UNIVERSITE DE NEUCHÂTEL
INSTITUT DE MICROTECHNIQUE

On the Design and Implementation of Steady-State Kalman Filters: An Approach for Spectrum Analysis of Dynamic Signals

THÈSE

PRÉSENTÉE À LA FACULTÉ DES SCIENCES
POUR OBTENIR LE GRADE DE DOCTEUR ÈS SIENCES

PAR

Peter Balsiger

Neuchâtel, Juin 1995

to Karin,

Eric-Maxime, and

Simona-Viktoria

IMPRIMATUR POUR LA THÈSE

Conception et intégration de filtres de Kalman stationnaires: Une approche pour l'analyse spectrale de signaux dynamiques

de M. Peter Balsiger

UNIVERSITÉ DE NEUCHÂTEL

FACULTÉ DES SCIENCES

La Faculté des sciences de l'Université de Neuchâtel sur le rapport des membres du jury,

MM. F. Pellandini, N. de Rooij , M. P. Kunt (Lausanne) et P. Gruber (Zoug)

autorise l'impression de la présente thèse.

Neuchâtel, le 1er septembre 1995

Le doyen:

H.-H. Nägeli

Acknowledgments

I am greatly indebted to Professor Fausto Pellandini for his encouragement and support.

I wish to thank Professor Nico De Rooij, Professor Murat Kunt, and Dr. Peter Gruber for accepting to co-examine this thesis.

I extend my thanks to Sandro Ponta, M.J.J. Valkering, Alexandre Heubi, and Louisa Grisoni for their help with the software implementation on DSP. Benedikt Guldimann and Christophe Oberson contributed to the software and ASIC implementations, respectively.

I am grateful to the many students at the Institut de Microtechnique, Université de Neuchâtel (IMT Uni NE) who have helped me during this project. I also appreciate the efforts of all the members of the Chair of Electronics and Signal Processing, IMT Uni NE. Their support and companionship was tremendous. Special thanks to Dr. Hanspeter Amann for his willingness to examine any problem and to Heinz Burri for his hardware and software support.

I was lucky to have the assistance of Marc A. Wyttenbach, a graduate of Cornell University in the United States. His editing and insight added immeasurably to the clarity and depth of the finished work.

Finally, I would like to express my gratitude to Karin, my wife, for her continuous love, her never ending patience, and her understanding. I was able to draw upon her acute knowledge of the inner workings of human beings to realize more than a purely scientific perspective.

SUMMARY

This thesis presents an approach for short-time spectrum analysis of dynamic signals measured in noise. This is accomplished via state estimation. Two different signal models are presented for the estimation of the Fourier coefficients, applying Kalman filtering theory. The resulting filter is a parallel bank which produces sliding estimates of the desired Fourier coefficients for a set of frequencies. The applied Kalman theory leads to more accurate estimations in the presence of noise. The approach offers a parametrization by the design tradeoff between noise rejection and convergence speed. The method can be seen as a generalization of Discrete Fourier Transform.

The first model is an Oscillator Signal Model and the second is a Random Walk Signal Model. A procedure is derived that reduces the computational complexity by pre-computing certain parameters leading to steady-state Kalman filters.

Additional considerations explore the behavior of sinusoidal signals in the time and frequency domains. A new realization structure of the steady-state Oscillator Signal Model calculates the magnitude of the estimated frequency directly. This structure has very low computational complexity.

Well-known classical spectrum analysis techniques, such as windowing short-time Discrete Fourier Transform, recursive running Discrete Fourier Transform, and Fast Fourier Transform (Bergland Algorithm), are compared and contrasted with the proposed Kalman approaches.

The steady-state Random Walk Signal Model is considered for software implementation on a Motorola DSP56001 Digital Signal Processor. A Digital Power Network Signal Analysis System has been realized; a study of power computation is

presented; and the nature of quasi-periodic voltage and current signals and harmonic distortions is incorporated.

The steady-state Oscillator Signal Model is considered for a VLSI integration. The basic structure for a given set of frequencies, where the Fourier coefficients are estimated, is realized in a CMOS standard cell layout technology. Performance results are analyzed.

This work was initiated by Landis & Gyr AG, Zug.

Contents

1. Introduction

The first chapter furnishes the motivation behind this work and gives an overview of each chapter's contents. Spectrum analysis of both arbitrarily and uniformly spaced sinusoidal signals measured in noise is considered.

The specific problem, spectral analysis of pseudo-stationary periodic signals, is presented and different design methodologies are briefly mentioned.

1.1. MOTIVATION

The clear trend towards increased use of Digital (discrete-time) Signal Processing over the past two decades provided the rationale for this work. To some extent, this can be attributed to the inherent flexibility and reliability of Digital Signal Processing. Two other important factors that have supported this trend are:

1. The rapid development of integrated circuit technology, i.e. the capability to manufacture fast, complex, and cheap digital circuits.
2. Advances in computer hardware and software.

Digital Signal Processing (DSP) has not only partially replaced traditional analog techniques, it has opened new areas of signal processing: Fresh products and markets are being created for new generations of equipment.

Faster digital hardware — small, inexpensive, and with low power consumption — is currently becoming available because of the continuing development of integrated circuit technology. Today's Computer Aided Design (CAD) tools offer the possibility to

design, synthesize, simulate, and realize on a very high abstraction level. Designers today handle very large schemes, containing as many as several millions of transistors. Large systems can be realized on the same waver and be simulated and verified before transferring to silicon.

Since the early 1990s, designs have become independent of technologies and even independent of the CAD environment. The designs describe a digital circuit in a Very high-speed IC Hardware Description Language (VHDL). Graphical front ends allow developers to enter the behavior, or functionality, of the considered application. The generated code is then interfaced to IC synthesis tools for synthesis and technology mapping.

Clearly, it is essential to use strategies that will quickly arrive at efficient, cheap, and correct solutions. Today's tools, however, do not make the right choice for how to realize a specific system or a considered function. This area will be reserved, for some time, to human experts.

This thesis emphasizes one very specific approach to estimating sliding Fourier coefficients for a given set of frequencies — the Kalman filter. This method is especially provided for spectrum analysis of signals where the frequency components are slowly time-varying and/or when only specific frequency bands are of interest. The approach can be seen as a signal tracker. On/off switched signals having a large dynamic behavior can be estimated.

Using the Kalman method, one is able to select the range of the frequency axis and the distance separating individual frequencies. As a special peculiarity, the sharpness or the shape, of the frequency response (noise rejection), can be parametrized separately for each frequency.

A new algorithm to estimate the sliding Fourier coefficients is proposed along with an ASIC design. Furthermore, the approach is used for a real-time implementation on digital signal processors. Close collaboration with the Swiss company, Landis & Gyr Betriebs AG, Zug, resulted in the development of a new type of digital power meter, with special functions for power network signal analysis.

The ideas presented are the consummation of many stimulating hours of work at the University of Neuchâtel and from the close partnership with Landis & Gyr. The author has participated in a

number of scientific reports [Bals91, Ober93, Ober94, Guld92, Pont92, Heub91], two publications [Bals92, Gris92], and several postgrade courses [Bals95a, Bals95b, Bals94, Bals93a, Bals93b, Bals93c], all of which have arisen from this project. The results were also influenced by the related works of Bitmead et al. [Bitm86] and Gruber et al. [Grub88].

Evidently, there are parts that have not been covered. The industrial project with Landis & Gyr was performed by two people, each of whom was simultaneously involved in other ventures, and by contributing students. Therefore, it is not a complete examination that is presented but a collection of ideas, algorithms, and results which simplify real-time implementation of sliding spectrum analysis. The exhibited concept is very interesting for system integration due to its low complexity, the algorithm, and the target architecture.

1.2. SPECTRUM ANALYSIS

In many practices, the Fourier coefficients of a given set of frequencies of a pseudo-stationary and quasi-periodic signal must be estimated in real-time. In this context, a quasi-periodic signal is a signal that consists of a given number of sinusoidal signals with known frequencies and unknown, time-varying amplitudes and phases, superimposed by some additive noise source, $w(k)$.

This special kind of periodicity arises, for instance, in speech signals, vibrational signals, and in electrical energy networks. In Figure 1.1, a frequency representation of typical such signals is presented. A continuous estimation of these parameters can be used for monitoring or diagnostic purposes. Another, more traditional application, is the on-line estimation of the distortion of pure sinusoidal signals. The emphasis in this thesis is on the latter type, where distortion of pure sinusoidal signals must be analyzed. An example of such a signal, $y(k)$, can be written as

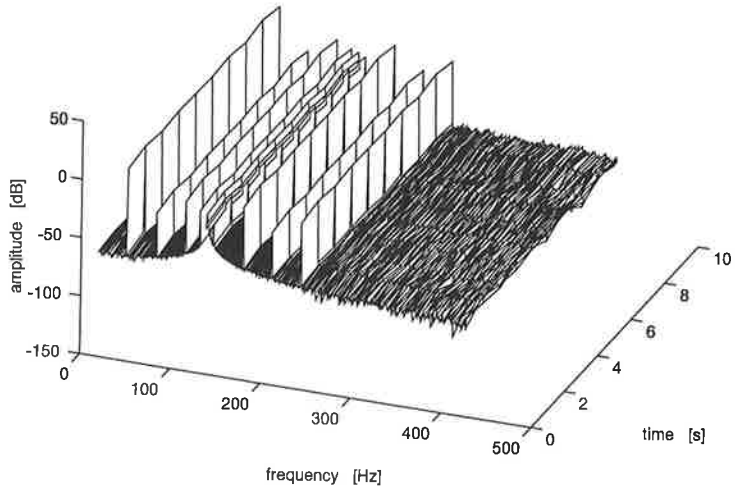


Figure 1.1. *Quasi-periodic signals corrupted by additive noise.*

$$y(k) = a_0(k) + \sum_{i=1}^{M} [a_i(k)\cos(\omega_i k T_s) + b_i(k)\sin(\omega_i k T_s)] + w(k) + \sum_{i=1}^{O} [a_{di}(k)\cos(\omega_{di} k T_s) + b_{di}(k)\sin(\omega_{di} k T_s)] \qquad (1.1)$$

The coefficient pairs — a_i, a_{di} and b_i, b_{di} — are, in general, time-varying. They correspond, respectively, to the *real part* and the *imaginary part* of the Fourier coefficient of the observed sinusoid, at frequency ω_i. The sampling frequency is $\omega_s = \frac{2\pi}{T_s}$. It is assumed that the sampling frequency is chosen such that all the frequency components of $y(k)$ are below half the sampling frequency, in concordance with Shannon's theorem.

Different methods can be used for real-time estimation of time-

varying Fourier coefficients. A commonly used process is *time-frequency analysis*. This is accomplished with the *sliding window*, or *short-time Discrete Fourier Transform*, which can be efficiently implemented by using the *recursive running discrete Fourier series* [Alle77, Papo77, Rabi75]. In recent times, operations other than sinus and cosine functions have been applied to enhance time-frequency resolution. Since 1989, the *Wavelet Transform* has received increased attention.

Over the past ten years, new methods have been developed, based upon statistics. The time-variation of the sinusoidal frequency is expressed by a *stochastic design model* which, together with a model of the additive noise source, is the foundation for the design of a *Kalman filter*.

Bitmead et al. use a bank of undamped oscillators with a specific set of frequencies (only multiples of a fundamental frequency, ω_o), which are driven by white noise sources [Bitm86]. This is their basis for the derivation of the Kalman-based Fourier Coefficient Estimator (KFCE). Wied uses the same model except that only the oscillator for ω_o is undamped, whereas all the others are slightly damped [Wied88]. Doraiswami et al. estimate the unknown with the same undamped oscillator, but constant coefficients of a sinusoid are corrupted by additive noise [Dora86]. Gruber and Toedtli have derived a model that uses a Kalman filter to directly estimate the Fourier coefficients [Grub88, Grub94].

1.3. SIGNAL MODELLING

There are currently many ways to find a Fourier coefficient estimator for a given frequency band — and there is no assertion that the scheme is complete. An overview is given in Figure 1.2. The main difference between the existing solution alternatives lies in the choice of the signal model that is used for the design of the estimator. Two avenues for determining Fourier coefficient estimators are treated in this thesis: The Static Signal Model and the Dynamic Signal Model.

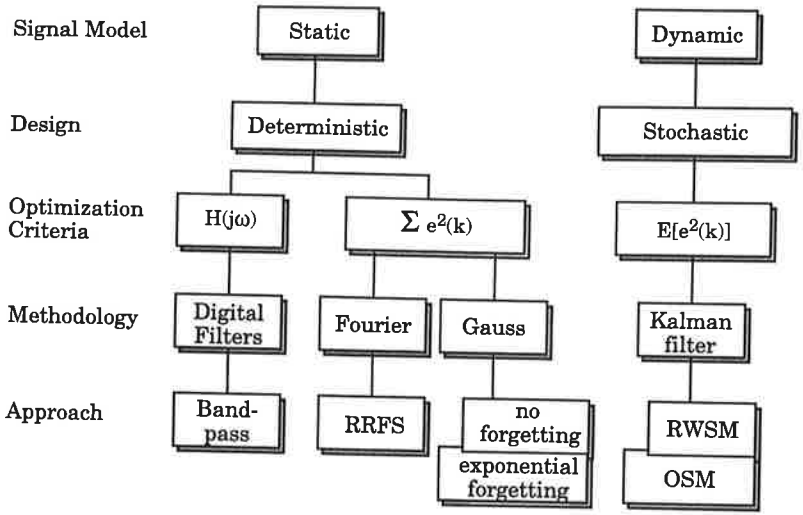


Figure 1.2. *Range of solutions for the estimation of the Fourier coefficients of a sinusoid.*

1.3.1. Static Signal Model

The *Static Signal Model* (SSM) assumes that the parameters to be estimated remain constant during the observation time. No a priori knowledge about the dynamics of the parameter is required.

One method in this category is based upon linear, time- invariant digital filters. For each of the desired frequencies, a band-pass filter is applied. The optimization criteria is, in general, the imposed frequency or the time response of the transfer function, $H(\omega)$. In the literature, many different structures and variants are described. The drawback to this method is the distortion of the signal when *Infinite Impulse Response* (IIR) filters are employed. If the phases of the considered signals have to be conserved, *Finite Impulse Response* (FIR) filters are used. FIR implies a higher filter order, leading to a more complex realization.

Discrete Fourier Transform (DFT), with its well-known variants, such as the recursive running discrete Fourier series and the *Recursive Least Square* (RLS) method, are predicated on a Static Signal Model. These routines were originally developed for the estimation of constant parameters. They fit a static model to the received data, $y(k)$, by minimizing the deterministic performance criteria — the sum of the squared errors, $\sum e^2$, between the sampled signal sequence, $y(k)$, and its estimate, $\hat{y}(k)$.

$$\sum_{i=0}^{n} e^2(i) = \sum_{i=0}^{n} (y(i) - \hat{y}(i))^2 = minimum \qquad (1.2)$$

In particular, the recursive running discrete Fourier series fits into a given number of frequencies, N, such that the sum of the squared error over the N data points, is zero. This fact is related to the assumption that the data sequence repeats itself periodically, after N points. The number of frequencies that can be analyzed is fixed by N.

In the context of classical Fourier transform, the assumption of correct spectrum analysis requires an adequate choice for the length of the observation window. An N-point (window length is N) Discrete Fourier Transform can be carried out to determine the N-unknown Fourier coefficients, a_i and b_i.

Furthermore, problems arise if the input sequence, $y(k)$, is corrupted by the aforementioned periodic disturbances, with frequencies ω_{di}. If the Fourier coefficients of the disturbances, a_{di} and b_{di}, are non-zero, the observed signal, $y(k)$, with period τ_y, will generally be periodic.

In this case, DFT produces a coefficient set that varies periodically because the frequency resolution of ω_1 does not sufficiently identify the time-variations. To achieve the required resolution for a_{di} and b_{di}, one must either increase the number of points or use a low pass filter, to reduce the time-variations. The bandwidth of the low pass filter must be carefully adapted to the observed frequencies.

In general cases, and in real world applications, the sampled signal, $y(k)$, is corrupted by noise. Therefore, DFT will generate a different set of parameters for each window block. Again, a low pass filter can make the time-variations more smooth.

The RLS method offers the freedom to choose many frequencies that will be estimated. The behavior of the RLS method, without the forgetting factor, is equivalent to the recursive running discrete Fourier series. In the case of slow, periodic, varying disturbances and the additional noise source, Equation 1.2 does not converge on zero. With noise, the RLS method needs more data points, to smooth the variation in the Fourier coefficients and to minimize Equation 1.2. In order to improve the RLS method, different forgetting methods have been introduced. A survey is given in [Toed83, Krau86].

1.3.2. Dynamic Signal Model

One problem with the Static Signal Model is its reliance on a specific number of data samples. Should new data become available, the process must be repeated. A procedure in which previously determined estimates are simply updated as new data arrive would be superior.

The recursive estimation technique introduced in this section, and discussed in detail in Chapter 3, is a powerful tool for estimating desired signals. A dynamic signal attempts to model the time-variation of the coefficients to be estimated. It is described by a set of linear difference equations.

The successive data samples, $x(k)=s(k)+n(k)$, contain a random signal, $s(k)$, with $E[s^2]=S$ to be estimated, as well as additive, uncorrelated noise samples, $n(k)$, with variance, σ_n^2. Both $s(k)$ and $n(k)$ are assumed to be zeromean random variables. Then, the output of a first-order recursive filter taking in successive $x(k)$ samples at the input can given by

$$y(k)=ay(k-1)+x(k) \qquad |a|<1 \tag{1.3}$$

This filter updates, adding a new data sample, $x(k)$, to a fraction of the previous output. Separating the signal term of the geometrical sequence from the noise term, we get

$$y(k)=s\frac{1-a^k}{1-a}+\sum_{j=1}^{k}a^{k-j}n_j \tag{1.4}$$

For large k ($|a|^k<<1$), the signal part of $y(k)$ approaches $s/(1-a)$, while the variance due to noise approaches $\sigma_n^2/(1-a)$. If one multiplies $y(k)$ by $(1-a)$, it becomes apparent that $(1-a)y(k)\rightarrow s$, for very large k, while the variance decreases — $[(1-a)y(k)\longrightarrow \sigma_n^2(1-a)/(1+a)]$. Thus, this device does provide some smoothing, as expected, producing an estimate that has a smaller variance, or mean-square variation, about s, than would be the case with only one sample: $s[\hat{s}=(1-a)y(k)]$. By increasing the effective time constant of the filter, a, the mean-square error of $s(k)$ is reduced.

The problem now is to find the *best* recursive linear estimate of $s(k)$. The term best refers to the particular estimate that minimizes the sum of the squared residuals. The estimator is found by adding two time-varying gains, $a(k)$ and $b(k)$ [Schw75, Hwang92]

$$\hat{s}(k)=a(k)\hat{s}(k-1)+b(k)x(k) \tag{1.5}$$

Note that $\hat{s}(k)$ is the weighted sum of the previous estimate, $\hat{s}(k-1)$, of parameter $s(k)$. The two time-varying gains, $a(k)$ and $b(k)$, must then be found such that the mean-square error, $e(k)$, is minimized

$$e(k)=E[[\hat{s}(k)-s(k)]^2] \tag{1.6}$$

Parameter $b(k)$ can be expressed in terms of $a(k)$. Equation 1.5 becomes

$$\hat{s}(k)=a(k)\hat{s}(k-1)+b(k)[x(k)-a(k)\hat{s}(k-1)] \tag{1.7}$$

The first term, $a(k)\hat{s}(k-1)$, represents the best estimate of $s(k)$, without any additional information, and is therefore a prediction based on past observations. The second term is a correction term, involving the difference between the new data sample, $x(k)$, and the updated estimate, with the variable gain factor, $b(k)$, appended.

Extending the signal model to include vector signals, multiple signal measurements, and signals with more complex dynamic behavior, prepares us to tackle the problem discussed at the beginning of this chapter. The Kalman filter we will consider in Chapters 3 and 4 requires a model for the measurement of the

vector signal as well as a model for the generation of sinusoidal signals.

The signal model can be described by the state equations, $\underline{x}(k)$, which are driven by noise sources, $\underline{v}(k)$. The a priori knowledge about the irregularity is put into the variances of these noise sources and into the parameters of the signal model. In this context, a Kalman filter approach can be used for the estimation of the Fourier coefficients. In Figure 1.2, the stochastic design succession is seen in relation to the deterministic design approach.

The Kalman filter determines, for an incoming data point, $y(k)$, at time instant k, the estimates, $\hat{\underline{x}}(k)$, of the states, $\underline{x}(k)$. The "one-step-ahead" prediction of $\underline{x}(k)$ is $\hat{\underline{x}}^+(k)$. The estimates are carried out, minimizing the square root of the estimated values and of the true values. The minimization criteria for the error covariance equation are

$$\underline{V}(k) = E[(\underline{x}(k) - \hat{\underline{x}}(k))(\underline{x}(k) - \hat{\underline{x}}(k))^T] \tag{1.8}$$

and

$$\underline{V}^+(k) = E[(\underline{x}(k) - \hat{\underline{x}}^+(k))(\underline{x}(k) - \hat{\underline{x}}^+(k))^T] \tag{1.9}$$

known as the predictor.

1.4. OUTLINE OF THE TEXT

The characteristics of deterministic and stochastic design signal models are introduced and compared in Chapter 2. General aspects of spectrum analysis are reviewed and an overview of well-known estimation techniques is presented. Classical Fourier transforms, the Wavelet Transform, and the Kalman filter are compared.

In Chapter 3, the Kalman-based spectrum estimation approach is carefully introduced. The design of two specific signal models is presented. For both signal models, the time-frequency behavior is discussed and a steady-state solution is obtained. The different results obtained using the stochastic and deterministic design signal models demonstrate the need to use Kalman theory for measured noisy signals.

The steady-state Kalman filter is derived in Chapter 4. The signal flow graph and the resulting arithmetic operations are compared for each signal model and Fourier-based approach. Their numerical properties, based on simulation results, are discussed. Finally, the algorithms are prepared for software implementation on DSPs and for silicon integration on ASICs.

The implementation on DSPs is treated in Chapter 5. Moreover, the presented approach is used for an industrial application. The Digital (electrical) Power Network Signal Analysis System is briefly addressed.

In Chapter 6, the steady-state Oscillator Signal Model is implemented on various ASICs. Particularly interesting results are derived. The findings allowed for a comparison of the power consumption, the execution speed, and the silicon area.

The conclusions in Chapter 7 give a summary of the main advantages and features of the Kalman filter. Additionally, propositions for future extensions and research in this field are offered.

REFERENCES

[Alle77] J.B. Allen, L.R. Rabiner : "A unified approach to short-time Fourier analysis and synthesis", Proc. IEEE, Vol.65, Nov. 1977, pp. 1558-1564.

[Bals95a] P. Balsiger, P. Gruber : "Echtzeit-Netzsignalanalyse basierend auf Kalman Filtern", VDI/VDE-GMA Aussprachetag, Messsignalverarbeitung und Diagnose Mittel zur Prozess- und Qualitätssicherung, Langen b. Frankfurt / M. 21/22. März 1995, Deutschland.

[Bals95b] P. Balsiger : "Digital Signal Processing in ASICs", Austria Mikro Systeme Int., Workshop 4-6 April 1995, Unterpremstätten, Austria.

[Bals94] P. Balsiger : "Kalman Filter für Real-time Spektralanalyse: Eine High-Level ASIC-Integration mit DSP Station", Solution Expo, Microswiss Zentrum Nord-Süd, Windisch, 21. April 1994, Switzerland.

[Bals93a] P. Balsiger, S. Ponta, P. Gruber : "Echtzeit-Netzsignalanalyse basierend auf Kalman Filter", Vortrag an der Eidg. Techn. Hochschule ETH-Zürich, Professur für Leistungselektronik, 23. Juni 1993.

[Bals93b] P. Balsiger, L. Grisoni, F. Pellandini : "High-Level Design of Digital Signal Processing Applications with COMDISCO's SPW Tool", Postgrade course at IMT Uni Neuchâtel, March 25-26, 1993.

[Bals93c] P. Balsiger, L. Grisoni, E. Jeanclaude, F. Pellandini : "High-Level Design of Digital Signal Processing Applications with Mentor's DSP Station Tool", Postgrade course at IMT Uni Neuchâtel, March 31 - April 2, 1993.

[Bals92] P. Balsiger, S. Ponta, F. Pellandini, P. Gruber, and J. Tödtli : "A Kalman Filter Approach to Power Network Signal Analysis", Proc. URSI Int'l Symposium on Signals, Systems and Electronics, ISSSE '92, Paris, France, Sept. 1992, pp. 474-477.

[Bals91] P. Balsiger, S. Ponta : "Etude des algorithmes d'analyse et de caractérisation en temps réel de signaux quasi-périodiques", Rapport IMT No 306 PE 11/91, Université de Neuchâtel, IMT, Novembre 1991.

[Bitm86] R.R. Bitmead, A.H. Tsoi, P.J. Parker : "A Kalman Filter Approach to Short-Time Fourier Analysis", IEEE Trans. on Acc. Syst. and Signal Processing, Vol.ASSP-34, No.6, December 1986, pp. 1493-1501.

[Bitm82] R.R. Bitmead : "On Recursive Discrete Fourier Transformation", IEEE Trans. on Acc. Syst. and Signal Processing, Vol.ASSP-30, No.2, April 1982, pp. 319-322.

[Dora86] R. Doraiswami, J. Jiang, R. Balasubramian : "A novel narrow band digital filter and its application to multivariable system identification", Automatica, Vol. 22, No. 6, 1986, pp. 733-737.

[Grub94] P. Gruber, J. Tödtli : "Estimation of Quasiperiodic Signal Parameters by Means of Dynamic Signal Models", IEEE Trans. on Signal Processing, Vol.42, No. 3, March 1994.

[Guld92] B. Guldiman : "Analyse et implantation sur DSP d'un système de détection du signal de signalisation du réseau électrique", Projet de sémestre hiver 91/92, IMT Uni NE, mars 1992 (assistant S. Ponta).

[Grub88] P. Gruber, J. Tödtli : "A Recursive Estimation for the Determination of Unknown Constant or Slowly Varying Fourier Coefficients", EUSIPCO-88, Forth European Signal Processing Conference, Grenoble, September 5-8, 1988.

[Heub91] A. Heubi, "Application d'un filtre de Kalman pour l'analyse des signaux du réseau électrique", travail de diplôme, décembre 1991, étudiant en Microtechnique, EPFL, (ass. P. Balsiger).

[Heub93] A. Heubi, M. Ansorge, F. Pellandini, "Architecture VLSI faible consommation pour le traitement numérique du signal", Proc. GRETSI-93, Juan-Les-Pins, France, Sept. 1993, pp. 1083-1086.

[Krau86] F. Kraus : "Das Vergessen in rekursiven Paremeterschätzverfahren", Diss. No. 8012, ETH Zürich, 1986.

[Ober94] C. Oberson, "Réalisation d'un circuit intégré dédié à l'analyse du réseau de distribution d'énergie électrique", Travail de diplôme, DMT-EPFL, février 1994, (ass. P. Balsiger).

[Ober93] C. Oberson, "Système d'analyse du réseau de distribution d'énergie électrique", Travail de semestre, juin 1993, DMT-EPFL, (ass. P. Balsiger).

[Papo77] A. Papoulis : "Signal Analysis", Mac Graw Hill, New York 1977.

[Pont92] S. Ponta, P. Balsiger, *Power network signal analysis system interface*, IMT Report No 317 PE 05/92, University of Neuchâtel, IMT, May 21, 1992.

[Rabi75] L. Rabiner, B. Gold : "Theory and application of digital signal processing", Prentice Hall, New York 1975.

[Schw75] M. Schwartz, L. Shaw : "Signal processing descrete spectral analysis, detection, and estimation", McGraw-Hill, New York, 1975.

[Toed83] J. Tödtli : " Rekursive Parameterschätzverfahren nach der Methode der kleinsten Quadrate", Scienta Electronica, Vol. 29, No. 4, 1983.

[Wied88] A. Wied : "Erfassung der Stromoberschwingungen von Lichtbogenöfen mit Hilfe des Kalman Filters", Automatisierungstechnik, Vol 36, No.1, 1988, pp. 21-25.

2. Survey of Fourier Analysis and Least-squares Estimation

This chapter is comprised of an introduction to spectrum analysis, followed by an historical overview of Fourier analysis. It is a deterministic design model, as opposed to a stochastic design model, e.g. the Kalman filter approach.

Symmetry relations between real, imaginary, and complex signals are reviewed before the Discrete Fourier Transform is discussed. Special attention is given to the windowing technique, applied for noise rejection enhancement. Time-frequency resolution, of primary interest for short-time Fourier transform, is carefully treated.

An alternative method for time-frequency analysis has recently surfaced — discrete Wavelet Transform. With Wavelet Transform, a signal can be decomposed into many different time and frequency scales, hence both low-frequency, long duration phenomena and high-frequency, short-duration phenomena can be studied within the same transform. Because Wavelet Transform is well-suited for the analysis of dynamic signals, a brief overview is given.

Subsequently, the Dynamic Signal Model is applied to spectrum analysis. The design of the signal model is stochastic and thus a Kalman filter is proposed. Its design methodology is briefly addressed.

2.1. HISTORY

The development of Fourier analysis has a long history, involving a select group of individuals and their investigation of many different physical phenomena. The concept of using trigonometric sums, that is, the sums of harmonically related sinusoids and cosines or periodic complex exponentials, to describe periodic phenomena goes at least as far back as the Babylonians. It was they who used ideas of this sort in order to predict astronomical events [Oppe83].

The "modern" history of this subject begins in 1748 with L. Euler, who was examining the motion of a vibrating string. He noticed that the configuration of a vibrating string at any point in time is a linear combination of the signals that are constructed by the fundamental mode. Furthermore, Euler demonstrated that one could directly calculate the coefficients for the linear combination at a later point in time based upon the coefficients at an earlier one.

In 1807, Jean Baptiste Joseph Fourier (1768-1830), who was born in Auxerre, France, presented his results on the phenomenon of heat propagation and diffusion to the Institut de France. In his work, Fourier found that a series of harmonically related sinusoids could be used to represent the temperature distribution trough a body. In addition, he speculated that any periodic signal could be depicted by such a series.

Fourier then proceeded further than any of his predecessors when he obtained a representation for aperiodic signals, not as weighted sums of harmonically related sinusoids, but as weighted integrals of sinusoids that are not all harmonically associated. The Fourier series and the Fourier integral, or Fourier transform, continue to be two of the most powerful tools for the analysis of linear time invariant systems.

2.2. FOURIER TRANSFORM

In order to calculate the information concerning how the signal, $x(t)$, is composed of sinusoidal signals at different frequencies, we have to determine if the signal is periodic or aperiodic in time.

A **periodic signal**, with period T_0, can be described by a linear combination of harmonically related complex exponentials. These complex exponentials have amplitude $\{c_k\}$, and frequencies $k\omega_o$ — a *discrete* set of harmonically related frequencies ($k = 0, \pm1, \pm2, \pm3, \ldots$).

$$x(t) = \sum_{k=-\infty}^{+\infty} c_k e^{jk\omega_0 t} \tag{2.1}$$

$$c_k = \frac{1}{T_o}\int_{T_o} x(t) e^{-jk\omega_o t} dt \tag{2.2}$$

Equation 2.1 is often referred to as the *synthesis equation* and Equation 2.2 as the *analysis equation*. The coefficients, c_k, are called the *Fourier coefficients*, or the *spectral coefficients*, of $x(t)$. These complex coefficients measure the portion of the signal, $x(t)$, that is at each harmonic, ω_k, of the fundamental component, ω_o. The spectrum is described by a *discrete number* of Fourier coefficients at each analyzed spectral line and at each harmonic that is a multiple of the fundamental component.

In the case of an **aperiodic signal**, Equation 2.1 utilizes an integral, assuming that $\omega_o \to 0$ as $T_o \to \infty$. By definition, as $\omega_o \to 0$, the sum becomes the integral of $X(\omega)e^{j\omega t}$. Therefore, Equations 2.1 and 2.2 become

$$x(t) = \frac{1}{2\pi}\int_{-\infty}^{+\infty} X(\omega) e^{j\omega t} d\omega \tag{2.3}$$

$$X(\omega) = \int_{-\infty}^{+\infty} x(t) e^{-j\omega t} dt \tag{2.4}$$

For aperiodic signals, the complex exponentials occur on a *continuum* of frequencies and, according to Equation 2.3, have amplitude $\frac{1}{2\pi} X(\omega) d\omega$.

2.2.1. Symmetry Relations in Time and Frequency Domains

In general, a complex signal, $x(t)$, is the sum of two real signals, $x_1(t)$ and $x_2(t)$, and their corresponding Fourier transforms, $X_1(\omega)$ and $X_2(\omega)$. The real and imaginary parts are denoted by $R_{X_k}(\omega)$ and $I_{X_k}(\omega)$. The Fourier transform, $X(\omega)$ of $x(t)$, is obtained by Equations 2.5 and 2.6

$$x(t) = x_1(t) + jx_2(t) \tag{2.5}$$

$$X_1(\omega) = \underbrace{R_{X_1}(\omega)}_{even} + \underbrace{jI_{X_1}(\omega)}_{odd} \;;\; X_2(\omega) = \underbrace{R_{X_2}(\omega)}_{even} + \underbrace{jI_{X_2}(\omega)}_{odd} \tag{2.6}$$

$$X(\omega) = X_1(\omega) + jX_2(\omega) = \underbrace{[R_{X_1(\omega)} - I_{X_2(\omega)}]}_{R_{X(\omega)}} + \underbrace{j[I_{X_1(\omega)} + R_{X_2(\omega)}]}_{I_{X(\omega)}} \tag{2.7}$$

The symmetry relations in the time and frequency domains for real, imaginary, and complex signals are summarized in Tables 2.1 through 2.3 [Pell94]. However, in most practical applications only real signals are measured.

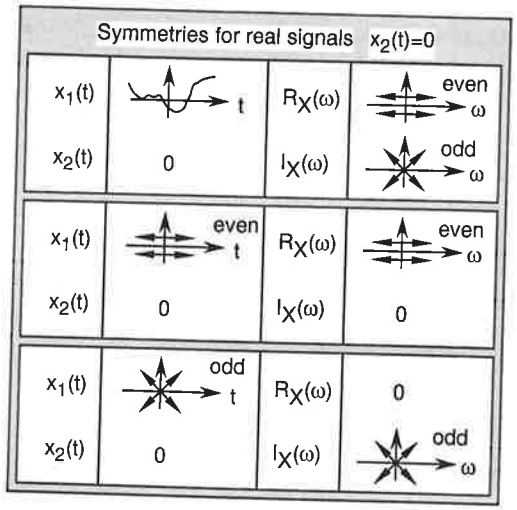
For real signals having odd or even symmetry in the time domain, the spectrum is reduced to a real part in the case of even symmetry and to an imaginary part for odd symmetry.

When there is information about the symmetry properties of signals in the time domain, the calculations are drastically reduced and thus the algorithms used to estimate the Fourier transform can be optimized. The same computing savings can be applied for imaginary signals, shown Table 2.2. The symmetry characteristics of complex signals, which consist of a real and an imaginary part, are summarized in Table 2.3.

2.2.2. Discrete Fourier Transforms

To analyze the frequencies that compose a sampled signal, methods based on Fourier transform (Equation 2.8) can be used.

