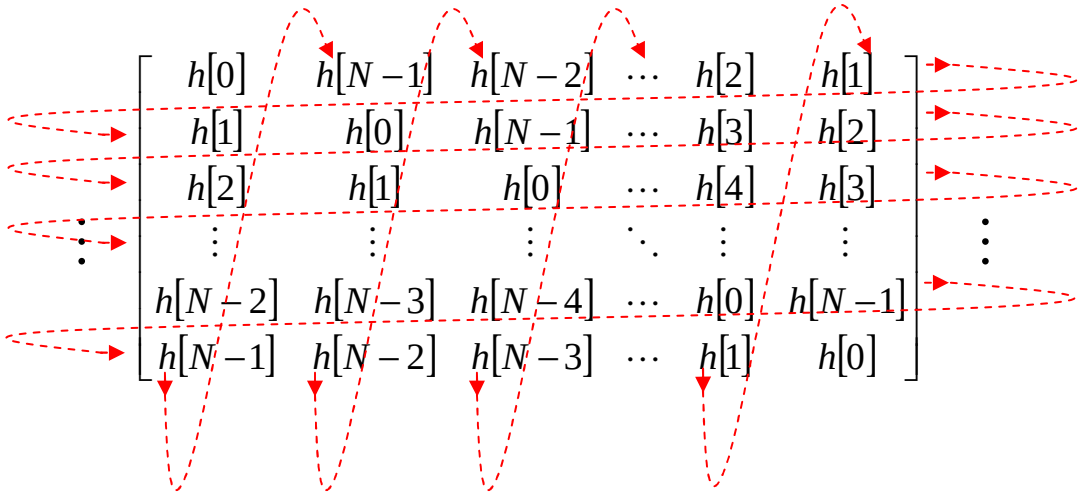

Figure 4.7 Circulant Matrix

In order to understand how to compute that matrix in an effective way, let see an example:

Suppose we have a signal x an filter h:

$$x=\{x[0],x[1],x[2],x[3],x[4],x[5]\}$$

$$h=\{h[0],h[1]\}$$

The signals lengths are $N_x = 6$ and $N_h = 2$, therefore $N = N_x + N_h = 8$

In order to compute the matrix $[H]$, x has to be padded by N_h and the filter h by N_x. As result of the padding operation the signals x_p and h_p are obtained.

$$x_p = \{x[0], x[1], x[2], x[3], x[4], x[5], 0, 0\}$$

$$h_p = \{h[0], h[1], 0, 0, 0, 0, 0, 0\}$$

Therefore the matrix $[H]$ is given by:

$$\begin{bmatrix} h[0] & 0 & 0 & 0 & 0 & 0 & 0 & h[1] \\ h[1] & h[0] & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & h[1] & h[0] & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & h[1] & h[0] & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & h[1] & h[0] & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & h[1] & h[0] & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & h[1] & h[0] & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & h[1] & h[0] \end{bmatrix}$$

Once we have $[H]$ the next step is to perform a point-by-point multiplication between the vector x_p and each row of the matrix $[H]$ to obtain the matrix $[X]$.

$$[X] = \begin{bmatrix} h_0 \bullet_N x_p \\ h_1 \bullet_N x_p \\ h_2 \bullet_N x_p \\ h_3 \bullet_N x_p \\ h_4 \bullet_N x_p \\ h_5 \bullet_N x_p \\ h_6 \bullet_N x_p \\ h_7 \bullet_N x_p \end{bmatrix} = \begin{bmatrix} h[0]\cdot x[0] & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ h[1]\cdot x[0] & h[0]\cdot x[1] & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & h[1]\cdot x[1] & h[0]\cdot x[2] & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & h[1]\cdot x[2] & h[0]\cdot x[3] & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & h[1]\cdot x[3] & h[0]\cdot x[4] & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & h[1]\cdot x[4] & h[0]\cdot x[5] & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & h[1]\cdot x[5] & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

In this process we had to perform N^2 multiplications, but as you can see there are only $N_x \cdot N_h$ values different from 0. We can take advantage of that to design an algorithm with a minimum number of multiplications.

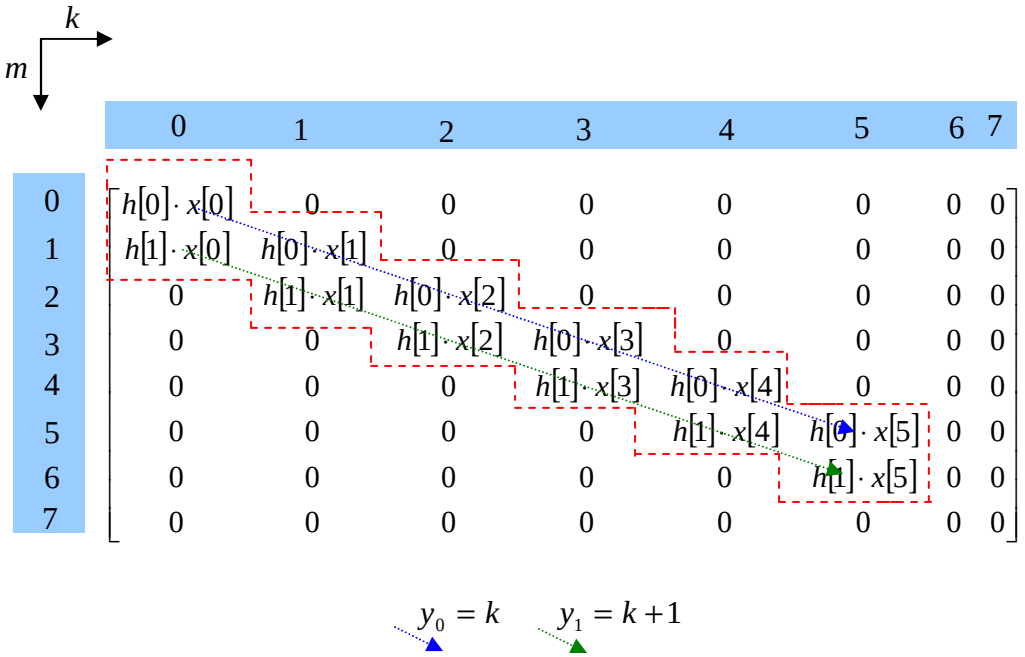

Figure 4.8 Non-zero elements

Figure 4.8 shows that all the nonzero points are in the diagonal lines $y_0 = k$ and $y_1 = k+1$, therefore :

$$y_m = \langle k+m \rangle_N$$

where k is the matrix column index and n the matrix row index. Each element $[X](y_m, k)$ of the matrix $[X]$ can be computed in the following manner:

$$[X](y_m, k) = h[m] \cdot x[k] \; ; \; m \in Z_2, k \in Z_6$$

In general for any value of N ,

$$[X](y_m, k) = h[m] \cdot x[k] \; ; \; m \in Z_{N_h}, k \in Z_{N_x}$$

Once we have computed all the non zero elements of $[X]$; the next step is to compute the DFT over each row using the Fast Fourier Transform (FFT) algorithm to obtain the CSTFT.

4.4 Cyclic Ambiguity Function

As presented in chapter 3 the CAF can be described by :

$$A_m = F_N\{x_m\} = F_N\{x \bullet_N h_m\} = F_N\{x \bullet_N S_N^{N-m}\{h\}\} = F_N\{x \bullet_N S_N^{N-m}\{R_N\{h\}\}\}$$

Similar to the CSTFT case, we can pre-compute $h_m = S_N^{N-m}\{R_N\{h\}\}$ for all values of m. In this case the matrix $[H]$ can be computed as follows:

$$[H] = \begin{bmatrix} h_0 \\ h_1 \\ \vdots \\ h_{N-1} \end{bmatrix} = \begin{bmatrix} S_N^0 \\ S_N^{N-1} \\ \vdots \\ S_N^1 \end{bmatrix} = \begin{bmatrix} h[0] & h[1] & h[2] & \cdots & h[N-2] & h[N-1] \\ h[1] & h[2] & h[3] & \cdots & h[N-1] & h[0] \\ h[2] & h[3] & h[4] & \cdots & h[0] & h[1] \\ \vdots & \vdots & \vdots & \iddots & \vdots & \vdots \\ h[N-2] & h[N-1] & h[0] & \cdots & h[N-4] & h[N-3] \\ h[N-1] & h[0] & h[1] & \cdots & h[N-3] & h[N-2] \end{bmatrix}$$

This kind of matrix is known as a skew-circulant matrix. Each column is an up shifted version of the previous.

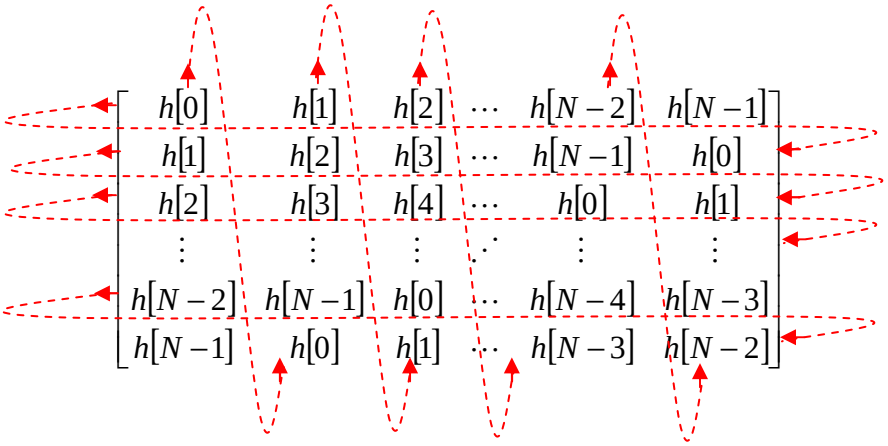

Figure 4.9 Skew-circulant matrix**.**

Let we have the same signal x and window filter h of the previous example. In this case, the matrix $[H]$ is given by:

$$\begin{bmatrix} h[0] & h[1] & 0 & 0 & 0 & 0 & 0 & 0 \\ h[1] & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & h[0] & 0 & 0 \\ 0 & 0 & 0 & 0 & h[0] & h[1] & 0 & 0 \\ 0 & 0 & 0 & h[0] & h[1] & 0 & 0 & 0 \\ 0 & 0 & h[0] & h[1] & 0 & 0 & 0 & 0 \\ 0 & h[0] & h[1] & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Therefore the matrix $[X]$ is presented in figure 4.10.

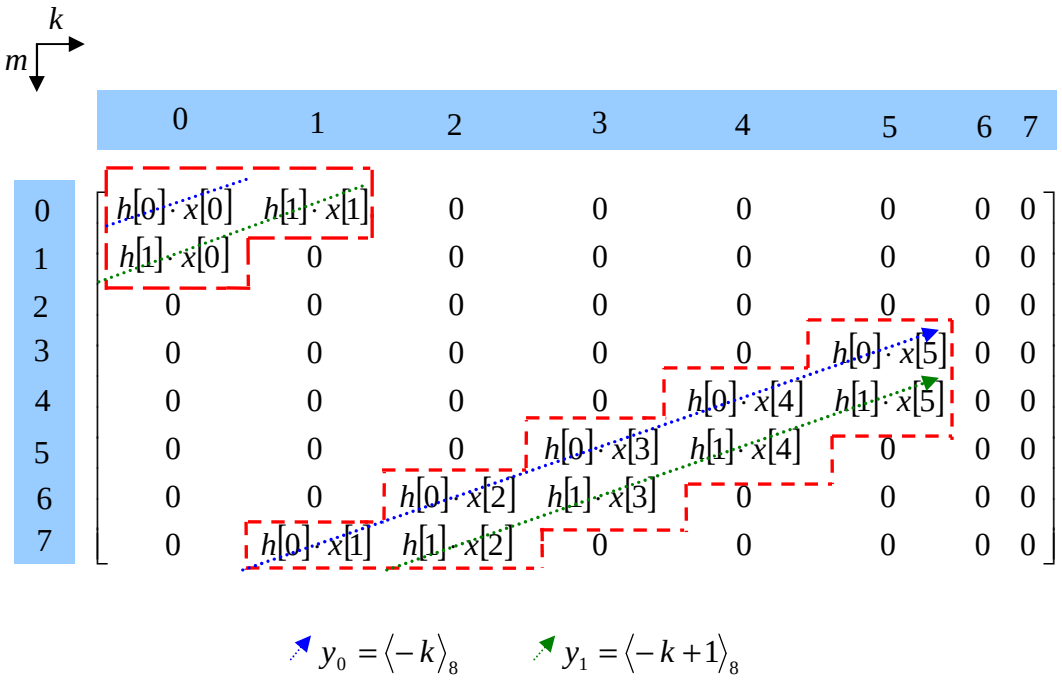

Figure 4.10 Non-zero elements

All the non-zero elements can be found over the diagonal lines:

$$y_0 = \langle -k \rangle_8$$
$$y_1 = \langle -k+1 \rangle_8$$

In general, for any value of N

$$y_m = \langle -k+m \rangle_N ; k \in Z_{N_x}, m \in Z_{N_h}$$

where k is the matrix column index and m the matrix row index. Each element $[X](y_m, k)$ of the matrix $[X]$ can be computed in the following manner:

$$[X](y_m, k) = h[m] \cdot x[k] ; \; m \in Z_2, k \in Z_6$$

In general for any value of N,

$$[X](y_m, k) = h[m] \cdot x[k] ; \; m \in Z_{N_h}, k \in Z_{N_x}$$

4.5 CSTFT and CAF Performance Characteristics

Using the multiplication schemes presented in sections 4.3 and 4.4 we can reduce the number of multiplications from N^2 to $N_x \cdot N_h$. In general for any value of N,

$$N_h = \alpha \cdot N$$
$$N_x = N - N_h = (1-\alpha) \cdot N$$

Where $0 < \alpha < \frac{1}{2}$, therefore :

$$N_x \cdot N_h = \alpha(1-\alpha) \cdot N^2$$

Figure 4.11-12 show the reduction curves for different values of α.