Table 2.1. *Symmetry Relations in Time and Frequency Domains for Real Signals.*



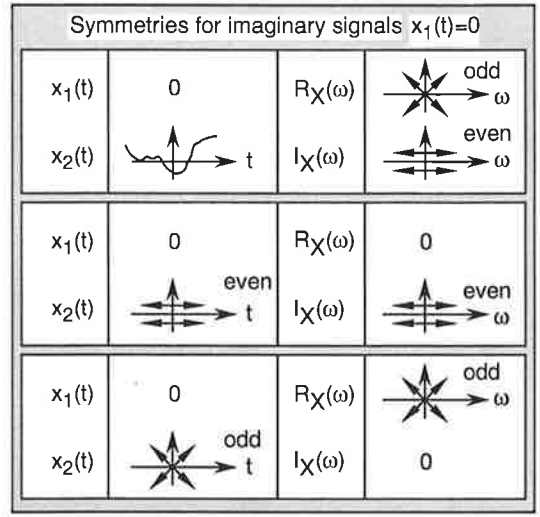
$$X(\omega) = \sum_{k=-\infty}^{\infty} x(k) e^{-j\omega \cdot k} \qquad (2.8)$$

T is the sampling period, $x_1 \ldots x_N$ are the signal samples, and $X(\omega)$ is the spectral amplitude, at frequency ω. Any spectral line can be computed, as long as the considered frequency is less than half the sampling frequency. Unfortunately, the side-lobe contributions, due to the finite number of samples, N, perturb the results.

If the frequencies to be analyzed are equidistant $\Delta\omega = 2\pi / NT$, ω becomes $2\pi k / NT$ and thus Discrete Fourier Transform (DFT) can be applied. Equation 2.8 becomes

$$X_k = \sum_{i=0}^{N-1} x_i e^{-j(2\pi/N)ki} \qquad k = 0,1,\ldots,N-1 \qquad (2.9)$$

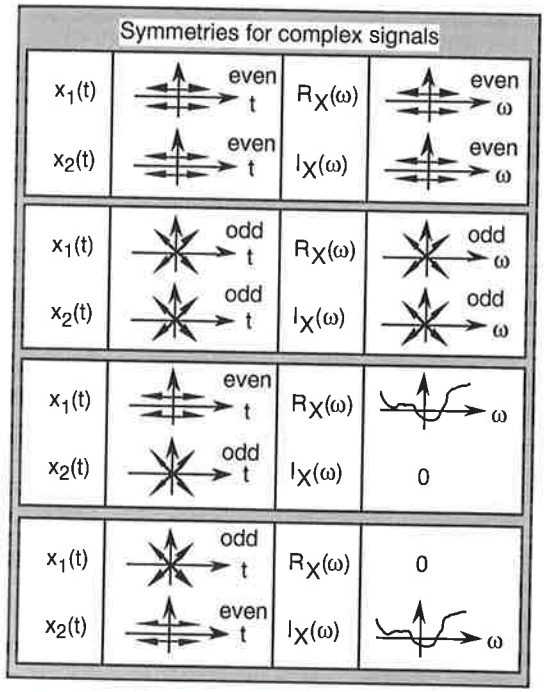
Table 2.2. *Symmetry Relations in Time and Frequency Domains for Imaginary Signals.*



There are no side-lobe contributions and so the problem is whether to perform a DFT algorithm or to explicitly compute, using Equation 2.9, the frequencies to analyze. Explicit computation of the DFT is used only when very few frequencies are to be analyzed.

In the former case, N, the number of points on which the Fourier transform is applied, can be selected so as to allow the use of Fast Fourier Transform (FFT) algorithms. This drastically reduces the number of real operations. The draw back, however, is that N must be in the set, 2^z (e.g., 2, 4, 8, 16, 32, ...). This method exposes the shortcoming of being restricted by N. This restriction limits the analysis to equidistant frequencies. Moreover, unless a windowing technique is applied, the noise rejection is poor.

Table 2.3. *Symmetry Relations in Time and Frequency Domains for Complex Signals: $x(t)=x_1(t)+j\,x_2(t)$.*



2.2.3. Noise Rejection Enhancement by Windowing

The sampled signal is commonly taken over a defined time frame, where N samples are considered. Such an operation is identified as *windowing*. It is the multiplication of the input sequence, $\{x_k\}$, by a square pulse, $w_r(k)$, which produces a convolution of the spectrum, $X(\omega)$, with a linear phase transform of the magnitude

$$|W_r(\omega)| = \left| \frac{\sin(\omega TN/2)}{\sin(\omega T/2)} \right| \qquad (2.10)$$

The major disadvantages of the rectangular window, $W_r(\omega)$, present in DFT, are that the peak side-lobe is only suppressed to -13 dB from the main-lobe level and that the roll-off is only -6 dB/octave (-20 dB/dec). This severely limits the ability of the rectangular window DFT to solve a weak signal in the presence of a stronger one, or to reject enough noise.

Several alternative window functions, $w(k)$, each improve upon the disadvantages of the rectangular window through the multiplication of the considered sample, $x(k)$, by the corresponding window correction factor, $w(k)$. The form is

$$X_k = \sum_{i=0}^{N-1} w(i)x(i)e^{-j(2\pi/N)ki} \qquad k = 0,\ 1,\ \ldots,\ N-1 \qquad (2.11)$$

Windowing further sacrifices spectral resolution to gain additional side-lobe attenuation. The consequence is an increased number of points, by a factor of two or three, on which the Fourier Transform is applied, when the same spectral resolution is required.

Hanning Window

$w_N(k)$ has a -18 dB/octave (-60 dB/dec) roll-off rate and a peak side-lobe level of -31 dB. The Hanning window is simply the square of the sine function, thus called the raised cosine [Jack86].

$$w_N(k) = \sin^2\left(\frac{\pi k}{N-1}\right) = \frac{1}{2}\left[1 - \cos\frac{2\pi k}{N-1}\right] \quad , \quad k = 0,1,\ldots,N-1 \qquad (2.12)$$

Hamming Window

$w_m(k)$ reduces the peak side-lobe level to -41 dB. The roll-off rate is only -6 dB/octave (-20 dB/dec), analogous to the rectangular window. The Hamming equation is

$$w_M(k) = 0.54 - 0.46\cos\left(\frac{2\pi k}{N-1}\right) \quad , \quad k = 0,1,\ldots,N-1 \qquad (2.13)$$

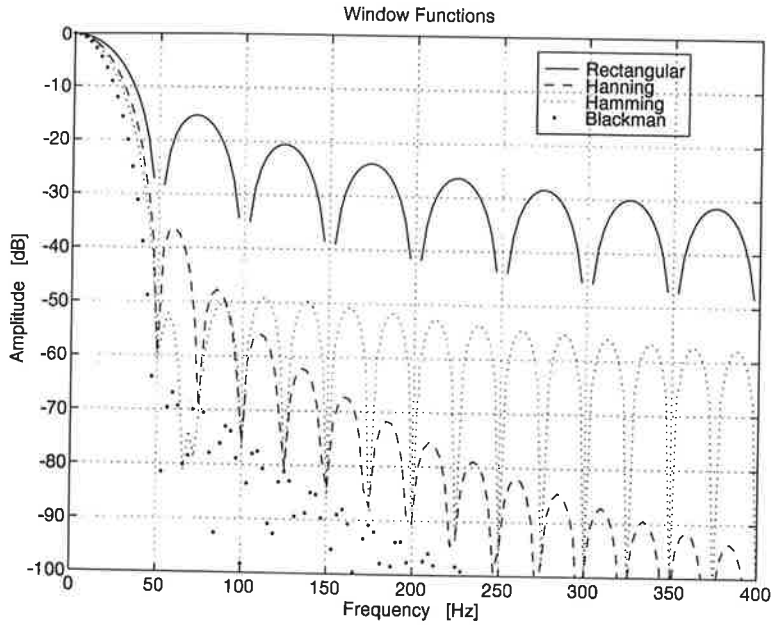


Figure 2.1. *Windowing functions for spectrum estimation: Rectangular window N=64, Hamming and Hanning N=128, Blackman N=192.*

Blackman Window

Side-lobe suppression is further improved by the Blackman window. The roll-off rate is -18 dB/octave (-60 dB/dec), while the peak side-lobe is -57 dB. However, the main-lobe width is now three times more than with the rectangular window and 50% greater than with the Hanning and Hamming windows.

Table 2.4. *Windowing Applied to Discrete Fast Fourier Transforms.*

Window Transform	*Side-lobe Peak Level*	*Roll-off*	*Main-lobe Width*	*Arithmetic Operations for FFT*
Rectangular	-13 dB	-6 dB/oct -20 dB/dec	$2\frac{2\pi}{NT}$	$N\log_2 N$
Hanning	-31 dB	-18 dB/oct -60 dB/dec	$2\frac{4\pi}{NT}$	$3N+N\log_2 N$
Hamming	-41 dB	-6 dB/oct -20 dB/dec	$2\frac{4\pi}{NT}$	$3N+N\log_2 N$
Blackman	-57 dB	-18 dB/oct -60 dB/dec	$2\frac{6\pi}{NT}$	$5N+N\log_2 N$

To calculate the DFT directly, N^2 complex multiplications and additions must be performed. With the FFT algorithm, on the other hand, $N \log_2 N$ operations are needed to get the rectangular window FFT. The number of operations is increased with respect to windowing. Table 2.4 summarizes the peak side-lobe level, the roll-off, the resulting main-lobe width, and the corresponding arithmetic operations.

In many applications, one would like the spectral resolution to be fixed at Δf. Δf defines the main-lobe width of the window function. Referring to Table 2.4, the rectangular window requires N' points, the Hanning and Hamming windows command $2N'$ points, and the Blackman window calls for $3N'$ points. Because the factor $3N'$ is not a power of 2, as it must be for FFT algorithms, $4N'$ points are used resulting in a main-lobe width that is slightly smaller than Δf.

One can see that the arithmetic operations, "#op", increase dramatically. Table 2.5 compares the computational complexity for fixed spectral resolution (main lobe width is constant).

A detailed analysis of the number of real multiplications and additions as a function of the blocklength, N, can be found in [Kunt91, Blah85].

Table 2.5 *Fixed Spectral Resolution and Windowing Applied to Discrete Fast Fourier Transforms.*

Δf fixed *Δf=fs/N*	*Rectang.*	*Hanning*	*Hamming*	*Blackman*
Δf=fs/64	N=64 #op=384	N=128 #op=1'280	N=128 #op=1'280	N=192->256 #op=3'328
Δf=fs/128	N=128 #op=896	N=256 #op=2'816	N=256 #op=2'816	N=384->512 #op=7'168
Δf=fs/256	N=256 #op=2'048	N=512 #op=6'144	N=512 #op=6'144	N=1536->2048 #op=32'768
Δf=fs/512	N=512 #op=4'608	N=1024 #op=13'312	N=1024 #op=13'312	N=3072->4096 #op=69'632

Within the scope of this thesis, no further studies about windowing possibilities are investigated. For a more in depth analysis, the reader can refer to [Jack86].

2.3. SHORT-TIME FOURIER ANALYSIS

We will now consider applications where signals are driven by dynamic processes, as seen in speech processing, power network signal analysis, and measurement equipment, amongst others. This spectrum will change with time. A time-frequency representation must be based on short-time Discrete Fourier Transform, which, of course, varies with time. It is obtained by a windowed segment, $w(n-m)$, that slides in time over the signal, $x(m)$, as depicted in Figure 2.2. At time instant n the spectrum is defined as

$$X(n,\theta) = \sum_m w(n-m)x(m)e^{-jm\theta} \quad , \quad \theta = \omega T \qquad (2.14)$$

The results can be interpreted in two distinct ways. First, if we assume that n is fixed, we note that $X(n,\theta)$ is simply the normal Fourier transform of the sequence $w(n-m)x(m)$. Therefore for fixed n, $X(n,\theta)$ has the same properties as the normal Fourier transform.

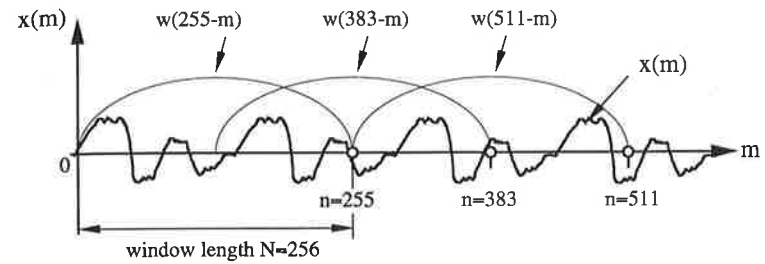


Figure 2.2. *Sliding window for short-time Fourier analysis.*

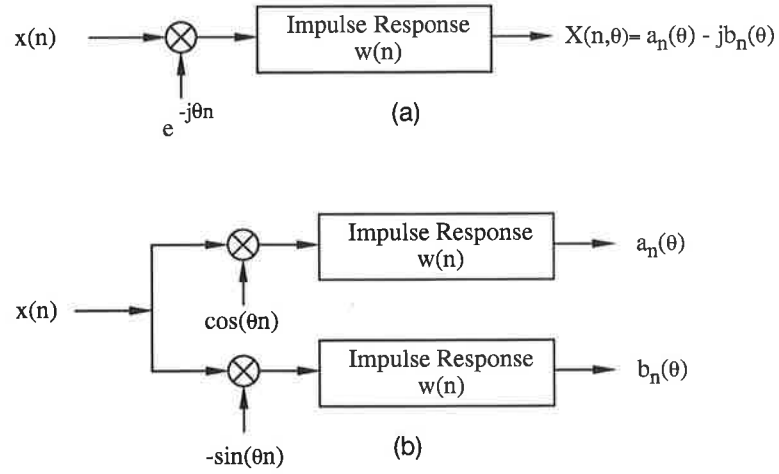


Figure 2.3. *Short-time Fourier analysis. Linear filtering interpretation (a) complex operations; (b) real operations only [Rabi78].*

The time dependent Fourier transform is clearly a function of two variables: the time index, n, which is discrete, and the frequency variable, $\theta = \omega T$. Thereby, not only is a pure spectrum analysis performed, but moreover, a time-frequency analysis around time n is obtained which spans over a time frame defined by the window length N.

The second interpretation follows by considering $X(n,\theta)$ as a function of time index n with θ fixed. In this case we observe that Equation 2.14 is in the form of a convolution, $X(n,\theta) = x(n)e^{-j\theta n} * w(n)$. This interpretation leads us to consider the time dependent Fourier representation in terms of linear filtering, shown in Figure 2.3. The window function, $w(n)$, plays the role of the impulse response of the linear shift invariant system. Thus the spectrum of $x(n)$ at frequency θ is shifted to zero frequency by $e^{-j\theta n}$. As a result of the modulation process by $e^{-j\theta n}$, the low pass filter becomes a very narrow passband. The cutoff frequency $f_c = f_s/N$ of the passband is given by the sampling frequency f_s and the window length N.

2.4. LIMITATIONS OF FOURIER TRANSFORM

For specific applications, this behavior must be balanced because errors occur during spectrum analysis. An analogy can be seen in the *uncertainty relation*, the famous quantum physics principle found in 1927 by Werner Heisenberg: *It is a priori impossible to determine the position and the impulse of a quantum particle at the same time.*

The Fourier approach is limited by:

1. The spectral resolution, defined as the number of points considered by the FFT algorithm.

   $\Delta f = (2\pi/NT) w_{type}$ $\quad$ w_{type}: - 1 Rectangular, - 2 Hamming & Hanning, - 3 Blackman

2. The time resolution, $\Delta t = NT$, which is defined by the window length. $\Delta t \Delta f = 2\pi w_{type} = const.$

By definition, the signal must be stationary over the considered time frame. The time behavior of the spectrum can be evaluated by sliding the window. A shorter time window introduces better time resolution. Constraints on the signal become less significant and thus signals with large dynamics can be analyzed (quasi-stationarity). The drawback, however, is that the spectral resolution is decreased because Fourier analysis has to be performed on fewer samples. A larger window introduces decreased time resolution, but with increased spectral resolution.

2.5. WAVELET TRANSFORM

Non-stationary signals are best described in the time-frequency plane (TF-plane). Time-frequency information can be obtained by a number of methods. In the previous paragraph, the uncertainty principle illustrated that there will be signals for which neither the time nor the frequency resolution is adequate.

The drawback of Fourier-based transform centers on the assumption that the analysis functions are cosine and sine signals which must be stationary over the analysis time frame. The Wavelet Transform is a relatively new approach that offers an alternative to the well known Fourier transform. An interesting parameter that can be changed is the time-frequency resolution, $\Delta t \Delta f = variable$.

The first published papers were written by French geo-physician Jean Morlet, in 1982. He derived this approach to describe the propagation of acoustic waves through the earth's ground. Yves Meyer and Ingrid Daubechies generalized the ideas to describe orthogonal wavelet families. In 1989, Stéphane Mallat, of New York University, published an efficient algorithm for the numerical computation of Wavelet Transform [Mall89].

Over the past four years, the application of Wavelet Transform has extended to speech and image processing applications and to various implementation structures. More detailed information can be found in [Barl92, Cheu92, Gopi92, Gopi93, Hwan92, Hanr92, Irin92, Joun92, Kova92, Luo92, Malv92, Moay92, Soma92, Sinh92, Petr92, Sait92, Tewf92].

The Wavelet Transform (WT) is a signal independent of time-frequency analysis, suited for instances when high frequency signal components have shorter durations than low frequency signal components. The considered analysis functions are obtained through translations and dilations of the wavelet function. One shrinks/stretches the wavelet to capture high/low frequency components of the signal [Gopi93]. The Wavelet Transform coefficients estimate the presence of a signal component at a certain point on the TF plane.

2.5.1. The Gauss-Wavelet Transform

The basic idea is a wavelet that is well-defined in both time and frequency, as shown in Figure 2.4. The depicted wavelet is a Gauss envelope modulated by a cosine wave — called the Gauss-wavelet.

$$\psi(t) = e^{-t^2/2} e^{j\omega t} \tag{2.15}$$

The resulting Wavelet Transform is described by

$$X(b,a) = |a|^{-1/2} \int_{-\infty}^{+\infty} x(t)\psi_{b,a}(t)dt \tag{2.16}$$

The time variable, t, can be substituted with $t \rightarrow (t-b)/a$. A family of Gauss-wavelets can be generated by changing the parameters, a and b. Parameter a is coupled with frequency ω. For $a>1$, the wavelet is submitted to a dilation (Figure 2.4d) For $a<1$, compression is achieved (Figure 2.4c). Parameter b corresponds to a time translation (Figure 2.4b).

In both short-time DFT and WT, the transform is obtained by multiplying the integral of the product of the signal by the analysis function. The main difference between the two is seen in the time resolution versus the frequency resolution. The difference is clarified graphically in Figure 2.5.

The frequency, or bandwidth, Δf, and time resolution, Δt, of the short-time DFT is independent of the frequency because the number of samples inside the time frame is fixed at N — Figure 2.5a. Thus, the error ellipses remain unchanged along the frequency axis.

Compared to the WT, the frequency resolution is much higher for low frequencies and becomes worse at higher frequencies. The relative frequency resolution, $\Delta f / f$, is fixed — in other words, the Q-factor remains constant. While the approach that must be used is certainly dependent on the specific application, it is often efficient to use a constant Q-factor rather than a constant bandwidth. A typical example is audio application, where the human hearing resolution is not constant.

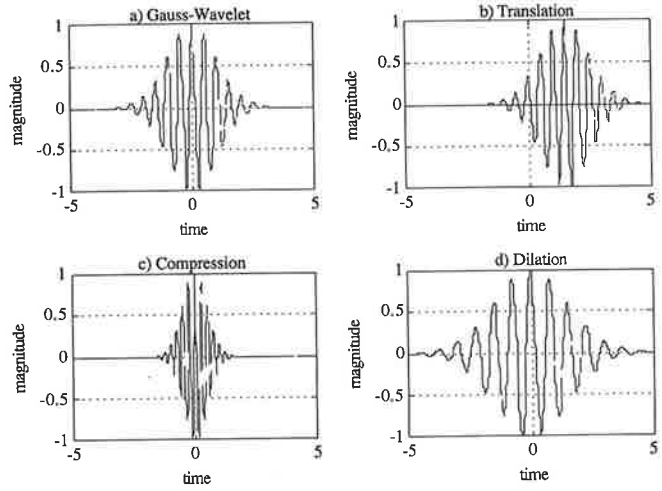


Figure 2.4. *Translation, compression, and dilation of a Gauss-wavelet.*

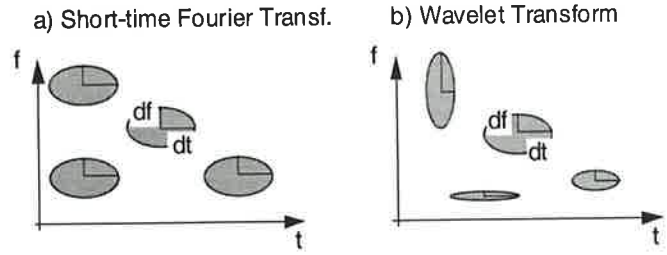


Figure 2.5. *Frequency versus time resolution (error ellipses): (a) short-time Fourier transform offering constant bandwidth; (b) Wavelet Transform offering relatively constant bandwidth, i.e. constant Q-factor.*

With Wavelet Transform, a signal can be decomposed into many different time and frequency scales, allowing both low-frequency, long duration phenomena, and high-frequency, short duration

aspects to be studied within the same transform. There are many applications for WTs, including image and speech coding and optimal data transmission over channels of unknown bandwidth. There also exist many other families which are not necessarily based on the Gauss function.

2.5.2. Fast Discrete Wavelet Transform

Computation-efficient WTs can be implemented with filter banks or hierarchical transforms [Mall89, Malv91, Malv92, Vett90, Luo92]. Orthogonal WTs with octave scales are usually generated from a tree-structured filter bank. With this form of decomposition a pyramid algorithm is obtained to compute the orthogonal WT.

The fast WT can be viewed as an example of sub-band coding. The hardware implementation can be performed by filters derived from the wavelet: A high-pass filter, and its corresponding low-pass filter. Both filters together are called the Quadrature Mirror Filter (QMF). To implement the above algorithm at maximum speed, a filter bank would be required for each octave.

The QMF is a pair of identical FIR filters that separate the signal components into a low-frequency part (LP) and a high-frequency part (HP). The frequency separation is located at f_s / 4, where f_s is the operation (local) sampling rate. The number of stages is application specific and splits the spectrum into octaves. In Figure 2.5, a four-stage WT pyramid algorithm is depicted.

The approximation coefficient, a^0, and the detail coefficients, b^0, b^1, b^2 *and* b^3, compose the Wavelet Transform of the input signal, $x(k)$ [Malv92]. The agreeable feature of the pyramid algorithm is that the LP and the HP Quadrature Mirror Filter pairs all have the same filter specifications. These filter coefficients are used for any number of stages, N. The octave splitting is achieved by the decimation factor of two (down-sampling) together with the half-band filter specifications.

To achieve the inverse WT, the data flow graph is inverted and each down-sampling is replaced by up-sampling by a factor of two. The number of operations increases proportionally with the filter length, N, as compared to $N \log_2(N)$ for the FFT algorithm.

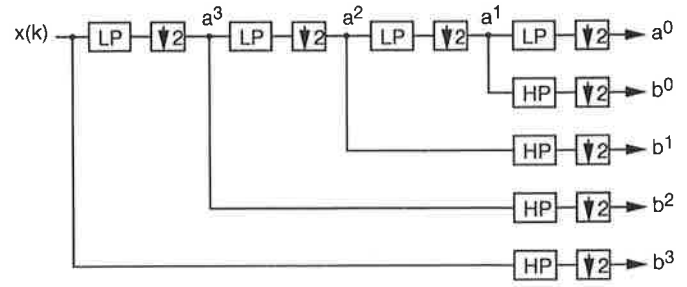


Figure 2.6. *Mallat pyramid algorithm based on QMF pairs.*

2.6. HISTORICAL SURVEY OF LEAST-SQUARES ESTIMATION

2.6.1. From Gauss to Kalman

Gauss: Method of least squares

The earliest motivation for the development of estimation theory was apparently provided by astronomical studies in which planet and comet motion was examined using telescopic measurement data. To solve problems concerning the revolution of astronomical bodies, the method of least squares was invented by Karl Friedrich Gauss. Gauss was 18 years old when he derived his revolutionary least-squares method in 1795 [Sore85, Sore70]. He suggested that the most appropriate values for the unknown but desired parameters are the *most probable values*, which he defined in the following manner: "The *most probable value* of the unknown quantities will be that in which the sum of the squares of the differences between the actually observed and the computed values multiplied by the numbers that measure the degree of precision is a minimum". The difference between the observed and computed measurement values is generally called the residual.

First, it is significant that he considered the problem from a probabilistic point of view and attempted to define the best estimate as the most probable value of the parameters. To make the discussion more precise, suppose that measurement quantities, $\mathbf{z}(k)$, are available at each time, k. Parameters $\mathbf{x}(k)$ are to be

determined from the data and are related according to

$$\mathbf{z}(k) = \mathbf{H}(k)\mathbf{x}(k) + \mathbf{v}(k) \tag{2.17}$$

where $\mathbf{v}(k)$ represent the measurement errors that occur at each observation time. Gauss assumed that the measurement data, $\mathbf{z}(k)$, and the parameters, $\mathbf{x}(k)$, are linearly related.

Denoting the estimate, $\hat{\mathbf{x}}(k)$, of $\mathbf{x}(k)$ based on the k data samples $\{\mathbf{z}(0),\mathbf{z}(1),\ldots,\mathbf{z}(k)\}$ as $\hat{\mathbf{x}}(k)$, the residual associated with the kth measurement is

$$\mathbf{r}(k) \stackrel{\Delta}{=} \mathbf{z}(k) - \mathbf{H}(k)\hat{\mathbf{x}}(k) \qquad k = 0,1,\ldots,\infty \tag{2.18}$$

The *most probable value* $\mathbf{x}(k)$, that is $\hat{\mathbf{x}}(k)$, is defined as the value that minimizes the sum of the squares of the residuals. Thus, choose $\mathbf{x}(k)$ so that

$$L(k) \stackrel{\Delta}{=} \frac{1}{2}\sum_{i=0}^{k}[\mathbf{z}(i) - \mathbf{H}(i)\mathbf{x}(i)]^T \mathbf{W}(i)[\mathbf{z}(i) - \mathbf{H}(i)\mathbf{x}(i)] \tag{2.19}$$

is minimized. The elements of the matrixes $\mathbf{W}(k)$ are selected to indicate the degree of confidence that one can place in the individual measurements.

He reasoned that errors in the measurements would be independent of each other, so the joint-probability density function of the measurement residuals can be expressed as the product of the individual density functions

$$f(\mathbf{r}(0),\mathbf{r}(1),\ldots,(\mathbf{r}(k)) = f(\mathbf{r}(0))f(\mathbf{r}(1))\ldots f(\mathbf{r}(k)) \tag{2.20}$$

Next, he argued that the density, $f(\mathbf{r}(k))$, would be a normal density

$$f(\mathbf{r}(k)) = \frac{\sqrt{\det \mathbf{W}(k)}}{(2\pi)^{m/2}} \exp[-\tfrac{1}{2}\mathbf{r}(k)^T \mathbf{W}(k)\mathbf{r}(k)] \tag{2.21}$$

although he recognized that one never obtains errors of infinite magnitude. Gauss proceeded by noting that the maximum of the probability density function is determined by maximizing the logarithm of this function. Thus, he anticipated the maximum likelihood method, which was introduced by R.A. Fisher in 1912 and

has been thoroughly investigated to the present time. Gauss is minimizing some function of the difference between estimate and observation, and thus independent of probability theory defined by Equation 2.19.

Wiener-Kolmogorov: Step to stochastic estimation

Independently, Kolmogorov (1941) and Wiener (1942) developed a linear minimum mean-square estimation technique that received considerable attention and provided the foundation for the Kalman filter theory. They considered the estimation problem when measurements are obtained continuously, as well as discretely.

Consider the problem of estimating a signal, $\mathbf{s}(k)$, possibly time-varying, from measurements data $\{z(0),\mathbf{z}(1),\ldots,\mathbf{z}(k)\}$, where the $\mathbf{s}(k)$ and the $\{\mathbf{z}(k)\}$ are related to the knowledge of the *cross-correlation functions*. Assume that the estimate of $\mathbf{s}(k)$, say $\hat{\mathbf{s}}(k)$, is to be computed as a linear combination of the measurements, $\mathbf{z}(i)$:

$$\hat{\mathbf{s}}(k) = \sum_{i=0}^{k}\mathbf{H}(k,i)\mathbf{z}(i) \tag{2.21}$$

The *filter gains*, $\mathbf{H}(k,i)$, are to be chosen such that the mean-square error is minimized; that is, choose the $\mathbf{H}(k,i)$ so that

$$\mathbf{M}(k) = E[(\mathbf{s}(k) - \hat{\mathbf{s}}(k))^T(\mathbf{s}(k) - \hat{\mathbf{s}}(k))] \tag{2.22}$$

is minimized. Wiener and Hopf derived the necessary and sufficient condition for $\hat{\mathbf{s}}(k)$ to minimize $\mathbf{M}(k)$: The error in the estimate $\tilde{\mathbf{s}}(k) \stackrel{\Delta}{=} (\mathbf{s}(k) - \hat{\mathbf{s}}(k))$ must be orthogonal to the measurement data. The expectation in a mean-squared sense is

$$E[\tilde{\mathbf{s}}(k)\mathbf{z}(i)^T] = 0 \qquad i = 0,1,\ldots,k \tag{2.23}$$

This is the Wiener-Hopf equation for stationary environments, which is frequently written as

$$E[\mathbf{s}(k)\mathbf{z}(i)^T] = \sum_{j=0}^{k}\mathbf{H}(k,j)E[\mathbf{z}(j)\mathbf{z}(i)^T] \qquad i = 0,1,\ldots,k \tag{2.24}$$

This equation must be solved for $\mathbf{H}(k,j)$ in order to obtain the

gains of the optimal filter.

The problem formulated and described here is significantly different from Gauss least-squares problem. First, no assumption that the signal is constant is imposed. The signal can be different at each k but can be described *statistically* by the *autocorrelation* and *crosscorrelation functions* of the *signal* and *measurement data*. Second, a *probabilistic version* of the least-squares method is chosen as the performance index rather than requiring that the estimate be the most probable.

Follin: Recursive stochastic estimation

In 1955, J.W. Follin at Johns Hopkins University suggested a recursive approach based on the idea that the measurements are described by

$$\begin{aligned}\mathbf{z}(k) &= \mathbf{s}(k) + \mathbf{w}(k) \\ &= \mathbf{H}(k)\mathbf{x}(k) + \mathbf{w}(k)\end{aligned} \tag{2.25}$$

where $\mathbf{w}(k)$ is a white-noise sequence. The system state vector, $\mathbf{x}(k)$, is assumed to be described as a dynamic system having the form

$$\mathbf{x}(k+1) = \mathbf{A}(k)\mathbf{x}(k) + \mathbf{v}(k) \tag{2.26}$$

where $\mathbf{v}(k)$ represents a white-noise sequence. It seems intuitively reasonable that estimates of $\mathbf{s}(k+1)$ (or $\mathbf{x}(k+1)$) could be derived, given a new measurement $\mathbf{z}(k+1)$, from $\hat{\mathbf{s}}(k)$ and $\mathbf{z}(k+1)$ rather than from $\mathbf{z}(0),\mathbf{z}(1),\ldots,\mathbf{z}(k),\mathbf{z}(k+1)$, since $\hat{\mathbf{s}}(k)$ is based on the data $(\mathbf{z}(0),\mathbf{z}(1),\ldots,\mathbf{z}(k))$. This approach gave the foundation for the developments that are now referred to as the Kalman filter. Follin's work provided a direct stimulus for the work of Richard Bucy, who subsequently collaborated with Kalman in the continuous-time version of the filter equation. Kalman published his first paper on discrete-time, recursive mean-square filtering in 1960. The developments, beginning with Wiener's work and culminating with Kalman's, reflect the fundamental changes that have occurred in control system theory during this period. In the "classical control theory", the emphasis was on the analysis and synthesis of systems in terms their input-output characteristics. The basic tools used for these problems were the Laplace and Fourier transforms.

The state-space approach makes use of difference and differential equations rather than the integral equations of the classical Laplace and Fourier transforms. It seems to be more satisfying to work with differential equations since dynamic systems are generally described in this manner.

2.6.2. Kalman Filter — A Perspective

At this point, let us summarize the Kalman filtering problem. Its solution will be detailed in Chapter 3. The Kalman filter equations provide an extremely convenient procedure for digital computer implementation [Kail68, Sore70]. Using the Kalman filter, one can develop a computer program *directly*, requiring little understanding of the theory that led to its development. There are well established numerical procedures for solving differential equations, so the engineer does not have to be worried about this problem. In contrast, the solution of the Wiener-Hopf equation and the implementation of the Wiener-Kolmogorov filter must be regarded as more difficult or there would have been no need for the Kalman filter. Since Gauss was very concerned with the computational aspects of least-squares applications, one can imagine that he would appreciate the computational benefits of the Kalman filter.

The Kalman filter, which assumes linear systems, has found its greatest application to nonlinear systems. It is generally used in these problems by assuming knowledge of an approximate solution and by describing the deviations from the reference by linear equations.

The Kalman filter, or Kalman estimator, is a particular state-space estimator which approximates the state of a discrete linear system that evolves dynamically. It is a stochastic design model, as opposed to a deterministic design model, e.g. the Fourier approach. Figure 2.7 illustrates the basic principles of a dynamic system.

Suppose the Dynamic Signal Model, with *discrete-time* k, is driven by two noise sources, measurement noise $w(k)$, and process noise $\underline{v}(k)$. The upper left portion of Figure 2.7 indicates the Dynamic Signal Model that is used to model the input samples, $y(k)$. In the lower portion, the Kalman filter uses the new input data, $y(k)$, to produce optimal estimates of the states of the system.

For both noise sources, the mean value is zero and the covariances are **R** and **Q**. The following conditions are given:

$$E\{\underline{x}(0)\} = \underline{x}_0 \quad Var[\underline{x}(0)] = P_0$$

$$E\{\underline{v}(0)\} = 0 \quad Cov[\underline{v}(k), \underline{v}(l)] = \mathbf{Q}\delta(k-l) = diag(q1, q2, \ldots)$$

$$E\{w(0)\} = 0 \quad Var[w(k), w(l)] = \mathbf{R}\delta(k-l) = r\delta(k-l)$$

$$Cov[\underline{v}(k), w(l)] = 0 \quad \text{for all k,l}$$

$$Cov[\underline{v}(k), \underline{x}(0)] = 0 \quad Cov[w(k), \underline{x}(0)] = 0 \quad \text{for all k}$$

$$\text{where} \quad \delta(i) = \begin{cases} 1 \text{ for i} = 0 \\ 0 \text{ for i} \neq 0 \end{cases}$$

The state equation, $\underline{x}(k+1)$; the modeled and measured input data, $y(k)$; and the desired output, $\underline{z}(k)$, are given by a set of three equations. They are expressed as

$$\underline{x}(k+1) = \mathbf{A}\underline{x}(k) + \mathbf{G}\underline{v}(k) \tag{2.27}$$

$$y(k) = C^T \underline{x}(k) + w(k) \tag{2.28}$$

$$\underline{z}(k) = \mathbf{H}\underline{x}(k+1)$$
$$(\ e.g.\ :\ \underline{z}(k)^T = [\underline{a}(k), \underline{b}(k), \underline{a}_d(k), \underline{b}_d(k)]^T\ ) \tag{2.29}$$

The goal is to define the optimal estimator of the state equation in a mean-squared sense, to determine the Fourier coefficients of the sinusoid of interest, $\underline{z}(k)^T = [\underline{a}(k), \underline{b}(k), \underline{a}_d(k), \underline{b}_d(k)]^T$. The input samples, $y(k)$, are fed into the Kalman filter that calculates the estimates, $\hat{\underline{x}}(k)$, of the states, $\underline{x}(k)$, of the Dynamic Signal Model. The Kalman filter provides two optimal estimates of the state of the signal model.

The first step is to calculate the one-step-ahead prediction, $\hat{\underline{x}}^-(k+1)$, based on the evaluations up to k estimates, $\hat{\underline{x}}(k)$. The two design parameters for the Kalman estimator are the covariance matrix, $Cov[\underline{v}(k), \underline{v}(l)]$, of the system noise, $\underline{v}(k)$, and the variance, $Var[w(k), w(l)]$, of the measurement noise, $w(k)$.

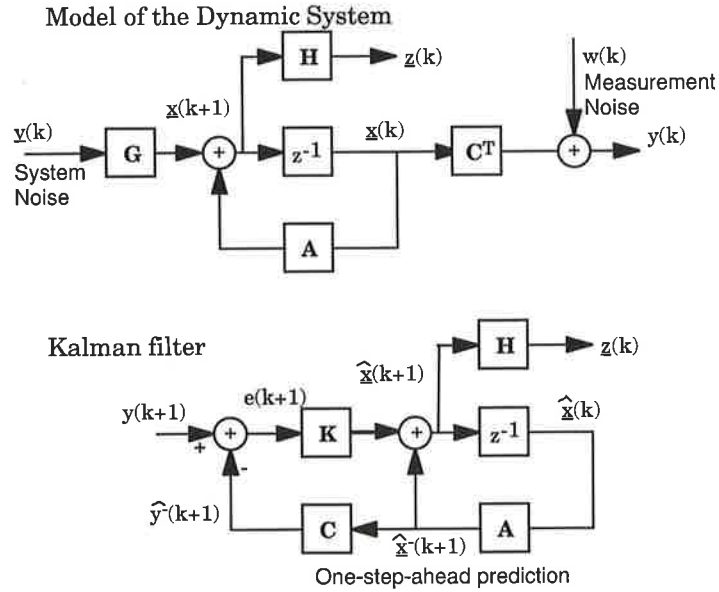


Figure 2.7. *Dynamic Signal Model and Kalman filter.*

The Dynamic Signal Model reappears, this time in the Kalman filter, but without the noise sources: $w(k)$ is no longer used and $\underline{v}(k)$ is replaced by new input. The Dynamic Signal Model generates, in addition to the output results, $\underline{z}(k)$, an estimation of the input vector, $\hat{y}^-(k+1)$. $\hat{y}^-(k+1)$ is the estimate of the input up to $k+1$. The estimated input is used to determine the residual error signal, $e(k+1)$. The error is the difference between the measured input vector, $y(k+1)$, and its one-step-ahead predicted value, $\hat{y}^-(k+1)$.

The error is multiplied by the Kalman Gain, $\mathbf{K}(k+1)$, which is a generalized P-controller for this loop. Depending on the Dynamic Signal Model used, the Kalman Gain will vary. The Kalman Gain is dependent on the design parameters. For a certain class of applications where the statistics of the noise sources is time invariant. the Kalman gain becomes stationary. This can often be done, but only if, for both noise sources, the covariances, **R** and **Q**, are time-invariant.

Various Dynamic Signal Models produce different matrices, **A**,**G**,**H**, and a different vector, **C**. For each of the models, the estimator is optimal, in the mean-square sense, only if the signal variation follows the assumed model.

2.7. SUMMARY

In this chapter we have surveyed basic techniques for spectrum analysis. The most common — DFT, FFT, short-time DFT, and WT — have been covered but not exhaustively examined.

Discussion has centered on the estimation of the spectral components of quasi-stationary sinusoidal signals measured in noise. The investigation is based on non-uniform spacing of estimated frequencies along the frequency axis. Specifically, time-frequency analysis and the complexity of the operations involved in the algorithms were studied.

Use of a Dynamic Signal Model rather than a Static Signal Model is preferable because signals measured in noise are stochastic by nature. A stochastic design (e.g. Kalman theory) incorporates the dynamics of the signal model by a system noise source $\underline{v}(k)$ and a measurement noise source $w(k)$ and therefore offers advantages over a deterministic design (e.g. Fourier analysis).

REFERENCES

[Barl92] M. Barlaud et al : "A pyramidal scheme for lattice vector quantization of wavelet transform coefficients applied to image coding", Proc. IEEE ICASSP-92, Int. Conf. on Accoustics, Speech, and Signal Processing, March 23-26, San Francisco, 1992.

[Blah85] R.E. Blahut : "Fast algorithms for digital signal processing", Addison-Wesley, New York 1995.

[Cheu92] A.C. Cheung et al : "Real-time detection of transient signals using spline wavelets", Proc. IEEE ICASSP-92, Int. Conf. on Accoustics, Speech, and Signal Processing, March 23-26, San Francisco, 1992.

[Gopi92] R.A. Gopinath, C.S. Burrus : "Wavelet-based lowpass/ bandpass interpolation", Proc. IEEE ICASSP-92, Int. Conf. on Accoustics, Speech, and Signal Processing, March 23-26, San Francisco, 1992.

[Gopi93] R.A.Gopinath, C.S. Burrus : "A tutorial overview of filter banks, wavelets and interrelations", Proc. IEEE of International symposium on circuits and systems, Chicago, May 3-6, 1993, pp.104-107.

[Hwan92] W.L. Hwang, S. Mallat : "Singularities and noise discrimination with wavelets", Proc. IEEE ICASSP-92, Int. Conf. on Accoustics, Speech, and Signal Processing, March 23-26, San Francisco, 1992.

[Hanr92] H.E. Hanrahan : "A family of wavelets which are dilatable by simple IIR filters", Proc. IEEE ICASSP-92, Int. Conf. on Accoustics, Speech, and Signal Processing, March 23-26, San Francisco, 1992.

[Irin92] T. Irino, H. Kawahara : "Signal reconstruction from modified wavelet transform - An application to auditory signal processing", Proc. IEEE ICASSP-92, Int. Conf. on Accoustics, Speech, and Signal Processing, March 23-26, San Francisco, 1992.

[Jack86] L.B. Jackson : "Digital Filters and Signal Processing", Kluwer Academic Publishers, Boston, 1986.

[Joun92] I. Jouny: "Target description using wavelet transform", Proc. IEEE ICASSP-92, Int. Conf. on Accoustics, Speech, and Signal Processing, March 23-26, San Francisco, 1992.

[Kail68] T. Kailath : "An Innovations Approach to Least-Squares Estimation, Part I; Linear Filtering in Additive Noise", IEEE Trans. Automat. Contr., vol.AC-13, pp.646-655, Dec.1968.

[Kova92] J. Kovacevic, M. Vetterli : "Design of multidimensional non-separable regular filter banks and wavelets", Proc. IEEE ICASSP-92, Int. Conf. on Accoustics, Speech, and Signal Processing, March 23-26, San Francisco, 1992.

[Kalm61] R.E. Kalman, R.S. Bucy . : "New Results in Linear Filtering and Prediction Theory", IEEE Trans. on Acc. Communications, Vol.COM-26, No.10, October 1983, pp. 1439-1446.

[Kalm61] R.E. Kalman, R.S. Bucy : "New Results in Linear Filtering and Prediction Theory", Trans. on the ASME, Journal of Basic Engineering, March 1961, pp. 95-108.

[Kalm60] R.E. Kalman : "A New Approach to Linear Filtering and Prediction Problems", Trans. on the ASME, Journal of Basic Engineering, March 1960, pp. 35-45.

[Kunt91] M. Kunt et al : "Techniques modernes de traitement numérique des signaux", Presse Polytechnique et Universitaires Romandes, Lausanne, 1991.

[Luo92] Y. Luo, D.N. Pinder, R.C. O'Driscoll : "Real-time implementation of wavelet transform in a dual DSP56001 system", Proc. IEEE ICASSP-92, Int. Conf. on Accoustics, Speech, and Signal Processing, March 23-26, San Francisco, 1992.

[Mall89] S.G. Mallat : "A theory for multiresolution signal decomposition: the wavelet representation", IEEE Trans. Pattern Anal. Machine Intell., vol.11, pp. 674-693, July 1989.

[Moay92] N. Moayeri, et al : "Wavelet transform image coding using trellis coded vector qunatization", Proc. IEEE ICASSP-92, Int. Conf. on Accoustics, Speech, and Signal Processing, March 23-26, San Francisco, 1992.

[Malv91] H.S. Malvar : "Extended lapped transforms: fast algorithms and applications", Proc. IEEE ICASSP-91, Int. Conf. on Accoustics, Speech, and Signal Processing, May 1991, Toronto, pp. 1797-1800.

[Malv92] H.S. Malvar : "Fast computation of wavelet transforms with the extended lapped transform", Proc. IEEE ICASSP-92, Int. Conf. on Accoustics, Speech, and Signal Processing, March 23-26, San Francisco, 1992.

[Open83] A .V. Oppenhaim, A.S. Willsky, I. T. Young : "Signals and Systems", Prentice-Hall, Englewood Cliffs, New Jersy, 1983.

[Pell94] F. Pellandini : "Signaux et Systems", Cours destiné aux étudiants en Microtechnique de 3ème année, EPFL-DMT, Lausanne, 1994.

[Petr92] A.P. Petropulu : "Detection of transients using discrete wavelet transforms", Proc. IEEE ICASSP-92, Int. Conf. on Accoustics, Speech, and Signal Processing, March 23-26, San Francisco, 1992.

[Rabi78] L.R. Rabiner, R.W. Schafer : "Digital Signal Processing of Signals", Prentice-Hall, Englewood Cliffs, New Jersey, 1978.

[Sait92] N. Saito, G. Beylkin: "Multiresolution representations using the auto-correlation function of compactly supported wavelets", Proc. IEEE ICASSP-92, Int. Conf. on Accoustics, Speech, and Signal Processing, March 23-26, San Francisco, 1992.

[Schw75] M. Schwartz, L. Shaw : "Signal processing discrete spectral analysis, detection, and estimation", McGraw-Hill, New York, 1975.

[Soma92] A.K. Soman, P.P. Vaidyanathan: "Paraunitary filter banks and wavelet packets", Proc. IEEE ICASSP-92, Int. Conf. on Accoustics, Speech, and Signal Processing, March 23-26, San Francisco, 1992.

[Sore85] H.W. Sorenson, "Kalman Filtering: Theory and Application", IEEE Press, New York, 1985, ISBN 0-87942-191-6.

[Sore70] H.W. Sorenson, "Least-square estimation: from Gauss to Kalman", IEEE Spectrum, vol. 7, July 1970, pp.63-68.

[Tewf92] A.H. Tewfik, M. Kim : "Fast multiscale statistical signal processing algorithms", Proc. IEEE ICASSP-92, Int. Conf. on Accoustics, Speech, and Signal Processing, March 23-26, San Francisco, 1992.

[Sinh92] D. Sinha, A. H. Tewfik : "Synthesis/coding of audio signals using optimized wavelets", Proc. IEEE ICASSP-92, Int. Conf. on Accoustics, Speech, and Signal Processing, March 23-26, San Francisco, 1992.

[Vett90] M. Vetterli, C. Herley : "Wavelets and filter banks: relationships and new results", Proc. IEEE ICASSP-90, Int. Conf. on Accoustics, Speech, and Signal Processing, April 1990, pp. 1723-1726.

3. A Kalman Filter Approach for Short-Time Spectrum Analysis

Chapter 1 presented the differences between the methodologies of Dynamic and Static Signal Models. In Chapter 2, the Kalman filter was introduced.

In this chapter, a stochastic design is used for spectrum analysis. A detailed discussion of the time and frequency characteristics of the Kalman filter is undertaken.

Sinusoidal signals corrupted by noise are then considered. For these signals, the stochastic approach to spectrum estimation is compared to the deterministic approach.

Two mathematical models for analyzing sinusoids are derived, which use a Kalman filter: An Oscillator Signal Model and a Random Walk Signal Model.

Lastly, time-frequency analysis is contrasted with windowed short-time Fourier analysis.

3.1. INTRODUCTION

In 1961, Kalman and Bucy published a powerful recursive estimation technique, now known as the Kalman filter [Kalm60, Kalm61]. This filter solves linear system problems: It is the optimal estimator of the Gaussian noise process and, when the assumption

of Gaussian noise no longer holds, it is the linear minimum-variance estimator.

The distinctive feature of a Kalman filter is that its mathematical formulation is based on state-space equations in the time domain [Hayk91]. Another important characteristic is that its solution is computed recursively — each updated estimate of a state is computed from combining its previous estimate with the new input. This is unique because only the previous estimate requires storage; there is no longer the need to store *all* of the past observed data.

The Kalman filter is computationally more efficient because the estimate is calculated directly and not at each step of the filtering process (block processing of data). The Kalman filter is ideal for real-time implementations and has been applied successfully to problems in many fields — notably in aerospace and aeronautics.

An *adaptive system* is one which allows changes in its parameters according to changes in the environment. Rudolph E. Kalman has shown that such changes affect only the values of the parameters and not the structure and order of the ideal filter [Kalm61].

3.2. KALMAN FILTER FORMULATION

Notation

For the most part, we will use conventional notation: $\tau, t, t_o, T, \ldots$ refer to time; $\alpha, \beta, \omega, \phi, \ldots$ are scalars; $\underline{x}, \underline{z}, \underline{y}, \underline{v}, \underline{w}, \ldots$ are vectors; and $\mathbf{A}, \mathbf{H}, \mathbf{G}, \mathbf{C}, \ldots$ are matrices; A prime ($'$) or a T in the exponent identifies the transposed matrix. For example, $x'y$ is the scalar (inner) product and xy' will produce a matrix made up of elements $x_i y_j$ (outer product) [Kalm61]. The expected value (ensemble average) is expressed by $E\{\underline{v}(t)\}$. Depending on which is more convenient, the covariance matrix of two vector-valued random variables can be denoted by either

$$E\{\underline{v}(t)\underline{w}'(\tau)\} - E\{\underline{v}(t)\}E\{\underline{w}'(\tau)\}$$

or

$$cov[x(t), y(\tau)]. \tag{3.1}$$

State-Space Signal Model

In order to apply a Kalman filter, one must be able to describe the system by a *set of state equations* (Equations 3.2-3.4). This linear stochastic system is driven by noise sources, illustrated by Figure 3.1.

$$\underline{x}(k+1) = \mathbf{A}(k)\underline{x}(k) + \mathbf{G}(k)\underline{v}(k) \tag{3.2}$$

$$y(k) = \mathbf{C}(k)\underline{x}(k) + w(k) \tag{3.3}$$

$$\underline{z}(k) = \mathbf{H}(k)\underline{x}(k+1)$$
$$(\ e.g.\ :\ \underline{z}(k)^T = [\underline{a}(k), \underline{b}(k), \underline{a}_d(k), \underline{b}_d(k)]^T\ ) \tag{3.4}$$

The *state-transition equation* (3.2) is a vector containing the values of the M parameters. These parameters define the state of the modeled system at each time-increment, k, and is therefore the *state vector.*

Matrix $\mathbf{A}(k)$, with dimensions $M * M$, is the *state-transition matrix.* Matrix $\mathbf{C}(k)$, with dimensions $L * M$, is the *observation matrix.* In this thesis, $\underline{C}(k)$ is reduced to a vector because only a one-dimensional input signal, $y(k)$, is measured; thus, $L = 1$.

The Gaussian noise processes, $\underline{v}(k)$ and $w(k)$, are zero-mean uncorrelated noise sources. The ***system noise source***, $\underline{v}(k)$, has the covariance matrix: $Cov[\underline{v}(k), \underline{v}(l)]$. The ***measurement noise source***, $w(k)$, is reduced, in this particular application, to a scalar. Its variance is: $Var[w(k), w(l)]$.

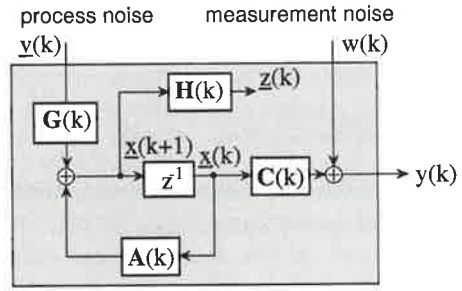


Figure 3.1. *The Stochastic Dynamic Signal Model.*

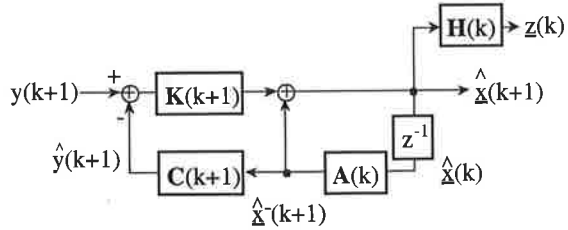


Figure 3.2. *The Kalman filter for the Dynamic Signal Model.*

Statement of the Problem

Since input data are available at each time-increment, k, we can describe the transition from k to $k+1$ by the following system dynamics:

1. Define, for each time-increment, k, the *a priori* error covariance, $\mathbf{P}^-(k+1)$. The dynamics of the system noise source $\underline{v}(k)$ is introduced by the covariance matrix $\mathbf{Q}(k)$:

$$\mathbf{P}^-(k+1) = \mathbf{A}(k)\mathbf{P}(k)\mathbf{A}^T(k) + \mathbf{G}(k)\mathbf{Q}(k)\mathbf{G}^T(k) \tag{3.5}$$

2. Compute the one-step ahead predicted estimate of each state variable, $\underline{\hat{x}}^-(k+1)$, i.e. before considering the new input data, $y(k)$:

$$\underline{\hat{x}}^-(k+1) = \mathbf{A}(k)\underline{\hat{x}}(k) \tag{3.6}$$

3. Determine the Kalman Gain, $\mathbf{K}(k+1)$, that depends on the a priori error covariance $\mathbf{P}^-(k+1)$ and the variance, $\mathbf{R}(k+1)$, of the measurement noise source $w(k+1)$:

$$\mathbf{K}(k+1) = \mathbf{P}^-(k+1)\mathbf{C}^T(k+1)\left[\mathbf{C}(k+1)\mathbf{P}^-(k+1)\mathbf{C}^T(k+1)+\mathbf{R}(k+1)\right]^{-1} \tag{3.7}$$

4. The update of the dynamics of both noise sources provide a correction to yield the net a posteriori error covariance, $\mathbf{P}(k+1)$. This is the effect of the system dynamics:

$$\mathbf{P}(k+1) = \mathbf{P}^-(k+1) - \mathbf{K}(k+1)\mathbf{C}(k+1)\mathbf{P}^-(k+1) \tag{3.8}$$

5. The final step is completed by the estimation of each state variable, $\underline{\hat{x}}(k+1)$, i.e. after considering the new input data, $y(k+1)$. The first order recursive estimator provides the corrected estimate up to time $k+1$. The corrected estimate is composed by the predicted estimate, $\underline{\hat{x}}^-(k+1)$, and the correction term inside the parenthesis. The correction term is found by the difference of the new input sample and the predicted estimated input sample. The correction term is weighted finally by the Kalman Gain. Together, Equations 3.5 through 3.9 describe the Kalman filter [Kalm61]:

$$\underline{\hat{x}}(k+1) = \underline{\hat{x}}^-(k+1) + \mathbf{K}(k+1)\left[y(k+1) - \mathbf{C}(k+1)\underline{\hat{x}}^-(k+1)\right] \tag{3.9}$$

To achieve the goal of spectrum analysis of sinusoidal signals, one must take into account the spectral components that are of interest. Usually, the amplitude of the signals is time-varying. In order to obtain accurate estimation results, it is important to track these time-variations. This can be achieved by selecting a Dynamic Signal Model that manifests the preferred characteristics of the selected input signal. In this thesis, two solutions are debated: The *Oscillator Signal Model (OSM)* and the *Random Walk Signal Model (RWSM)*.

3.3. OSCILLATOR SIGNAL MODEL

This approach to sinusoid modeling utilizes hyper-stable oscillators, each of which corresponds to a well-defined frequency, f_n [Bitm86]. The model generates two signals — a cosine signal (real part) and a sinusoidal signal (imaginary part) — both with the same amplitude and frequency.

The state variables of the Oscillator Signal Model, x_{n_1} and x_{n_2}, at frequency f_n, discrete-time k, and sampling frequency f_s, can be expressed by the state-transition matrix, $\mathbf{A}_n$, and the system noise source, $\underline{v}_n(k)$. Equation 3.2 now becomes

$$\begin{bmatrix} x_{n1}(k+1) \\ x_{n2}(k+1) \end{bmatrix} = \underbrace{\begin{bmatrix} \cos\frac{2\pi f_n}{f_s} & -\sin\frac{2\pi f_n}{f_s} \\ \sin\frac{2\pi f_n}{f_s} & \cos\frac{2\pi f_n}{f_s} \end{bmatrix}}_{\mathbf{A}_n} \cdot \begin{bmatrix} x_{n1}(k) \\ x_{n2}(k) \end{bmatrix} + \underbrace{\begin{bmatrix} 1 \\ 0 \end{bmatrix}}_{\mathbf{G}_n} \underline{v}_n(k) \tag{3.10}$$

$\mathbf{C}$ is the observation vector and $\mathbf{H}(k)$ is the output matrix that produces the desired Fourier coefficients. If M frequency lines are estimated the order of $\mathbf{A}$ is $2M * 2M$ and the order of $\mathbf{C}$ is $1 * 2M$. The observation and state transition matrix are time independent.

$$C_n = [1 \quad 0]^T \qquad \mathbf{C} = \underbrace{[1 \quad 0 \quad 1 \quad 0 \quad . \quad . \quad .]^T}_{2M} \tag{3.11}$$

$$\mathbf{H}_n(k)=\begin{bmatrix}\cos\frac{2\pi f_n}{f_s}k & \sin\frac{2\pi f_n}{f_s}k\\ \sin\frac{2\pi f_n}{f_s}k & -\cos\frac{2\pi f_n}{f_s}k\end{bmatrix} \tag{3.12}$$

To obtain the Fourier coefficients, $a_n(k)$ and $b_n(k)$ multiply the estimated state vector, $\underline{\hat{x}}(k)$, by the output matrix $\mathbf{H}(k)$ according to Equation 3.4. One obtains

$$\underline{z_n}(k)^T=[\hat{a}_n(k),\hat{b}_n(k)]^T \tag{3.13}$$

$\mathbf{H}(k)$ is time varying and has dimension $2M*2M$.

$$\mathbf{H}(k)=\begin{bmatrix}H_0(k) & 0 & .. & 0\\ 0 & H_1(k) & 0 & ..\\ .. & .. & H_{..}(k) & 0\\ 0 & 0 & 0 & H_{M-1}(k)\end{bmatrix} \qquad \mathbf{A}=\begin{bmatrix}\mathbf{A}_0 & 0 & .. & 0\\ 0 & \mathbf{A}_1 & 0 & ..\\ .. & .. & \mathbf{A}_{..} & 0\\ 0 & 0 & 0 & \mathbf{A}_{M-1}\end{bmatrix}$$

$$\mathbf{C}=[\mathbf{C}_0 \quad \mathbf{C}_1 \quad \ldots \quad \mathbf{C}_{M-2} \quad \mathbf{C}_{M-1}]$$

M: Number of estimated frequency lines

3.4. RANDOM WALK SIGNAL MODEL

With this procedure, each Fourier coefficient is a state of the system. The model is simplified to

$$\begin{bmatrix}a_n(k+1)\\ b_n(k+1)\end{bmatrix}=\begin{bmatrix}x_{n1}(k+1)\\ x_{n2}(k+1)\end{bmatrix}=\underbrace{\begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix}}_{\mathbf{A}_n}\begin{bmatrix}x_{n1}(k)\\ x_{n2}(k)\end{bmatrix}+\underbrace{\begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix}}_{\mathbf{G}_n}\begin{bmatrix}v_{n1}(k)\\ v_{n2}(k)\end{bmatrix} \tag{3.14}$$

and

$$\underline{z_n}(k)=\underbrace{\begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix}}_{\mathbf{H}_n}\begin{bmatrix}a_n(k+1)\\ b_n(k+1)\end{bmatrix} \tag{3.15}$$

The dynamic part of Equation 3.14 is the *Random Walk Signal Model.* Two states, $a_n(k)$ and $b_n(k)$, correspond, respectively, to the real part and the imaginary part of the Fourier coefficient. Both parts are driven by the independent noise sources, $v_{n1}(k)$ and $v_{n2}(k)$. In fact, $\mathbf{A}_n$, $\mathbf{G}_n$, and $\mathbf{H}_n$ are unit matrices. This produces a reduced set of Kalman equations. The time-variation, that had been in $\mathbf{H}_n$, is now in the observation vector, $\mathbf{C}_n$. Each element of the observation vector is a periodic signal with amplitude's $a_n(k)$ and $b_n(k)$. This is shown as

$$y_n(k)=\underbrace{\begin{bmatrix}\cos\frac{2\pi f_n}{f_s}k & \sin\frac{2\pi f_n}{f_s}k\end{bmatrix}}_{\mathbf{C}_n^T(k)}\cdot\underbrace{\begin{bmatrix}a_n(k)\\ b_n(k)\end{bmatrix}}_{\underline{x}_n(k)}+w_n(k) \tag{3.16}$$

$n=0,1,2,\ldots,M-1$

In both models, $\mathbf{Q}$ and $\mathbf{R}$ are diagonal matrices.

$$\mathbf{Q}=\begin{pmatrix}q_0\\ \vdots\\ q_{2M-1}\end{pmatrix}\mathbf{I} \;;\; \mathbf{R}=\begin{pmatrix}r_0\\ \vdots\\ r_{2M-1}\end{pmatrix}\mathbf{I} \tag{3.17}$$

$$\mathbf{H}=\begin{bmatrix}H_0 & 0 & .. & 0\\ 0 & H_1 & 0 & ..\\ .. & .. & H_{..} & 0\\ 0 & 0 & 0 & H_{M-1}\end{bmatrix} \qquad \mathbf{A}=\begin{bmatrix}\mathbf{A}_0 & 0 & .. & 0\\ 0 & \mathbf{A}_1 & 0 & ..\\ .. & .. & \mathbf{A}_{..} & 0\\ 0 & 0 & 0 & \mathbf{A}_{M-1}\end{bmatrix}$$

$$\mathbf{C}(k)=[\mathbf{C}_0(k) \quad \mathbf{C}_1(k) \quad \ldots \quad \mathbf{C}_{M-2}(k) \quad \mathbf{C}_{M-1}(k)]$$

M: Number of estimated frequency lines

3.5. ERROR COVARIANCE & KALMAN GAIN

In this section, the statistical behavior of the system noise and the measurement noise on the error covariance, $\mathbf{P}(k)$, and the Kalman Gain , $\mathbf{K}(k)$, is considered. But first, clearly understand the meaning of covariance, we will consider an example that is easy to follow.

Visualize an automobile moving with some velocity, in traffic. The car will accelerate or decelerate, as the driver steps on the gas or presses down on the brake pedal, according to the randomness of the traffic. The variations in velocity (the "signal" in this case) depend on two parameters, the overall system response time (combination of the driver and the car) and the random velocity perturbations, introduced by the changes in traffic speed. When noise processes are assumed to be stationary, both the variance of the car and the variance of the traffic are assumed to be independent of time.

In practice, noise processes are commonly assumed to be stationary, i.e., their statistics do not vary with time — $\mathbf{Q}$ and $\mathbf{R}$ are not related to k. Furthermore, the noise processes are uncorrelated, resulting in diagonal matrices.

The error covariance matrix is found recursively, using Equations 3.5, 3.6, 3.7 and 3.8. In our application the covariance matrix of the system noise, $\mathbf{Q} = E[(\underline{v}(k)\underline{v}'(k)]$, and the variance of the measurement noise, $\mathbf{R} = E[(w(k)w'(k)]$, are stationary.

As shown in Figure 3.3, the error covariance of the Oscillator Signal Model converges on a constant value as the Random Walk Signal Model converges on a constant value that is modulated by a periodic function. The periodic component is explained by the observation vector, $\mathbf{C}(k)$, described by sine and cosine functions with frequency, f_n described by Equation 3.16. Figure 3.3 is a graphical representation for q/r=0.01 of the 500 Hz component (see paragraph 3.6)

The Kalman Gain (KG), $K(k+1)$, expressed by Equation 3.7, becomes stationary as $k \to \infty$. Figure 3.4 is a graphical representation of KG. As k increases, KG grows asymptotically, until a stationary value is found. KG becomes a time-invariant, diagonal matrix when applying the OSM and a stationary, periodic, diagonal matrix concerning the Random Walk Signal Model.

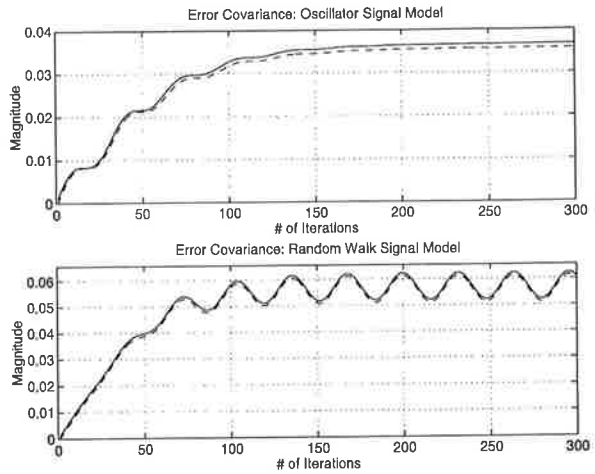


Figure 3.3. *Error covariance for stationary noise processes. The dotted line is the a priori error covariance; the solid line is the a posteriori error covariance.*

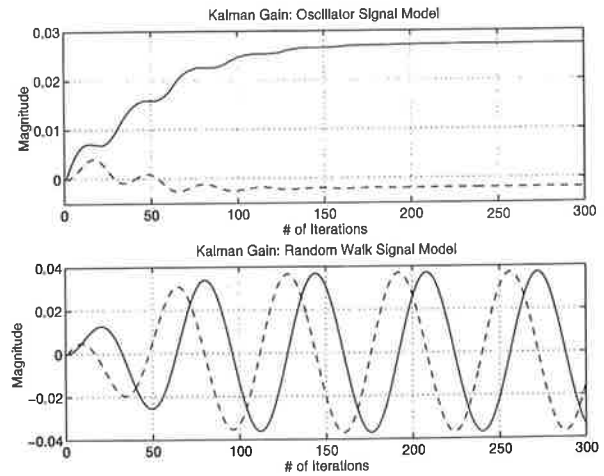


Figure 3.4. *Kalman Gain for stationary noise processes. The dotted line is the gain for x_{n1}; the solid line is the gain for x_{n2} .*

3.6. FREQUENCY DOMAIN

In this segment, the frequency behaviour is studied in detail. To compare the Kalman technique to the Fourier approach, the frequencies are spaced equidistant. The frequency response of a Kalman estimator, configured to estimate the first 21 spectral components, is shown in Figure 3.5.

An example of the set-up for the time-frequency analysis we will now discuss, is:

- sampling frequency: 3.2 kHz
- estimated frequencies: 0, 50, 100, ... 1000 Hz (21 freq.)
- Kalman filter order: 42
- design parameters q/r: 0.1, 0.01, 0.001, 0.0001, 0.00001

The side-lobe suppression is tunable, as a function of the design parameter, q / r, and does not depend on the filter order. Figure 3.5 is a plot of the frequency response of the Oscillator Signal Model at 0 Hz, 250 Hz, 500 Hz, and 750 Hz. For each plot, the design parameter decreases from 0.1 to 0.00001, and with each reduction the Kalman filter suppresses more of the side-lobe effect. This is true for any type of signal model, but the amount of suppression differs for each model.

The frequency responses produced by the Kalman filter can now be compared with the responses associated with classical Discrete Fourier Transform. A 64-point DFT with a rectangular window has the same frequency curve as the Kalman filter at design parameter 0.1.

Hence, the Kalman approach offers superior tunable side-lobe suppression. Furthermore, the tunable shaping of the curve is more effective than the windowing functions that can be applied with the DFT (refer to Figure 2.1).

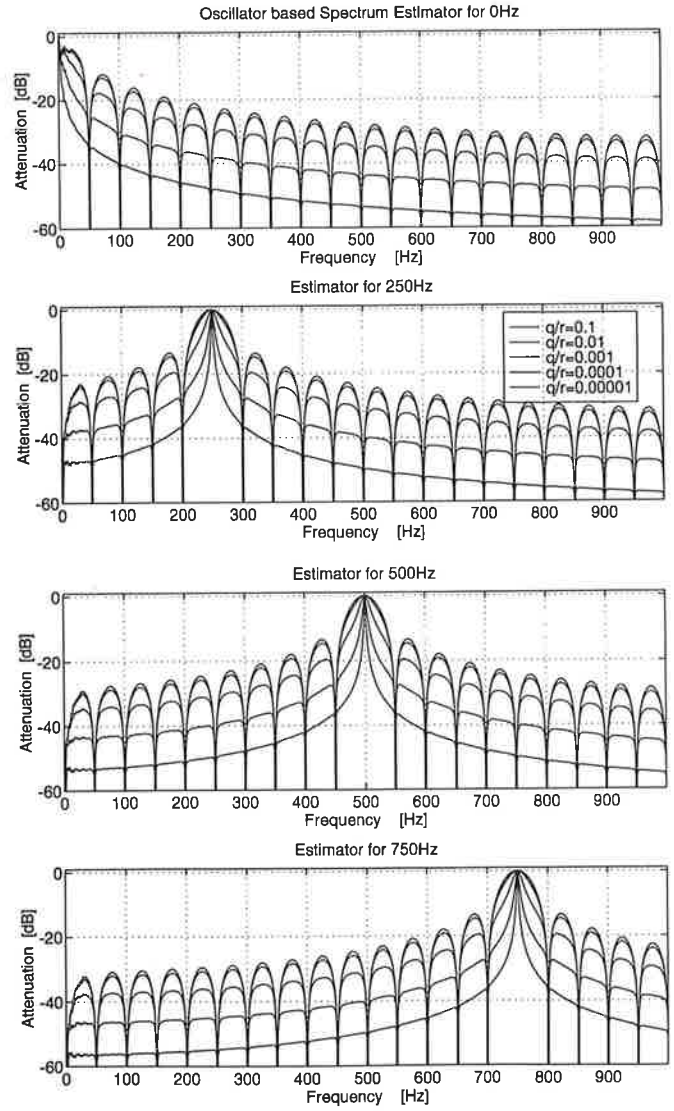


Figure 3.5. *Frequency response of the Oscillator Signal Model. Design parameters: 0.1, 0.01, 0.001, 0.0001, and 0.00001.*

Random Walk Signal Model versus Oscillator Signal Model

Changes in the design parameter dramatically affect the curvature, of the main-lobe especially. As q / r becomes smaller, the main-lobe becomes thinner and narrower, focusing in on the considered frequency. This can be seen in Figure 3.6. The Oscillator Signal Model requires a larger q / r value than the Random Walk Signal Model.

For a given set of covariances, the Oscillator Signal Model attains better side-lobe suppression than the Random Walk Model. Furthermore, this difference increases as the design parameter becomes smaller. In Figure 3.6, the side-lobe gain, when the design parameter is 0.00001, is approximately 5 dB greater with the oscillator. When the time domain is examined, it will become apparent that this relationship is inverted.