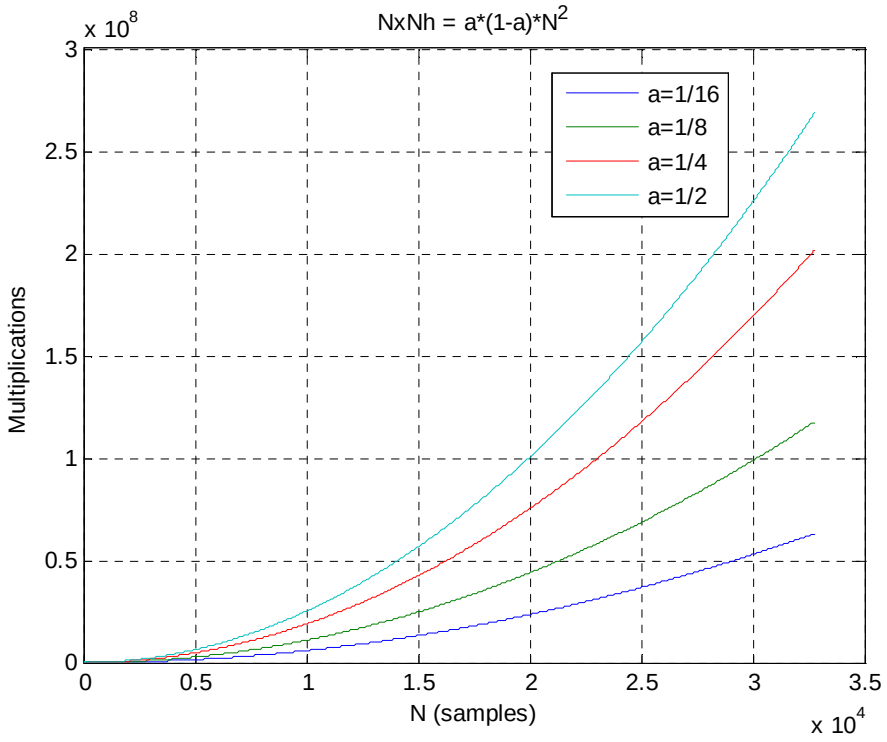

Figure 4.11 Number of Multiplications for different values of α

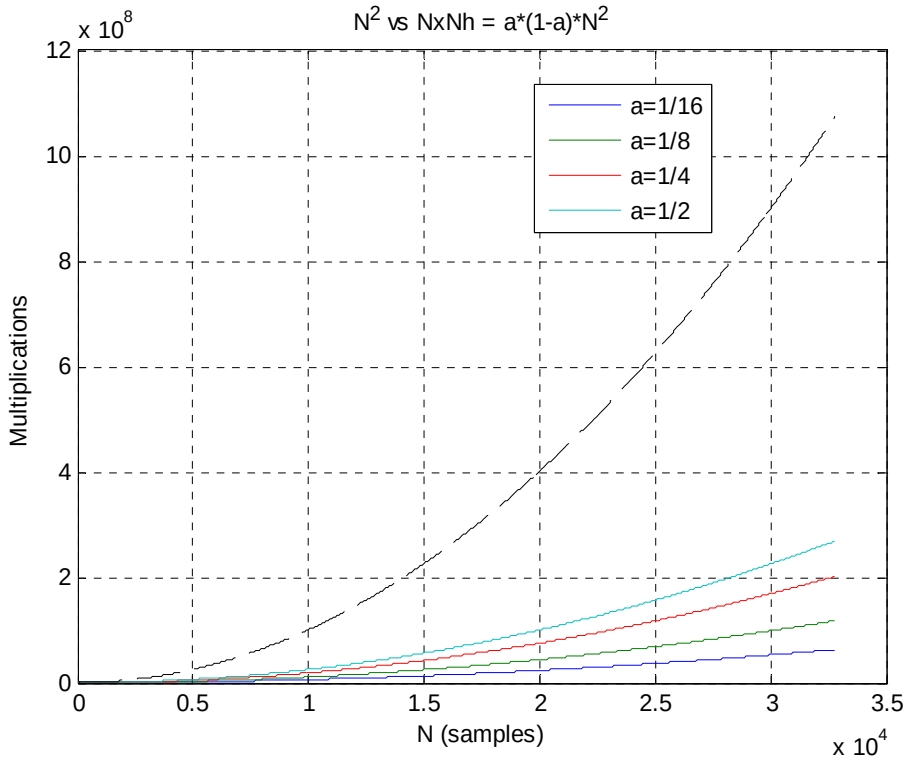

Figure 4.12 Number of Multiplications with and without the reduction method.

To measure the reduction in the number of multiplications of the CSTFT and CAF algorithms, we can define a matrix multiplication reduction index:

$$\mu_R = (1-R)\cdot 100\%$$

Where $R = \frac{N_x N_h}{N^2}$, for the examples of section 4.2 and 4.3 we only perform $N_x \cdot N_h = 6 \cdot 2 = 12$ multiplications instead of $N^2 = 8^2 = 64$.

$$R = \frac{12}{64} = 0.1875$$
$$\mu_R = (1-R)\cdot 100 = 81.75\%$$

That means that the number of multiplications has been reduced in a 81.75%. Figure 4.13 shows the multiplication reduction index curve. The reduction percentage becomes smaller as α increase.

$$R = \frac{N_x \cdot N_h}{N^2} = \frac{\alpha \cdot (1-\alpha) \cdot N^2}{N^2} = \alpha - \alpha^2$$
$$\mu_R = (1-R)\cdot 100 = \left(\alpha^2 - \alpha + 1\right)\cdot 100 \; ; \; 0 < \alpha < \frac{1}{2}$$

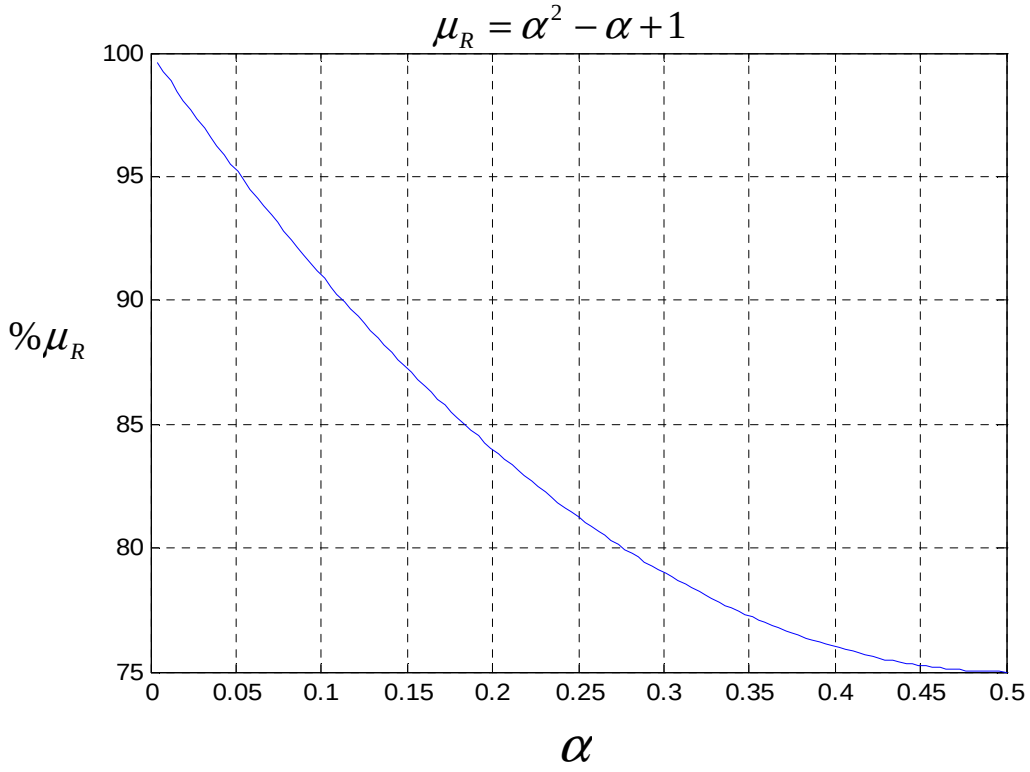

Figure 4.13 Multiplication Reduction Index Curve

5 Experimental Results

This chapter presents the experimental results. The algorithms under test were: the CSTFT, CAF and, the EPLL; all of them implemented in hardware, using a TMS320C6713.

5.1 CSTFT and CAF

The CSTFT and CAF were implemented using the computational framework developed in chapter 3. The algorithms were proved for some time-frequency signals, and performance measurements were taken. Figure 5.1 shows a 60 Hz sinusoidal signal and a Hanning filter.

These signals were used to compute the CSTFT and CAF in figure 5.2. As time advances the frequency maintains constant.

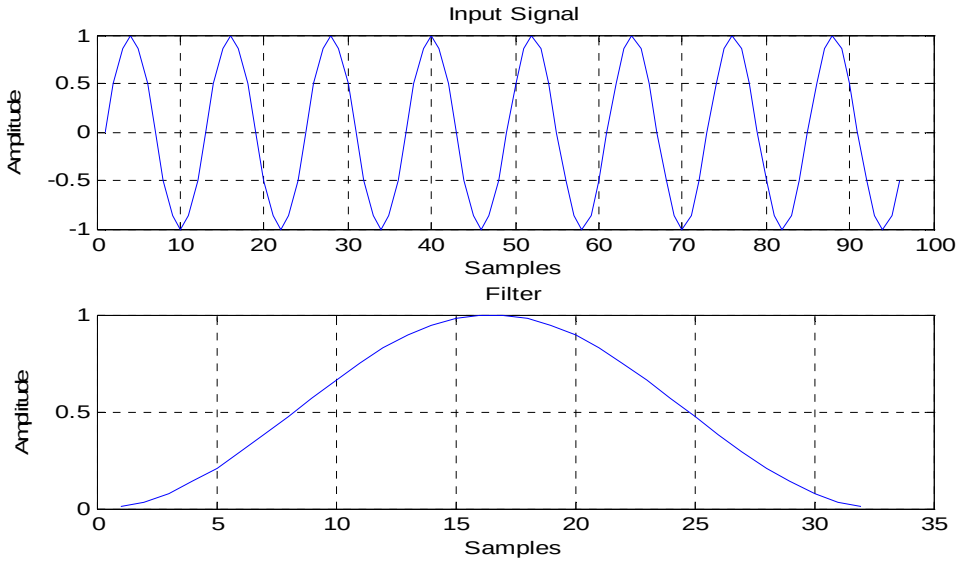

Figure 5.1 60 Hz Sinusoidal signal and hanning window filter

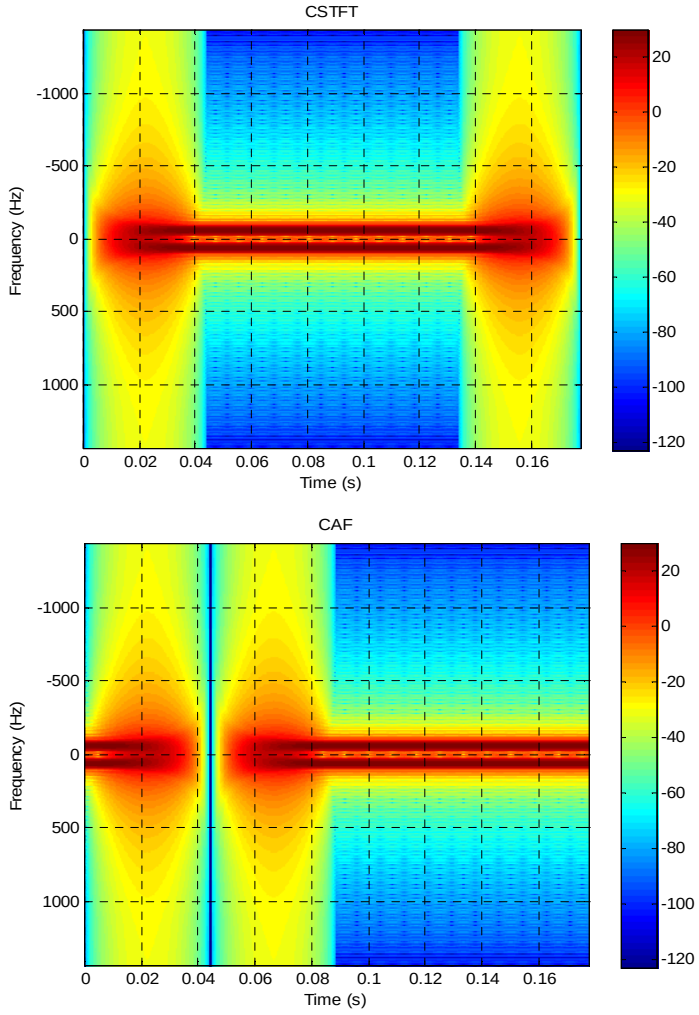

Figure 5.2 CSTFT and CAF for a 60 Hz sinusoidal signal using a hanning window filter

The same procedure has been made with some other possible harmonic signals. Figure 5.3 show the CSTFT for a harmonic signal $x(t)$

Where ,

$$x(t)=\sin(2\pi f_0 t)+\frac{1}{5}\sin(2\pi f_5 t)+\frac{1}{7}\sin(2\pi f_7 t)\ ; f_1=60Hz, f_5=5f_0, f_7=7f_0$$

Figure 5.4 to 5.5 shows the CSTFT for two time-varying frequency signals.

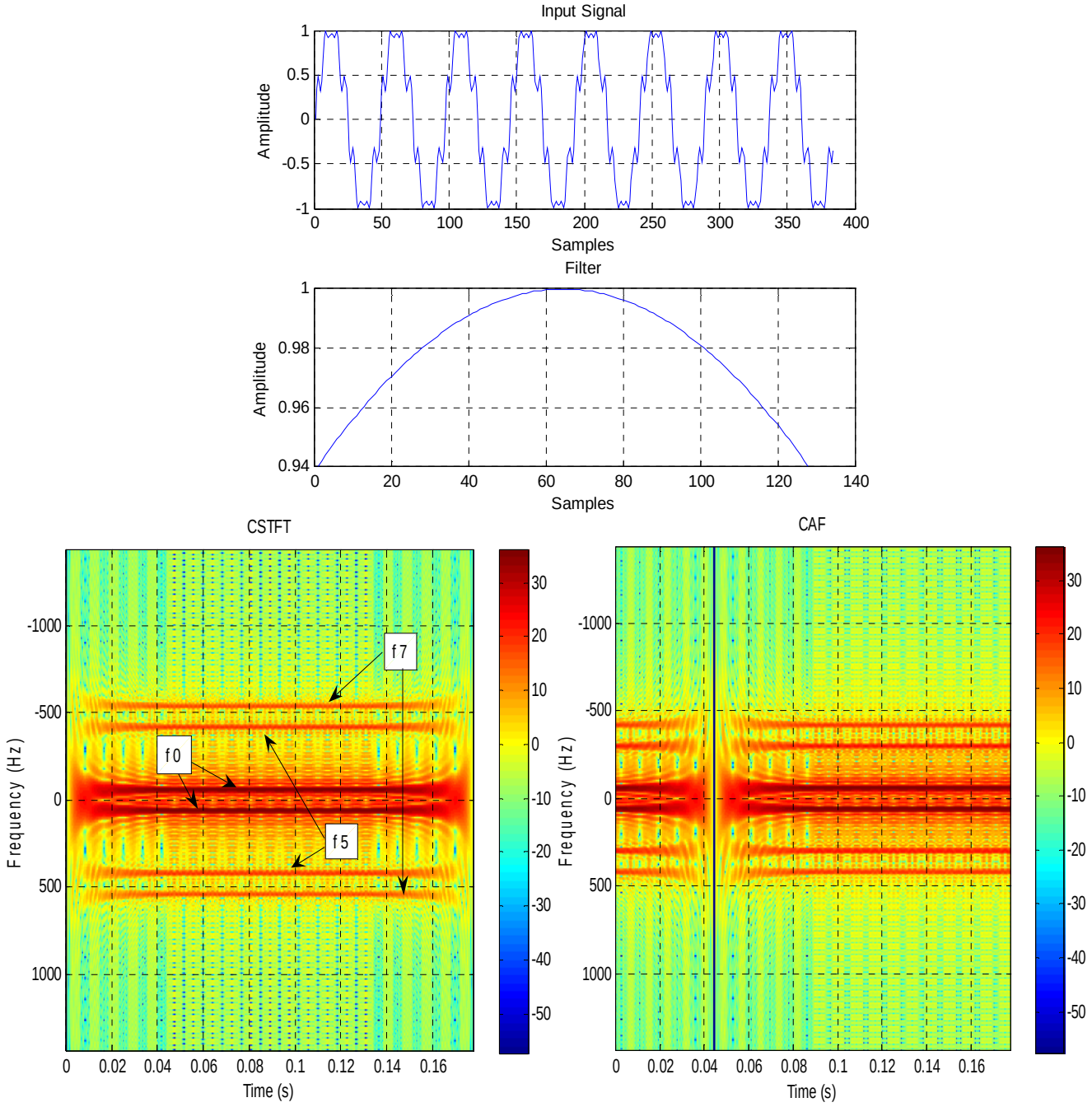

Figure 5.3 CSTFT and CAF , signal containing the 1st ,5th and 7th harmonics with a Kaiser window

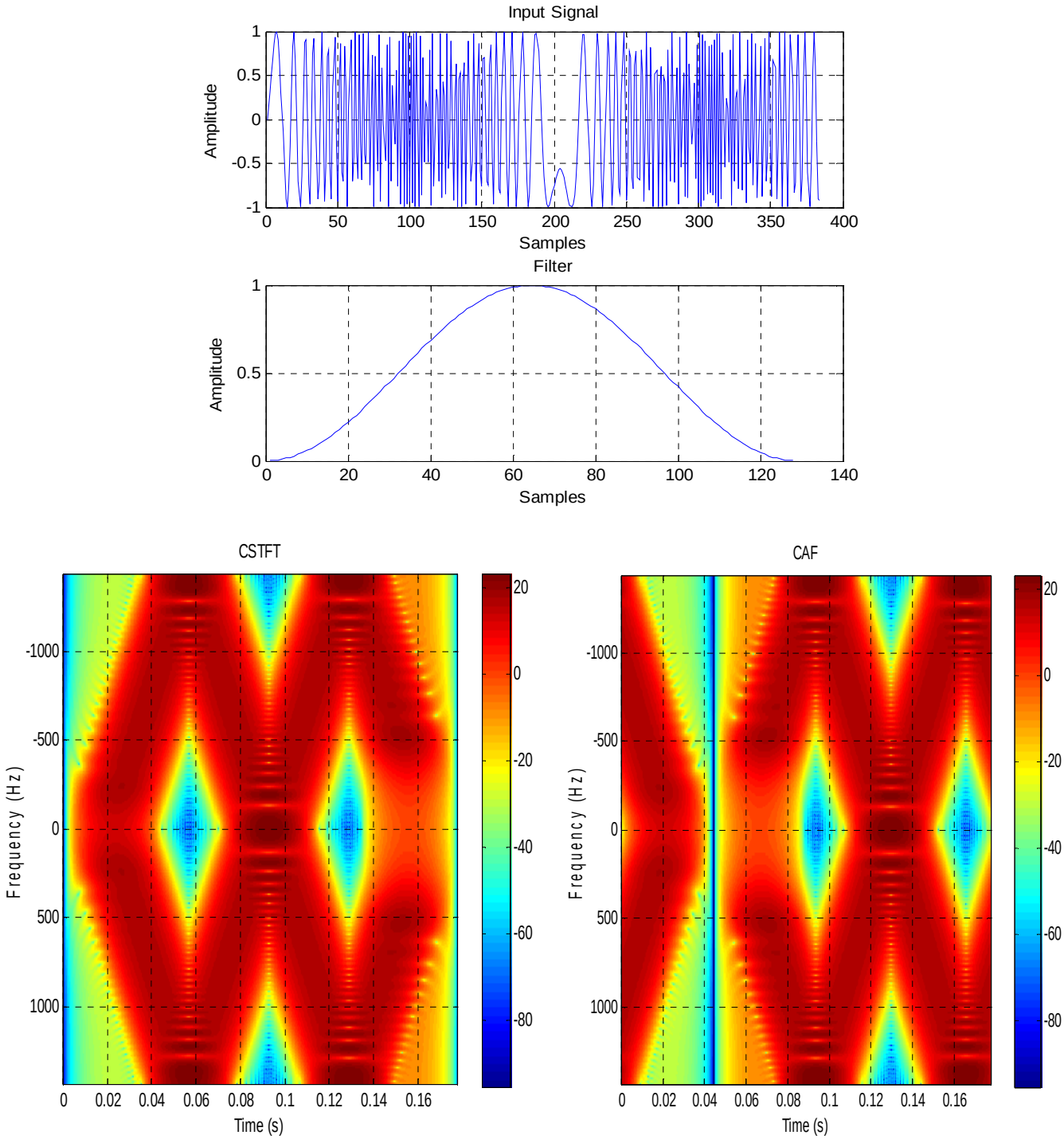

Figure 5.4 CSTFT and CAF for a double chirp signal using a Hanning window.

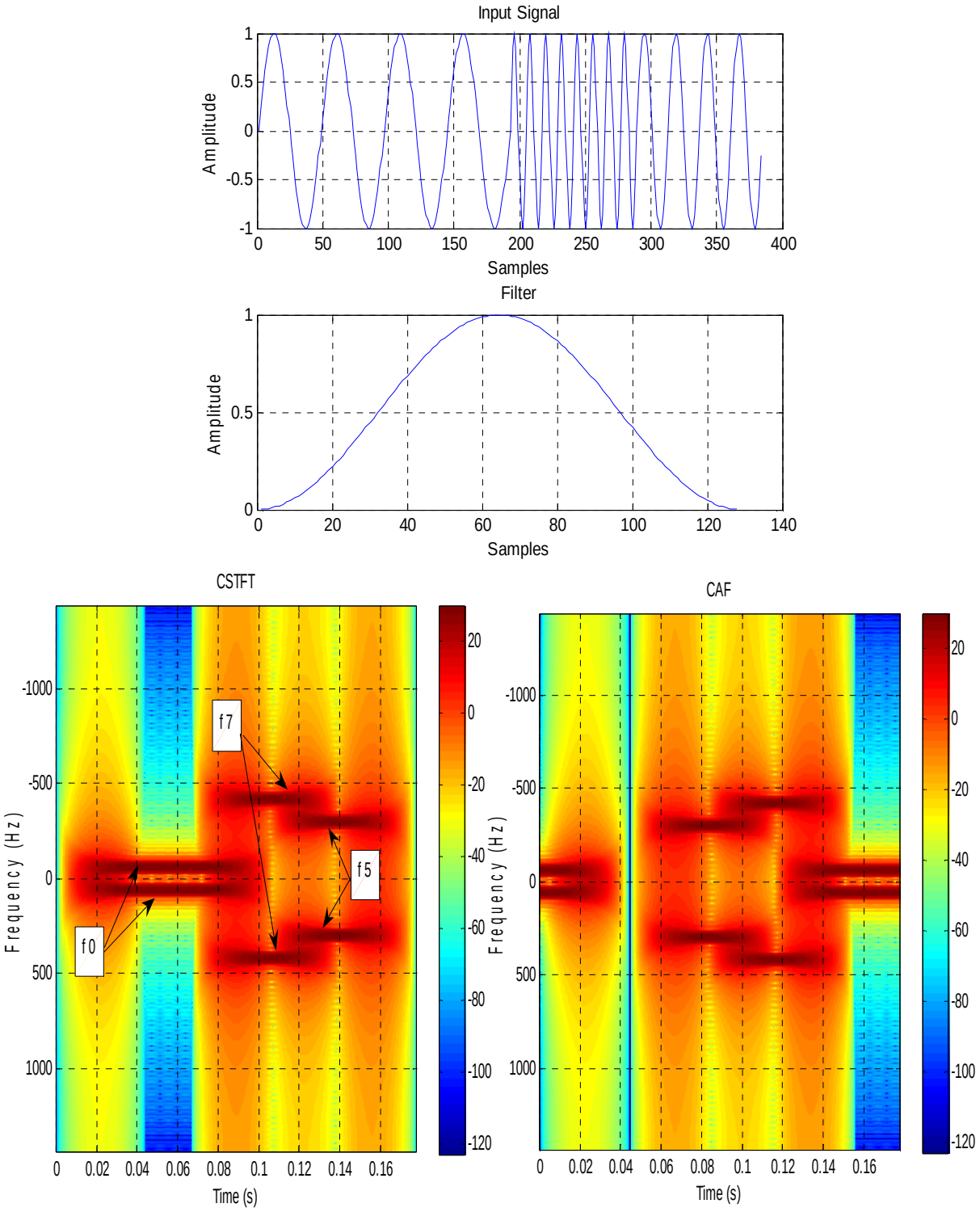

Figure 5.5 CSTFT and CAF for a frequency varying signal.