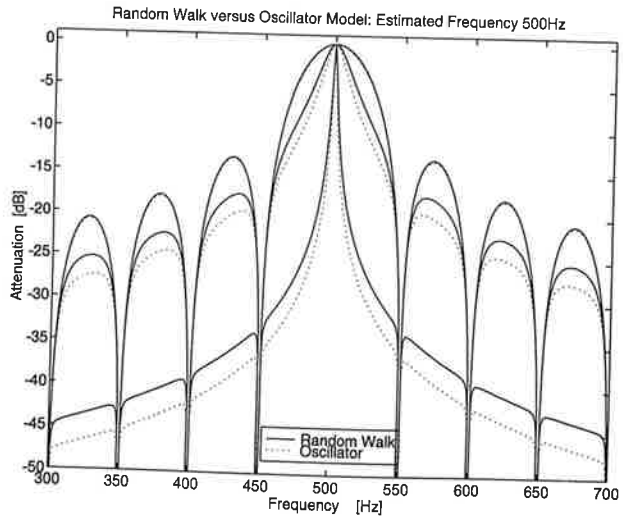


Figure 3.6. *Frequency response: Oscillator Signal Model versus Random Walk Signal Model. Estimated frequency is 500Hz; q/r= 0.1, 0.001, and 0.00001.*

Roll-off Rate

The *roll-off rate* is the slope of the decrease in the side-lobes. The roll-off rate is independent of the design parameter, q / r. To compare the roll-off rates of the Kalman and Fourier approaches, the frequency response is drawn in a logarithmic scale for the 0 Hz (DC) component. As with the Fourier-based approach, which uses the rectangular and Hamming windows (see Table 2.4), the roll-off rate with the Kalman approach is -20 dB/decade.

The peak side-lobe level is tunable, ranging from -16 dB for q/r=0.1 to -37 dB for q/r=0.00001. Side-lobe suppression can be further improved as q / r continues to decrease. But, the Kalman Gain coefficients will become very small, possibly complicating finite arithmetic problems for fixed point implementations.

At design parameter 0.1, the Kalman approach corresponds to the 64-point DFT rectangular window; and 0.00001 coincides with the 128-point DFT Hamming window. The 128-point DFT Hanning window achieves a better roll-off rate of -60 dB/dec, but its peak side-lobe level reaches only -31 dB. However, the Random Walk Signal Model has a tunable side-lobe maximum, that is -37 dB for q/r=0.00001, and can be improved even further if the design parameter is reduced. The windowed DFT approach, with its fixed characteristics, cannot be enhanced through tuning.

3.7. TIME DOMAIN

The foregoing discussion illustrates the parametrization of the design parameter by adapting the covariances of the noise sources to the Kalman approach. As q / r decreases the convergence (time it takes to determine the correct results) increases. The number of iterations necessary to estimate the convergence on the magnitude of the sinusoid is called the *tracking time*. This is analogous to the time needed to estimate the real part and the imaginary part of the Fourier coefficients.

Figure 3.8 shows the tracking of both the RWSM and the OSM for a sinusoid with amplitude 1, and frequency 500 Hz. Three different design parameters are exemplified.

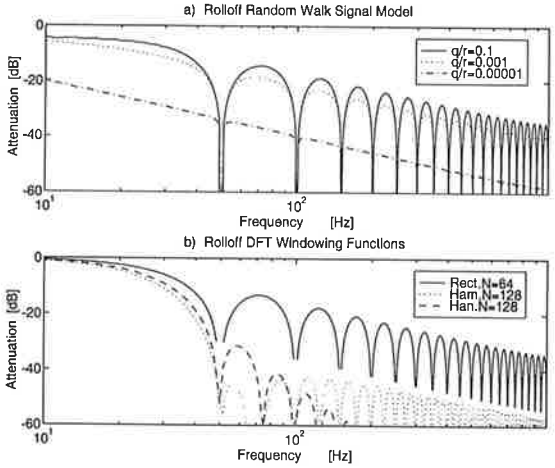


Figure 3.7. *Roll-off rate: (a) Random Walk Signal Model. (b) windowing functions.*

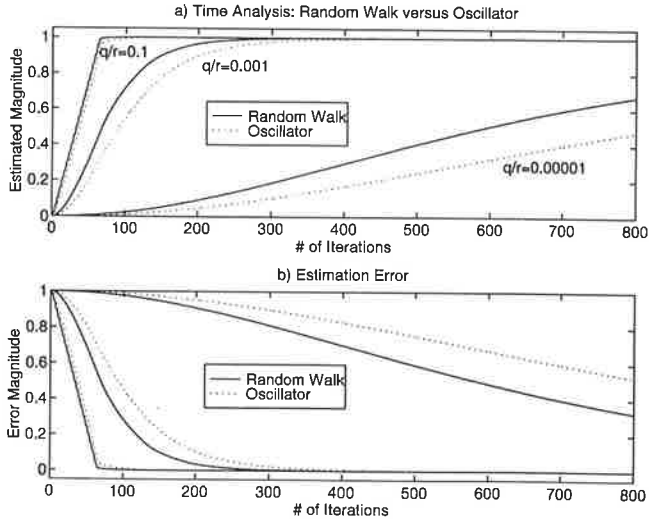


Figure 3.8. *Tracking of a sinusoidal signal: Oscillator Signal Model versus Random Walk Signal Model (frequency = 500 Hz).*

Table 3.1. *Number of Iterations Required to Track a Sinusoid (estimation error fixed at 0.01)*

[# of Iter.]	q/r=1E-1	q/r=1E-2	q/r=1E-3	q/r=1E-4	q/r=1E-5
RWSM	74	125	260	760	2380
OSM	105	150	355	1070	3360
speed factor	1.41	1.2	1.36	1.41	1.41

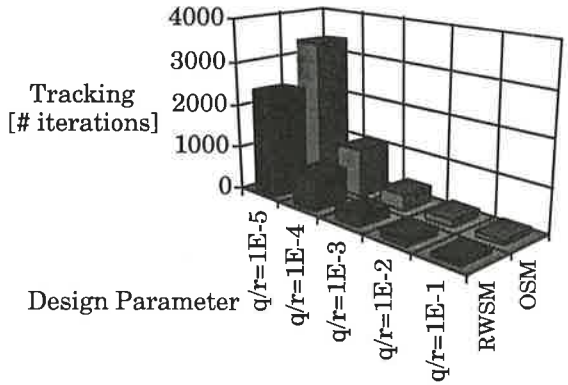


Figure 3.9. *Graphical representation of Table 3.1.*

For a given set of design parameters, the Random Walk Signal Model converges to correct results more quickly than the Oscillator Signal Model. Table 3.1 and Figure 3.9 summarize the number of iterations required to track a sinusoid, at fixed estimation error 0.01. One possible interpretation of the faster tracking capability can be found in the second noise source, introduced inside the Random Walk Signal Model. That second noise source introduces a factor of $\sqrt{2}$ compared to the Oscillator Model. In Table 3.1 the faster tracking is shown by the speed factor.

3.8. ESTIMATION IN NOISE

In order to compare the Dynamic Signal Model to the Static Signal Model (see Figure 1.2), a detailed study of sinusoidal signals

Table 3.2. *Mean-Square Estimation Error (EE_{ms}) as a Function of the Signal-to-Noise Ratio.*

Noise [EE_{ms}]	-105dB	-85 dB	-65 dB	-45 dB	-25 dB	-5 dB
RWSM						
q/r=1e-1	8.1e-7	1.1e-5	7.3e-5	1.1e-3	1.0e-2	8.1e-2
q/r=1e-3	6.1e-7	8.6e-6	5.6e-5	8.6e-4	7.9e-3	5.9e-2
q/r=1e-5	4.1e-7	3.0e-6	4.7e-5	3.3e-4	1.4e-3	1.8e-2
OSM						
q/r=1e-1	7.9e-7	1.0e-5	7.2e-5	1.1e-3	1.0e-2	7.9e-2
q/r=1e-3	5.2e-7	7.8e-6	5.1e-5	7.7e-4	6.7e-3	5.1e-2
q/r=1e-5	3.8e-7	3.3e-6	5.0e-5	2.8e-4	1.2e-3	**1.4e-2**
64-pt. short-time DFT rectangular window	9.0e-7	1.0e-5	7.9e-5	9.4e-4	9.0e-3	1.0e-1
128-point short-time DFT Hamming wind.	8.5e-7	8.3e-6	7.8e-5	8.0e-4	7.9e-3	7.4e-2
192-point short-time DFT Blackman wind.	7.2e-7	7.6e-6	7.4e-5	7.7e-4	6.8e-3	**6.9e-2**

corrupted by noise has been conducted. The research is based on a sinusoidal signal at 500 Hz, corrupted by an additional, random, zero-mean, white noise source. The noise level varies between -105 dB and -5 dB, with increments of 20 dB. Table 3.2 is a summary of the simulation results obtained with the MATLAB simulation tool, run on a SUN SPARC 10 computer. An equal number of frequency lines has been chosen to allow for comparisons between the Kalman approach and short-time Discrete Fourier Transform. The sampling frequency is the same as before, 3.2 kHz; the frequency lines are placed along the frequency axis between 0 kHz and 1.6 kHz (Δf= 50 Hz); and the main-lobe width is fixed at $\pm\Delta f$= 50 Hz.

Three different windowing functions are used for the short-time DFT: The 64-point rectangular window, 128-point Hamming window, and 192-point Blackman window. The window sizes that have been selected all have the same main-lobe width.

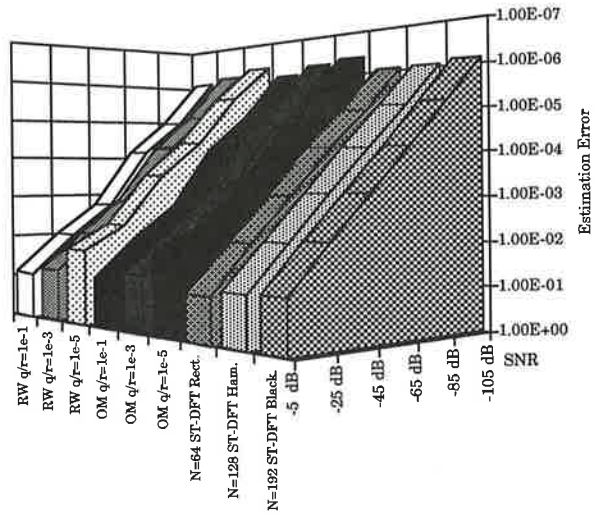


Figure 3.10. *Graphical representation of Table 3.2: Kalman approach versus short-time DFT.*

Identical input signal are applied to the Kalman estimator and the windowed short-time DFT.

The design parameter selected allows for a comparison to be made between these results and those obtained in the time and frequency domains.

The EE_{ms} for the short-time DFT is computed for 48 overlapped frames (50% overlap) while the Kalman EE_{ms} is computed for 256 stationary samples.

The Blackman window has the lowest estimation error and is thus the preferred short-time DFT window. In Figure 2.1, we saw that its suppression of the side-lobe's is greatest and its main-lobe curve is the narrowest. There is only a minimal difference between the estimation errors for the Hamming and rectangular windows.

Of the Kalman approaches, the Oscillator Signal Model is only

somewhat better than the Random Walk Model. Nevertheless, both are superior to the short-time DFT windows. The tuning of the noise source covariances, simultaneously altering the design parameter, effects an increase in noise smoothing. This leads to better estimation results.

For example, when q/r = 0.00001 and the noise is at -5 dB (strongly noisy measurements), the Oscillator Signal Model produces an estimation error of 0.014. The Blackman window achieves an error of only 0.069 under the same conditions. Thus, the exceptional performance of the Kalman approach, in this case almost five times more effective in noise, is highlighted.

3.9. SUMMARY

This chapter focused on the design of Kalman filters and spectrum estimation of sinusoidal signals corrupted by noise. After introducing the stochastic approach and its design, two signal models were applied to the Kalman filter. Both were analyzed in the time and frequency domains. Then, a detailed study of windowing functions was undertaken to prepare for a comparison of the stochastic approach to the well-known Discrete Fourier Transform.

An important characteristic of the classical Fourier approach is that the window length, N, and the applied window function, rectangular, Hamming, Hanning, Blackman, or others, completely determine the time and frequency behavior. The goal of the window is to minimize both the peak side-lobe level and the roll-off rate.

The main-lobe width is a function of the window length, N. The number of samples considered in the window frame determines the number of equidistant frequency lines that will estimated.

The estimation error as function of the noise level was then explored. So that results could be compared, the main-lobe width was fixed for the three windowing functions. Results show that the Blackman window achieves the best estimate, surpassing the alternative rectangular and Hamming windows. A 30% gain in estimation error is attained by the -60 dB/dec roll-off rate of the

Blackman window.

The classical DFT gives correct results only after the computation of a complete window length (block data processing). The tracking parameter, in this case, is the window-length, N. As more samples are taken into account, the tracking time will be increased. The Blackman window requires $3N$ samples for a fixed main-lobe width; the Hamming window, $2N$; and the rectangular window, N. The relationship between tracking time, t_{track} , and window-length, N_w, where τ_s is the sample period, is

$$t_{track} = N_w \tau_s \tag{3.18}$$

Even when only a few frequencies will be estimated, an application could require a high N to reach the desired resolution. Besides, when FFT is implemented, N has to be in the set, 2^z, for all integers, z; otherwise, a DFT algorithm will have to be used.

The interesting characteristics of the Kalman estimator are the tracking speed and the frequency response. They depend on the covariance matrices, $\mathbf{Q}$ and $\mathbf{R}$, or, more specifically, on the design parameter. The Kalman estimator is tracking on the true values more slowly than DFT windowing, though its frequency response is completely tunable. Additionally, there are no constraints on the range of frequencies or the distances separating them. At each estimated frequency, a design parameter, q/r, shapes the selectivity of the main-lobe and the amount of noise rejection (roll-off rate of the side-lobe): A small design parameter increases selectivity and side-lobe rejection, but with a slower tracking speed — this necessitates a stationary signal. The selectivity of the frequency response is also determined by the relative position of its neighboring frequencies. Thus, increased selectivity can be achieved, even for higher values of q / r.

REFERENCES

[Bitm86] R.R. Bitmead , A.H. Tsoi, P.J.Parker : "A Kalman Filter Approach to Short-Time Fourier Analysis", IEEE Trans. on Accoustics, Speech, and Signal Processing, Vol.ASSP-34, No.6, December 1986, pp. 1493-1501.

[Hayk91] S. Haykin : "Adaptive filter theory", Prentice-Hall Int. Editions, Englewood Cliffs, 1991, ISBN 0-13-005513-1.

[Jack86] L.B. Jackson : "Digital Filters and Signal Processing", Kluwer Academic Publishers, Boston, 1986.

[Kalm61] R.E Kalman, R.S. Bucy : "New Results in Linear Filtering and Prediction Theory", IEEE Trans. on Acc. Communications, Vol.COM-26, No.10, October 1983, pp. 1439-1446.

[Kalm61] R.E. Kalman, R.S. Bucy : "New Results in Linear Filtering and Prediction Theory", Trans. on the ASME, Journal of Basic Engineering, March 1961, pp. 95-108.

[Kalm60] R.E. Kalman : "A New Approach to Linear Filtering and Prediction Problems", Trans. on the ASME, Journal of Basic Engineering, March 1960, pp. 35-45.

[Kunt91] M. Kunt et al : "Techniques modernes de traitement numérique des signaux", Presse Polytechnique et Universitaires Romandes, Lausanne, 1991.

[Open75] A .V. Oppenhaim, R.W. Schafer : "Digital Signal Processing", Prentice-Hall, Englewood Cliffs, New Jersy, 1975.

[Open83] A .V. Oppenhaim, A.S. Willsky, I. T. Young : "Signals and Systems", Prentice-Hall, Englewood Cliffs, New Jersy, 1983.

[Schw75] M. Schwartz, L. Shaw : " Signal Processing: Discrete Spectral Analysis, Detection, and Estimation", Mac Graw-Hill, New York, 1975.

4. Steady-State Kalman Estimator

In Chapter 3, the statistical behavior of the noise sources was not limited in any way. The technique that is presented in this chapter assumes that the statistic is time-invariant. This condition leads to a steady-state Kalman filter which is computationally more efficient.

The steady-state Kalman Gain is extracted and remarkably simple signal flow graphs are achieved. The reduced number of operations is measured against classical FFT algorithms.

The steady-state Kalman algorithms are studied for finite arithmetic implementation. The optimized signal flow graphs will be used for real-time implementations in Chapters 5 and 6.

4.1. INTRODUCTION

It was shown in Chapter 2 that a dynamic system is driven by noise sources. Scrutinizing Equations 3.5-3.8, it is clear that the error covariance matrices and the Kalman Gain do not depend on the input signal, $y(k)$.

Referring once again to the traffic example from section 3.5, each type of traffic condition, from a Formula 1 auto race to city circulation, has its own time-invariant covariance, determined by the randomness of the situation. Similarly, the sinusoidal signals that are estimated in many applications exhibit this stationary random process.

Dynamic systems considered in this work have a time invariant stochastic nature. Consequently, this is expressed by a constant covariance of the system noise source $\underline{v}(k)$ and a constant variance of the measurement noise source $w(k)$ leading to time invariant $\mathbf{Q}$ and $\mathbf{R}$ matrices.

The stability condition is that the state-transition matrix, $\mathbf{A}$, is asymptotically stable. Normally, the error covariance, $\mathbf{P}(k+1)$, is time-varying. However, under this assumption, the Lyapunov Equation has a steady-state solution for large k. This steady-state covariance satisfies the Lyapunov Equation

$$\mathbf{P} = \mathbf{APA}^T + \mathbf{GRG}^T \tag{4.1}$$

As k increases, the state, $x(k)$, becomes a stationary Markov process. This is called the *statistical steady state*. When the stability condition does not hold, the state-transition equation is unstable and therefore the covariance, $\mathbf{P}(k+1)$, increases without bound.

4.2. STEADY-STATE KALMAN FORMULATION

The Kalman filter has a transient part and a steady-state part. The transient part is the number of iterations needed to track the results. When the correct estimates have been found, the steady-state part has been achieved. The steady-state Kalman estimator offers a set of steady-state Kalman Gain (steady-state KG) coefficients, $K_{n1}(k)$ and $K_{n2}(k)$, for each estimated frequency line, with index, n.

Under the assumption of stationary random process, the error covariances tend to stationary functions. Consequently, the error covariance matrices and the Kalman Gain can be computed off-line (in advance). In particular, the matrix multiplications and the matrix inversion can be transferred from real-time to off-line calculation. Thus, the real-time computational complexity of the steady-state Kalman estimator is exorbitantly reduced.

The a priori and a posteriori error covariance equations and the Kalman Gain equation can be computed in advance. All three equations are independent of the input data. The number of

equations needed to determine the estimates has been reduced from five to two. The remaining set of equations for the steady-state Kalman estimator are:

Time-update (affect on system dynamics)

- Predicted estimate: $\underline{\hat{x}}^-(k+1) = \mathbf{A}(k)\underline{\hat{x}}(k)$ (4.2)

Measurement-update (effect of new input data, $y(k)$)

- Corrected estimate:

$$\underline{\hat{x}}(k+1) = \underline{\hat{x}}^-(k+1) + \mathbf{K}(k+1)\left[y(k+1) - \mathbf{C}(k+1)\underline{\hat{x}}^-(k+1)\right] \tag{4.3}$$

Extraction of the Steady-State Kalman Gain

Using the Oscillator Signal Model, the steady-state KG coefficients are constant; using the Random Walk Signal Model, the coefficients consist of a linear combination of functions,

$$\cos\frac{2\pi f_n}{f_s}k, \quad \sin\frac{2\pi f_n}{f_s}k \tag{4.4}$$

Depending on which signal model is being considered, the extraction of the steady-state KG differs. With the OSM, the extraction is straight-forward because the KG grows asymptotically until the steady-state is achieved.

With the RWSM, there exist two methods for extracting the steady-state KG because the steady-state KG is a set of sine and cosine functions:

1. The steady-state KG can be rebuilt from at least one-fourth of a period, stored in a look-up table. Bear in mind, the phase and amplitude of each estimator must be carefully matched.
2. Decompose the steady-state KG into the sum of two harmonic signals (Equation 4.6). One can see that this

formula is simply the Fourier series of the corresponding signal

$$\Phi = 2\pi \frac{f}{f_s} \quad ; \; f_s = \textit{sampling frequency} \tag{4.5}$$

$$a\sin(\Phi k + \varphi_1) + b\cos(\Phi k + \varphi_2) = \sqrt{c^2 + d^2}\,\sin(\Phi k + \varphi) \tag{4.6}$$

$$c = a\sin(\varphi_1) + b\cos(\varphi_2) \tag{4.7}$$

$$d = a\cos(\varphi_1) - b\sin(\varphi_2) \tag{4.8}$$

$$\varphi = arctg\frac{c}{d} \quad and \quad \varphi = \arcsin\frac{c}{\sqrt{c^2 + d^2}} \tag{4.9}$$

To identify the amplitude of the steady-state KG, φ_1 and φ_2 are assumed to be zero, and β and α are substituted for a and b, respectively

$$\alpha\cos(\Phi k) + \beta\sin(\Phi k) = \sqrt{c^2 + d^2}\,\sin(\Phi k + \varphi) \tag{4.10}$$

$$c = b = \alpha \tag{4.11}$$

$$d = a = \beta \tag{4.12}$$

$$\varphi = arctg\frac{\alpha}{\beta} \quad and \quad \varphi = \arcsin\frac{\alpha}{\sqrt{\alpha^2 + \beta^2}} \tag{4.13}$$

For real-time implementations, the second option is best because only two coefficients are stored in the look-up table for each estimated frequency.

The overall steady-state KGs, for the RWSM and for the OSM, are tabulated in the appendix at the end of the chapter. A careful examination of the results reveals that the magnitude of the Kalman Gain decreases for smaller design parameters.

4.3. STEADY-STATE OSCILLATOR SIGNAL MODEL

The signal flow graph of the steady-state Oscillator Signal Model is shown in Figure 4.1. The set of equations that describe the signal flow graph is

$$error(k) = y(k) - \underbrace{\sum_{i=0}^{M-1} w_{1i}(k)}_{\hat{y}^{-}(k)} \tag{4.14}$$

$$w1i(k+1) = error(k)\cdot K1i + w1i(k)\cdot\cos(\Omega i) - w2i(k)\cdot\sin(\Omega i) \tag{4.15}$$

$$w2i(k+1) = error(k)\cdot K2i + w1i(k)\cdot\sin(\Omega i) + w2i(k)\cdot\cos(\Omega i) \tag{4.16}$$

$$\hat{a}i(k) = w1i(k+1)\cdot\cos(k\Omega i) + w2i(k+1)\cdot\sin(k\Omega i) \tag{4.17}$$

$$\hat{b}i(k) = w1i(k+1)\cdot\sin(k\Omega i) - w2i(k+1)\cdot\cos(k\Omega i) \tag{4.18}$$

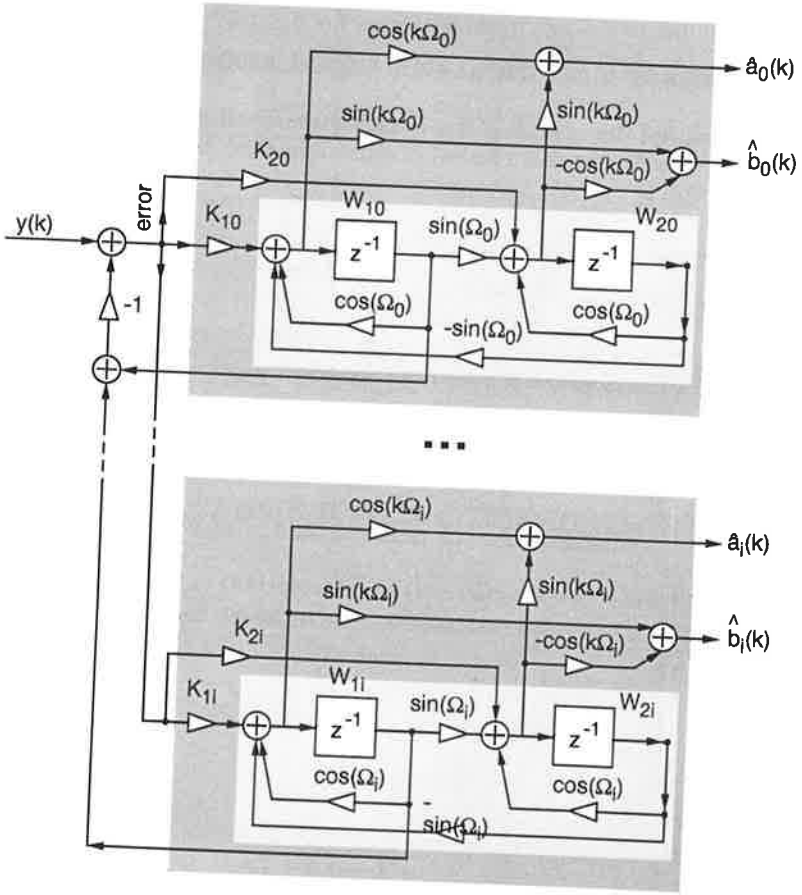


Figure 4.1. *Signal flow graph of the steady-state Oscillator Signal Model (estimated sinusoid: $\Omega_i = 2\pi f_i / f_s$).*

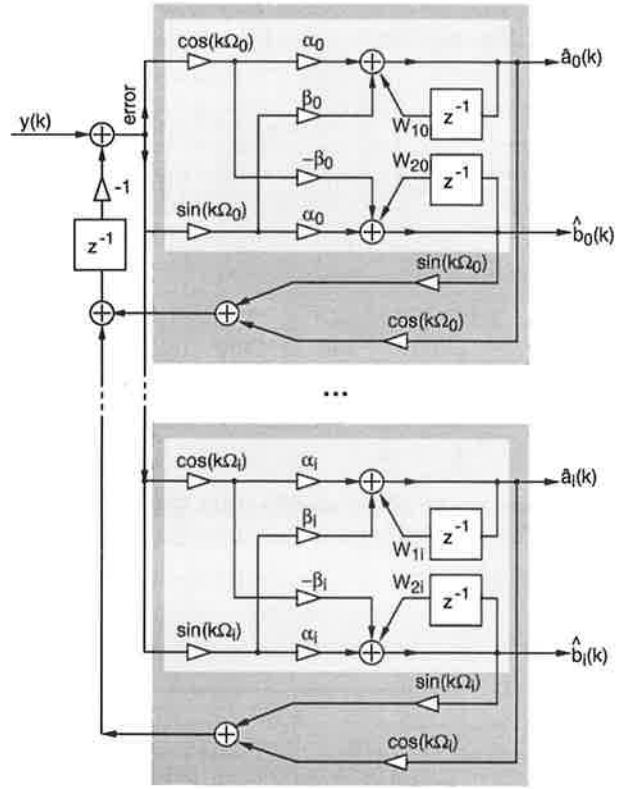


Figure 4.2. *Signal flow graph of the steady-state Random Walk Signal Model.*

4.4. STEADY-STATE RANDOM WALK SIGNAL MODEL

For each estimated frequency, a periodic set of steady-state Kalman Gains, $K_{1i}(k)$ and $K_{2i}(k)$, is defined. A linear combination of two functions, $\cos(k2\pi f_i / f_s)$ and $\sin(k2\pi f_i / f_s)$, with amplitudes α_i and β_i, respectively, form the KGs. By applying Equations 4.10-4.13, the KGs can be calculated.

Table 4.1. *Number of Iterations Required to Track a Sinusoid (estimation error is fixed at 0.01).*

	Number of Iterations				
Design Parameter	0.1	0.01	0.001	0.0001	0.00001
RWSM	53	118	257	757	2369
Steady-State RWSM	44	105	226	660	2087
OSM	95	142	353	1067	3351
Steady-State OSM	78	126	310	929	2913

The signal flow graph of the steady-state RWSM is pictured in Figure 4.2. The set of equations that describe the signal flow graph is

$$error(k) = y(k) - \underbrace{\sum_{i=0}^{M-1} w_{1i}(k)\cdot\cos(k\Omega_i) + w_{2i}(k)\cdot\sin(k\Omega_i)}_{\hat{y}^{-}(k)} \tag{4.19}$$

$$\hat{a}_i(k) = w_{1i}(k+1) = error(k)\cdot\cos(k\Omega_i)\alpha_i + error(k)\cdot\sin(k\Omega_i)\beta_i + w_{1i}(k) \tag{4.20}$$

$$\hat{b}_i(k) = w_{2i}(k+1) = error(k)\cdot\sin(k\Omega_i)\alpha_i - error(k)\cdot\cos(k\Omega_i)\beta_i + w_{2i}(k) \tag{4.21}$$

The frequency characteristic of each steady-state Kalman model is comparable that of its corresponding on-line algorithm. However, the time characteristic is not analogous. Due to the pre-calculated KG coefficients, the algorithms have faster tracking speeds.

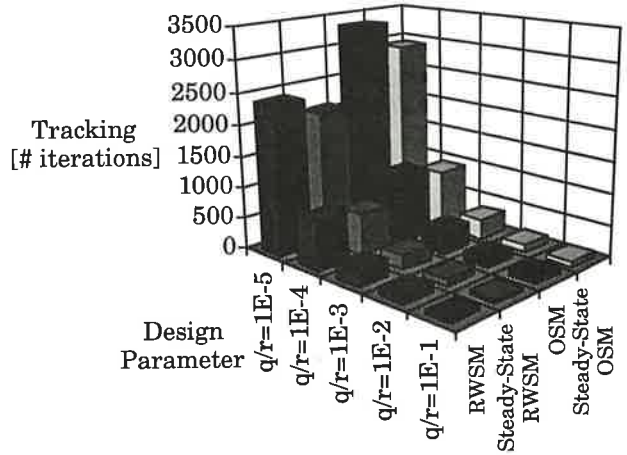


Figure 4.3. *Graphical representation of Table 4.1.*

4.5. COMPUTATIONAL PERFORMANCE

There are a number of important points to remember when real-time algorithms are involved. For DSP and ASIC implementations, the computational requirement of each steady-state Kalman estimator is extracted. Using Equations 4.14-4.21, the number of variables, coefficients, and arithmetic operations necessary to execute the estimators can be enumerated. The results are summed in Table 4.2 (M refers to the number of spectral components).

The most important advantage, which distinguishes the Kalman models, is that the computational requirements are *linearly* related to M. For all DFT algorithms, the number of arithmetic operations is *exponentially* related to M.

Table 4.2. *Arithmetic Operations and Memory Requirements for Variables and Coefficients.*

	Oscillator Signal Model	*Random Walk Signal Model*
# operations	10 Mult + 7 Add = 17 M	8 Mult + 6 Add=14 M
# variables	$2M+1$	$2M+1$
# coefficients	$4M$ sin, cos table	$2M$ sin, cos table

Because computation with the steady-state RWSM is 11% lower than with the OSM, it is recommended for high-performance applications (those with many frequency lines and a high sampling frequency).

The Kalman approach is an iterative calculation method, while most Fast Fourier Transforms are block data methods. Because no input data block has to be stored, the steady-state Kalman estimator features low memory occupation. Both models require sine and cosine tables and thus memory space must be reserved for them. However, this drawback is often negated because most general purpose Digital Signal Processors contain the tables on-chip, in a dedicated ROM. If the target implementation is ASIC, the tables are not available (memory space must be provided).

In Table 4.3 computational performance of both steady-state Kalman estimators and two very efficient Fast Fourier Transform algorithms, Radix 2 and split-radix FFT, is tabulated. Remember, to obtain a certain number of lines, M, a $N=2M$-point FFT is required. The frequency lines are spaced equidistant on the frequency axis, ranging from 0 to $f_s/2$. More detailed information about FFT algorithms can be found in [Kunt91].

A computational procedure that somewhat more efficient than the direct method or the FFT algorithms is called the Goertzel algorithm [Oppe75]. It is a method commonly used when only a small number of frequency lines is desired. The computational performances are comparable to the OSM but can be seen still as a block oriented computation method. Results are available only after the complete

Table 4.3. *Number of Additions and Multiplications as a Function of the Number of Estimated Frequency Lines.*

# estimated lines ($N/2$ for FFTs)	16	32	64	128	256	512	1024
Steady-St. RWSM	112	224	448	896	1792	3584	7168
Steady-St. OSM	136	272	544	1088	2176	4352	8704
Radix 2 FFT *	496	1296	3216	7696	17926	40976	92176
Split-Radix FFT *	456	1160	2824	6664	15368	34824	77832

* *results from [Kunt91]*

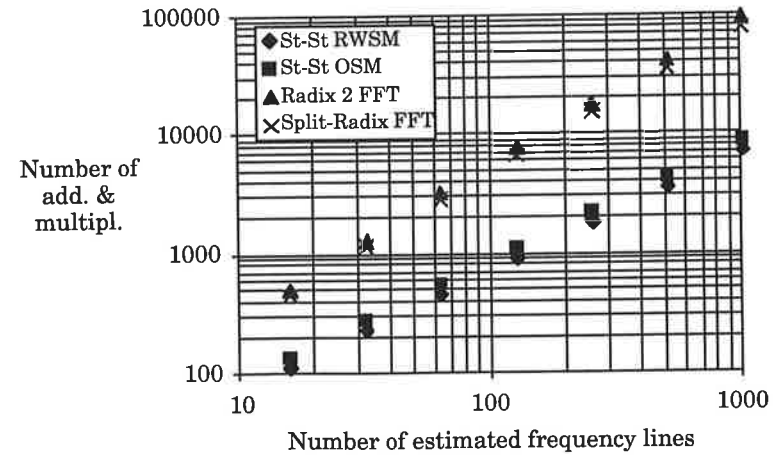


Figure 4.4.(a) *Graphical representation of Table 4.3. (a) FFT algorithms versus steady-state Kalman estimaters. Logarithmic scale.*

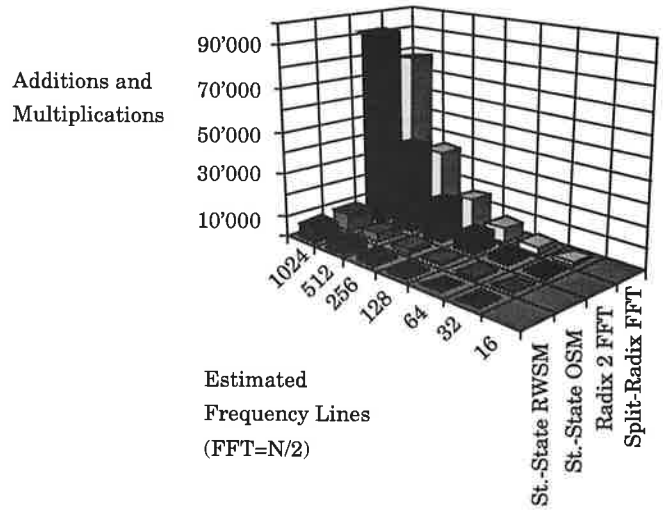


Figure 4.4.(b) *Graphical representation of Table 4.3. (b) FFT algorithms versus steady-state Kalman estimators. Linear scale.*

computation of a data block, N. The manner how the results are computed is done in a recursive way. However, because this formulation is recursive, there are difficulties due to quantization effects. In particular, the variance of the computational noise grows linearly with time. Thus, the Goertzel algorithm must be reseted after a certain time. Thus, on-line running is not affordable. This draw-back leads us to not consider the Goertzel algorithm in this work. Detailed studies about Goertzel algorithm can be found in [Cove91].