5.2 Enhanced Phase Locked Loop

The Enhanced phase locked loop presented in chaper 3 was implemented in Matlab/Simulink and a TMS320C6317 DSP. The core unit is shown in Figure 5.6, for a nominal frequency of

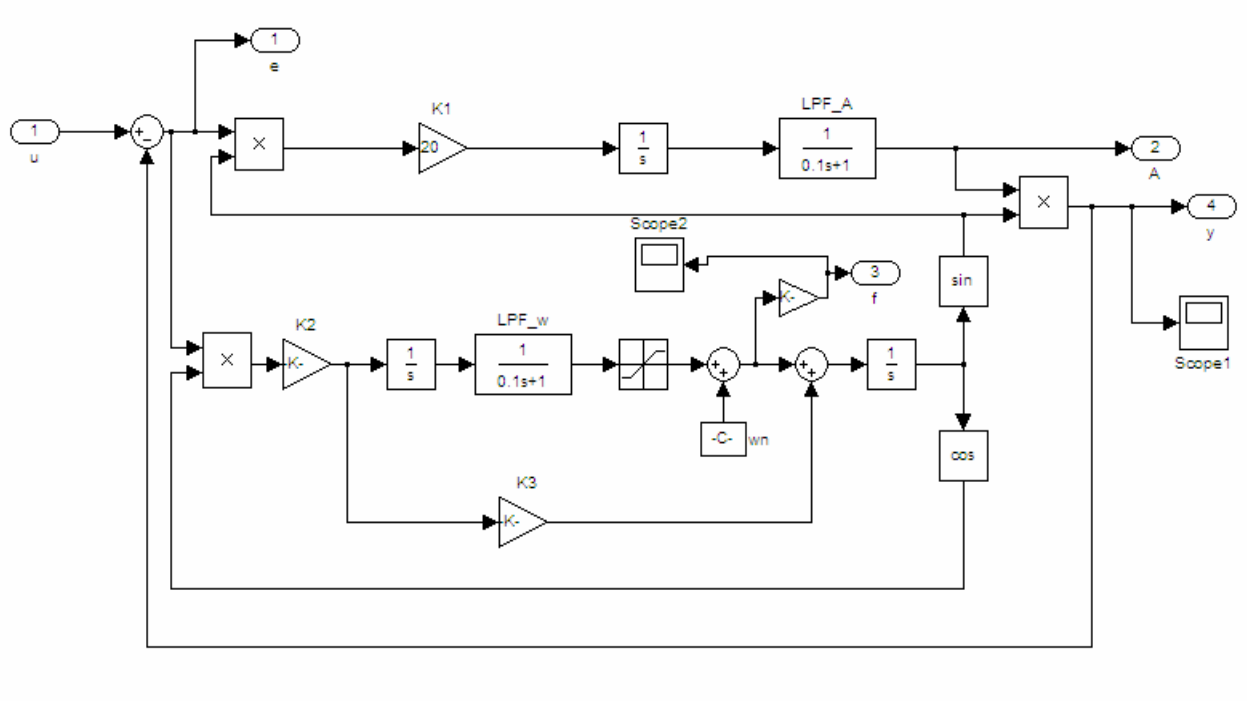

Figure 5.6 EPLL core unit

f = 60 Hz. A cascade configuration to extract different harmonic components is presented in Figure 5.7. A signal

$$u(t) = \sin(2\pi 63t) + \frac{1}{5}\sin(10\pi 63t) + \frac{1}{7}\sin(14\pi 63t) + n(t)$$

Was used to observe the performance of the proposed method, where $n(t)$ represent a Gaussian noise with variance $\sigma^2 = (0.1)^2 / 2$. Figure 5.3 shows the extracted sinusoidal component of 63 Hz.

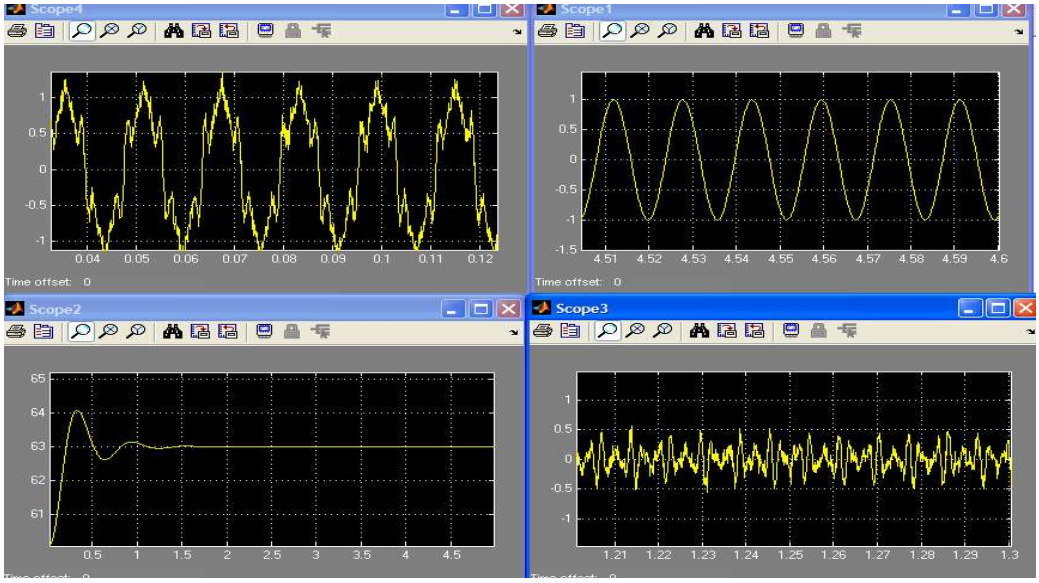

Figure 5.7 EPLL: harmonic input signal, Estimated 60 Hz sinusoidal component, frequency tracking performance and, residual error.

Figure 5.7 shows the input signal $u(t)$ on the upper left, the extracted sinusoidal component on the upper right, the frequency tracking performance and the error signal.

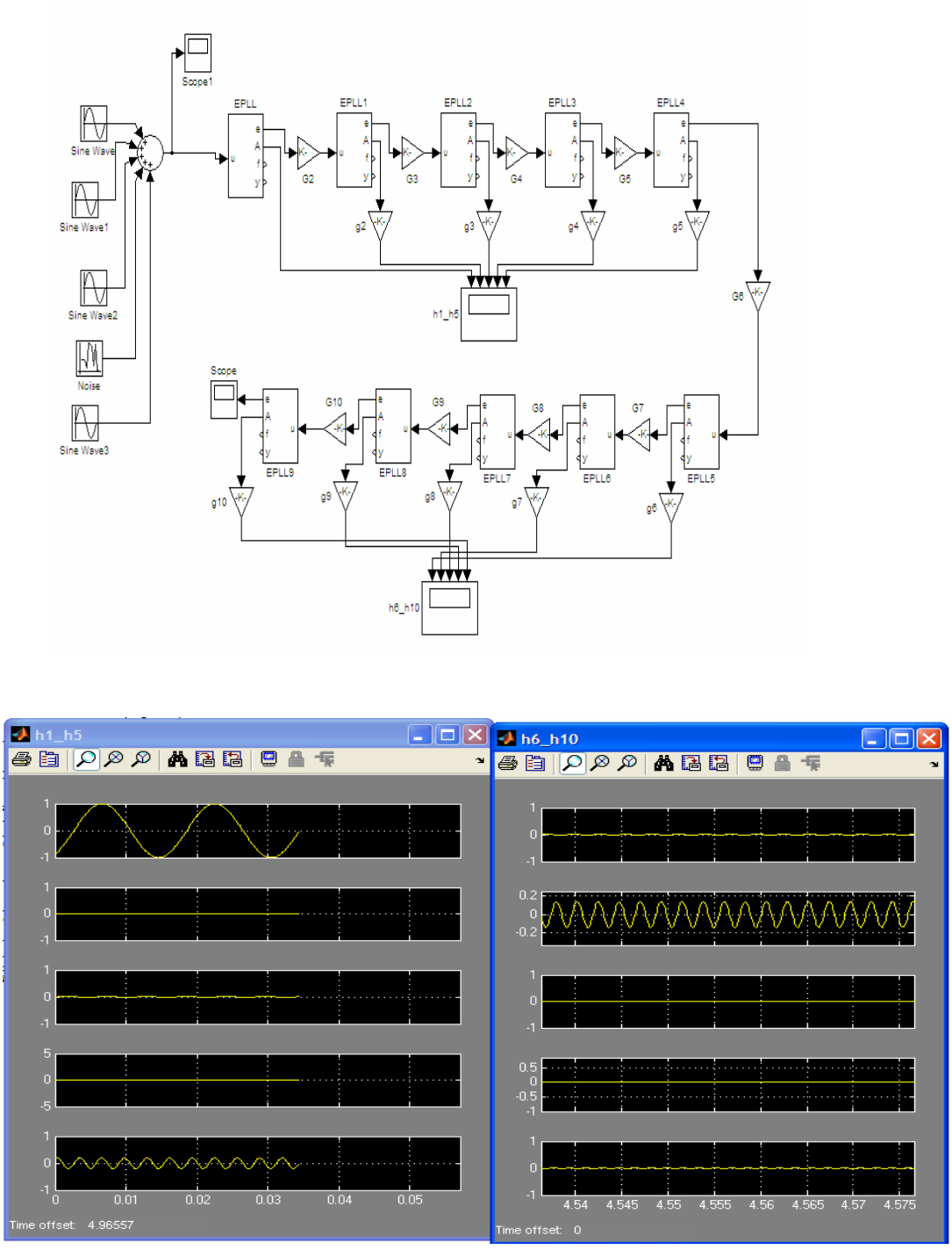

Figure 5.8 EPLL cascade configuration and harmonic estimation.

Figure 5.8 shows the tracking performance of a cascade configuration of ten units. The system clearly extracts the fundamental component and the fifth and seventh harmonic.

The EPLL was implemented on the TMS320C6713, using a discrete version of the third order dynamical system presented in this section. The discrete equations are the following:

$$
\begin{aligned}
A[n] &= A[n-1] + \mu_1 T_s\, e[n-1] \sin\phi[n-1] \\
\omega[n] &= \omega[n-1] + \mu_2 T_s e[n-1] \cos\phi[n-1] \\
\phi[n] &= \phi[n-1] + \omega[n] T_s + \mu_2 \mu_3 T_s\, e[n-1] \cos\phi[n-1] \\
y[n] &= A[n] \sin\phi[n] \\
e[n] &= u[n] - y[n]
\end{aligned}
$$

Where $u[n]$ the discrete input is signal, $e[n]$ and $y[n]$ are the error signal and estimated fundamental/harmonic component respectively. The set composed by $A[n], \omega[n], \phi[n]$ are the discrete version of the third order dynamical system.

The parameters μ_1, μ_2, μ_3 control the convergence rate of the amplitude, frequency and phase equations. The parameter T_s represents the sampling time. Next we consider some examples to observe the performance of this dynamical system on estimating the fundamental component of a time-varying frequency signal. Let we have a signal with a stepped frequency variation as shown in figure 5.9.

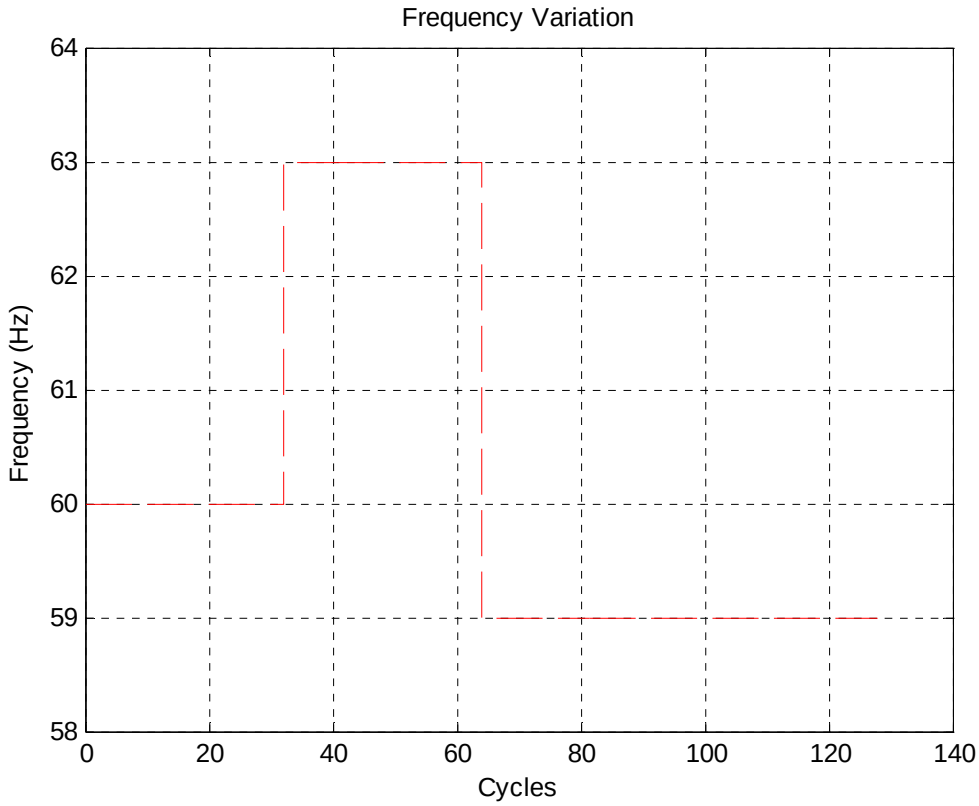

Figure 5.9 Frequency variation of an EPLL input signal

As we can see the frequency has a step change from 60 Hz to 63 Hz and then a negative frequency step from 63 Hz to 59 Hz. The performance of the dynamical system to track this frequency characteristic as well as the signal amplitude is shown in figure 5.10. The dynamical system is able to track the frequency variations. At the moment the frequency steps occurs the system do not track the frequency immediately, some cycles later the systems track the frequency effectively. Figures 5.11-13 show the system performance on some other input signals with different time-varying frequency characteristics.

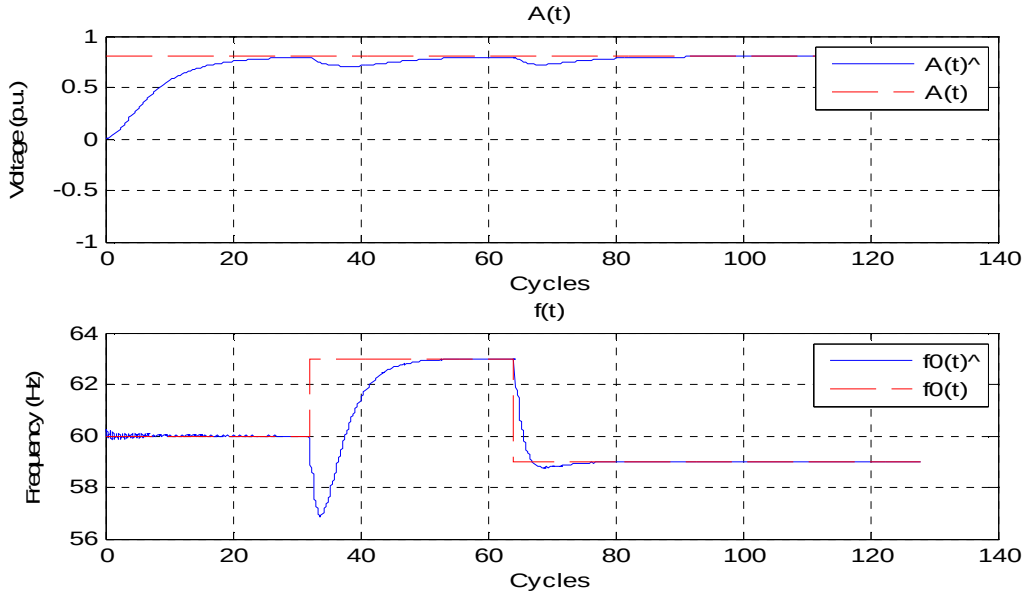

Figure 5.10 EPLL amplitude and frequency tracking performance example

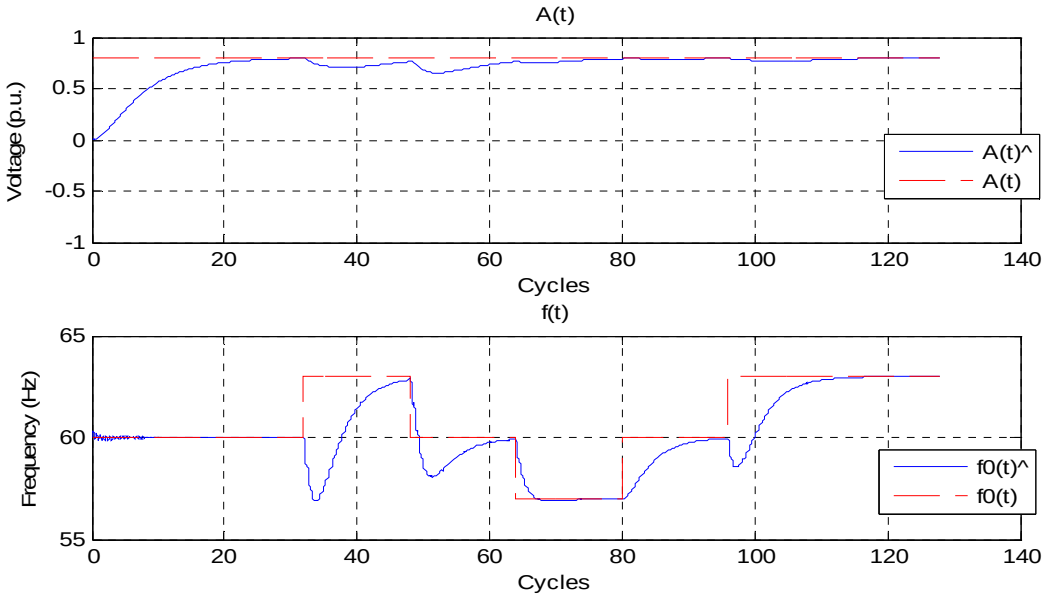

Figure 5.11 EPLL amplitude and frequency tracking performance example

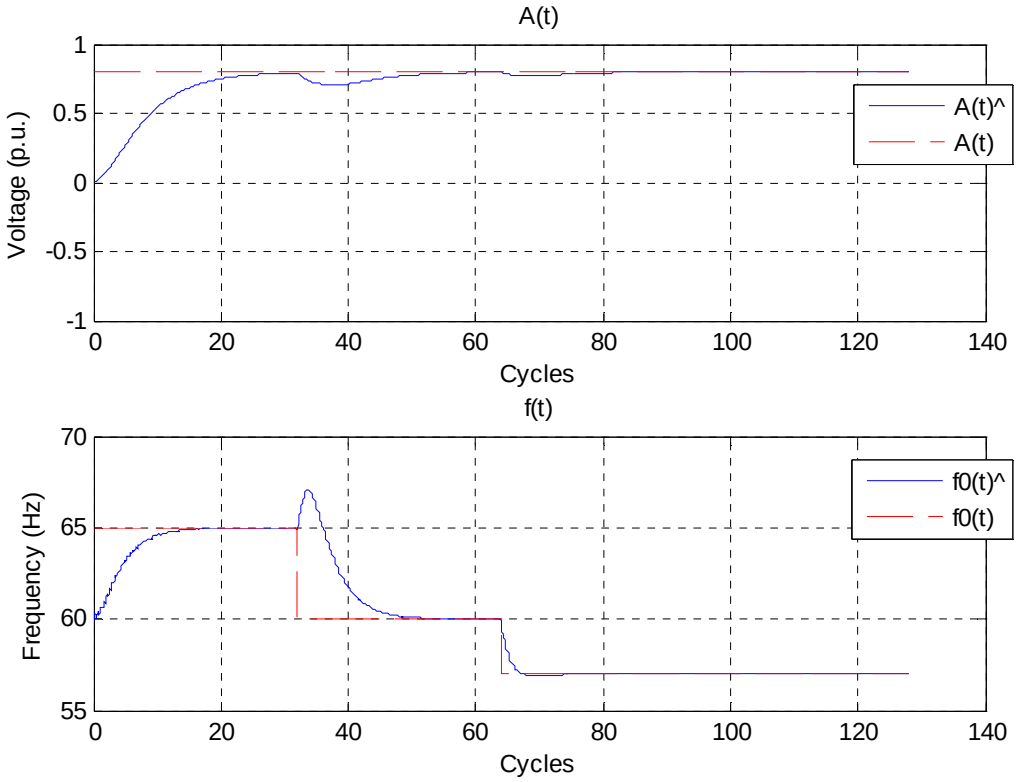

Figure 5.12 EPLL amplitude and frequency tracking performance example

5.3 EPLL Case Study: Six Diode Rectifier Bridge

A six diode rectifier bridge is commonly used at the converter section of a standard three-phase Pulse Width Modulator (PWM) drive. This kind of circuit introduces certain harmonics like 5^{th}, 7th, 11th, 13th, 17th, 19th, 23th and, 25^{th} as shown in figure 5.13. In general the number of harmonic orders in a wave for a six diode bridge is given by:

$$h = 6k \pm 1$$

Where h is the harmonic order and k is an integer.

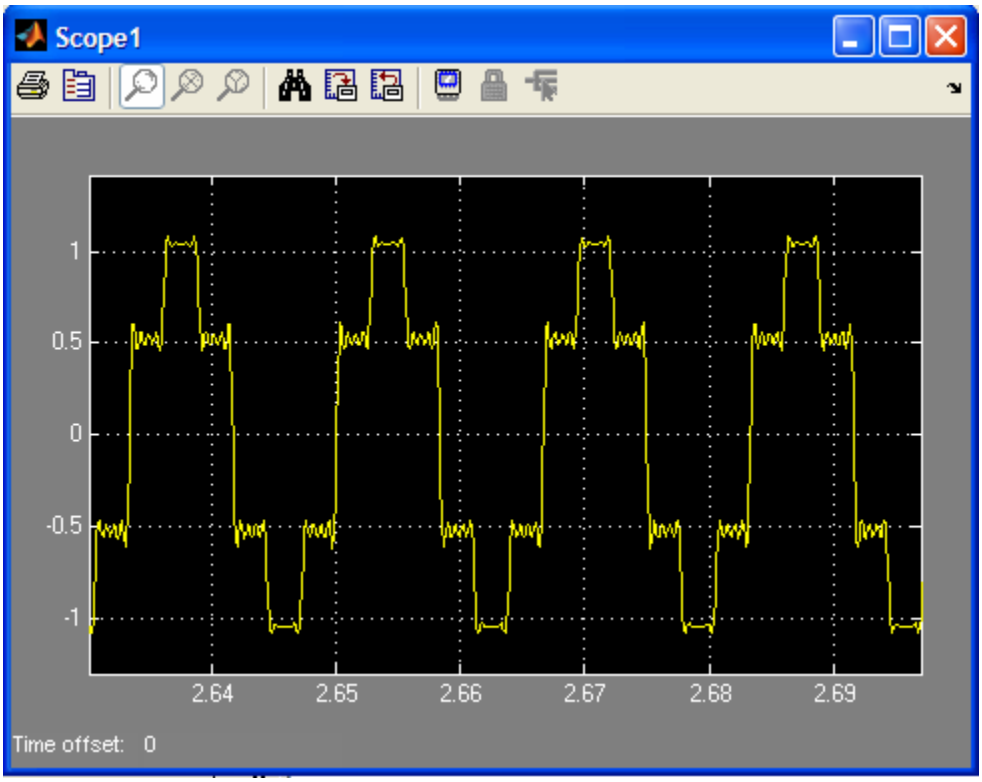

Figure 5.13 Six Pulse Rectifier Signal

An EPLL cascade configuration has been used to detect each one of the signal components of the six pulse rectifier signal. Figure 5.14 shows the cascade configuration using 25 EPLL core units. The orange blocks represent the core units that extract the corresponding sinusoidal components. Figure 5.15- 17 shows the extracted sinusoidal components.