The tracking time for the steady-state Kalman estimator corresponds to the length of the window used in classical short-time Fourier transform. There exists a direct positive relationship between the number of arithmetic operations gained and the number of frequency lines estimated, comparing the steady-state RWSM to the split-radix FFT.

The gain is 4 for 16 estimated frequency lines, increasing to 10.8 for 1024 estimated frequencies (see Figure 4.4). The gain variation is a result of the linear dependence of the Kalman approach and of

the logarithmic dependence of the FFT algorithms [Kunt91, Jack86].

4.6. FINITE WORDLENGTH ANALYSIS

For any application, the goal is to find a solution that can be realized in fixed-point arithmetic units. In relation to a floating-point, the chip area is smaller and the execution speed is faster. This thesis focuses on fixed-point silicon (e.g. ASIC) and fixed-point DSP software implementations. In most cases, an ASIC implementation on silicon is an optimized solution. To minimize the finite wordlength effects, and thus increase arithmetic precision, many bits are needed. However, increased silicon area is very expensive. Therefore, a trade-off must be made between silicon area and finite wordlength effects.

When working with a finite number of bits, overflow and/or quantization occurs. To simulate the finite wordlength effects, the high level language that is utilized must offer the possibility to apply various overflow and quantization characteristics. The DSP specification languages that have been developed over the past ten years are: Assembly language provided with the preferred DSP processor; hardware description languages, VHDL and Verilog; and high-level languages, Pascal, C, and ADA. Unfortunately, all of these *general purpose languages* have specific problems and cannot offer a solution to all of the major requirements imposed by the nature of DSP algorithms. An alternative would be to use one of the hybrid languages that some hardware and software vendors have been developing. But, none of these provides a complete solution, either.

Data Flow Language (DFL) covers the most important requirements. It not only offers a range of powerful constructs for DSP algorithm specification, but furthermore, it is a front end for hardware synthesis and assembly code generation for both fixed-point and floating-point processing.

4.6.1. Introduction to Data Flow Language

Data Flow Language is used as the behavior specification language for DSP Station™. It is an extension of the *silage* language, version 2.0, that was developed in 1984, at the University of California at Berkeley, by P. Hilfinger, as the DSP specification language for silicon compilation tools. Silage has been used in advanced research on silicon compilation at Berkeley (Philips Research Laboratories), Eindhoven (the Netherlands), and at IMEC (Leuven, Belgium). Research has extended the DFL to a point where it has become a powerful tool for expressing data flow in DSP algorithms.

It has become popular to graph DSP algorithms, forming a signal flow graph. This is exemplified in Figure 4.5.

Nodes represent instances of primitive functions (such as addition, multiplication, and delay) and lines connecting the nodes represent the path followed by data.

The language that is chosen must be able to describe the data path, and not the execution order of the operations. In the latter case, control flow semantics are implied, that is, an imperative sequence of operations performed on memory locations. An advantage of the data flow representation is that no explicit statements are made about the order or concurrence of the operations.

A common textual specification of data flow semantics is an *applicative language:* A language whose fundamental operations are primitive functions, and which has no variables or assignment operator. DFL is one such language, with signals as its basic data objects. A DFL program consists of a set of signal definitions, not individual signal assignments.

A DFL program, derived from the data flow graph in Figure 4.5, is given.

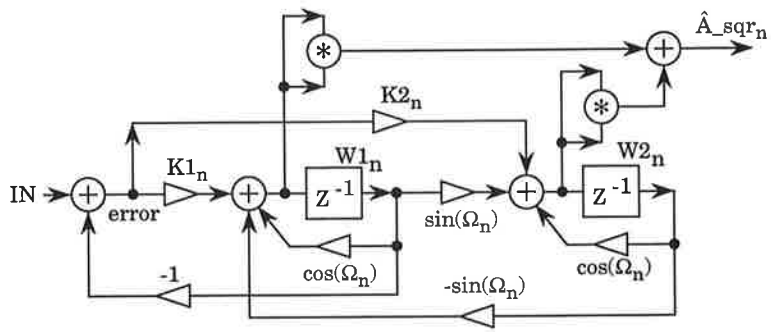


Figure 4.5. *Signal flow graph of the steady-state Oscillator Signal Model.*

Corresponding DFL program:

```
/* signal & coefficient wordlength definition*/
#define IW      fix<20,18>
#define CW      fix<20,18>
#define AW      fix<20,18>
#define OW      fix<20,18>
/* coefficients*/
#define    cos(Ωn)  CW( 0.99518472667220 )
#define    sin(Ωn)  CW( 0.09801714032956 )
#define    K1n      CW( 0.14080423564252 )
#define    K2n      CW( -0.03365454047474 )
/* main DFL function */
func kalman_osc ( IN:IW ) Â_sqrn: OW =
begin
  error = AW( IN - sum_w1@1 );
  w1n = AW( error * K1n - sin(Ωn) * w2n@1 + cos(Ωn) * w1n@1 );
  w2n = AW( error * K2n + cos(Ωn) * w2n@1 + sin(Ωn) * w1n@1 );
  Â_sqrn = OW( w1n * w1n + w2n * w2n );
end;
```

Each signal is an ordered sequence of values sampled in the time domain. Delayed signals are denoted by @k, where k indicates the number of time frames that the signal is delayed. Normally, each value is taken on a regular-time basis. A DFL algorithm describes a

synchronous data flow system. A synchronous data flow system imposes the additional constraint that all data arrive at the inputs at the same time. The arrival frequency of the data is called the rate. Thus, all data arrive at the same rate in a synchronous data flow. Thereafter, a compiler can determine the order of the operations in the program. The set of values, derived from different input signals that arrive at the same moment, is called a frame. For each execution of the DFL algorithm, a new frame is defined. The concept of synchronous data flow can be easily extended to include multirate signals and irregular rate signals. DFL generally accepts many other types of casts — "float", "int," and "unsigned" are examples. For more information on these and other casts, refer to [DFL92].

The keyword, "*fix*<*w*,*d*>," optionally followed by the length parameters for the bit representation, is between angle brackets. "*fix*<13,9>" is shown in Figure 4.6. The length parameters indicate a wordlength, *w*, which corresponds to the total number of binary digits, and a fractional length, *d*, for the binary digits to the right of the binary point (fractional bits). "*fix*<*w*,*d*>" can be used to represent values of the set $2^{-d}n$, where $-2^{w-1} < n < 2^{w-1}$.

Overflow Characteristics

Overflow, the effects on the Most Significant Bits (MSBs) of a signal, is shown in Figure 4.7. The largest acceptable input signal is the largest possible input signal that does not cause overflow inside the algorithm. To find this input signal, a norm calculation must be performed. A *norm* is a well-defined criterion that helps to make predictions about the necessary wordlength of an observation signal. The norm calculation estimates the maximum signal excursion for an observation signal. Consequently, the number of MSBs that are needed to avoid overflow can be determined.

The impulse response, $h_{ij}(n)$, is defined as the response of the observation signal, j, when the node, i, is driven by a discrete-time impulse. The frequency response, $H_{ij}(j\omega)$, between i and j is the transfer function, $H_{ij}(z)$, evaluated on the unit circle.

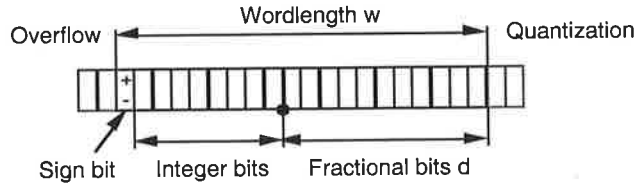


Figure 4.6. *Fixed-point bit representation.*

$\mathbf{L_1}$ norm is the worst-case norm. The signals observed inside the algorithm cannot have a greater magnitude. It is the upper bound for the signal magnitude excursion for all observed signals. The applied input stimulus is a discrete-time impulse.

$$L_{1_{ij}} = \sum_{n=0}^{\infty} |h_{ij}(n)| \tag{4.22}$$

$\mathbf{L_2}$ norm is the upper bound for the signal energy of the observation signal. The applied input stimulus is a discrete-time impulse.

$$L_{2_{ij}} = \sum_{n=0}^{\infty} |h_{ij}(n)|^2 \tag{4.23}$$

$\mathbf{L_\infty}$ norm is the upper bound for the signal magnitude excursion of the observation signal. The input signal is a sinusoidal stimulus, with amplitude 1. The L_∞ norm is calculated in the frequency band, $[\omega_1, \omega_2]$.

$$L_{\infty_{ij}} = Max|H_{ij}(j\omega)| \quad ; \text{ with } \omega_1 \, , \, \omega \, , \, \omega_2 \tag{4.24}$$

The $\mathbf{L_1}$ and $\mathbf{L_2}$ norms are calculated in the time domain; the $\mathbf{L_\infty}$ norm in the frequency domain. For instances when the positive numbers are outside the range of values that can be represented, signals must be scaled and/or the overflow characteristics must be defined. However, overflow correction and scaling both require extra hardware on chip at each operation where overflow occurs. Scaling calls for extra instruction cycles; overflow correction introduces one extra bit for each add and subtract operation. Consequently, the circuit wordlength is increased and logic, for overflow detection and correction, will be added.

If no overflow characteristic is specified, a signed fixed-point two's complement arithmetic will automatically wrap around if overflow does appear. If this happens, the sign bits are wrapped around and large scale limit cycles are created inside a DSP algorithm, as shown in Figure 4.7.

The input values are situated on the x-axis and the overflow corrected values on the y-axis. For each graph of Figure 4.7, the ideal characteristic is represented by the diagonal line. The keyword, "saturating," specifies that the output is saturated to the largest possible positive magnitude in the case of positive overflow and to the largest negative magnitude in the case of negative overflow. The error that is produced is small and no large scale limit cycles are generated. For fixed-point two's complement arithmetic implementations, saturation is recommended. The keyword, "zero-saturating," means that the output is forced to zero in both cases, positive and negative overflow.

Quantization Characteristics

Customarily, signal processing of digital systems is intended for linear operations. Inside an algorithm, the required linearity can only be achieved to a certain degree. The amount of deviation from the linear behavior can be minimized by selecting sufficiently long binary wordlengths for the signals inside the algorithm. One of these large scale non-linearities was previously discussed, alongside overflow. Yet, typical finite signal wordlength effects remain, which cause a digital system to generate noise, due to quantization effects. These small scale limit cycles can affect the dynamic range and, even worse, the stability of a digital system.

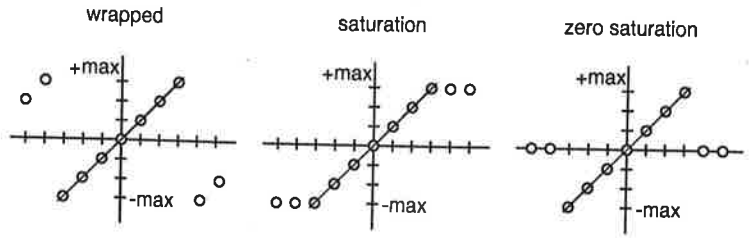


Figure 4.7. *Overflow characteristics.*

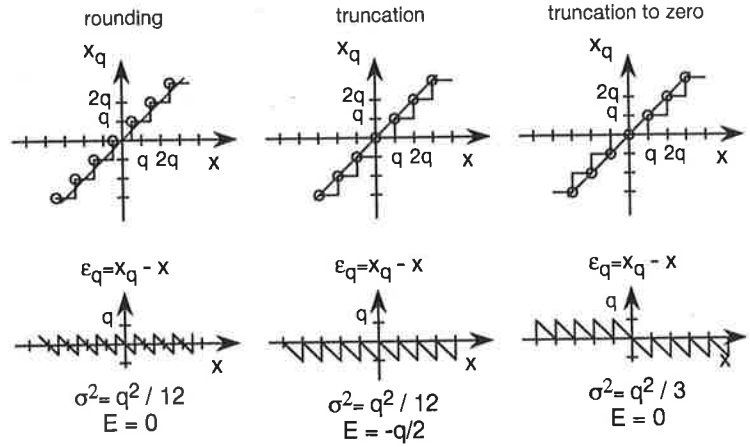


Figure 4.8. *Quantization characteristics.*

Therefore, it is standard procedure to investigate these finite signal wordlength effects by performing a bit-true simulation in the time domain. The behavior of the digital system depends on the type of quantization characteristics, whether or not limit cycles occur, and at which level the remaining quantization noise is situated. Two commonly used quantization methods are rounding and truncation.

Rounding: The truncated bit in the highest bit position is the "round-bit." It is added to the next Least Significant Bit (LSB) with an adder block.

Truncation: Truncation can be realized by eliminating the extraneous bits from the LSB side. Truncation to zero is called *sign-magnitude truncation*. For negative values, it must be checked if one or more of the truncated bits differ from zero. When this situation arises, a "1" must be added to the signal at the position of the new LSB.

4.6.2. Numerical Performance of the Steady-State OSM

The discussion now will focus on the analysis of the numerical performance of the steady-state Oscillator Signal Model. The algorithm will be chosen for a bit-serial Mistral 1 ASIC implementation and presented in details in Chapter 6.

The design parameter for the steady-state OSM has been chosen at 0.1. The sampling frequency is at 3200 Hz. Input and output wordlength is of type *fix*<16,15>. The considered target architecture uses fixed-point arithmetic. Therefore, quantization occurs and internal noise sources will alter the signal. The extraction of the analysis results is done inside *DSP Station*™ using *DSPlab*™.

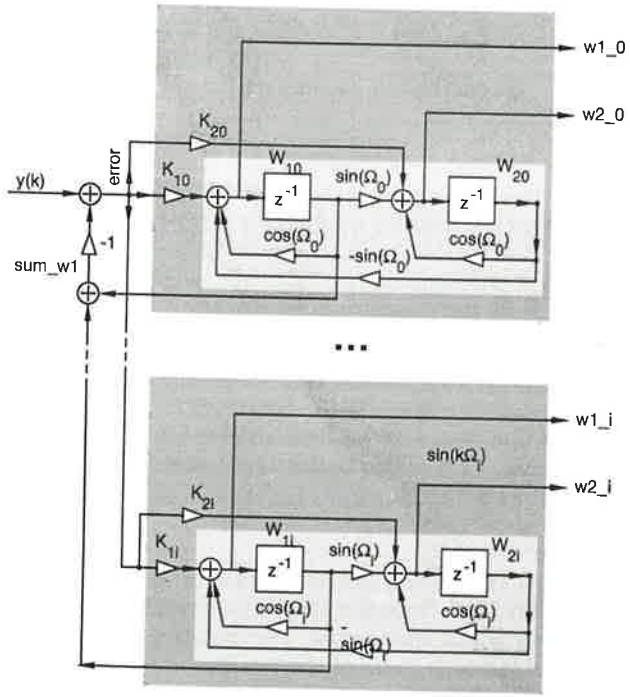


Figure 4.9. *Block implementation of modified steady-state OSM.*

Table 4.4. *Calculation of the Time Domain Norms L1, L2 of the Unit Impulse Response and Frequency Domain Norms L∞ from Node y(k) to the Observation Nodes.*

Observation node	L_1 norm	L_2 norm	L_∞ norm
error	6.14601e+00	1.66829e+00	4.33645e-04
sum_w1	5.14601e+00	6.68294e-01	4.33645e-04
w1_0	1.21999e+00	2.32089e-02	8.78127e-04
w1_50	1.34385e+00	2.38336e-02	3.91973e-04
w2_50	1.29454e+00	1.79332e-02	3.81253e-04
w1_100	1.37735e+00	2.45453e-02	3.55175e-04
w2_100	1.29981e+00	1.90577e-02	3.51776e-04
w1_150	1.50048e+00	2.23066e-02	4.70827e-04
w2_150	1.36834e+00	1.64536e-02	4.65599e-04
w1_175	1.56710e+00	1.95701e-02	6.96360e-04
w2_175	1.42539e+00	1.33347e-02	6.98218e-04
w1_188	1.55234e+00	1.64869e-02	7.73648e-04
w2_188	1.56790e+00	1.66057e-02	7.83854e-04
w1_200	1.67754e+00	1.91066e-02	5.70587e-04
w2_200	1.79368e+00	2.73839e-02	5.56453e-04
w1_250	1.56217e+00	3.11521e-02	3.23597e-04
w2_250	1.65094e+00	3.67977e-02	3.35022e-04
w1_300	1.73855e+00	4.12260e-02	3.07994e-04
w2_300	1.82114e+00	5.01047e-02	3.20195e-04
w1_350	2.33793e+00	8.67807e-02	2.45943e-04
w2_350	2.50851e+00	1.03049e-01	2.60892e-04

Mapping of the algorithm can happen on the target architecture when the numerical performance of the algorithm is examined on bit-true level. The results emphasis an optimization of the algorithm. The optimization is always a trade-off between numerical performance and silicon complexity on the chip. A increased performance of the numerical computation increase the silicon area and often decrease the execution speed of the integrated circuit.

The state variables, $\underline{\hat{x}}(k+1)$, used to calculate the magnitude, A_n, of spectral line n is illustrated in Figure 4.9. In Figure 4.9 input and output signals are represented in signed fractional twos-complement notation.

Table 4.5. *Noise and Small-Scale Limit Cycle Norms: Values from the Noise Sources to the Observation Nodes.*
R : (RMS value of statistical noise)
N : (RMS value of maximal statistical noise)
E : (worst case value of statistical noise)

Observation node	R norm	N norm	E norm
error	2.40959e-04	4.33645e-04	0.00000e+00
sum_w1	2.40959e-04	4.33645e-04	7.18634e-03
w1_0	**8.37131e-04**	8.78127e-04	8.70803e-04
w1_50	1.88072e-04	3.91973e-04	1.03915e-03
w2_50	5.69735e-05	3.81253e-04	1.13815e-03
w1_100	1.01032e-04	3.55175e-04	1.03224e-03
w2_100	5.52679e-05	3.51776e-04	1.06932e-03
w1_150	7.66891e-05	4.70827e-04	1.32904e-03
w2_150	6.31775e-05	4.65599e-04	1.33424e-03
w1_175	9.09725e-05	6.96360e-04	1.96344e-03
w2_175	7.98260e-05	6.98218e-04	1.98631e-03
w1_188	1.07451e-04	7.73648e-04	2.17736e-03
w2_188	9.04683e-05	7.83854e-04	2.23921e-03
w1_200	1.04022e-04	5.70587e-04	1.62959e-03
w2_200	8.94575e-05	5.56453e-04	1.65310e-03
w1_250	7.37883e-05	3.23597e-04	9.22577e-04
w2_250	6.00773e-05	3.35022e-04	9.67745e-04
w1_300	6.64079e-05	3.07994e-04	8.58693e-04
w2_300	5.81202e-05	3.20195e-04	8.99485e-04
w1_350	5.94492e-05	2.45943e-04	6.95124e-04
w2_350	5.71132e-05	2.60892e-04	7.17041e-04

Achieved Performance

A norms analysis provide the necessary information on the signal magnitude excursion inside the signal flow graph.

Internal overflow can be avoided by scaling specific nodes. The analysis is done for the L_1, L_2 and L_∞ norms. Table 4.4 summarizes the achieved results. As it has been stated before the L_1 norms is pessimistic and would introduce a increase of the data wordlength. A good compromise is the L_2 norms that measures the signal energy instead the signal magnitude. Only one node, namely the error signal can cause some overflow. All the other nodes are well scaled. A bit-level representation of the norms is given in Table 4.6.

Table 4.6. *Templates for the Selected Observation Nodes.*

Observation node	Bit position: type fix<16,15> 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
error	S # # # # # # # # # # # # N R R R
sum_w1	S S # # # # # # E E E E N R R R
w1_0	**P P P P** # # # # # # # R R R R R
w1_50	L P P P # # # # # # E E N R R R
w2_50	P P P P # # # # # # E E N N N R
w1_100	L P P P # # # # # # E E N N R R
w2_100	L P P P # # # # # # E E N N N R
w1_150	L P P P # # # # # # E E N N R R
w2_150	L P P P # # # # # # E E N N R R
w1_175	P P P P # # # # # E E N N N R R
w2_175	P P P P P # # # # E E N N N R R
w1_188	L P P P # # # # # E E N N N R R
w2_188	L P P P # # # # # E E N N N R R
w1_200	P P P P # # # # # # E N N N R R
w2_200	P P P P # # # # # # E N N N R R
w1_250	P P P P # # # # # # # E N N R R
w2_250	P P P P # # # # # # # E N N N R
w1_300	P P P P # # # # # # # E N N R R
w2_300	P P P P # # # # # # # E N N N R
w1_350	P P P # # # # # # # # E N N N R
w2_350	P P P # # # # # # # # E N N N R

Templates legend (in decreasing order of priority)

- R: R norm (RMS value of statistical noise)
- N: N norm (RMS value of maximal statistical noise)
- E: E norm (worst case value of statistical noise)
- #: L2 norm (RMS signal value)
- P: L∞ norm (worst case signal value with sinusoidal stimulus)
- L: L1 norm (worst case signal value with impulse stimulus)
- S: unused signal bits (usable for sign encoding)

Quantization occurs inside the signal flow graph. Each quantization node introduces an internal noise source that affect the desired signal. The sum of all internal noise sources determine the

signal-to-noise ratio (SNR). The noise analysis takes into account the small-scale limit cycles as well, shown in Table 4.5. The most affected node by noise is w1_0, having a noise magnitude of 8.3e-04. The maximum achievable SNR is limited at 60 dB using the format type *fix*<16,15>. In other words, the last 5 bit of the output signal w1_0 is affected by noise that is generated inside the algorithm. A bit-level representation of the templates is given in Table 4.6.

A carefully look at the templates presented in Table 4.6 allows to analyze the accomplished results. The most critical nodes that are affected most by quantization noise in decreasing order are; w1_0, sum_w1, error, w1_50. The state variable, w1_0, for the 0 Hz (DC) component is most critical. Compared to all other nodes two more bit are lost due to quantization and limit-cycles.

Additionally, sinusoidal signals placed close together generate increased internal noise. This can be observed for the 175 Hz and 188 Hz component that are placed inbetween 150 Hz and 200 Hz.

4.7. SUMMARY

Under the assumption of stationary random processes and for the considered application, the error covariances tend to stationary functions. The error covariance matrices and the Kalman Gain can be computed in advance. Specifically, the matrix multiplications and the matrix inversion are transferred from real-time to off-line calculation. The number of equations needed to determine the on-line estimates has been reduced from five to two, thus exorbitantly abbreviating the real-time computational complexity of the steady-state Kalman estimator.

There is no noticeable difference between the frequency characteristic of each steady-state Kalman estimator, as compared with its on-line algorithm. However, the steady-state Kalman estimator has a faster tracking speed than its on-line algorithm.

Comparing the steady-state RWSM to the steady-state OSM, computation is decreased by 11%. In both cases, the number of arithmetic operations is linearly related to the number of frequency lines — the filter order is linearly dependent on the number of

analyzed frequencies. This is the most significant improvement that has been achieved thus far.

Using the steady-state Kalman estimator, each input sample creates an estimate of the output. This iterative calculation procedure can be seen as short-time spectrum analysis. The time needed to track correct estimates is comparable to the window function used in classical short-time Fourier transform.

The number of arithmetic operations gained by using the steady-state RWSM, instead of Radix 2 and split-radix FFT, two very efficient Fast Fourier Transform algorithms, is notable. The gain variation is the difference between the linear dependence of the steady-state Kalman estimator and the logarithmic dependence of the FFT algorithms. The gain over the split-radix FFT increased from 4 for 16 estimated frequencies to 10.8 for 1024 estimated frequencies.

Because the steady-state Kalman estimator uses an iterative calculation method to estimate the Fourier coefficients, no data block has to be stored and thus memory occupation for data remains low. On the other hand, steady-state Kalman estimators require a sine and a cosine table. If the ratio of the lowest estimated frequency to the sampling frequency is small, the tables could be large, engaging more space.

In conclusion, FFT algorithms are best suited for applications that necessitate high tracking speed, where noise rejection is secondary. The distance between neighboring frequencies is fixed. FFT algorithms determine the entire spectrum and results are available only after calculation of the complete data block.

The steady-state Kalman estimator offers better quality estimations, especially for signals measured in noise. Because the stationarity of the sinusoid is related to the tracking-time, the Kalman filter is well-suited for quasi-stationary signals. An important advantage of the Kalman filter is that estimates are available at each sample, for any set of frequencies anywhere along the frequency axis.

Appendix

The following tables contain the steady-state Kalman Gains for the Random Walk Signal Model and for the Oscillator Signal Model. The gains are for three design parameters, 0.1, 0.001, and 0.00001. The setup which was used for the noise analysis in Section 3.8 is again implemented.

Steady-State Kalman Gain for the Oscillator Signal Model: Spectrum Estimation of 32 Equidistant Frequency Lines (f_s= 3.2 kHz).

		OSM. q/r=1E-1	OSM. q/r=1E-3	OSM. q/r=1E-5
DC	K_1	3.0056e-02	1.9171e-02	3.0016e-03
	K_2	0	0	0
50 Hz	K_1	2.9770e-02	1.9124e-02	3.0014e-03
	K_2	-4.1338e-03	-1.3331e-03	-2.4054e-05
100 Hz	K_1	2.9982e-02	1.9159e-02	3.0014e-03
	K_2	-2.1104e-03	-6.6502e-04	-1.1911e-05
150 Hz	K_1	3.0024e-02	1.9166e-02	3.0014e-03
	K_2	-1.3921e-03	-4.3668e-04	-7.8099e-06
200 Hz	K_1	3.0039e-02	1.9168e-02	3.0015e-03
	K_2	-1.0217e-03	-3.1996e-04	-5.7198e-06
250 Hz	K_1	3.0045e-02	1.9169e-02	3.0015e-03
	K_2	-7.9252e-04	-2.4800e-04	-4.4330e-06
300 Hz	K_1	3.0049e-02	1.9170e-02	3.0015e-03
	K_2	-6.3431e-04	-1.9841e-04	-3.5463e-06
350 Hz	K_1	3.0051e-02	1.9170e-02	3.0015e-03
	K_2	-5.1661e-04	-1.6156e-04	-2.8871e-06
400 Hz	K_1	3.0053e-02	1.9170e-02	3.0015e-03
	K_2	-4.2406e-04	-1.3259e-04	-2.3691e-06
450 Hz	K_1	3.0054e-02	1.9170e-02	3.0015e-03
	K_2	-3.4806e-04	-1.0882e-04	-1.9442e-06
500 Hz	K_1	3.0055e-02	1.9170e-02	3.0015e-03
	K_2	-2.8341e-04	-8.8600e-05	-1.5831e-06
550 Hz	K_1	3.0055e-02	1.9170e-02	3.0015e-03
	K_2	-2.2673e-04	-7.0877e-05	-1.2666e-06
600 Hz	K_1	3.0055e-02	1.9171e-02	3.0015e-03

	K_2	-1.7571e-04	-5.4926e-05	-9.8161e-07
650 Hz	K_1	3.0056e-02	1.9171e-02	3.0015e-03
	K_2	-1.2869e-04	-4.0225e-05	-7.1875e-07
700 Hz	K_1	3.0056e-02	1.9171e-02	3.0015e-03
	K_2	-8.4384e-05	-2.6377e-05	-4.7112e-07
750 Hz	K_1	3.0056e-02	1.9171e-02	3.0015e-03
	K_2	-4.1783e-05	-1.3060e-05	-2.3318e-07
800 Hz	K_1	3.0056e-02	1.9171e-02	3.0015e-03
	K_2	2.3882e-17	3.9447e-18	1.5621e-18
850 Hz	K_1	3.0056e-02	1.9171e-02	3.0015e-03
	K_2	4.1783e-05	1.3060e-05	2.3318e-07
900 Hz	K_1	3.0056e-02	1.9171e-02	3.0015e-03
	K_2	8.4384e-05	2.6377e-05	4.7112e-07
950 Hz	K_1	3.0056e-02	1.9171e-02	3.0015e-03
	K_2	1.2869e-04	4.0225e-05	7.1875e-07
1000 Hz	K_1	3.0055e-02	1.9171e-02	3.0015e-03
	K_2	1.7571e-04	5.4926e-05	9.8161e-07
1050 Hz	K_1	3.0055e-02	1.9170e-02	3.0015e-03
	K_2	2.2673e-04	7.0877e-05	1.2666e-06
1100 Hz	K_1	3.0055e-02	1.9170e-02	3.0015e-03
	K_2	2.8341e-04	8.8600e-05	1.5831e-06
1150 Hz	K_1	3.0054e-02	1.9170e-02	3.0015e-03
	K_2	3.4806e-04	1.0882e-04	1.9442e-06
1200 Hz	K_1	3.0053e-02	1.9170e-02	3.0015e-03
	K_2	4.2406e-04	1.3259e-04	2.3691e-06
1250 Hz	K_1	3.0051e-02	1.9170e-02	3.0015e-03
	K_2	5.1661e-04	1.6156e-04	2.8871e-06
1300 Hz	K_1	3.0049e-02	1.9170e-02	3.0015e-03
	K_2	6.3431e-04	1.9841e-04	3.5463e-06
1350 Hz	K_1	3.0045e-02	1.9169e-02	3.0015e-03
	K_2	7.9252e-04	2.4800e-04	4.4330e-06
1400 Hz	K_1	3.0039e-02	1.9168e-02	3.0015e-03
	K_2	1.0217e-03	3.1996e-04	5.7198e-06
1450 Hz	K_1	3.0024e-02	1.9166e-02	3.0014e-03
	K_2	1.3921e-03	4.3668e-04	7.8099e-06
1500 Hz	K_1	2.9982e-02	1.9159e-02	3.0014e-03
	K_2	2.1104e-03	6.6502e-04	1.1911e-05
1550 Hz	K_1	2.9770e-02	1.9124e-02	3.0014e-03
	K_2	4.1338e-03	1.3331e-03	2.4054e-05

Steady-State Kalman Gain for the Random Walk Signal Model: Spectrum Estimation of 32 Equidistant Frequency Lines (f_s= 3.2 kHz).

		Random Walk Signal Model q/r=1E-1	Random Walk Signal Model q/r=1E-3	Random Walk Signal Model q/r=1E-5
DC	α	0	0	0
	β	2.1726e-02	1.6138e-02	2.9414e-03
50 Hz	α	2.4165e-02	1.7289e-02	2.9688e-03
	β	1.8977e-02	1.4898e-02	2.9137e-03
100 Hz	α	3.0670e-02	2.2807e-02	4.1597e-03
	β	-1.8521e-03	-8.4301e-04	-1.9312e-05
150 Hz	α	2.0847e-02	1.5742e-02	2.9324e-03
	β	-2.2571e-02	-1.6524e-02	-2.9503e-03
200 Hz	α	-8.9391e-04	-4.0557e-04	-9.2743e-06
	β	-3.0713e-02	-2.2819e-02	-4.1597e-03
250 Hz	α	-2.2211e-02	-1.6358e-02	-2.9464e-03
	β	-2.1231e-02	-1.5914e-02	-2.9363e-03
300 Hz	α	-3.0721e-02	-2.2821e-02	-4.1597e-03
	β	5.5467e-04	2.5150e-04	5.7493e-06
350 Hz	α	-2.1405e-02	-1.5992e-02	-2.9381e-03
	β	2.2043e-02	1.6282e-02	2.9447e-03
400 Hz	α	3.7074e-04	1.6807e-04	3.8416e-06
	β	3.0723e-02	2.2822e-02	4.1597e-03
450 Hz	α	2.1940e-02	1.6235e-02	2.9436e-03
	β	2.1510e-02	1.6040e-02	2.9391e-03
500 Hz	α	3.0725e-02	2.2822e-02	4.1597e-03
	β	-2.4776e-04	-1.1231e-04	-2.5669e-06
550 Hz	α	2.1586e-02	1.6074e-02	2.9399e-03
	β	-2.1866e-02	-1.6201e-02	-2.9428e-03
600 Hz	α	-1.5360e-04	-6.9622e-05	-1.5912e-06
	β	-3.0725e-02	-2.2822e-02	-4.1597e-03
650 Hz	α	-2.1806e-02	-1.6174e-02	-2.9422e-03
	β	-2.1647e-02	-1.6102e-02	-2.9405e-03
700 Hz	α	-3.0726e-02	-2.2822e-02	-4.1597e-03
	β	7.3763e-05	3.3434e-05	7.6414e-07
750 Hz	α	-2.1701e-02	-1.6126e-02	-2.9411e-03
	β	2.1752e-02	1.6149e-02	2.9416e-03
800 Hz	α	3.7097e-15	2.7444e-15	4.7879e-16
	β	3.0726e-02	2.2822e-02	4.1597e-03
850 Hz	α	2.1701e-02	1.6126e-02	2.9411e-03

	β	2.1752e-02	1.6149e-02	2.9416e-03
900 Hz	α	3.0726e-02	2.2822e-02	4.1597e-03
	β	7.3763e-05	3.3434e-05	7.6414e-07
950 Hz	α	2.1806e-02	1.6174e-02	2.9422e-03
	β	-2.1647e-02	-1.6102e-02	-2.9405e-03
1000 Hz	α	1.5360e-04	6.9622e-05	1.5912e-06
	β	-3.0725e-02	-2.2822e-02	-4.1597e-03
1050 Hz	α	-2.1586e-02	-1.6074e-02	-2.9399e-03
	β	-2.1866e-02	-1.6201e-02	-2.9428e-03
1100 Hz	α	-3.0725e-02	-2.2822e-02	-4.1597e-03
	β	-2.4776e-04	-1.1231e-04	-2.5669e-06
1150 Hz	α	-2.1940e-02	-1.6235e-02	-2.9436e-03
	β	2.1510e-02	1.6040e-02	2.9391e-03
1200 Hz	α	-3.7074e-04	-1.6807e-04	-3.8416e-06
	β	3.0723e-02	2.2822e-02	4.1597e-03
1250 Hz	α	2.1405e-02	1.5992e-02	2.9381e-03
	β	2.2043e-02	1.6282e-02	2.9447e-03
1300 Hz	α	3.0721e-02	2.2821e-02	4.1597e-03
	β	5.5467e-04	2.5150e-04	5.7493e-06
1350 Hz	α	2.2211e-02	1.6358e-02	2.9464e-03
	β	-2.1231e-02	-1.5914e-02	-2.9363e-03
1400 Hz	α	8.9391e-04	4.0557e-04	9.2743e-06
	β	-3.0713e-02	-2.2819e-02	-4.1597e-03
1450 Hz	α	-2.0847e-02	-1.5742e-02	-2.9324e-03
	β	-2.2571e-02	-1.6524e-02	-2.9503e-03
1500 Hz	α	-3.0670e-02	-2.2807e-02	-4.1597e-03
	β	-1.8521e-03	-8.4301e-04	-1.9312e-05
1550 Hz	α	-2.4165e-02	-1.7289e-02	-2.9688e-03
	β	1.8977e-02	1.4898e-02	2.9137e-03

REFERENCES

[Bitm86] R.R. Bitmead, A.H. Tsoi, P.J. Parker : "A Kalman Filter Approach to Short-Time Fourier Analysis", IEEE Trans. on Acc. Syst. and Signal Processing, Vol.ASSP-34, No.6, December 1986, pp. 1493-1501.

[Cove91] M. Covell, J. Richardson : "A new, efficient structure for the short-time Fourier Transform, with an application in code-division sonar imaging", IEEE-ICASSP, Int. Conf. on Accoustics, Speech, and Signal Processing, May 14-17, Toronto, 1991, pp. 2041-2044.

[Jack86] L.B. Jackson : "Digital Filters and Signal Processing", Kluwer Academic Publishers, Boston, 1986.

[Kunt91] M. Kunt et al : "Techniques modernes de traitement numérique des signaux", Presse Polytechnique et Universitaires Romandes, Lausanne, 1991.