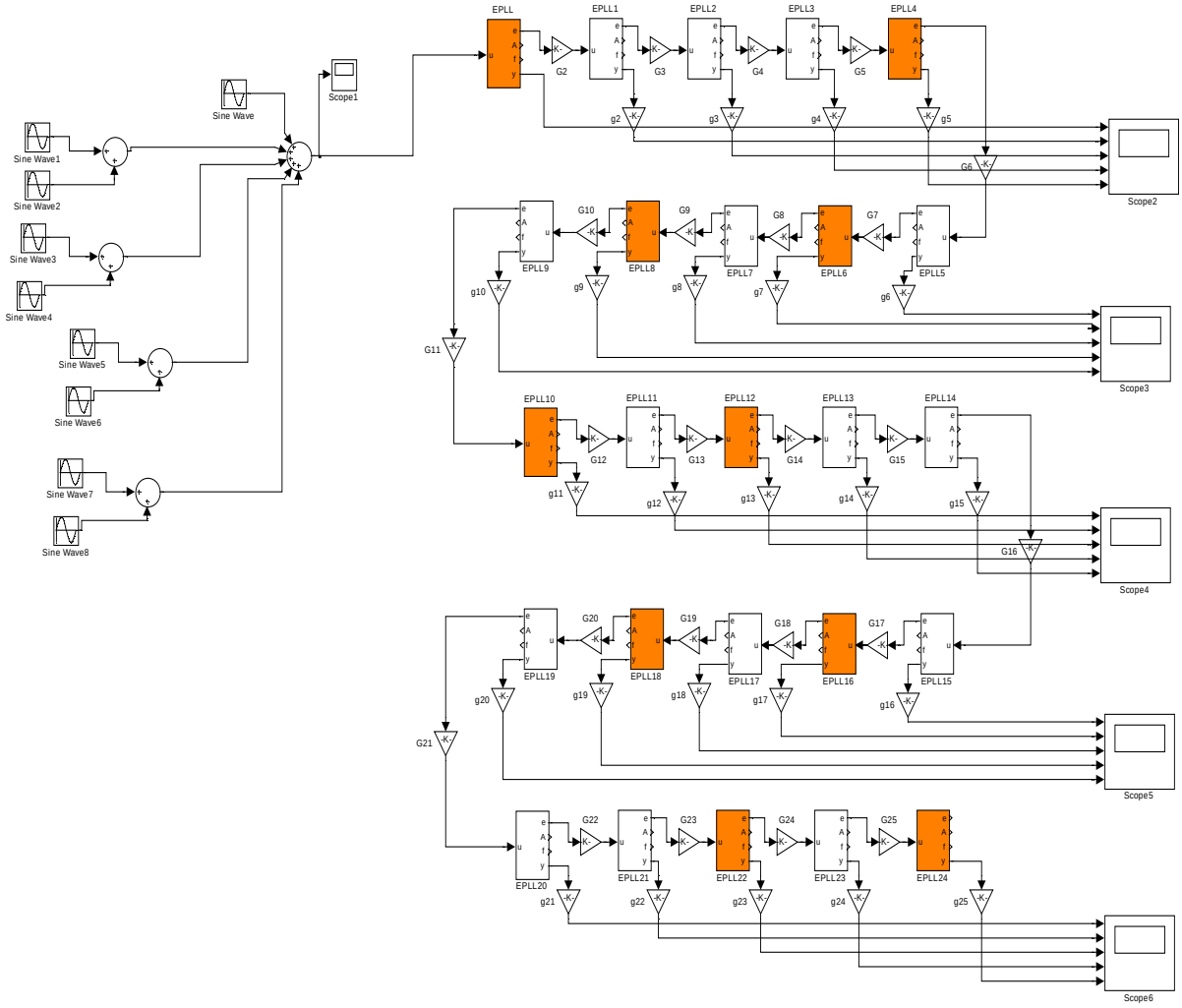

Figure 5.14 EPLL Cascade Configuration using 25 core units

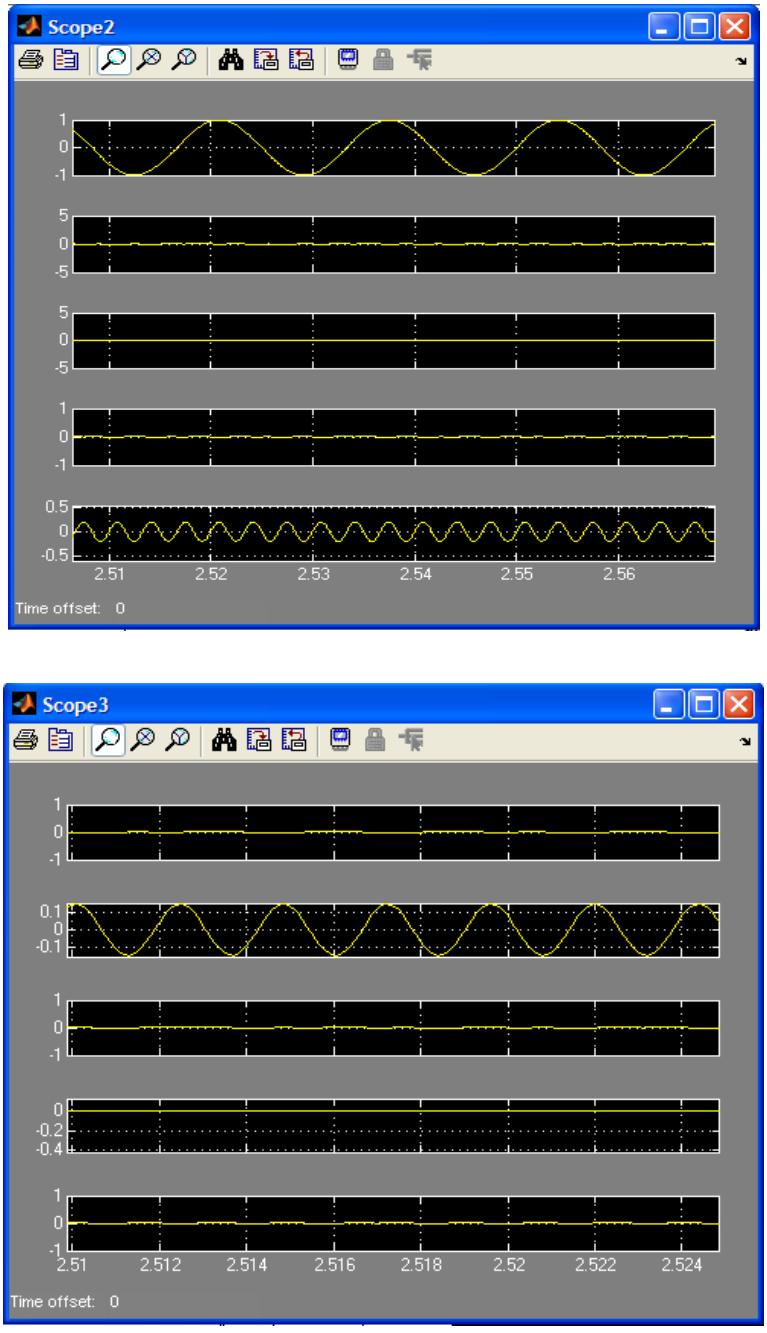

Figure 5.15 EPLL harmonic estimators, 1st – 10th. The fundamental, 5th and 7th component are present.

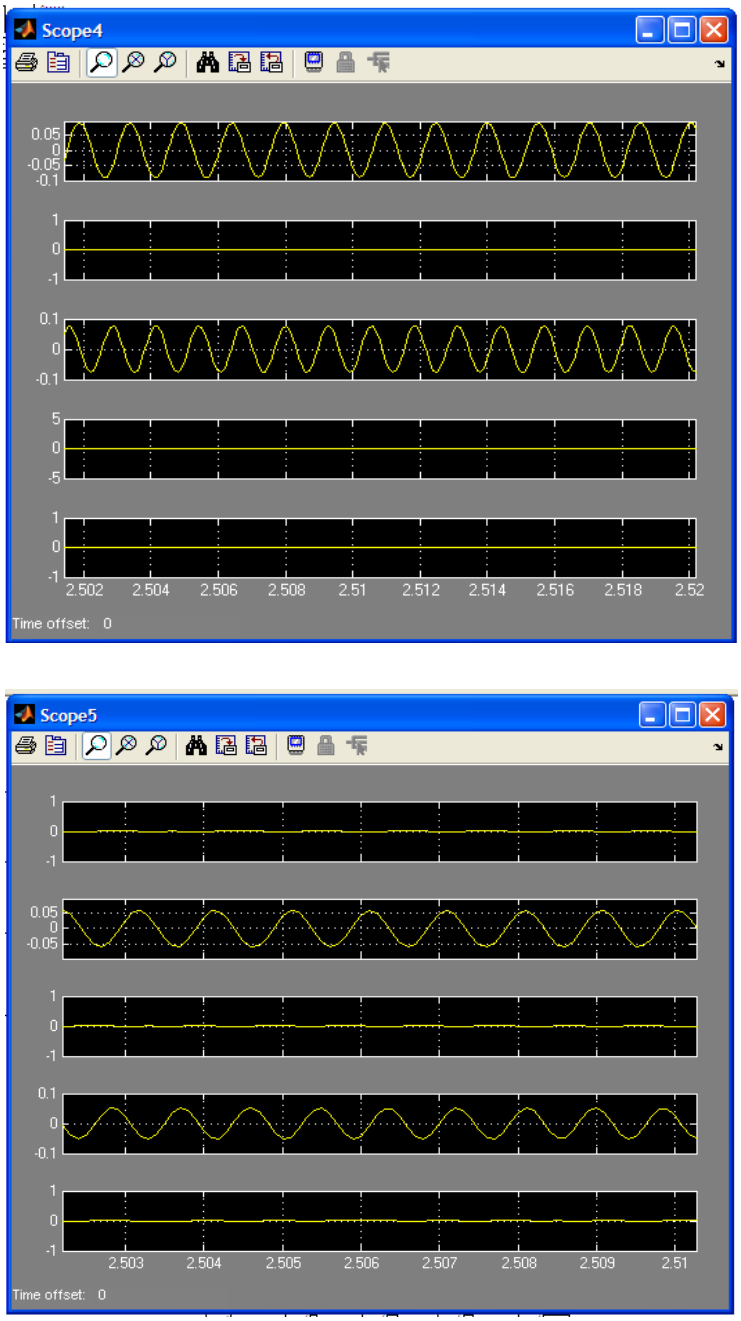

Figure 5.16 harmonic estimators, 11^{th} – 20^{th}. The 11^{th}, 13^{th}, 17^{th} and 19^{th} component are present.