[Schw75] M. Schwartz, L. Shaw : " Signal Processing: Discrete Spectral Analysis, Detection, and Estimation", Mac Graw-Hill, New York, 1975.

5. DSP Implementation Results

The Kalman filter spectrum estimation approaches that have been theoretically established and justified are physically applied in this chapter.

First, each steady-state Kalman estimator and a split-radix FFT algorithm are implemented on a general purpose signal processor, produced by Motorola. The implementation results are discussed and compared.

Second, the Random Walk Signal Model is used as a spectrum analyzer in order to build a Digital Power Network Signal Analysis System. The derived results were immediately applied industrially, by Landis & Gyr AG, the collaborating company.

5.1. INTRODUCTION

A *software* implementation is investigated using a second generation Digital Signal Processor (DSP) chip produced by Motorola, the DSP56001. In the context of this chapter, Sections 5.2 through 5.5 cover the implementation of each Kalman approach and of the split-radix FFT algorithm. For the Kalman algorithms, optimal handwritten assembly codes are developed. The split-radix assembly code was provided by Motorola [Berg68, Moto]. The results, including the number of instruction cycles, the execution time, and the memory requirements, are presented. In Section 5.6, the Random Walk Signal Model is used for an industrial application.

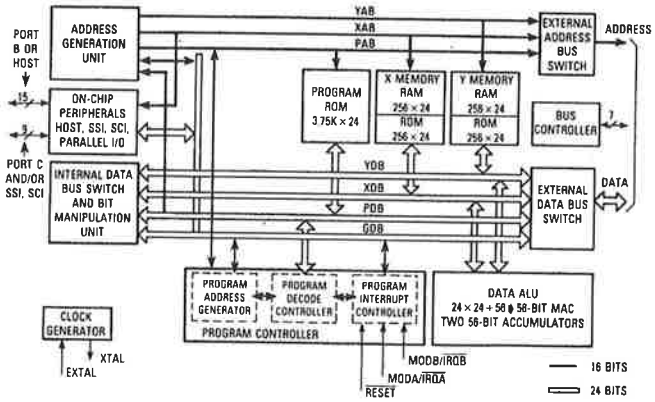


Figure 5.1. *DSP56001 architecture.*

The DSP56001 is a fixed-point arithmetic, general purpose Digital Signal Processor. Its block diagram, given in Figure 5.1, has a 24-bit fixed-point format. To guarantee margin against overflow, the accumulator has a 56-bit format.

The architecture is composed of three execution units, operating in parallel:

1. The data ALU performs arithmetic and data manipulation.
2. The address generation unit generates all addresses useful for operand and coefficient fetches.
3. The program controller pre-fetches, decodes, and executes instructions.

The DSP56001, with its Harvard parallel architecture, is well-suited for the realization of both steady-state Kalman estimators. The memory is divided into three spaces (x-data, y-data, and program code) that can be accessed concurrently, within the same instruction cycle. The data transfer and the arithmetic operations are performed simultaneously. Thus, the total number of instruction cycles nearly equals the number of arithmetic operations.

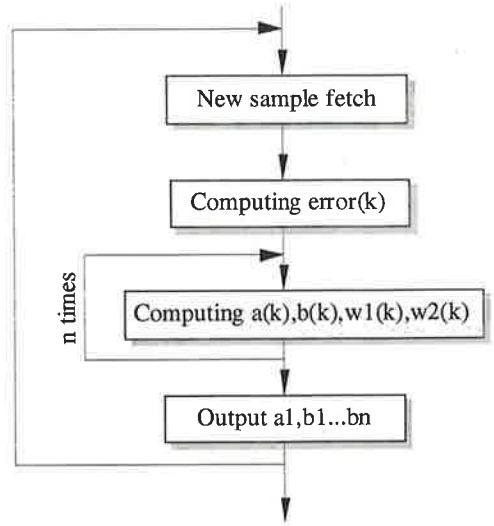


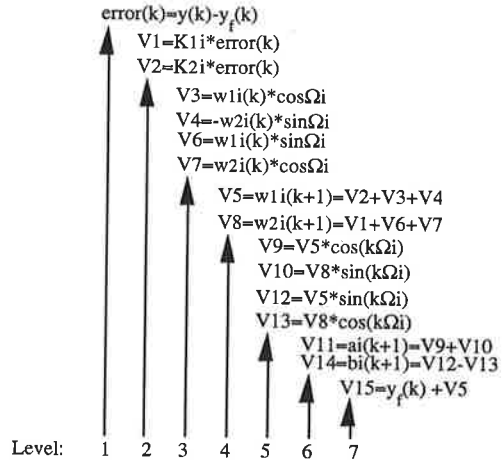
Figure 5.2. *DSP implementation flow graph.*

The first step to forming an assembly code that provides good memory organization is the elaboration of the precedence equations. To attain optimal implementation, the precedence equations should be fixed during this step. The code is optimal when the number of instructions is minimum. Inside each level, the order in which the precedence equations, V_i, are calculated is inconsequential. The programmer can choose the solution that best adheres to the DSP programming rules (see Table 5.1 and Appendix 5.A). This process is summarized in Figure 5.2.

5.2. OSCILLATOR SIGNAL MODEL

The Oscillator Signal Model requires more operations than the Random Walk Signal Model (refer to Table 4.4). Consequently, two additional equations have to be performed, which is approximately 11% more than the steady-state RWSM [Bals93, Bals92, Bals91, Gris92].

Table 5.1. *List of Precedence Equations for the Steady-State OSM*



The implementation requires $7+18M$ instructions. The number of frequency lines that are estimated is M. Clock cycle frequencies, f_c, of today's DSPs typically range from 20 MHz to 80 MHz. A general relationship has been derived to estimate the number of frequency lines that can be processed in real-time on a DSP56001. Defining f_s as the sampling rate, we have

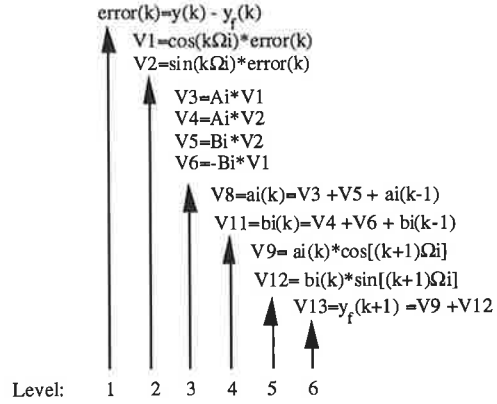
$$M \le \frac{1}{36}\left(\frac{f_c}{f_s} - 14\right) \qquad (5.1)$$

The precedence equations form the basis from which the assembly code can be written. Seven levels must be computed, sequentially, but the selected order of calculations inside each level can be arbitrary. The obtained assembly code, and its memory organization, is given in Appendix A, at the end of this chapter.

5.3. RANDOM WALK SIGNAL MODEL

The RWSM is implemented by $7+16M$ instruction cycles. The number of clock cycles (1 instruction cycle = 2 clock cycles), f_c,

Table 5.2. *Precedence Equations for the Steady-State Random Walk Signal Model.*



used by these instructions is $(14+32M)/f_c$ [Bals93, Bals92, Bals91, Gris92]. The number of frequency lines that can be calculated in real-time is

$$M \le \frac{1}{32}\left(\frac{f_c}{f_s} - 14\right) \qquad (5.2)$$

The signal flow graph of the RWSM is derived from Equation's 4.19, 4.20, and 4.21. The precedence equations, extracted from Figure 4.2, are summarized in Table 5.2. Again, the obtained assembly code, and its memory organization, is given at the end of this chapter, in Appendix B.

5.4. SPLIT-RADIX FAST FOURIER TRANSFORM

When a FFT is implemented, the *entire* spectrum of frequencies is automatically computed. N is the number of data samples. The number of frequencies that can be estimated using the Fourier approach is $M = N/2$. This is efficient for spectrum analysis but not for spectral analysis. In the latter case, a specific set of frequencies is of interest but cannot be determined separately. With only one

exception, $N=2^z$, for all integers, z. If not, alternative DFT algorithms will have to be implemented. This increases the number of instruction cycles.

Of the numerous derived FFT algorithms, the split-radix FFT algorithm is best suited for the present study — full benefits of the modern, two-data-bus DSP architecture are exploited [Kunt91, Berg68]. The split-radix, based on real signals, was derived by G. Bergland. The Bergland algorithm sequentially splits an N-point FFT into as many base-8, base-4, and base-2 stages as is possible (s is the number of stages required). $N/2$ frequency components are computed, from 0 to $f_s/2$. The frequency components are separated by f_s/N.

The tracking speed is directly related to the window length, while the frequency resolution is inversely related to the window length. However, the frequency resolution and the noise rejection (roll-off rate) are also affected by the window function that is selected.

Implementation of the Bergland algorithm is rather complex, counting up to three nested loops. Its assembly code can be found in [Moto]. The implementation of a N-point Bergland algorithm requires I instructions, where I is

$$I = 28 + \frac{5N}{4} + 11\cdot\left(\frac{N}{4}-1\right) + \sum_{s=1}^{(\log 2N)-3} 14 + \frac{5N}{2^{s+2}} + (2^s-1)\left[\left(8\cdot\frac{N}{2^{s+2}}\right)+9\right] \qquad (5.3)$$

5.5. IMPLEMENTATION RESULTS

The steady-state Kalman approaches clearly outperform the fastest known FFT algorithm (for real signals). When estimating 16 or more frequency lines, the number of instruction cycles is always lower for both Kalman methods, as compared to the Bergland FFT algorithm, and this difference increases exponentially for additional frequency lines (Table 5.3 and 5.4). The low memory requirement for data storage is also achieved (see Appendix A at the end of the chapter for proof). These results support the predictions made in Section 4.5.

Table 5.3. *Number of DSP56001 Instruction Cycles.*

	# of Instruction Cycles			% Difference
Estimated Frequency Lines	Steady-State RWSM	Steady-State OSM	Bergland Algorithm	RWSM versus Bergland
8	135	151	130	+3 %
16	263	295	319	-21
32	519	583	756	-45
64	1031	1159	1753	-70
128	2055	2311	3998	-94
256	4103	4615	8995	-119
512	8199	9223	20008	-144
1024	16391	18439	44077	-168
2048	32775	36871	96306	-193

Another incredible efficiency of the steady-state Kalman estimators is the elimination of the need to quantize intermediate calculations, thus maximizing precision of the computations.

Table 5.4. *Benchmarking on DSP56001*

Number	*Instruction Cycle Time*					
of	*100 ns*	*60 ns*	*100 ns*	*60 ns*	*100 ns*	*60 ns*
Estimated	*Steady-State OSM*		*Steady-State RWSM*		*Bergland FFT (N=2M)*	
M Lines	[ms]					
128	0.2311	0.13866	0.2055	0.1233	0.3998	0.23988
256	0.4615	0.2769	0.4103	0.24618	0.8995	0.5397
512	0.9223	0.55338	0.8199	0.49194	2.0008	1.20048
1024	1.8439	0.50634	1.6391	0.98346	4.4077	2.6446

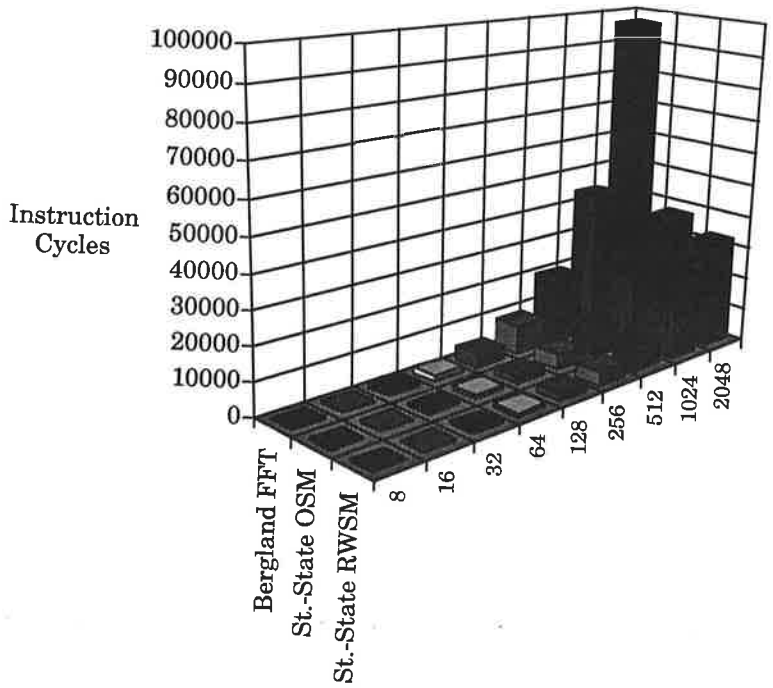


Figure 5.3. *Graphical representation of Table 5.3.*

Finally, because of the new linear address system that has been developed, the complicated address generation of FFT algorithms is no longer a limiting factor.

5.6. DIGITAL POWER NETWORK SIGNAL ANALYSIS SYSTEM

5.6.1. Motivation

Increasingly, power distribution networks are becoming polluted by consumer generated harmonic disturbances.

Power system stability is one of the most important problems in the power industry [Ague81]. The evaluation of the response to disturbances in this area determines the stability of the system that faces different situations, which could be present in power system operations, planning, and power measurement [Ilic90, Jave89, Leos89, Nwan90].

These harmonics cause energy losses and other undesirable effects. Exact knowledge about the harmonics gives valuable information that is used to monitor power networks. Naturally, then efficient and cost-effective real-time methods for measuring the harmonics would be useful to utility companies, in their efforts to reduce them [Ilic90, Jave89].

The ideal monotone network would be a power distribution system with a constant voltage amplitude and a frequency of 50 Hz (Europe) or 60 Hz (U.S. and other countries). However, because harmonic disturbances are random, a power network is stochastic. For example, periodic on-off switching, cut-off phase devices (dimmers), and rectifier bridges for electrical motors generate large harmonic disturbances. Periodic on-off switching creates sub-harmonic signals on a network. Cut-off phase devices produce thousands of Hertz of harmonic signals (see Figure 5.4) [Wied88, Will74].

Cyclo-converters, creating sub-harmonics and pseudo-harmonics, are the principal source of harmonic disturbances. Rectifier bridges form harmonics with ranks, $n = kp \pm 1$, where k is an integer and p is the number of phases used by the rectifier bridge (six for an hexa-phase bridge, twelve for a dodeca-phase bridge). The amplitude of harmonic n, I_n, decreases while n increases [Leos89].

$$I_n = \frac{I_1}{(n - 5/n)^{1.2}} \qquad n>5 \qquad (5.4)$$

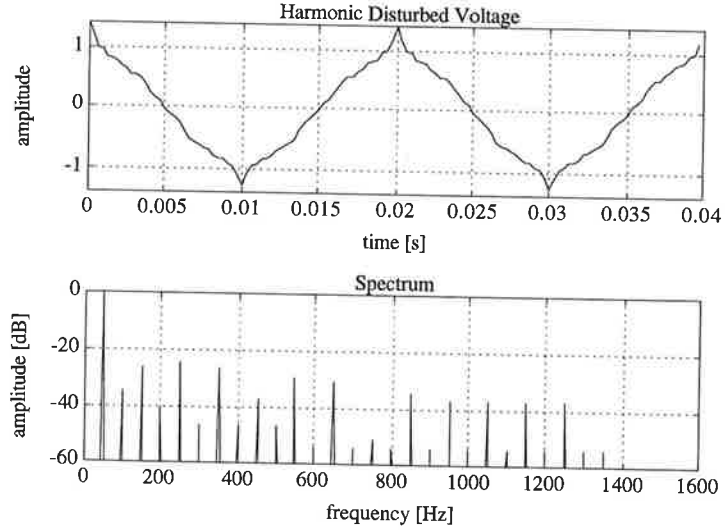


Figure 5.4. *Supply voltage in time domain and frequency domain.*

Harmonics of even rank are injected by dissymetric rectifier bridges, equipped with diodes and thyristors. *Pseudo-harmonics* are fabricated by speed-controlled, alternative current motors and by arc ovens. These arc ovens form an electrical noise with a large spectrum, where the amplitudes are decreasing as the frequencies increase.

Today's energy measurement meters are designed for networks where information about the harmonic contribution is not available. The meters are based upon the analog power measurement system, a commonly used method for computing power consumption.

The proposed measurement system, however, is based on digital signal processing — spectral analysis of current and voltage network signals. The advantage of spectral decomposition of the voltage and current signals is the ability to determine the harmonic contribution of each load. Supplementary information about the type of load and the time consumption is also available.

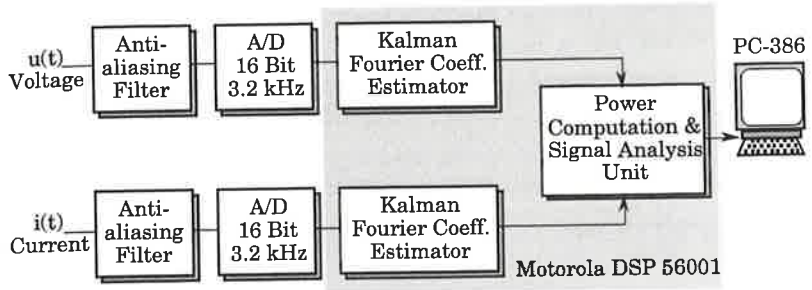


Figure 5.5. *Digital Power Network Signal Analysis System.*

The Digital Power Network Signals Analysis System (DPNSAS) is presented in Figure 5.5. First, the measured network signals are band-limited by two analog low-pass filters. Second, the current and voltage signals are sampled by analog-to-digital (A/D) converters. Third, a spectral analysis is performed on each signal. A detailed examination of the harmonic influences on the electrical power computation is the final step [Bals93, Bals92, Bals91, Gris92, Ober93, Pont92].

5.6.2. Signal Models

The instantaneous voltage, $u(t)$, and current, $i(t)$, at time t, can be expressed by the sum of M cosine functions, with additional noise sources, $u_w(t)$ and $i_w(t)$

$$u(t) = \sum_{n=0}^{M-1} U_n \cos(n\omega t + \alpha_n) + u_w(t) \tag{5.5}$$

$$i(t) = \sum_{n=0}^{M-1} I_n \cos(n\omega t + \beta_n) + i_w(t) \qquad (n \text{ is an integer}) \tag{5.6}$$

U_n and I_n are the corresponding amplitudes and $\varphi_n = \alpha_n - \beta_n$ is the phase-shift occurring between each voltage and current harmonic. The period is given by $T_n = 2\pi / \omega n$. $u_w(t)$ and $i_w(t)$ are uncorrelated, Gaussian noise sources.

The effective, or Root Mean Square (RMS), voltage and current are given by

$$I_{RMS} = \sqrt{\frac{1}{T}\int_0^T i^2(t)dt} \qquad (5.7)$$

$$U_{RMS} = \sqrt{\frac{1}{T}\int_0^T u^2(t)dt} \qquad (5.8)$$

The signal models derived in Chapter 3 accurately describe the signals present in the power network. Thus, the approach that was presented is now used for spectral analysis of both network signals. In Equations 5.7 and 5.8, the RMS value must be carefully filtered (noise) if accurate results are to be obtained.

5.6.3. Implementation Results

The DPSNAS, which uses the steady-state Random Walk Signal Model as the spectral analyzer, was developed in collaboration with Landis & Gyr AG, Zug, Switzerland. The Digital Signal Processor performs, in real-time, the functions shown in Figure 5.5. The broadcast detector (ripple control) signals, sinusoidal signals that remote control specific loads (e.g. street lamps, boilers, washing-machines, etc.), are also taken into account. These ripple control signals are *not* multiples of the network fundamental and can be composed of several frequencies.

Based on the estimated Fourier coefficients, the power, the energy, and the distortion are computed in real-time. The results are transferred through the DSP host interface to a PC-AT and are then shown on specially constructed windows.

The assembler implementation of the DPNSAS is executed in 725 instruction cycles, corresponding to an execution time of 72.5 μs. This is equivalent to 25% of the available execution time (A/D sampling frequency, 3.2 kHz; DSP instruction cycle time, 100 ns). The DPNSAS has the following specifications:

1. On the voltage signal, ten frequencies are estimated. From these, eight equidistant frequencies are placed from 0 Hz to 350 Hz, with separations of 50 Hz. Two additional ripple control signals are placed at 175 Hz and at 188 Hz.
2. On the current signal, eight equidistant frequencies, separated by 50 Hz are placed from 0 Hz to 350 Hz.
3. Active, reactive, and apparent power is calculated, based on the Fourier coefficient for each of the eight frequency lines.
4. The average value for active, reactive, and apparent power is determined.
5. Effective voltage and current, including two square-root computations (23-bit precision), is extracted.

To test the accuracy of the DPNSAS, numerical input signals are generated on the DSP itself, containing 40% of the harmonics for each, voltage and current (23-bit precision). The dynamic range applied for the current signal was decreased from 1 to 1/1024 of the current amplitude; the voltage signal was kept at its maximum for the duration of the test. The maximum error over the indicated dynamic range, for the estimated active power, depends on the design parameter, q/r. For q/r = 0.01, the error is 0.02%; for q/r = 0.0001, the error increases to 0.3%. No numerical instabilities due to quantization effects were observed during real-time tests.

5.7. SUMMARY

Two steady-state Kalman algorithms and a Bergland FFT were implemented on a Motorola DSP56001 signal processor. To allow for comparison, the same specifications were used for each method. It was shown that the steady-state Kalman approaches require more instruction cycles for all $M \leq 8$, with M depicting the number of estimated frequency lines. However, for all $M \geq 16$, the Kalman techniques are faster.

The FFT approaches feature several major drawbacks. The frequency resolution depends on the number of points and on the applied window function. In order to increase the frequency resolution, a large N is needed. This is further complicated by the fact that N is limited to the set, 2^z, for all integers, z. Furthermore, the complete spectrum is automatically calculated, even if only a

certain frequency band is of interest. Finally, the algorithm's complicated addressing of data and coefficients, based on block data processing, requires that the entire data block be stored.

These drawbacks are irrelevant for the Kalman approaches. Only the desired frequencies have to be implemented and thus, for applications where only a band of frequencies needs to be analyzed, Kalman is executed more efficiently. Also, the data and coefficient addressing is unambiguous, with no pre-address calculations required.

The steady-state Random Walk Signal Model was used for an industrial application — a Digital Power Network Signal Analysis System, implemented on the DSP56001. The DSP performs the estimation of the Fourier coefficients for the power network signals and determines the power, the distortion, and the ripple control signals. The DSP board, interfaced to the PC screen, communicates the various findings.

It is now clear that the steady-state Kalman approach is appropriate for real-time implementations. The regular structure of the signal flow graph and the sequential addressing of coefficients and variables lead to a very efficient assembly code. There is no need to program in nested loops, which invariably increases the processing time.

Appendix A

Kernel Implementation on DSP56001: Steady-State Oscillator Signal Model

```
    move      x:inp_vol,a
; error = y - y_est
    move      y:y_est_vol,x0
    sub  0,a
    move      a,x:error
; clr y_est
    clr  a
    move      a,y:y_est_vol
; loop (estimates states variables for each frequency)
    do        #n_freq,end_comp_vol
; address and step for the sin table
    move      x:(r0),r5
    move      #offset,n5
; V1 = error*k1
    move      y:(r4)+,y0 x:(r3),x0
    mpy       x0,y0,a    x:(r2)+,x1 y:(r4)+,y1
; V2 = error*k2
    mpy       x0,y1,b    x:(r2)-,x0  y:(r4)+,y0
; V2+V3 = V2+W1*cos(Fi)
    mac             x1,y0,b                  y:(r4)+,y1
; V5 = V2+V3+V4 = V2+V3-W2*sin(Fi)
    macr      -x0,y1,b
; V1+V6 = V1+W1*sin(Fi)
    mac       x1,y1,a    b,x:(r2)+   y:(r5)+n5,y1
;V8 = V1+V6+V7 = V1+V6+W2*cos(Fi)
    macr      x0,y0,a    b,x1        y:(r5)-n5,y0
;V9 = V5*cos(k*Fi)
    mpy       x1,y0,b    a,x0
    move      x:(r1)+,n5
;V11 = V9+V10 = V9+V8*sin(k*Fi)
    macr      x0,y1,b    a,x:(r2)+
;V12 = V5*sin(k*Fi)
    mpy       x1,y1,a                  b,y:(r6)+
;V14 = V12+V13 = V12-V8*cos(k*Fi)
    macr      -x0,y0,a                 y:y_est_vol,b
```

```
;V15 = y*+V5
      add       x1,b                a,y:(r6)+
      clr       a                   (r5)+n5
      move      b,y:y_est_vol
;save position in sin table
      move      r5,x:(r0)+
end_comp_vol
```

Memory Organization

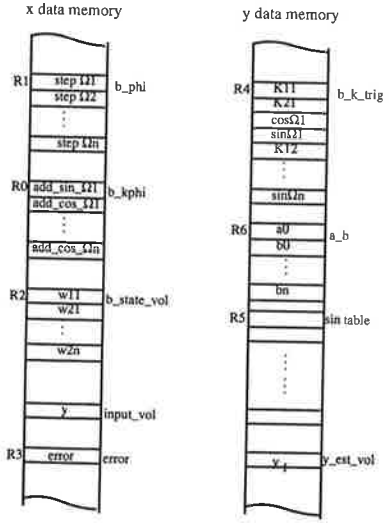


Figure A.1. *Memory organization for the Oscillator Signal Model.*

Appendix B

Kernel Implementation on DSP56001: Steady-State Random Walk Signal Model

```
;error= y - y_est
      move      y:y_est_vol,x0
      sub       x0,a
      move      a,x:error
; clr y_est
      clr   a
      move      a,y:y_est_vol
; state variables access
      move      #b_state_vol,r2
; loop (estimate states variables for each frequency)
      do        #n_freq,end_comp_vol
; address and step for the cosinus
      move      y:(r4)+,r5
      move      x:(r1),n5
; address and step for the sinus
      move      y:(r4)+,r6
      move      x:(r1)+,n6
      move      x:inp_vol,a         ; error
; V1 = error*cos(k*phi[i])
      move      y:(r5)+n5,y0        x:(r3),x0
      mpyr      x0,y0,a             x:(r0)+,x1 y:(r6)+n6,y0
; V2 = error*sin(k*phi[i])
      mpyr      x0,y0,b             a,y0       x:(r2)+,a
; V3 = apha[i]*V1 + a[i](k-1)
      mac       x1,y0,a             b,y1       x:(r0)+,x0
; a[i](k) = beta[i]*V2 + V3
      macr      x0,y1,a             x:(r2)-,b
; V4 = -beta[i]*V1 + b[i](k-1)
      mac       -x0,y0,b            a,x:(r2)+  y:(r5),y0
; b[i](k) = alpha[i]*V2 + V4
      macr      x1,y1,b             a,x1       y:y_est_vol,a
; feedback: V5= a[i](k)*cos((k+1)*phi[i])+b[i](k)*sin((k+1)*phi[i])
      mac       x1,y0,a             b,x1       y:(r6),y0
      macr      x1,y0,a             b,x:(r2)+
      move      a,y:y_est_vol
```

```
    move        r5,y:(r4)+
    move        r6,y:(r4)+
end_comp_vol
```

Memory Organization

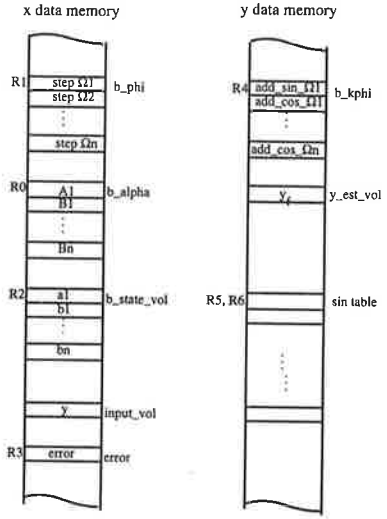


Figure B.1. *Memory organization for the Random Walk Signal Model.*

Appendix C

Electrical Power Measurement Based On Fourier Coefficients

C.1. INTRODUCTION

The aim of this appendix is to highlight the most important physical properties of one-phase sinusoidal voltage networks. Power systems are designed to satisfy consumer loads and therefore voltage and current must be available when the consumer desires it. Electrical power is provided by using either a constant voltage, a one-phase sinusoidal voltage, or a three-phase sinusoidal voltage network. A one-phase sinusoidal voltage network is the least complicated. The following equations are valid only for one-phase networks, but can be easily extended to three-phase networks.

Voltage and current sensors are, in general, sophisticated transducers that deliver a signal in the time domain. However, to determine the harmonic contribution, knowledge about its frequency components is essential. To analyze these components, the signals must be transformed from the time domain to the frequency domain. This can be achieved by applying a Fourier transform to the acquired signal. The signals are assumed to be real and periodic; thus, the spectrum is discrete.

The Fourier series of a real, periodic function, $u(t)$, with period T, is the sum of the sinus and cosine functions, each weighted by a Fourier coefficient. b_n corresponds with the sinus, a_n with the cosine

$$u(t) = a_o + \sum_{n=1}^{\infty} [a_n \cos(n\omega_o t) + b_n \sin(n\omega_o t)] \tag{5.9}$$

The complex expressions are

$$u(t) = \sum_{n=-\infty}^{\infty} c_n e^{jn\omega_o t} \tag{5.10}$$

The complex coefficient, c_n, is found by

$$c_n = \frac{1}{2}(a_n - jb_n) = \frac{1}{T}\int_{-T/2}^{T/2} u(t)e^{-jn\omega_o t}dt \qquad \omega_0 = 2\pi / T \tag{5.11}$$

$n = 0, \pm 1, \pm 2, \ldots$

The amplitude, A_n, phase, ϕ_n, of the n-th spectral line is

$$A_n = \sqrt[2]{a_n^2 + b_n^2}, \qquad \phi_n = arctg\frac{-b_n}{a_n} \tag{5.12}$$

C.2. INSTANTANEOUS POWER — $p(t)$

The *instantaneous power*, $p(t)$ is composed of two terms. The first term, $U_n I_n \cos(\varphi_n)$, is time-invariant and depends on the phase-shift and the amplitude. The second term, $\cos(2n\omega t + \alpha_n + \beta_n)$ is time-varying, with frequency, 2nω, and the 50 Hz fundamental. This term expresses the oscillating energy, flowing between the source and the charge. Instantaneous power is given by

$$p(t) = u(t)i(t) = \frac{1}{2}\sum_{n=0}^{M-1} U_n I_n[\cos(\varphi_n) + \cos(2n\omega t + \alpha_n + \beta_n)] \tag{5.13}$$

Replacing β_n with $\alpha_n - \varphi_n$, Equation 5.13 becomes

$$p(t) = \frac{1}{2}\sum_{n=0}^{M-1}\{U_n I_n \cos(\varphi_n)[1 + \cos(2n\omega t + 2\alpha_n)] + U_n I_n \sin(\varphi_n)\sin(2n\omega t + 2\alpha_n)\} \tag{5.14}$$

(5.14)

If $\varphi_n = 0$, the load is a pure, resistive charge: $\cos\varphi_n = 1$, $\sin\varphi_n = 0$, and the average instant power stabilizes at its maximum value. Average power becomes

$$p(t) = cte. = \frac{1}{2}\sum_{n=0}^{M-1} U_n I_n = \bar{p}.$$

On the other hand, if $\varphi_n = \pm\pi/2$, the load is a pure, reactive

charge. This means the charge is either capacitive or inductive. In this case, $\cos\varphi_n = 0$, $\sin\varphi_n = \pm 1$, and the instantaneous power is

$$p(t) = \frac{1}{2}\sum_{n=0}^{M-1} U_n I_n \sin(2n\omega t) \qquad \bar{p} = 0$$

C.3. ACTIVE POWER — P

The active energy absorbed by a consumer is the equivalent, or mechanical, or thermal, energy. The *active power*, P, is the average of the instantaneous power, calculated over one period of the fundamental network frequency. Considering the sinusoidal nature of the signals, corresponding to full set of orthogonal functions and taking into account Parseval relation, it should be noted that only the spectral lines which have the same frequencies for current and voltage contribute to the active power [Fisch82].

$$P = \frac{1}{T}\int_0^T p(t)dt \qquad \text{[Watt]} \tag{5.15}$$

$$= \sum_{n=0}^{M-1}\frac{1}{T}\int_0^T p_n(t)dt = \sum_{n=0}^{M-1}\frac{U_n I_n}{2}\cos(\varphi_n) \tag{5.16}$$

The maximum active power is obtained for $\varphi_n = 0$. The active power is measured with a Wattmeter and then charged to the consumer. The active power, expressed by the Fourier coefficients, a_n and b_n, for the spectral line, n, becomes

$$U_n(t) = U_n \sin(\omega_n t + \alpha_n) = U_n \sin(\omega_n t)\cos(\alpha_n) + U_n \cos(\omega_n t)\sin(\alpha_n)$$

$$= {}^{u}\mathbf{a}_n \cos(\omega_n t) + {}^{u}\mathbf{b}_n \sin(\omega_n t) \tag{5.17}$$

$$I_n(t) = I_n \sin(\omega_n t + \beta_n) = I_n \sin(\omega_n t)\cos(\beta_n) + I_n \cos(\omega_n t)\sin(\beta_n)$$

$$= {}^{i}\mathbf{a}_n \cos(\omega_n t) + {}^{i}\mathbf{b}_n \sin(\omega_n t) \tag{5.18}$$

The Fourier coefficients, expressed by Equations 5.17 and 5.18, are

$$ {}^{u}a_n = U_n \sin(\alpha_n) \qquad {}^{u}b_n = U_n \cos(\alpha_n) \tag{5.19} $$

$$ {}^{i}a_n = I_n \sin(\beta_n) \qquad {}^{i}b_n = I_n \cos(\beta_n) \tag{5.20} $$

After trigonometric transformation, the active power expressed by the Fourier coefficients is given by

$$ P = 0.5 U_n I_n \cos(\alpha_n - \beta_n) $$

$$ = 0.5 U_n I_n [\cos(\alpha_n)\cos(\beta_n) + \sin(\alpha_n)\sin(\beta_n)] \tag{5.21} $$

$$ P = 0.5 [\sum_{n=0}^{M-1} ({}^{u}a_n {}^{i}a_n + {}^{u}b_n {}^{i}b_n)] \tag{5.22} $$

C.4. THE REACTIVE POWER — P_q

The *reactive power*, P_q, characterizes the "not converted" energy flow of reactive charges. This "not converted" reactive energy, flowing between the power source and the charge (consumer) is equal to zero, after an even number of periods. For the power industry, the consumer who creates reactive power charges the distribution network, producing important losses due to the Joule effect. Undoubtedly, the electrical power industry would like to maximize active power transmission and minimize reactive power.