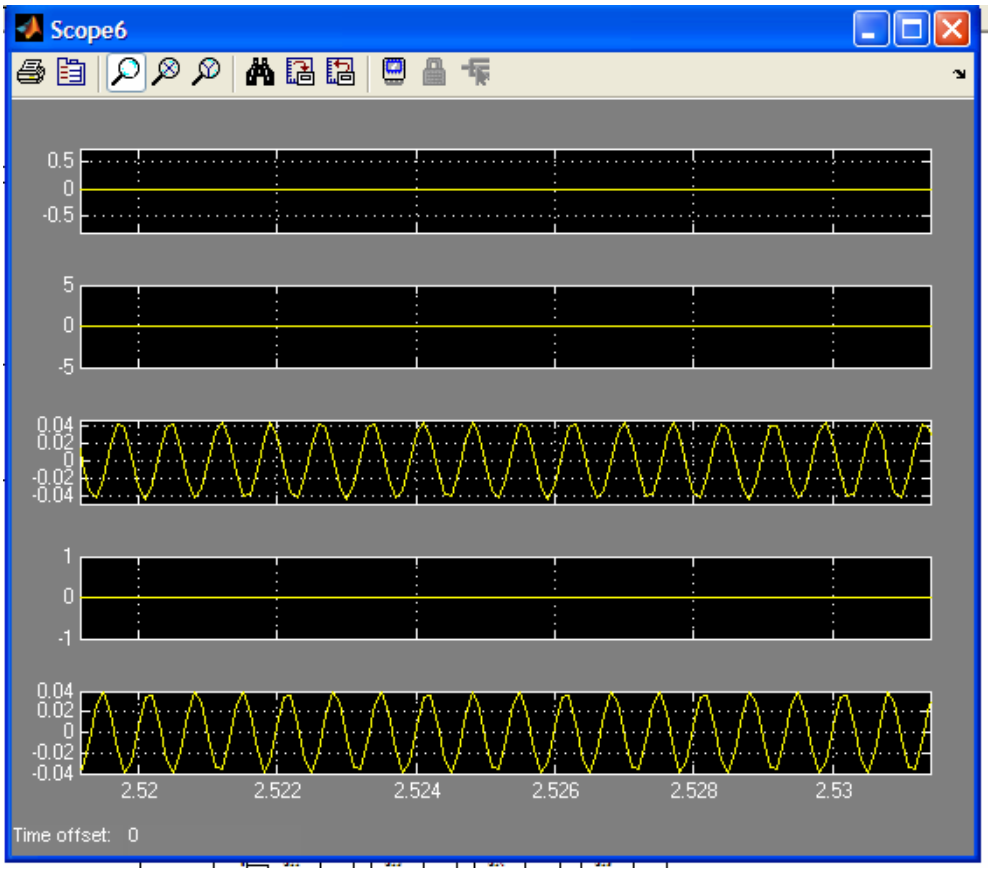

Figure 5.17 EPLL harmonic estimators, 21st – 25th. The 23rd and 25th components are present.

6 Conclusions and Future Work

Efficient time-frequency algorithms have been developed using a signal algebra operators approach for power quality signal analysis. A hardware system for the automated processing of sensor based signal data information generated in industrial and residential applications has been designed, developed and implemented. The system performs time-frequency analysis of power line signals and is named Signal Imaging and Monitoring for Power Line Environment (SIMPLE). A SIMPLE prototype has been tested with generated and acquire data. A special signal conditioning system, that converts a power line voltage level to audio signal range has been developed, which could be the first step in a new concept of power quality monitoring that we call “Hearing the Power Lines”. An Enhanced Phase Looked Loop algorithm has been simulated and implemented in Simulink and a DSP unit. The EPLL has been tested using generated signals.

As future work, we can extend the signal operators algebra to other time-frequency representations like the Wigner-Ville Distribution, Wavelets and the Reduced Interference Distribution (RID). To use different hardware computational structures like FPGA’s or multiple core DSPs. To improve the Signal Conditioning System (SCS) dynamic range using active components. To use an embedded computer instead of a personal computer for the SIMPLE prototype.

REFERENCES

[Arti04] Artioli M. Low Cost DSP Based Equipment for the Real Time Detection of Transients in Power Systems. IEEE Transactions on Instrumentation and Measurements. Vol 53, NO. 4, August 2004.

[Bail05] Bailey D., Harmonic Distortion Measurements Using Spectral Warping. Proceedings of the Third IEEE International Workshop on Electronic Design, Test and Applications (DELTA) 2005.

[Boll06] Bollen M. H. J and Gu I. Y. H., Signal Processing of Power Quality Disturbances. 2006

[Diaz06] Diaz A.H and Rodriguez D., One Dimensional Cyclic Convolution Algorithms with Minimal Multiplicative Complexity. ICASSP 2006

[Dura03] Durak L. and Arikan O. Short Time Fourier Transform: Two Fundamental Properties and an Optimal Implementation. IEEE Transactions on Signal Processing, VOL 51, NO. 5, May 2003.

[Gu00] Gu Y.H. and Bollen M.H.J., Time-Frequency and Time-Scale Domain Analysis of Voltage Disturbances. IEEE Transactions on Power Delivery, VOL 15, NO.4, October 2000.

[Herr00] Herraiz S. Power Quality indices for Aperiodic Voltages and Currents. IEEE Transactions on Power Delivery, VOL 15, NO 2, April 2000.

[Heyd98] Heydt G.T. and Jewell W.T. , Pitfalls of Electric Power Quality Indices. IEEE Transactions on Power Delivery, VOL 13, NO. 2 , April 1998 .

[Jeon92] Jeong J. et al. , Kernel Design for Reduced Interference Distributions. IEEE Transactions on Signal Processing. VOL 40, NO . 2, February 1992.

[Kari02] Karimi-Ghartemani M, and Iravani R. A Nonlinear Adaptive Notch Filter for Online Signal Analysis in Power Systems: Applications. IEEE Transactions on Power Delivery, Vol 17,NO 2, April 2002.

[Kari03] Karimi-Ghartemani M. and Iravani R. A Signal Processing Module for Power System Applications. IEEE Transactions on Power Delivery , Vol 18, NO. 4, October 2003.

[Kari04_1] Karimi-Ghartemani M. and Ziarani A. K. A Nonlinear Time-Frequency Analysis Method. IEEE Transactions on signal processing vol 52, NO. 6, June 2004.

[Kari04_2] Karimi-Ghartemani M et al. A Magnitude/Phase-Locked Loop System Based on Estimation of Frequency an In-Phase/Quadrature-Phase Amplitudes. IEEE Transactions on Industrial Electronics, Vol 51,NO. 2, April 2004.

[Kari04_3] Karimi-Ghartemani et al. A Signal Processing System for Extraction of Harmonics and Reactive Current of Single Phase Systems. IEEE Transactions on Power Delivery, Vol 19, NO. 3 July 2004.

[Kari05] Karimi-Ghartemani M. and Iravani R. Measurement of Harmonics/Inter-harmonics of time-varying frequencies. IEEE Transactions on Power Delivery, Vol 20, NO. 1, January 2005

[Lin05] Lin T. and Domijan A. On Power Quality indices and Real Time Measurements. IEEE Transactions on Power Delivery. Vol 20, NO. 4, October 2005.

[Lu05] Lu. S., Applications of DFT filter bank to power frequency harmonic measurement. IEEE Proc. Gener. Transm. Distrib. , VOL 152, NO.1 January 2005.

[McNa07] McNamara D.M and Ziarani A.K. A New Technique of Measurement of Nonstationary Harmonics. IEEE Transactions on Power Delivery Vol 22, NO 1 January 2007.

[Rive05] Rivera I, et al. A Time-Frequency Signal Analysis System for Power Quality Assessment. IEEE MWSCAS 2005 August 2005

[Shin99] Shin Y.J. et al . , Time –Frequency Analysis of Power System Disturbance Signals for Power Quality. 1999.

[Shin06] Shin, Y.J. et al., Power quality indices for transient disturbances, Power Delivery, IEEE Transactions on Power Delivery, Vol.21, NO. Jan. 2006.

[Std95] IEEE Std 1159-1995. IEEE Recommended Practice for Monitoring, Electric Power Quality. 1995

[Teix95] Texeira M. and Rodriguez D., A class of Fast Cyclic Convolution Algorithms Based on Block Pseudocirculats. IEEE Signal Processing Letters VOL.2 , NO 5, May 1995.

[Wats00] Watson N.R et al., A Global Power Quality Index for Aperiodic Waveforms. 2000.

[Wang04_1] Wang M. and Mamishev A.V. Classification of Power Quality Events Using Optimal Time Frequency Representations, Part 1: Theory. IEEE Transactions on Power Delivery, Vol 19, NO. 3, July 2004.

[Wang04_2] Wang M. et al. Classification of Power Quality Events Using Optimal Time Frequency Representations, Part 2: Application. IEEE Transactions on Power Delivery, Vol 19, NO. 3, July 2004.

[Ziar02] Ziarani A.K. et al. Experimental Verification of a Novel Method of extraction of nonstationary sinusoids.

.

[Ziar05] Ziarani A.K. and Karimi-Ghartemani M. On the equivalence of Three Independently Developed Phase-Locked Loops. IEEE Transactions on Automatic Control, Vol 50,NO 12 December 2005.