The reactive power identifies a consumers as either inductive or capacitive. An inductive charge is detected if $\varphi_n > 0$ (positive reactive power); a capacitive charge if $\varphi_n < 0$ (negative reactive power).

$$ P_q = \sum_{n=0}^{M-1} \frac{1}{T} \int_0^T p_n dt \quad [\text{var}] \tag{5.23} $$

The reactive power for sinusoidal signals is defined by the amplitude of the time-varying component of the instantaneous power. It can now be computed with the Fourier coefficients

$$ P_q = \sum_{n=0}^{M-1} U_n I_n \sin(\varphi_n) \qquad where \varphi_n = \alpha_n - \beta_n $$

$$ = 0.5 \sum_{n=0}^{M-1} U_n I_n [\sin(\alpha_n)\cos(\beta_n) - \cos(\alpha_n)\sin(\beta_n)] \tag{5.24} $$

$$ P_q = 0.5 \sum_{n=0}^{N} ({}^{u}a_n {}^{i}b_n - {}^{i}a_n {}^{u}b_n) \tag{5.25} $$

C.5. APPARENT POWER — P_s

The *apparent power*, P_s, is the time-varying portion of the instantaneous power. It is defined as the product of effective voltage and effective current. This product is a power definition, but does not necessarily contribute to work. For facility purposes, the unity is [VA] for Volt Ampere.

$$ P_s = U_{RMS} I_{RMS} \quad [\text{VA}] \tag{5.26} $$

The apparent power is related to the active and reactive power

$$ P_s = \sqrt{P^2 + P_q^2} \tag{5.27} $$

Equations 5.26 and 5.27, however, do not produce the same estimate of P_s — in Equation 5.26, the distortion is taken into account [Fisch82]. The apparent power for sinusoidal signals, expressed by the Fourier coefficients, is

$$ P_s = 0.5 \sum_{n=0}^{M-1} \sqrt{\{({}^{u}a_n)^2 + ({}^{u}b_n)^2\}\{({}^{i}a_n)^2 + ({}^{i}b_n)^2\}} \tag{5.28} $$

REFERENCES

[Alle77] J.B. Allen, L.R. Rabiner : "A unified approach to short-time Fourier analysis and synthesis", Proc. IEEE, Vol.65, Nov. 1977, pp. 1558-1564.

[Ague81] M. Aguet, Morf J. J. : "Energie électrique", Presse polytechnique romandes, CH-Lasuanne 1981, Traité d'électricité, Vol. XII.

[Bals95a] P. Balsiger, P. Gruber : "Echtzeit-Netzsignalanalyse basierend auf Kalman Filtern", VDI/VDE-GMA Aussprachetag, Messsignalverarbeitung und Diagnose Mittel zur Prozess- und Qualitätssicherung, Langen b. Frankfurt, 21/22. März 1995, Deutschland.

[Bals95b] P. Balsiger : "Digital Signal Processing in ASICs", Austria Mikro Systeme Int., Workshop 4-6 April 1995, Unterpremstätten, Austria.

[Bals93] P. Balsiger, S. Ponta, P. Gruber : "Echtzeit-Netzsignalanalyse basierend auf Kalman Filter", Seminarvortrag an der Eidg. Techn. Hochschule ETH-Zürich, Professur für Leistungselektronik, 23. Juni 1993.

[Bals92] P. Balsiger, S. Ponta, F. Pellandini, P. Gruber, and J. Toedli : "A Kalman Filter Approach to Power Network Signal Analysis", Proc. URSI Int'l Symposium on Signals, Systems and Electronics, ISSSE '92, Paris, France, Sept. 1992, pp. 474-477.

[Bals91] P. Balsiger, S. Ponta, *"Etude des algorithmes d'analyse et de caractérisation en temps réel de signaux quasi-périodiques"*, Rapport IMT No 306 PE 11/91, Université de Neuchâtel, IMT, Novembre 1991.

[Berg68] G. D. Bergland : *"A Fast Fourier Transform Using Base 8 Iterations"*, Math. Comp., vol 22, 1968, pages 275-279.

[Bitm86] R.R. Bitmead, A.H. Tsoi, P.J.Parker : "A Kalman Filter Approach to Short-Time Fourier Analysis", IEEE Trans. on Accoustics, Speech, and Signal Processing, Vol.ASSP-34, No.6, December 1986, pp. 1493-1501.

[Bitm82] R.R. Bitmead : "On Recursive Discrete Fourier Transformation", IEEE Trans. on Acc. Syst. and Signal Processing, Vol.ASSP-30, No.2, April 1982, pp. 319-322.

[Blah85] R.E. Blahut : "Fast algorithms for digital signal processing", Addison-Wesley, New York 1995.

[DFL92] "DSP Architect DFL User's and reference manual" Mentor Graphics Corporation, Wilsonville, Oregon 97070, USA 1992.

[Fisch82] H.D. Fischer : "Bemerkungen zu Leistungsbegriffen bei Strömen und Spannungen mit Oberschwingungen", Archiv für Elektrotechnik, Vol.64, Springer Verlag 1982, pp. 289-295.

[Gris92] L. Grisoni, S. Ponta, P. Balsiger, A. Heubi, F. Pellandini : "Short-time Spectral Analysis Based on Kalman Filters", Proc. Int'l Conf. on Signal Processing, Applications and Technology, Boston'92, Cambridge, MA-USA, Nov. 1992, pp. 145-153.

[Grub88] P. Gruber, Tödtli J. : " A Recursive Estimation for the Determination of Unknown Constant or Slowly Varying Fourier Coefficients", EUSIPCO-88, Forth European Signal Processing Conference, Grenoble, September 5-8. 1988.

[Heub91] A. Heubi, "Application d'un filtre de Kalman pour l'analyse des signaux du réseau électrique", travail de diplôme, décembre 1991, IMT Uni NE, Switzerland, 1991.

[Ilic90] M.D. Ilic : "Nonlinear Monotone Networks and their Role in the Electric Power Systems", Proc. ICASSP 90, pp. 148-151.

[Jave89] J.L. Javerzac, Léost J.Y. : "Measurement of disturbances and compatibility levels on the network: overview and prospects", EDF Bulletin de la Direction des Etudes et Recherches, Serie 8, Reseaux Electriques Materiels Electriques, No.3 1989, pp. 13-18.

[Jack86] L.B. Jackson : "Digital filters and signal processing", Kluwer Academic Publishers, Boston 1986.

[Kunt91] M. Kunt et al : "Techniques modernes de traitement numérique des signaux", Presse Polytechnique et Universitaires Romandes, Lausanne, 1991.

[Leos89] J.Y. Léost : "Les perturbations à basse fréquence de l'alimentation électrique", RGE Journal, No.2, February 1989.

[Moto] Motorola GMBH, Schatzbogen 7, 8000 München 82, Modem mail_box # 0049 89 92 103 111.

[Nwan90] C.O. Nwankpa, Shahidehpour S.M. : "A Class of Power Network Reliability Problems via Colored Noise Modelling", Proc. ICASSP 90, pp. 164-167.

[Ober93] C. Oberson, "Système d'analyse du réseau de distribution d'énergie électrique", Travail de semestre, été 1993, IMT Uni NE, Switzerland, 1993.

[Pont92] S. Ponta, P. Balsiger, *Power network signal analysis system interface*, IMT Report No 317 PE 05/92, IMT Uni NE, Switzerland, May 21, 1992.

[Rabi75] L. Rabiner, B. Gold : "Theory and application of digital signal processing", Prentice Hall, New York 1975.

[Schr91] R. Schreiber : "Digitaler Spannungs-Analysator (DSA), Messgerät zur Analyse von Netzrückwirkungen", Bulletin SEV/VSE 82(1991)15, 7. August 1991.

[Schw75] M. Schwartz, L. Shaw : "Signal processing discrete spectral analysis, detection, and estimation", McGraw-Hill, New York, 1975.

[Shep74a] W. Shepherd, P. Zakihani : "Suggested Definition of Reactive Power for Nonsinusoidal Systems", Proc. IEE, Vol.121, No.5, May 1974, pp. 389-390.

[Shep74b] W. Shepherd, P. Zakihani : "Reactive-Power Definitions and Power-Factor Improvement in Nonlinear Systems", Proc. IEE, Vol.120, No.5, May 1974, pp. 390-391.

[Shar73] D. Sharon : "Reactive-Power Definitions and Power-Factor Improvement in Nonlinear Systems", Proc. IEE, Vol.120, No.6, June 1973, pp. 704-706.

[Toed83] J. Tödtli : " Rekursive Parameterschätzverfahren nach der Methode der kleinsten Quadrate", Scienta Electronica, Vol. 29, No. 4, 1983.

[Wied88] A. Wied : "Erfassung der Stromoberschwingungen von Lichtbogenöfen mit Hilfe des Kalman Filters", Automatisierungstechnik, Vol 36, No.1, 1988, pp. 21-25.

[Will74] A.N. Willson : "Nonlinear Networks: Theory and Analysis", IEE Press Book, 1974.

6. ASIC-Design Results

The ASIC-design methodology, based on silicon compilers and VHDL-logic synthesis, is examined in this chapter. Attributes of the design process are discussed, algorithm-to-architecture mapping is introduced, mapping alternatives are offered, and the basic design strategy of silicon compilers is exposed, including a summary of the most well-known compilers.

The target silicon compiler, Mistral 1, has been chosen for bit-serial hard-wired ASIC synthesis. Design flow road map is analyzed, beginning at the algorithm level and extending to algorithm mapping, DSP architecture synthesis, and the generation of structured VHDL code.

The steady-state OSM algorithm has been preferred for silicon implementation. The algorithm has been studied in details for spectral power estimation. In this work, one has discovered a modified steady-state OSM signal flow graph, that reduces the number of arithmetic operations and offers very few memory requirements.

The modified steady-state OSM is used for four integrated circuits that have been designed down to the layout level. Each estimates the squared magnitude of a specific set of frequency lines. The first chip performs spectral power estimation of 2 frequency lines, the second 4 lines, the third 6 lines and the fourth chip calculates 10 lines.

6.1. INTRODUCTION

The evolution of silicon processing technology has made it feasible to integrate entire systems on a single chip. Today's, commercial digital CMOS chips contain up to 7'000'000 transistors. This is sufficient for implementing digital applications in real-time that were formerly impossible to build. Applications that were built using analog components, are continually being digitized.

Moreover, innovative digital applications are being created. In recent years, digital electronics has infiltrated society. Examples include the Compact Disk (CD), High Definition Television (HDTV), Integrated System Digital Network (ISDN), digital radio, desk top publishing, digital copiers, mobile telephones (Natel-D), and Global Positioning Systems (GPS, D-GPS). These new applications demonstrate that there exists a market for integrated digital products. However, due to high development costs and short life span, full-custom, hand-crafted Integrated Circuit (IC) design is often not affordable in this competitive market.

To overcome these problems, complex systems have to be integrated with Application Specific Integrated Circuits (ASICs) much faster and with minimal efficiency losses in silicon area and operation speed, compared to hand-crafted chips. To bridge the gap between system design and silicon design, new methodologies, suited for automation and leading to expensive and sophisticated Computer Aided Design (CAD) environments, must be proposed.

As design complexity increases, persistent advances in design methodologies are necessary. Over the past three years, logic synthesis on a Very High-Speed IC Hardware Description Language (VHDL) level has played an increasingly important role in the ability to apply to large design projects, resulting in an advancing evolution of the ASIC design process. The current issue in the evolution is an advanced design process which is distinguished by the following features:

1. Use of logic synthesis to provide accurate design, with process independence for simulation, verification, and implementation.

2. Use of silicon compilers, translating a behavioral system specification directly to the structural description, provided for physical implementation of a design at the Register

Transfer Level (RTL) within a VHDL environment.

3. Incorporation of Design-for-Test (DfT) features, including serial scan, within the RTL descriptions.

4. Seamless integration of RTL and structural simulation, including Random Test Pattern Generation (RTPG) for validation of synthesis.

5. Use of a CAD software engineering environment to control the design process and maintain the design database.

One of the advantages of VHDL is that the design process is technology-independent. The designer builds the functional design and defers the technology-dependent details. Design decisions and architectural trade-offs can be made without considering the target technology. The power of logic synthesis allows a single VHDL design to be automatically targeted to various technologies.

Another advantage of high-level description design is that the functionality of the design can be easily modified. Using a bottom-up, gate-level design method, a change in a functional specification in the midst of the design process can cause tremendous delays. However, with a top-down design method, and the employment of VHDL synthesis, functional changes can be made rapidly and verified trough simulation, because the path to gates is automated. Thus, high-level synthesis produces architectures that enable logic synthesis. The designer can optimize in itarative way a variety of architectures and process technologies to find to optimal solution.

6.2. MAPPING OF DSP-ALGORITHMS

A design task for digital IC implementation is composed of many steps. First, an appropriate architecture must be selected. Then, the function blocks of this architecture should be designed. Finally, the DSP-algorithm is mapped and scheduled on the hardware. The largest portion of the design effort is the translation of system behavior into appropriate architecture, which is an interconnection of execution units, e.g. adders, multipliers, PLAs, RAMs, ROMs, and registers. Categorically, mapping can be split into general purpose architecture synthesis and dedicated silicon compilers.

General Purpose Architecture Synthesis

Commencing with a behavioral system specification, the synthesis generates a resource allocation and scheduling of function blocks on a high abstraction level. Subsequently, dedicated logic synthesis tools produce structured VHDL codes. This design package focuses on logic synthesis, state assignment, PLA synthesis, controller generation, ALU synthesis, timing and scheduling optimization of gate-level circuits, and technology mapping. They make silicon implementation transparent to the system designer, because they allow chip design without detailed silicon expertise.

Some commercial performances include:
- System Design Station, AutoLogic (Mentor Graphics)
- Synopsys
- Data Path & Chip Compiler, Asic Synthesizer (Compass)

Dedicated Silicon Compilers

The behavioral system specification is mapped on a dedicated target silicon compiler offering a fixed architecture [Gajs88, Gajs92, Park93, Kung85, Vanh93]. Structural VHDL codes are generated for logic synthesis. Compared to general purpose architecture synthesis, *dedicated silicon compilers* produce optimized resource shearing and scheduling, leading to select layouts that exploit three properties:

1. Addressing a specific application domain allows for integrate

and exploit domain expertise on all design levels.

2. Inside the selected application domain, an appropriate target architecture can be selected. The layout style can be adapted to the architecture and its requirements.

3. Each application-specific chip is designed for a particular task, thus avoiding the overhead inherent in programmable devices. Optimum area, speed, and power consumption solutions can be found.

6.2.1. Silicon Compilers

Currently, silicon compilers dedicated to architecture synthesis are popular research topics. As a result of research in these fields at various institutions, different solutions to the algorithm-to-architecture mapping problem have been proposed. Some are specifically for DSP algorithms while others are more general in scope. The more general systems are often too inefficient to provide satisfactory results for DSP-ASIC integration. Furthermore, the difficult task of scheduling respective data flow in DSP algorithms (multirate signal processing) renders some systems less suitable for DSP applications.

Dedicated silicon compilers for DSP applications are greatly desired and a large number of compiler tools have been proposed over the past ten years. Some of them include all phases, i.e. parts for algorithm mapping, architecture synthesis, realization, and implementation; others emphasize the realization phase. The following list of tools is neither complete nor extensive but it is comprised of tools that have influenced the work presented in this thesis.

Hard-wired Silicon Compiler for Inner Products

The first in a series of synthesis systems developed at IMT Uni NE, Switzerland, this silicon compiler was developed for high-speed computation of inner products [Sjös93]. State-space algorithms (i.e. inner products) can be mapped on that distributed arithmetic based architecture. This work was realized in a Compass CAD

environment and is restricted to VTI's cmn20, cmn16, and cmn12 CMOS technologies.

Low-power Micro-programmed Silicon Compiler

The dedicated architecture is a micro-programmed DSP where scheduling and resource allocation is especially adapted for minimum power consumption [Heub93]. Research for low-power, general-purpose DSP applications is performed at IMT Uni NE, Switzerland. This work was realized in a Compass CAD environment and is restricted to VTI's cmn20, cmn16, and cmn12 CMOS technologies and to ALP2 CMOS technology from Microelectronics Marin, Switzerland.

Cathedral I (Mistral 1™)

Catholic University of Leuven, at IMEC, Leuven, Belgium, developed four application-specific architecture synthesis methodologies with corresponding compilers [Vanh93]. Each compiler, namely *Cathedral I* through *IV*, addresses applications in a particular range of throughput requirements and system complexity. The compiler systems are built upon the SILAGE language (or Data Flow Language) used for both synthesis and simulation. The final DSP algorithm described by this language is mapped onto the architectures using strict mapping rules. In 1992, the first commercially available silicon compiler design environment for ASICs was introduced — EDC Mentor Graphics' Mistral 1™ inside DSP Station™.

The objectives behind *Cathedral I (Mistral 1™)* were to treat fixed coefficient DSP algorithms and implement them on an optimized bit-serial hard-wired architecture. Multiplication's are implemented by a shift-add and shift-subtract approach using canonical signed digit coded coefficients. Very few building blocks are required, and the methodology is impressively efficient from an area point of view. All of the main design steps are treated in this system. However, it only covers applications with a low to moderate speed and time-invariant algorithms. Typical sample rates reach 5 MHz. This is the silicon compiler that has influenced the implementation of the steady-state Kalman approach.

Cathedral II (Mistral 2™)

The *Cathedral II (Mistral 2™)* architecture is based on the classical Harward processor concept. It is a compiler with medium throughput requirements and a complex control flow. Typical applications require sample rates up to 2 MHz, covering the domain of complex audio and speech processing. Many data paths can be generated and different solutions can be compared. directly. The moderate throughput allows one to time-share operations for area efficiency reasons, controlled by a centralized micro-program. The system is commercialized by EDC Mentor Graphics under the name Mistral 2™ inside DSP Station™.

Cathedral III

Cathedral III was designed to meet the high throughput requirements of DSP applications with regular control flow. Typical sample rates of up to 50 MHz are achieved. This compiler addresses video and radar filtering and image processing. It uses a weakly programmable, pipelined, bit-parallel architecture, with local control decision making. The approach is, in principle, limited to fixed-point arithmetic DSP algorithms. Parts of the system are commercialized by EDC Mentor Graphics and integrated with Mistral 2™ and DSP Station™.

Cathedral IV

Cathedral IV is a compiler for highly pipelined, regular DSP algorithms. It is aimed at front-end image, video, and radar applications. The system is used by Philips but is not commercially available at the moment.

AMICAL

This silicon compiler has been developed in the research laboratories of Techniques of Information and Microelectronics for Computer Architectures (TIMA), Grenoble, France [Park93]. AMICAL is an assistant for architectural compilation of control dominated circuits. It is organized as an architectural synthesis environment, allowing the combination of automated, manual, and interactive synthesis. AMICAL performs both scheduling and

allocation. It starts from a behavioral VHDL description and generates the detailed architecture of a circuit in VHDL. The output can feed existing logic synthesis tools acting at the logic and register transfer level. The compiler is built upon a dedicated library. It contains standard operators (addition, subtraction, etc.) as well as complex functional units such as memories, cache memories, input/output units, etc. The association between function units and operations is made trough names. The operations of these function units are referred through VHDL functions and procedures. In this library, the description of each function unit contains some physical parameters and the I/O ports. Each operation is broken down into basic transfers (clock cycles). One of the functional advantages of AMICAL is the fact that the number of cycles necessary for the execution of a function unit is not limited in any way.

6.3. MAPPING ON MISTRAL 1 SILICON COMPILER

6.3.1. Basic Concept

Mapping is the translation of the signal flow graph (e.g. Figure 6.6) onto a target architecture. Bit-serial implementation is an architectural strategy for implementing an algorithm on a piece of hardware, where all bits of the a signal word are processed in consecutive clock periods using the same piece. Bit-serial implementation is slower than bit-parallel implementation, but the hardware is relatively cheap. Each operation of the signal flow graph is mapped onto a separate, bit-serial hardware operator. Because bit-serial operators are small, area-efficient solutions are obtained. With bit-parallel architectures, all signal bits are processed in the same or few (pipelined) clock periods using different pieces of hardware.

Mistral 1™ is a silicon compiler that generates an optimum hard-wired bit-serial DSP architecture. The DSP algorithm mapping is based on twenty-nine primitive operators, described in a structural VHDL code for further logic synthesis.

Multiplication's are implemented by a shift-add and shift-subtract approach using Canonical Signed Digit (CSD) coded coefficients (Figure 6.1). It is a base-2 numeration system which makes that a constant can only have one unique representation. To ensure that the representation is unique, the product of the two leftmost digits must not equal one (1). The nature of CSD representation is such that the number of non-zero digits is minimized and there exist no adjacent non-zero digits. A CSD format, indicated by a "T" is a series of one or more ternary digits (0, 1, or -1) optionally containing one binary point.

Example: 0.011111 *becomes* 0.100000 T, *in CSD format.*

If there is one constant input specified by the CSD format, the multiplication operation is mapped onto a nested multiplier. If that is not the case, a data-by-data multiplier is utilized. A nested multiplier consists of a chain of adders or subtractors, and

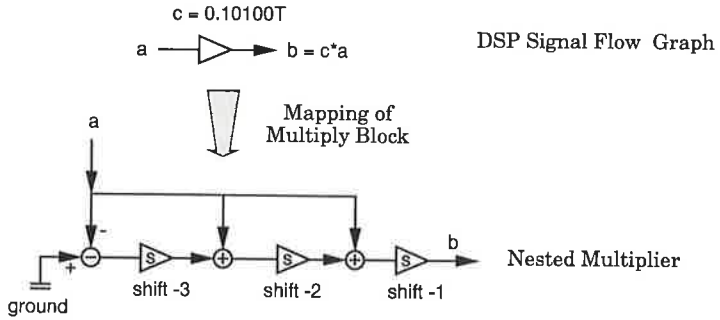


Figure 6.1. *Mapping of multiplier operation: Nested multiplier.*

scalers (see Figure 6.1). For each non-zero bit in the coefficient there is an adder or a subtractor. A one (1) gives rise to an adder; a T to a subtractor. A scaler is added with a shift value equal to the distance between the mapped non-zero bit and the next non-zero bit, starting from the LSB position. The position of the MSB non-zero bit determines the shift value of the scaler at the end of the nested multiplier. It is important to minimize the number of non-zero bits in the coefficients.

The selected operators and the expected scheduling depends on the number of one's (1's) present in the coefficient wordlength. Optimization of the coefficients with the goal function of a minimum number of one's (1's) achieves a reduced amount of operators.

The example used in Figure 6.2 is a bit-serial implementation of the sum of two signals, A and B: $C = fix < 6,4 > (A + B)$.

The hardware illustrated in Figure 6.2 is reduced to a full adder and two flip-flops, denoted by FF. The bits are processed one-by-one, beginning on the LSB side of the input words. Only one full adder and two flip-flops are needed. The entire operation is repeated six times to determine the output word.

Using that basic concept, a DSP signal flow graph can be mapped by replacing each operation in the signal flow graph with its corresponding hardware operator. Timing constraints of delayed signals may occur when signals have to be synchronized inside the

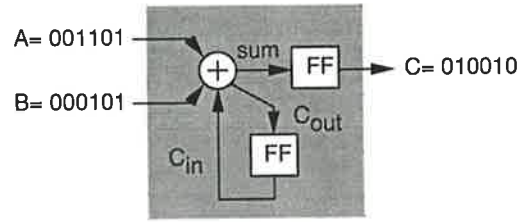


Figure 6.2. *Bit-serial implementation of a sum.*

signal flow graph. A delay graph is extracted to manage these delays. After the mapping process has been completed, the internal and external word-level controllers are processed in order to obtain a circuit with superior results, following delay management. The controller produces the necessary control signals, ensuring that the data path works correctly. The control circuit consists of a bit-level controller and an optional word-level controller, added for time-dependent flow graphs, that is, for circuits which have been hardware multiplexed and/or converted from multirate to monorate.

6.3.2. Design Flow for Mistral 1™

Architecture synthesis begins with an algorithm (signal flow graph) described by DFL language. The synthesis is performed through four mapping levels:

Mapping Level I

The Optimized Signal Flow Graph (OSFG) validates the algorithm written in DFL. This step establishes that the circuit can be implemented.

Mapping Level II

The conversion from OSFG to ESFG (Expanded Signal Flow Graph) implies the mapping of complex operations onto simpler elements. Multipliers are mapped to a network of adders, scalers, and subtractors. Overflow and quantization characteristics are expanded, too. It may be necessary to shift inverters through the

network. Inversion management reduces the number of inverters in a circuit and minimizes the chip area. Inverters may be eliminated by combining adders and sign inverters into subtractors. The ESFG remains an implementation-independent representation of the original signal flow graph.

Mapping Level III

Conversion from ESFG to AFG (Architecture Flow Graph) transforms the implementation-independent circuit into a bit-serial architecture. The operations are now defined in terms of bit-serial primitives. The primitives are read from the cell library, which contains only 29 primitives in VHDL for further logic synthesis and architecture simulation. Each of these primitives is described at a very low level that means on a detailed logical description. For simulation and implementation, the structural description can be interfaced to IC design tools.

To achieve the algorithmic behavior of the operations, delay management of the internal signals is applied. The final scheduling is performed by a bit-level controller, a chain of shift registers. The addition of multiplexing hardware, and a word-level controller to drive these multiplexer operations, enables multirate digital algorithms to be translated into single rate structures. This way those algorithms can be mapped too.

Mapping Level IV

The Cell Flow Graph (CFG) level generates a scan path, an extra path connecting all operations with a built-in memory function. It is used to test the circuit together with test vectors. For further implementations, EDIF, VHDL, and GN netlist are written.

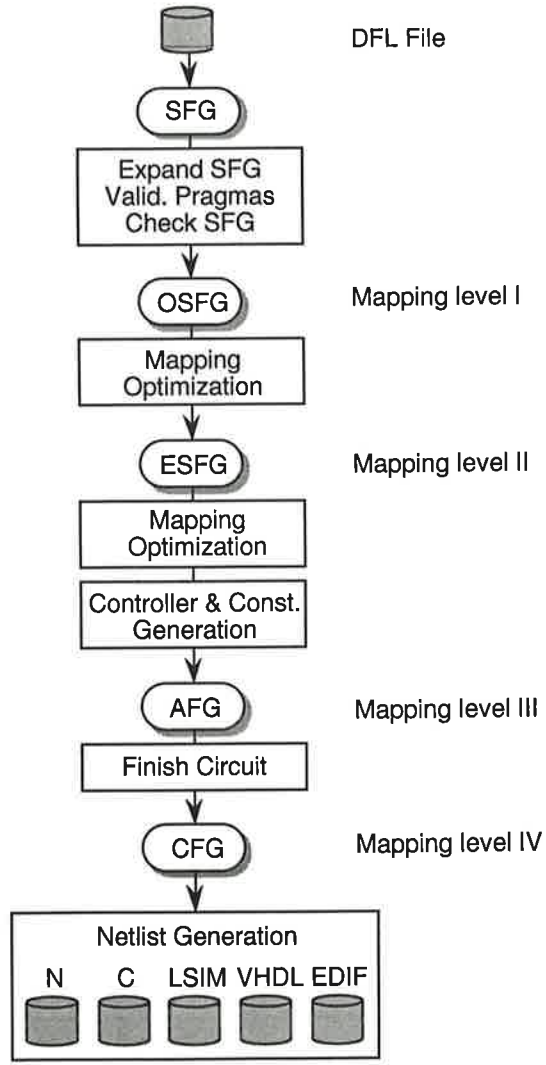


Figure 6.3. *Design flow: Mistral 1™ architecture.*

6.3.3. Verification and Simulation

Functional simulation must be carefully executed before the physical implementation. To simplify a bit-serial simulation of the design, additional VHDL models are employed. The models are:

1. A controller generating the clock and the bit rate. The controller induces the bit-level controller pulses as well as the clock of the Mistral 1 design.

2. A parallel-to-serial converter for interfacing the input signals to the design under test, as shown in Figure 6.4. The parallel-to-serial converters load the word that is present at the input, when the start input is high and the clock is active (positive edge). The word is then serially shifted through the output of the parallel-to-serial converter, from the LSB to the Mistral 1™ design. After loading in the value, the LSB of the input is immediately available at the output and can be read.

3. A serial-to-parallel converter for the output signals. The serial-to-parallel converter shifts the bits present at the output of the Mistral 1™ design and discharges the stored result to the output, when the start pulse is high and the clock is active (positive edge). The timing is shown in Figure 6.5.

The input word "input" is processed in w clock cycles. Scheduling is performed by the controller signals that generate the Mistral 1™ bit-level controller, "blc_in", and the test control signal, "tc". An external view of functional verification of the timing of the control signals in the top level VHDL description is depicted in Figure 6.5 . The clock generated by the controller originates with a value of zero (0). The reset pulse must include at least one active edge of the clock, that is, the positive edge. On the first active edge of the clock that the reset signal is low, the start signal goes high again every wordlength number of clock cycles. The parallel-to-serial converter reads a new value from its inputs and then produces the LSB of "ms1_IN" immediately, when both the start pulse and clock are active. This is the case before the falling edge of start. The serial-to-parallel converter outputs the value stored in its internal registers at the same time. Next, it reads in the bit value present at its input.

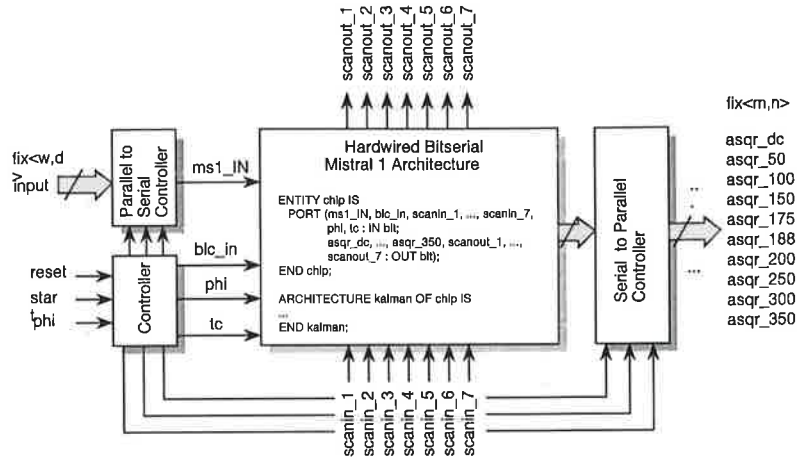


Figure 6.4. *Functional verification using top-level VHDL interface.*

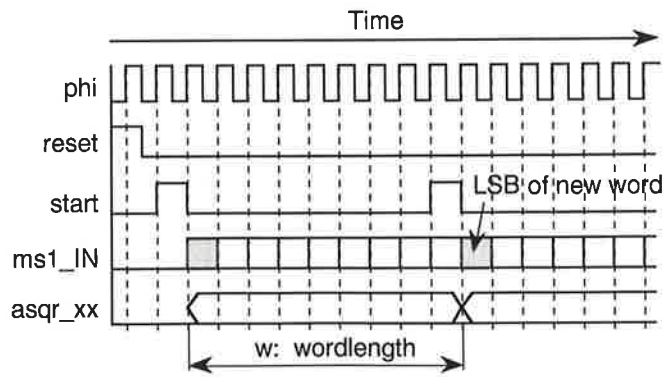


Figure 6.5. *Timing of top-level VHDL control signals.*

6.4. DESIGN OF SPECTRAL ANALYZER ASICS

As Kalman filtering theory has entered many fields over the past ten years, emphasis has been put on VLSI implementation of various Kalman filter algorithms. If interested, one can find publications presenting different implementation strategies and design methodologies in [Azim91, Chen87, Falc78, Gast88, Gent81, Hen86, Jove86, Kung85, Kung87, Kung91, Lee88, Rao90, Seitz85, Shpa89, Sung87, Yeh88, Zhan90].

In Chapter 5, the software implementation of a Digital Power Network Analysis System was demonstrated. The discussion now focuses on the design and implementation of spectral analysis based on a steady-state Oscillator Signal Model.

The same specifications have been assigned for a bit-serial Mistral 1 ASIC realization. The selected design parameter for the steady-state Oscillator Signal Model is 0.1; the sampling frequency is 3,200 Hz; input and output wordlength are fixed at 16-bit, offering true 10-bit resolution. Four bits are lost due to quantization inside the algorithm. The maximum signal-to-noise ratio that is achieved is 60 dB. Input and output signals are represented in signed fractional two's-complement notation. The signal word type is *fix*<16,15>.

The magnitude, $\hat{A}_n(k)$, of spectral line n, is the square-root of the squared real and imaginary parts of the estimated Fourier coefficients, $\hat{a}_n(k)$ and $\hat{b}_n(k)$.

$$\hat{A}_n(k) = \sqrt{\hat{a}_n^{\,2}(k) + \hat{b}_n^{\,2}(k)} \qquad (6.1)$$

The estimated Fourier coefficients are obtained by multiplying the estimated state variables, $\underline{\hat{x}}(k)$, by the output coupling matrix, $\mathbf{H}(k)$ (refer to Chapter 3).

$$\underline{\hat{z}}(k) = \mathbf{H}(k)\underline{\hat{x}}(k)$$
$$(\ e.g.\ :\ \underline{\hat{z}}(k)^T = [\underline{\hat{a}}(k), \underline{\hat{b}}(k)]^T\ ) \qquad (6.2)$$

Because $\mathbf{H}(k)$ is only a matrix rotation, the calculation of the magnitude of a frequency line becomes independent of $\mathbf{H}(k)$.

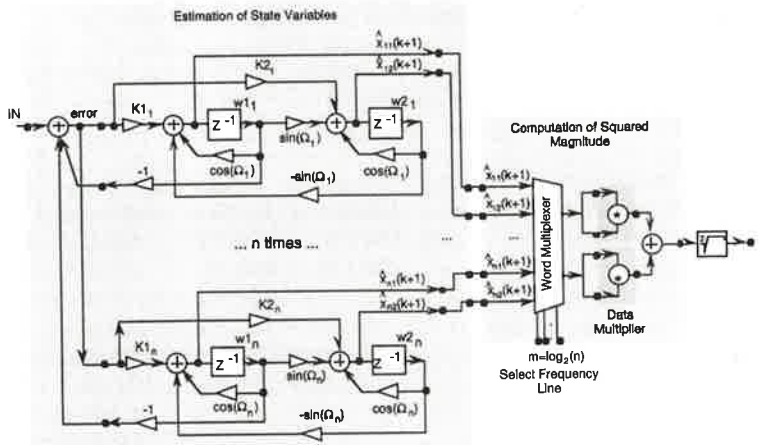


Figure 6.6. *Spectral analysis of signal flow graph of modified steady-state OSM: Magnitude estimation, $\hat{A}_n(k)$, of n frequency lines.*

In fact, this computational savings also reduces the memory requirements for the look-up tables, which are required in order to save the sine and cosine tables present inside $\mathbf{H}(k)$ (see Equation 3.12). Equation 6.1 now becomes

$$\hat{A}_n(k) = \sqrt{\hat{x}_{n1}(k)^2 + \hat{x}_{n2}(k)^2} \qquad (6.3)$$

Figure 6.6 presents the signal flow graph of the modified steady-state OSM. The memory occupation is greatly reduced with the algorithm because any sine and cosine tables present inside the output coupling matrix have to be stored in memory (see Table 4.2). The corresponding DFL file is listed in Appendix A at the end of this chapter.

The balance of this chapter is restricted to the design and implementation of the modified steady-state OSM approach. In particular, power spectral analysis of a chosen set of frequencies is performed (shown in Table 6.1). The first set of frequencies calculates the squared magnitudes of 2 frequency lines; the second set calculates 4 lines; the third 6 lines, and the fourth 10.

Table 6.1. *Power Spectral Analyzer Chips: Four Sets of Frequencies Calculating State Variables.*

q/r= 1E-1	Chip 1 2 Lines	Chip 2 4 Lines	Chip 3 6 Lines	Chip 4 10 Lines
Estimation of state variables	50 Hz 150 Hz	50 Hz 100 Hz 150 Hz 200 Hz	50 Hz 100 Hz 150 Hz 200 Hz 250 Hz 300 Hz	0 Hz 50 Hz 100 Hz 150 Hz 175 Hz 188 Hz 200 Hz 250 Hz 300 Hz 350 Hz

The estimation is limited to the calculation of the squared magnitude, because Mistral 1 architecture is a hard-wired implementation strategy, meaning that it cannot compute a square-root. The squared magnitudes present the spectral power of the input signal.