APPENDIX A. MATLAB PROGRAMS

```
%EPLL.m
%-------------------------------------------------------------------------
 f = 60;
% w = 2*pi*f;
N = 4096; % numero de puntos de la senal
m = 128; % numero de cyclos de la senal
%Fs = (N/m)*f;
Fs = 8000;
Ts = 1/Fs;
t = 0:Ts:(N-1)*Ts;
%u = 0.8*sin(w*t)+ 0.5*sin(2*w*t)+ 0.1*rand(1,N);
%fv = [60*ones(1,N/4),(60+3)*ones(1,N/4),(60-1)*ones(1,N/2)];
%fv = [65*ones(1,N/4),(60)*ones(1,N/4),(57)*ones(1,N/2)];
fv = 63;
p = 0.8*sin(2*pi*fv.*t);
K1 = 20;
K2 = 2000;
K3 = 0.08;
wn = 2*pi*60;
%-------------------------------------------------------------------------
  N = length(y);
  Ts = 1/Fs;
  V = N*Ts;
  t = 0:Ts:V-Ts;
 %-------------------------------------------------------------------------
 u = p;
 A(1) = 0;
 phi(1) = 0;
 w(1) = wn;
 y(1) = 0;
 ys(1)= 0;
 e(1) = u(1);
 tau = 0.04;
 Sa(1) = 0;
 %-------------------------------------------------------------------------
 for i = 2:N

    A(i) = A(i-1) + K1 * Ts* e(i-1) * sin(phi(i-1));
    Sa(i) = Sa(i-1)+(A(i-1)-Sa(i-1))*Ts/tau;
    w(i) = w(i-1) + K2 * Ts * e(i-1) * cos(phi(i-1));
    phi(i) = phi(i-1)+ w(i-1)*Ts + K2 * K3* Ts * e(i-1) * cos(phi(i-1));
    y(i) = A(i) * sin(phi(i));
    ys(i)= Sa(i)*sin(phi(i));
    e(i) = u(i) - y(i);
 end
ref = 0.8*ones(1,N);
subplot(2,1,1)
plot(t*f,Sa,'b',t*f,ref,'r--'),grid
title('A(t)')
xlabel('Cycles')
```

```
ylabel('Voltage (p.u.)')
%axis([0,140,-1,1])
legend('A(t)^','A(t)')
subplot(2,1,2)
plot(t*f,w/(2*pi),'b',t*f,fv,'r--'),grid
title('f(t)')
xlabel('Cycles')
ylabel('Frequency (Hz)')
legend('f0(t)^','f0(t)')

figure
plot(t*f,fv,'r--'),grid
%axis([0,140,58,64])
title('Frequency Variation')
xlabel('Cycles')
ylabel('Frequency (Hz)')

figure
plot(t,u),grid
```

```
function X = dstft(x,h,L,Fs)
% Discrete Short Time Fourier Transform
% Transform Method
%-------------------------------------------------------------------------

Nx = length(x);
Nh = length(h);
N = 2^ceil(log2(Nx+Nh-1));
xp = [x,zeros(1,N-Nx)];
hp = [h,zeros(1,N-Nh)];

Ts = 1/Fs;
V = (N/L)*Ts;
t = 0:Ts:V-Ts;
f =-Fs/2:1/V:Fs/2-1/V;

hf = fliplr(hp(2:N));
hr = [hp(1),hf];

for m = 0:N/L-1
hm(m+1,:) = circshift(hr,[0,m*L]);
xm = xp.*hm(m+1,:);
Xm = fft(xm);
X(m+1,:) = fftshift(abs(Xm));
end

%-------------------------------------------------------------------------
```

```
subplot(2,1,1)
imagesc(t,f./Fs,X.'),grid
title('DSTFT')
xlabel('Time (s)')
ylabel('Normalized Frequency')
subplot(2,1,2)
plot(t,xp,'b--'),grid
title('Signal')
xlabel('Time (s)')
ylabel('Amplitude')

figure
mesh(hm)
```

```
function A = daf(x,h,L,Fs)
% Discrete Ambiguity Function
% Transform Method
%-------------------------------------------------------------------------

A = [];
Nx = length(x);
Nh = length(h);
N = 2^ceil(log2(Nx+Nh-1));
xp = [x,zeros(1,N-Nx)];
hp = [h,zeros(1,N-Nh)];

Ts = 1/Fs;
V = (N/L)*Ts;
t = 0:Ts:V-Ts;
f =Fs/2-1/V:-1/V:-Fs/2;


for m = 0:N/L-1
hm = circshift(hp,[0,mod(-m*L,N)]);
xm = xp.*hm;
Am = fft(xm);
A(:,m+1) = abs(Am);
end
A = fftshift(A);
contour(t,f./Fs,A),grid
```

```
% Programa que crea los header files necesarios para usar
% el dsp TMS320C6713

function [x,hp,xpadd,y]= dsp6713(Nh,Nxy,f)
clc
m = 3;
V= m/f;
```

```
Ts = V/Nxy;
t = 0:Ts:V-Ts;
w = 2*pi*f;
x = cos(w*t);
h = hanning(Nh).';
a = 200000;
wch = 2*pi*(a*t + f);
y = cos(wch.*t);
N = 2^ceil(log2(Nh+Nxy));

x = x.';
y = y.';
hp = [h,zeros(1,N-Nh)].';
xpadd = zeros(1,N).';
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% This Program identifies the number of overvoltages  %
% for long duration  variations in the line.          %
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

pu=117;  %Voltage Nominal Value to normalize the measurements
Tp=1/60; %Standard Time Period
cycles=(N*Ts)/Tp; %Number of Cycles
ovvolt=0; %Initializing a variable to count the number of overvoltages
n=1;
[mar,olg]=size(x);
for n=1:olg-1,
if x(n) >= 1.1*pu & x(n)<= 1.2*pu
   if cycles > 3600
       ovvolt=ovvolt+1;
   end
end
n=n+1;
end
longdv_over

set(overvolt,'string',num2str(ovvolt));
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% This Program identifies the number of undervoltages   %
% for long duration  variations in the line.            %
%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

pu=117;  %Voltage Nominal Value to normalize the measurements
Tp=1/60; %Standard Time Period
cycles=(N*Ts)/Tp; %Number of Cycles
undvolt=0; %Initializing a variable to count the number of undervoltages
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% This program calculates the Oscilatory Transients in the line for
% Low Frequency. The Energy Ratio is then calculated, where
%                    ER = (sum |y[n]|^2)/(sum |x[n]|^2)
% If the ER is 5% or higher, we conclude that some frequency content was
% found in the line.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
N=1666;
fs=48000; % Sampling Frequency in Hertz
fc1=65; %
fc2=5000;  % Filter Cut-off frequency in Hertz
wc1=fc1/fs; % Filter cut off frequency in radians/sec.
wc2=fc2/fs;
finc=fs/N;
f=-(fs/2):finc:(fs/2)-finc;
Wn=[wc1/fs wc2/fs];  % Filter cut off frequency in rad/samples
n=4;  %filter order
h=fir1(n,Wn,boxcar(n+1));
[H,w]=freqz(h,1,N);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Making the vectors the same length and a power
% of two for the computation of the fft
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

x=[x zeros(1,length(h)-length(x))];
h=[h zeros(1,length(x)-length(h))];
xf=fft(x);
hf=fft(h);
yf=xf.*hf;
y=ifft(yf);

clear ER_s
pu=117;                 %Voltage Nominal Value to normalize the
measurements
Fs=48000;               %Sampling Frequency
Ts=1/Fs;                %Sampling Period
Tp=1/60;                %Standard Time Period
N=1666;                 %Number of Samples
cycles=(N*Ts)/Tp;       %Number of Cycles
n=0;
k=0;
[m,inputs]=size(x);
for n=0:inputs-1,
    X(k)=x(n)^2;
    k=k+1;
end
xs=sum(X);

[p,exits]=size(y);
k=0;
for n=0:exits-1,
```

```
   Y(k)=y(n)^2;
   k=k+1;
   end
ys=sum(Y);
%=========================

ER=ys/xs;
ERabs=abs(ER);
save ERabs ERabs

Osctran_low
set(ER,'string',num2str(ERabs));
if ERabs >= 0.5,
   set(txtER,'string',str2mat('Some Frequency Content was found in the
line'));
else
   set(txtER,'string',str2mat('No Frequency Content was found in the
line'));
end

n=1;
[mar,olg]=size(x);
for n=1:olg-1,
if x(n) >= 0.8*pu & x(n)<= 0.9*pu
   if cycles > 3600
       undvolt=undvolt+1;
   end
end
n=n+1;
end
longdv_under

set(underv,'string',num2str(undvolt));
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% This Program identifies the number of momentary interruptions
% for short duration variations in the line.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

N=1666;
Fs=48000;
Ts=1/Fs;
pu=117;     %Voltage Nominal Value to normalize the measurements
Tp=1/60;    %Standard Time Period
cycles=(N*Ts)/Tp;   %Number of Cycles
momint=0;   %Initializing a variable to count the number of momentary
interruptions
n=1;
[a,b]=size(x);
for n=1:b-1,
if x(n) < 0.1*pu
   if cycles >= 0.5 & cycles <= 180
       momint=momint+1;
```

```
   else
   end
else
end
n=n+1;
end
mom_interruption
set(sdvmomint,'string',num2str(momint));
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% This Program identifies the number of
% temporary interruptions for short duration variations
% in the line.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

pu=117;  %Voltage Nominal Value to normalize the measurements
Tp=1/60; %Standard Time Period
cycles=(N*Ts)/Tp; %Number of Cycles
temint=0; %Initializing a variable to count the number of temporary
interruptions
n=1;
[m,p]=size(x);
for n=1:p-1,
if x(n) < 0.1*pu
    if cycles > 180 & cycles <= 3600
    temint=temint+1;
    end
end
n=n+1;
end
temp_interruption

set(tempint,'string',num2str(temint));
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% dcoffset.m
% Program to calculate the dc component
% of a signal.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

dcvalue=mean(x);
save dcvalue dcvalue
DCoffset
set(dc_value,'string',num2str(dcvalue));
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% harmonics.m
% Program to calculate the harmonics of a signal
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

Fs=48000;
N=1666;
finc=Fs/N;
fho=60;    %The value of the first harmonic is 60 Hz.
Xf=fft(x);
%Kho=fho/finc;
%Xf(Kho+1);
numharm=0;
for m = 1:N,
   fhm=m*fho;
   Kho=fhm/finc;
   if Kho>length(Xf),
     results(17,4)=0;
    else
   Harm=Xf(round(Kho+1));
   if Harm ~= 0
    numharm=numharm+1;
   end
end
m=m+1;
end
save numharm numharm
wave_Harmonics
set(num_harm,'string',num2str(numharm));
```

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% interharmonics.m
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Fs=48000;
N=1666;
finc=Fs/N;
fho=60;    %The value of the first harmonic is 60 Hz.
m=1;
Xf=fft(x);
%Kho=fho/finc;
%Xf(Kho+1);
numinterharm=0;
for m = 1:150,
   fhm=m*fho;
   Kho=fhm/finc;
   if Kho>length(Xf),
     results(18,4)=0;
     results(19,4)=0;
    else

   Interharm=Xf(round(Kho+1));
```

```
   if Interharm == 0
    numinterharm=numinterharm+1;
   end
end
m=m+1;
end
save numinterharm numinterharm
wave_Interharmonics
set(num_interharm,'string',num2str(numinterharm));
```

APPENDIX B C PROGRAMS

```
/*-------------------------------------------------------------------------------------

AFPLL.C

Description:

This program executes a dynamical system to track the amplitude, phase and frequency

of a time-varying fundamental component of a given signal.

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

Author:

Ivan Rivera

Advisor:

Domingo Rodriguez

-------------------------------------------------------------------------------------*/

#include "dsk6713_aic23.h"

Uint32 fs=DSK6713_AIC23_FREQ_8KHZ;

#include <stdio.h>

#define N 8128

#include "math.h"

//#include "u_0.8sin2pi63t_8KHz_508.h"

//#include "u_0.8sin2pi63t_8KHz_4064.h"

//#include "u_0.8sin2pi63t_8KHz_2032.h"

//#include "u_0.8sin2pi63t_8KHz_8128.h"

//#include "u_0.8sin2pifvt_8KHz_8128.h"

#include "u_sin2pifvt_harmonics57_8KHz_8128.h"

FILE *fp;
```

```
void main()
{
  double A=0,phi=0,y=0,e=0;
  double w;
  double Ts,Fs,K1,K2,K3,PI,f;
  int i;
  PI=3.1416;
  f=60;
  w=2*PI*f;
  K1=50;
  K2=1000;
  K3=0.08;
  Fs=8000;
  Ts = 1/Fs;
  fp = fopen("EPLL.txt","w");
  for(i=0;i<N;i++)
   {
     A += K1 * Ts * e * sin(phi);
     w += K2 * Ts * e* cos(phi);
     phi += w*Ts + K3 * K2 *Ts *e * cos(phi);
     y = A * sin(phi);
     e = u[i] - y;
     fprintf(fp,"%lf\t" "%lf\t" "%lf\t" "%lf\t" "%lf\t %lf\n",A,w,phi,y,e,u[i]);
   }//end for

    fclose(fp);
```

```
   return;
}//end main
/*---------------------------------------------------------------------------
 AIP, Laboratory.
 STFT.C
 DESCRIPTION
 This program compute the Discrete Short Time Fourier Transform
 (DSTFT) of a given signal,using the transform method.
 DEVICE: T.I. TMS320C6713
-----------------------------------------------------------------------------
--------
 Coments:
 1) Necesita un .h con la variable x(M puntos)
    que es la senal de entrada.
 2) Necesita un .h con la variable xpadd(P puntos) llena
    de ceros, que es la senal de entrada padeada.
 3) Necesita un .h con la variable h(L puntos) padeada hasta P puntos
    que es el filtro padeado.
-----------------------------------------------------------------------------
---------
 HISTOR
      Rev 1.00 - Feb/2007 Created By Ivan Rivera
          Dr. Domingo Rodríguez - Advisor
-----------------------------------------------------------------------*/
#include <stdio.h>
#include <stdlib.h>
```

```
#define N  512          // Total Shifts: SN^m, m =0,1...N-1
#define P  1024         // Points of the signal padded: xp
#define L  128          // Points of the filter without padding: h
#define M  P-L          // Points of the signal without padding: x
#include "math.h"
#include "dataTW1024.h"       // Twiddle Factors to compute a P-pointFFT.
//#include "sin_896.h"
#include "chirp_896.h"
#include "hanning_128_1024.h"
#include "xpadd_1024.h"
far double hm[N][2*P]; // far - location, of the STFT External memory
allocation
                                    //xk[Rows][ComplexColumns].
int m,n;
FILE *fp;
/*-------------------------------------------------------------------------
-------*/
void FFT_TI (void)
{
int m;
double *pointer1;



for (m=0; m<N; m++)
      {

       pointer1 = hm[m];
```

```
        DSPF_sp_cfftr2_dit((double*) pointer1,(float*) w, P);
        bit_rev((double *)pointer1, (2*P)>>1);

      }

}// End FFT_TI



/*-------------------------------------------------------------------------
-*/

void Shift_and_Haddamard(double *h, double *xpadd)
{
int m,n,y;

for (m=0; m<N; m++)
      {
              for (n=0;n<P;n++)
                  {
                      y = m-n;

                   if (y<0)
                       y = y+P;

                         hm[m][2*n]= h[y]*xpadd[n]; // h: padded filter, P
points
```

```
                        hm[m][2*n+1]= 0;
                }
        }
}//end Shift_and_Haddamard


/*-------------------------------------------------------------------------
-*/

void Input_Signal_Padding(double *x)
{
int n;

for (n=0; n<M; n++)
                {
                        xpadd[n] = x[n]; // xpadd: header file with P zeros
                                                // P = M+L
                }
}//end Input_Signal_Padding


/*-------------------------------------------------------------------------
--*/

void main()
{
Input_Signal_Padding((double*)x);    // Input padding
```

```
Shift_and_Haddamard((double*)h, (double*)xpadd); //Shift and Haddamard
operations

FFT_TI();                                  // FFT over the rows of hm

fp = fopen("Sh.txt","w");

for(m=0;m<N;m++)
{
 for(n=0;n<2*P;n++)
 {
  fprintf(fp,"%lf\t",hm[m][n]);
 }//end for n
  fprintf(fp,"\n");
}//end for m

 fclose(fp);

}//end Main
```