Each set is mapped onto the Mistral 1™ architecture. Four integrated circuits (chips) are designed for the calculation of the state variables, shown on the left side of Figure 6.6. Each frequency line is a precise copy of each other line, but with differing coefficients. As has already been shown in Chapter 4, the number of arithmetic operations is linearly dependent upon the number of frequency lines. As ASIC implementation based on that hard-wired DSP architecture is concerned, the required amount of hardware is also theoretically linear dependent on the number of frequency lines.

The square function appears for every frequency line, and thus each can be realized separately. This is also shown on the left side of Figure 6.6. The selection of a specific frequency line is achieved by choosing the state variables, $\hat{x}_{n1}(k+1)$ and $\hat{x}_{n2}(k+1)$, with the word de-multiplexer. The number of select signals, $m = \log_2(M)$, of the word de-multiplexer depends on the base 2 logarithm of the number of estimated frequencies, M. Separate placement of the square function reduces silicon area on the coast of time multiplexing the output results.

6.5. TOP-DOWN ASIC DESIGN WITH LOGIC SYNTHESIS

With the recent trend towards the integration of Microsystems comes the design of new electrical, mechanical, optical, and biomedical sensors. In addition, a variety actuators are being developed—electromagnetic and electrostatic motors, hydraulic and pneumatic, with power electronics interfaces. Already frequently used, sensors will become even more common as soon as manufacturing techniques correspond with those allowing the manufacture of microelectronics. Not only will manufacturing benefit from downsizing, integration, and mass production, but it will become possible to integrate the sensors and actuators with computing. Because of the vast range of application possibilities (automotive, home electronics, medicine, biomedical environment, food industry), Microsystems will develop rapidly in the near future[Cour93].

As the application specific integrated circuit design process becomes more complex due to increased design complexity and shorter design cycles, logic synthesis and optimization tools will become vital to the design process. These tools help shorten the time between design and production, thus offering a reliable method of meeting the demands of the marketplace.

6.5.1. Design Methodology

Mapping of DSP Algorithms

Figure 6.7 is a flowchart illustrating where logic synthesis and optimization occur in the design process. It is assumed that the behavior for a fixed-point implementation is analyzed and optimized before algorithm-to-DSP architecture mapping is performed. The *algorithm optimizations* have been performed using Matlab (Mathworks) and DSPlab (DSP Station, Mentor Graphics) CAD-tolls.

The output description of the Mistral 1™ *silicon compiler* is a detailed, structured VHDL that is well-suited for logic synthesis.

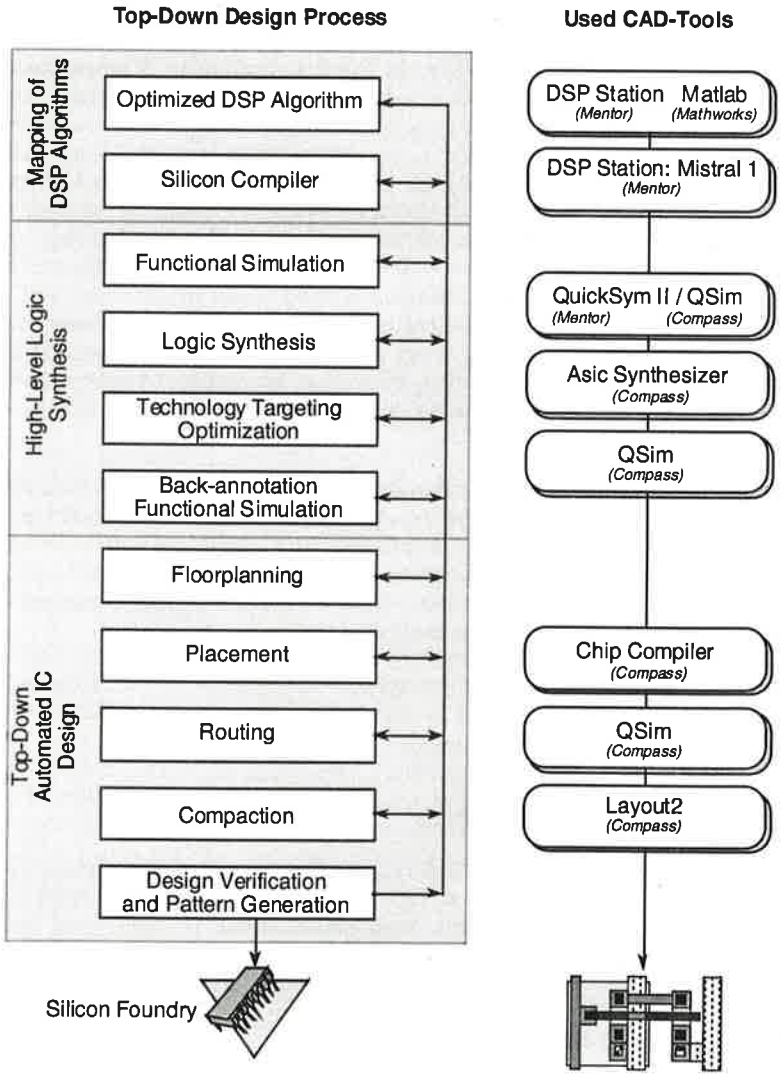


Figure 6.7. *Top-Down Design Process with Logic Synthesis. Left: Design process. Right: Design flow inside CAD environment.*

High-Level Logic Synthesis

Today, designers can choose from a variety of silicon foundries proposing specific technologies, such as ECL, TTL, Bipolar, PMOS, NMOS, CMOS, and GaAs. Various processes offer power supplies ranging from 1 Volt to 30-50 Volts. Furthermore, the trend towards deep sub-micron technologies (0.2 µm) offers the opportunity to integrate many millions of transistors on a single IC. It has thus become common to integrate entire systems.

For each technology, dedicated layout techniques have been developed to enhance the design process and to simplify automated IC design. Such techniques are based on gate arrays, or sea-of-gates, where only the last few mask layouts have to be generated. This pre-diffused approach is cost efficient and preferred for high volume productions.

Another layout technique is the so-called semi-custom design approach. It is based on standard cell libraries: Depending on the desired application field, dedicated libraries are provided for area or speed optimized designs. Recently, libraries have appeared on the market for low-voltage and low-power applications used in the field of portable products [CSEM92].

Of the technologies and layout techniques that have been introduced thus far, only CMOS processes and standard cell techniques will be discussed further. Information about IC design, technological trends, and CAD environments can be found in [Cour93].

The IC design flow starts with structural VHDL description of the DSP architecture that is interfaced to logic synthesis. This is the step in the design process where the design is implemented at the gate level and optimized in the desired technology to meet the specific needs.

Logic synthesis has been performed using Asic Synthesizer (Compass). The selected technology (VLSI Technology Inc.) is: CMOS 0.8 µm, double metal, single poly. The layout technique is based on a vsc450l standard cell library. It is a medium-voltage (2.7 V < Power Supply < 3.6 V), high performance standard cell library with a clock edge switching time from VDD to VSS and VSS to VDD in 0.5 ns.

Top-Down Automated IC Design

Floorplanning is the process of estimating the chip area that will be used for each standard cell or block in the design. A floorplan is a topological structure comprised of rows and shapes and used as guides for arranging cells. Layout partitioning is important so that one can be sure that the layout will fit into a given area before placement and routing of cells is started.

Placement is the process of installing standard cells or blocks into the floorplan. *Routing* is the process of placing and connecting signal and power paths between the standard cells and blocks.

Compaction is the process of minimizing the size of a completed layout. Compaction can be done for either the entire layout or for a specific area. Freedom to compact in several directions and towards any point that have been chosen are proposed.

A Chip Compiler (Compass) that performs both automatic and interactive floorplanning, placement and routing, and compaction of a design has been chosen for the organization of the modified steady-state OSM spectral analyzer chips.

The schematic netlist that is generated is interfaced to the *layout editor*, Layout2 (Compass). Layout2 is a graphical tool used for manual cell design or chip assembly. It is equipped to perform complete chip layout tasks. Design Rule Check (DRC) is conducted to detect any layout rule violations. The final layout generates *.cif* or GDSII files that describe the mask specific data. The silicon foundry obtains the *.cif* file for masks and silicon production. The results presented here is culminates with the generation of the layout, the mask data, and the functional verification. Due to time and money constraints, the silicon production phase has been foregone.

6.6. IMPLEMENTATION RESULTS

In Section 6.4, algorithmical aspects for optimal implementation results were introduced. Reviewing Figure 6.6, the left block estimates the state variables while the right block calculates the squared magnitude of a single frequency line. To select the desired frequency line, a de-multiplexer has been added to the right block. The Mistral 1™ implementation maps the left block onto a set of four chips, each computing a specific number of frequency lines. The kernel of the right block contains a data-by-data multiplier which has been mapped onto a fifth chip. The five chips have been designed according to the process shown in Figure 6.7 (CAD-tools are indicated on the right side of the figure). The layouts are then presented in Figure 6.8. Detailed layout photographs can be found in Appendix B at the end of this chapter.

The vsc450l standard cell library has been chosen from VLSI Technology Incorporation, USA. The technology is CMOS 0.8 µm, working at a medium power supply voltage (2.7 V < Vdd < 3.6 V). The results are summarized in Table 6.2.

The implementation of the Kalman approach does verify predictions made in Chapter 4. Particularly interesting results obtained with the modified steady-state OSM are:

1. Top-down design process could be applied in a heterogeneous CAD environment. Thus, VHDL described designs are easily transferred from one CAD environment to another, allowing for shorter design time and efficient standard cell layouts.

2. The modified steady-state OSM Kalman Filter applied for power spectral analysis offers uncomplicated arithmetic, leading to efficient ASIC integrations.

3. The Mistral 1™ chip clock rate is theoretically independent of the number of frequency lines. The chip clock rate depends on the signal wordlength and the CSD coded coefficient wordlength. For 16-bit input and output wordlength, the chip clock frequency is 32 times the sampling frequency. Thus, real-time power spectral analysis can be performed at very high sampling rates (e.g. vsc450l, t_{max}=100 ns, max. sampling rate ≤ 310 kHz).

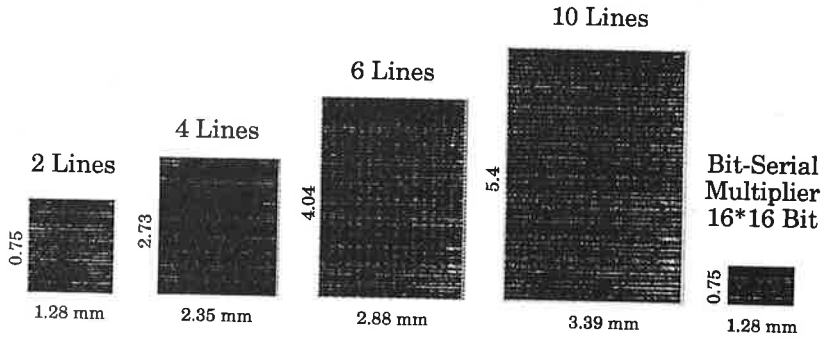


Figure 6.8. *Power spectral analyzer ASICs, modified steady-state OSM: 2 lines, 4 lines, 6 lines, 10 lines; 16-bit data-by-data multiplier; standard cell vsc450l, VLSI CMOS cmn08 technology.*

4. The theoretical prediction stated above (3.) is not true in practice. The maximum sampling rate is limited by the standard cell library, the place and route efficiency, and by the technology. Results have shown that during the mapping procedure delay management becomes more difficult as the number of frequency lines increases. The delay management optimization routing increases the internal signal wordlength (number of clock cycles needed to process an input sample). Theoretically, the maximum internal signal wordlength (sum of the input wordlength and the coefficient wordlength) increased when the number of lines was greater than or equal to ten. The bit synchronization of the error signal inside the frequency estimation blocks (shown in Figure 6.6) became difficult and thus caused the increased internal signal wordlength.

5. The number of standard cells (transistors) depends theoretically linear on the number of frequency lines. Upon examination of the achieved implementation results, one can see that the number of standard cells (transistors) increases exponentially. The reason for this is seen in the signal flow graph (Figure 6.6)—the error signal is distributed to each frequency estimation block and this specific signal feeds many inputs.

Table 6.2. *Power Spectral Analyzer ASICs, Steady-State Oscillator Signal Model: Standard Cell vsc450, VLSI cmn08 Technology, CMOS 0.8 µm, 2 Metal, 1 Poly-silicon, 3.3 Volt Power Supply.*

	Chip 1 2 Lines	Chip 2 4 Lines	Chip 3 6 Lines	Chip 4 10 Lines	Chip 5 16*16 bit multiply
Core Size [mm]	1.67*1.89	2.35*2.73	2.88*4.04	3.39*5.4	1.28*0.75
Area [mm^2]	3.14	6.44	11.67	18.34	0.97
Standard Cells	1453	2739	4285	6753	518
Transistors	24617	44364	69078	109461	8337
Density [tr./mm^2]	7755	6880	5915	5965	8594
Power estimation at 3.3 V [mW/MHz]	5.1	10.4	18.1	29.4	1.5

The silicon consequence is that this error signal must be pre-buffered using pre-buffering strategy because one output of a standard cell can only drive a certain number of input cells. During logic synthesis, a relatively high clock frequency, 15 MHz, was chosen. The effect of pre-buffering can be seen by examining Figures 6.10 and 6.11. The purpose is an increased core size, shown in Figure 6.9.

6. Theoretically, the layout density (transistors/mm^2) should be constant for all four chips. In reality, however, while the number of standard cells increases, the density decreases. This effect, due to the place and rout optimization process inside the Chip Compiler, can be studied in Figure 6.12. Routing becomes more difficult because rows have to be crossed, via feed-trues, in standard cells. But, the number of feed-trues in each row is limited. And furthermore, the length of rows that imposes to enlarge the routing channel (see layout in Appendix B at the end of this chapter).

Table 6.3. *Benchmarking: Motorola versus ASIC. The estimated results are based on Tables 5.2 and 6.2; the extrapolation is linear for the ASIC approach. (*) These values are not feasible in practice.*

	Motorola DSP 56002 steady-state OSM 24*24 bit arithmetic	Mistral 1 ASIC mod. steady-state OSM 24*24 bit arithmetic
Technology	0.65 µm CMOS full custom	0.8 µm CMOS standard cells VLSI vsc450l
Power supply voltage	5 V (3.3 V mid '95)	2.7 V - 3.6 V
max. chip clock	66 MHz (30 ns instr. cycle)	15 MHz
max. sampling rate	[kHz]	is almost constant because it is fixed by 24*24 bit arithmetic (48 cycles)
32 lines	57.0	
64 lines	28.0	
128 lines	14.0	
256 lines	7.0	
512 lines	3.6	208 kHz
1024 lines	1.8	
Power consumption	is almost constant	[mW/MHz]
32 lines		96
64 lines	≈ 150 mA at 5 V	192
128 lines	750 mW at 66 MHz	384
256 lines		768
512 lines		1536
1024 lines		3072
Chip area	no indications from the vendor	[mm²]
32 lines		58
64 lines		117
128 lines		234 (*)
256 lines		470 (*)
512 lines		939 (*)
1024 lines		1878 (*)

Because the error signal must be passed to each frequency estimation block, routing becomes more difficult as the number of lines increases.

Table 6.4. *Benchmarking: Motorola Throughput Fixed at 100%. Mistral 1 ASIC's are adapted to the same throughput; the extrapolation is linear for the Mistral 1 ASIC approach.*

	Motorola DSP 56002 steady-state OSM 24*24 bit arithmetic	Mistral 1 ASIC mod. steady-state OSM 24*24 bit arithmetic
32 lines	57.0 kHz 750 mW	2.7 MHz 260 mW
64 lines	28.0 kHz "	1.3 MHz "
128 lines	14.0 kHz "	672.0 kHz "
256 lines	7.0 kHz "	336.0 kHz "
512 lines	3.6 kHz "	173.0 kHz "
1024 lines	1.8 kHz "	87.0 kHz "

7. Benchmarking of programmable DSPs versus ASICs. The discussions related to ASIC integration of the modified steady-state OSM Kalman approach have shown that all parameters exhibit an exponential dependency. In order to simplify the benchmarking of programmable signal processors versus Mistral 1 ASIC approaches, a linear extrapolation has been chosen. The results are presented in Table 6.3. The inexact values are always too low for Mistral 1 ASIC approaches, but give an approximation of what results could be expected.

   Today's DSPs are implemented on high quality technologies offering very low power, high speed, and small size. In contrast to Mistral 1 ASICs, the chip clock speed of DSPs is certainly optimized for speed (full custom layout technique).

   The maximum possible sampling rate is much higher for Mistral 1™ ASICs. This advantage is obtained with hard-wired bit-serial implementation strategy. The relatively low chip clock speed is more than compensated for. Mistral 1™ ASICs perform power spectrum estimation theoretically independent of the number of frequency lines and achieves more than 200 kHz sampling rate compared to poor 1.8 kHz for 1024 frequency lines on the DSP56002.

The estimations of power consumption expose very interesting trends, presented in Tables 6.2, 6.3, and 6.4, and in Figure 6.13. The new Motorola chip, the DSP56002, runs at 40 to 66 MHz (50 to 30 ns instruction cycle time) and consumes, at 5 V, approximately 750 mW at 66 MHz. In Table 6.3, the Mistral 1™ ASICs run at the maximum possible chip clock speed. One can see that consumption is clearly less for low to moderate number of frequencies. Break-even is achieved for 256 frequency lines.

The results of a comparison that utilizes the same throughput for both demonstrates that the Mistral 1™ ASICs consume one-third of power of the Motorola DSP. The power consumption is constant for both and independent of the number of frequencies.

Regard the Mistral 1™ ASICs, the explanation is that the chip clock speed for bit-serial hard-wired Mistral 1™ architectures is independent of the number of frequencies and thus can be lowered and adjusted to the sampling rate. The number of standard cells, the area, and consequently the capacitance, C, are increasing. The power consumption, P, is linearly related to the chip clock frequency, f, and is exponentially related to the power supply voltage ,V, that is fixed for the applied technology.

$$P = C * f * V^2 \qquad (6.4)$$

Finally, estimations of the chip area highlight the main drawback of hard-wired bit-serial implementations. The signal flow graph is mapped directly onto hardware. As the signal flow graph becomes more complex, the layout becomes very large. Chip area greater than 200 mm^2 is expensive and difficult to integrate. Applications that require more than 64 lines have to be time-multiplexed.

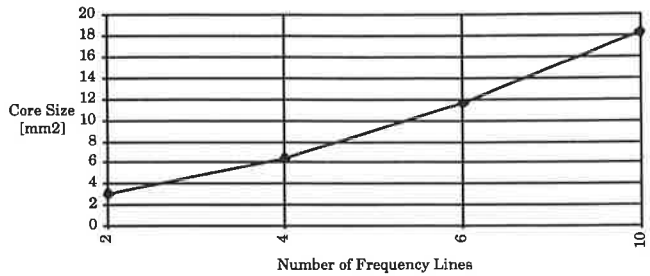


Figure 6.9. *Core Size: Graphical representation of Table 6.2, row 2.*

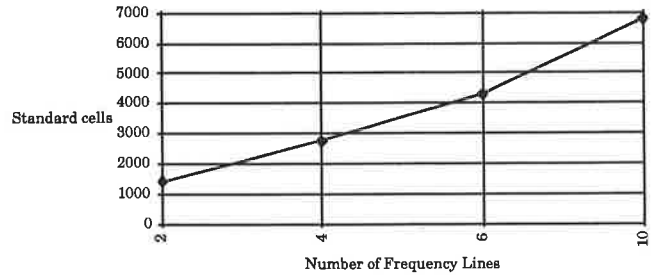


Figure 6.10. *Number of Standard Cells: Graphical representation of Table 6.2, row 3.*

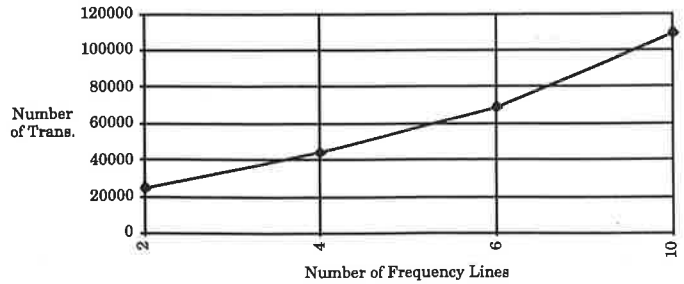


Figure 6.11. *Number of Transistors: Graphical representation of Table 6.2, row 4.*

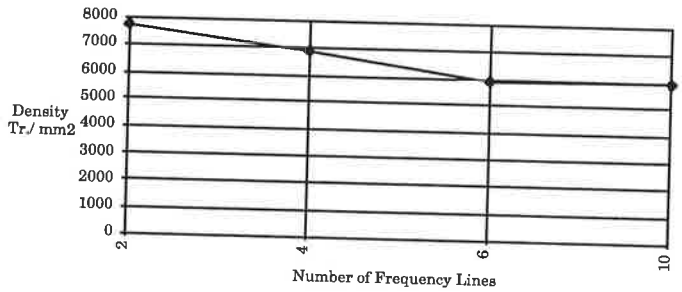


Figure 6.12. *Density: Graphical representation of Table 6.2, row 5.*

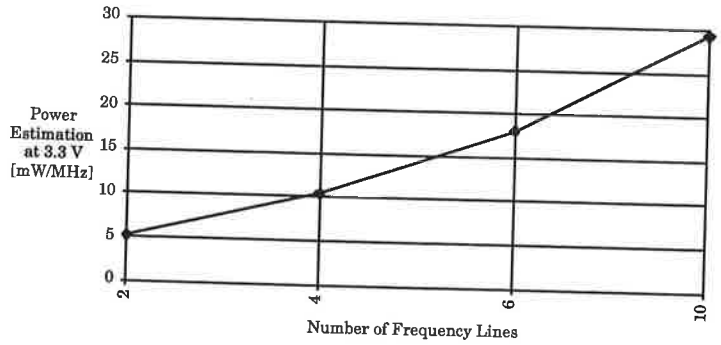


Figure 6.13. *Power estimation: Graphical representation of Table 6.2, row 6.*

6.7. SUMMARY

This chapter explored the ASIC design results. Its design methodology is based on silicon compilers and on VHDL logic synthesis. Algorithm-to-architecture mapping was introduced, mapping alternatives were described, and the design strategy of silicon compilers was exposed. The target silicon compiler, Mistral 1™, was chosen for bit-serial hard-wired DSP-architecture synthesis. Finally, design flow road mapping was discussed.

The steady-state OSM algorithm, selected for silicon implementation, has been applied to spectral power estimation. A modified steady-state OSM signal flow graph that reduces the number of arithmetic operations and offers very few memory requirements was discovered.

The modified steady-state OSM is used for four integrated circuits that have been designed to the layout level. The modified steady-state OSM Kalman filter, applied for power spectral analysis, offers straightforward arithmetic leading to efficient ASIC integrations.

The Mistral 1™ chip clock rate depends on the signal wordlength and the CSD coded coefficient wordlength. Thus, real-time power spectral analysis can be performed at very high sampling rates. Results have shown that, in practice, the maximum sampling rate is limited by the standard cell library, the place and route efficiency, and by the technology.

The implementation has shown an exponential dependency of the implementation complexity. The reason can be found in the error signal that is distributed to each frequency estimation block. This error signal must be pre-buffered using pre-buffering strategy. Furthermore, routing becomes more difficult, because rows have to be crossed, via feed-trues, in standard cells.

The benchmarking of programmable DSPs versus ASICs has been investigated. In sum, the maximum achievable sampling rate is much higher for Mistral 1™ ASICs. This advantage is obtained through the use of hard-wired bit-serial implementation strategy. Special attention has been given to the power consumption. A comparison that holds the throughput constant clearly favors the Mistral 1™ ASIC's, consuming much less power. The power savings can be explained by the fact that the chip clock speed for bit-serial hard-wired Mistral 1™ architectures are unaffected by the number of frequencies and thus can be lowered and adjusted to the sampling rate.

Finally, estimations of the chip area accent the drawback of hard-wired bit-serial implementations. The signal flow graph is mapped straight forward into hardware. The result is a increasing chip area with increasing complexity of the algorithm. Applications that require more than 64 lines have to be time-multiplexed.

Appendix A

DFL File: Spectral Analyzer ASICs

```
/*********************************************************/
/* Steady-state oscillator signal model for
   Kalman Fourier Coefficient Estimator*/
/*********************************************************/
/* Copy Right:          Peter Balsiger
                        IMT Uni NE
                        Tivoli 28
                        CH-2003 Neuchâtel-Serrières, Switzerland
                        Phone:  ++41 38 30 16 33
                        Fax:    ++41 38 30 18 45
                        EMail:  peter.balsiger@imt.unine.ch

   Date:                December 12, 1994
   Last Modifications:  December 24, 1994
   Filename:            spec_analysis.dfl
   DSP Station Version: 8.4_2

   Sampling rate:       3200 Hz
   Number of Lines:     10
   Estimated Frequencies:
   DC, 50, 100, 150, 175, 188, 200, 250, 300, 350 Hz
   Design Parameter q/r:    0.1                    */
/*********************************************************/

#include <dsp_base_lib.hdfl>
#include <dsp_func_lib.hdfl>

/* signal & coefficient wordlength */
#define IW fix<16,15>
#define CW fix<16,15>
#define SW fix<16,15>
#define AW fix<16,15>
#define OW fix<16,15>
#define PW fix<16,15>

/* coefficients for q/r= 0.1 */
#define   kal1_0      CW( 0.14477037289943 )
#define   cos_50      CW( 0.99518472667220 )
#define   sin_50      CW( 0.09801714032956 )
#define   kal1_50     CW( 0.14080423564252 )
#define   kal2_50     CW( -0.03365454047474 )
#define   cos_100     CW( 0.98078528040323 )
#define   sin_100     CW( 0.19509032201613 )
```



```
#define   kal1_100    CW( 0.14109885531625 )
#define   kal2_100    CW( -0.03239712792647 )

#define   cos_150     CW( 0.95694033573221 )
#define   sin_150     CW( 0.29028467725446 )
#define   kal1_150    CW( 0.14477036922300 )
#define   kal2_150    CW( 0.00003262629215 )

#define   cos_175     CW( 0.94154406518302 )
#define   sin_175     CW( 0.33688985339222 )
#define   kal1_175    CW( 0.14078399536883 )
#define   kal2_175    CW ( -0.03373910961227 )

#define   cos_188     CW( 0.93263902314309 )
#define   sin_188     CW( 0.36081082648764 )
#define   kal1_188    CW( 0.09263766727536 )
#define   kal2_188    CW( -0.11125072346381 )

#define   cos_200     CW( 0.92387953251129 )
#define   sin_200     CW( 0.38268343236509 )
#define   kal1_200    CW( -0.01119036586004 )
#define   kal2_200    CW( -0.14433723213834 )

#define   cos_250     CW( 0.88192126434836 )
#define   sin_250     CW( 0.47139673682600 )
#define   kal1_250    CW( 0.04207775782813 )
#define   kal2_250    CW( -0.13852047922815 )

#define   cos_300     CW( 0.83146961230255 )
#define   sin_300     CW( 0.55557023301960 )
#define   kal1_300    CW( 0.01361107928693)
#define   kal2_300    CW( -0.14412910667206)

#define   cos_350     CW ( 0.77301045336274 )
#define   sin_350     CW ( 0.63439328416365 )
#define   kal1_350    CW ( -0.05894857567481)
#define   kal2_350    CW ( -0.13222528614207 )

/* main function that is used for mistral1 synthesis */
func osc_syn (IN:IW) amp_dc,amp_50,amp_100,amp_150,
amp_175,amp_188,amp_200,amp_250,amp_300,amp_350: OW =
begin
sum_w1=AW(w1_0+w1_50+w1_100+w1_150+w1_175+w1_188+w1_200 +
          w1_250+w1_300+w1_350);
error=AW(IN-sum_w1@1);
w1_0=AW(error*kal1_0+w1_0@1);

w1_50=AW(error*kal1_50- sin_50*w2_50@1+cos_50*w1_50@1);
w2_50=AW(cos_50*w2_50@1+ sin_50*w1_50@1+error*kal2_50);

w1_100=AW(error*kal1_100-sin_100*w2_100@1+cos_100* w1_100@1);
w2_100=AW(cos_100*w2_100@1+ sin_100*w1_100@1+error* kal2_100);
```

```
w1_150=AW(error*kal1_150- sin_150*w2_150@1+cos_150* w1_150@1);
w2_150=AW(cos_150*w2_150@1+ sin_150*w1_150@1+error* kal2_150);
w1_175=AW( error*kal1_175- sin_175*w2_175@1+ cos_175* w1_175@1);
w2_175=AW( cos_175*w2_175@1+ sin_175*w1_175@1+ error* kal2_175);
w1_188=AW( error*kal1_188- sin_188*w2_188@1+ cos_188* w1_188@1);
w2_188=  AW(cos_188*w2_188@1+ sin_188*w1_188@1+ error* kal2_188);
w1_200=AW( error*kal1_200- sin_200*w2_200@1+ cos_200* w1_200@1);
w2_200=AW( cos_200 * w2_200@1+ sin_200 * w1_200@1+ error * kal2_200);
w1_250=AW( error*kal1_250- sin_250*w2_250@1+cos_250* w1_250@1);
w2_250=AW(cos_250*w2_250@1+ sin_250*w1_250@1+error* kal2_250);
w1_300=AW(error*kal1_300- sin_300*w2_300@1+ cos_300* w1_300@1);
w2_300=AW(cos_300*w2_300@1+ sin_300*w1_300@1+error* kal2_300);
w1_350=AW(error*kal1_350- sin_350*w2_350@1+ cos_350* w1_350@1);
w2_350=AW(cos_350*w2_350@1+ sin_350*w1_350@1+error* kal2_350);

/* computation of the magnitude square */
amp_dc=AW(w1_0*w1_0);
amp_50=AW(w1_50*w1_50+w2_50*w2_50);
amp_100=AW(w1_100*w1_100+w2_100*w2_100);
amp_150=AW(w1_150*w1_150+w2_150*w2_150);
amp_175=AW(w1_175*w1_175+w2_175*w2_175);
amp_188=AW(w1_188*w1_188+w2_188*w2_188);
amp_200=AW(w1_200*w1_200+w2_200*w2_200);
amp_250=AW(w1_250*w1_250+w2_250*w2_250);
amp_300=AW(w1_300*w1_300+w2_300*w2_300);
amp_350=AW(w1_350*w1_350+w2_350*w2_350);
end;

/* this function allows to verify the magnitude by taking the square of the
amp_xx. Not usable for Mistral 1 synthesis ! */
func racine(asqrt_dc,asqrt_100,asqrt_150,asqrt_175,asqrt_188,
asqrt_200,asqrt_250,asqrt_300,asqrt_350:SW)
amp_dc,amp_50,amp_100,amp_150,amp_175,amp_188,amp_200,amp_250,
amp_300,amp_350 :SW =

begin
asqrt_dc=SW(sqrt(amp_dc));
asqrt_50=SW(sqrt(amp_50));
asqrt_100=SW(sqrt(amp_100));
asqrt_150=SW(sqrt(amp_150));
asqrt_175=SW(sqrt(amp_175));
asqrt_188=SW(sqrt(amp_188));
asqrt_200=SW(sqrt(amp_200));
asqrt_250=SW(sqrt(amp_250));
asqrt_300=SW(sqrt(amp_300));
asqrt_350=SW(sqrt(amp_350));
end;
```


Appendix B

B1: Spectral Analyzer ASIC: 2 Lines

# Estimation of state variables for:	50 Hz and 150 Hz
input wors	fix<16.15>
output words	fix<16.15>
Sampling Rate: fs	3200 Hz
Clock rate chip	32 * fs
# General Information:	
VLSI Tools Version	v8r4.6.4
tool	Chip Compiler
technology	cmn8a3
# Nets And Cells:	
nets	1642
cell_areas	1
standard_cells	1453
connectors	126
# Total Area (excluding scribe lines) :	
chip_size	4186.00 X 4740.50 lambda (65.92 X 74.65 mil)
transistors	24617
core_density	.1241E-02 trans/sq-lam (1.2505 gates/sq-mil)
# Post Route:	
CM2CM2	6485
metal	525952 lambda
metal2	603969 lambda
unroutes_in_cell_routing	0
unroutes_in_block_routing	0
# Cell Areas:	
size	4186 X 4740.5 lambda
routing_factor	.73
transistors	24617
density	.1242E-02 trans/sq-lam (1.2517 gates/sq-mil)
standard_cells	1453
row_end_cells	56
feed_thru_cells	217

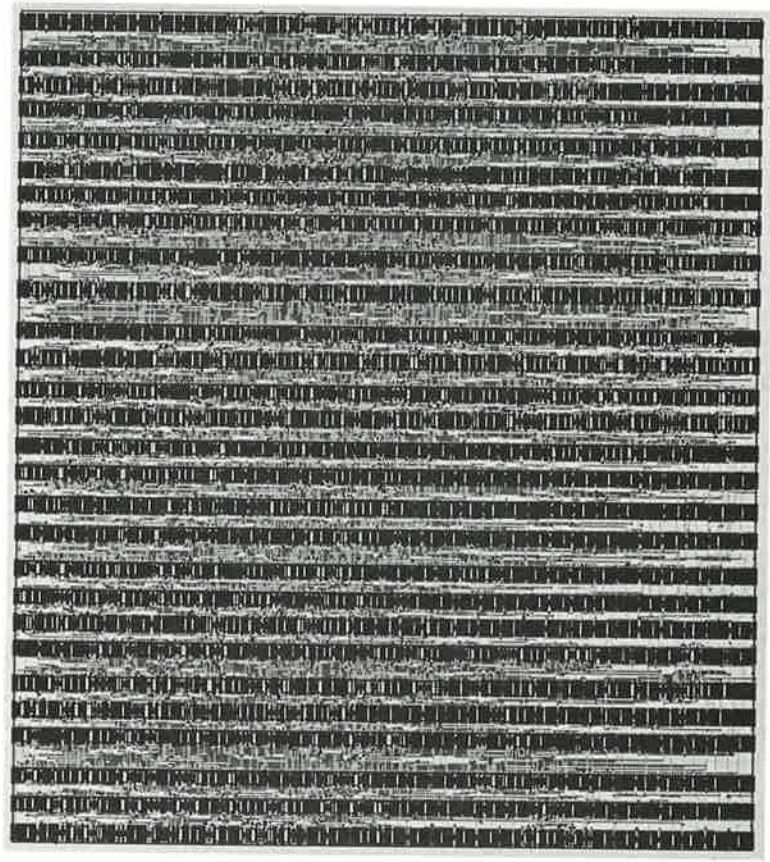


Figure 6.B1. *Layout spectral analyzer: 2 frequency lines. Technology cmn8a3, standard cell library vsc450l.*

B2: Spectral Analyzer ASIC: 4 Lines

# Estimation of state variables for:	50 Hz, 100 Hz, 150 Hz, 200 Hz.
input wors	fix<16.15>
output words	fix<16.15>
Sampling Rate: fs	3200 Hz
Clock rate chip	32 * fs

# General Information:	
VLSI Tools Version	v8r4.6.4
tool	Chip Compiler
technology	cmn8a3

# General Information:	
VLSI Tools Version	v8r4.6.4
tool	Chip Compiler
technology	cmn8a3

# Nets And Cells:	
nets	3095
standard_cells	2739
connectors	180

# Total Area (excluding scribe lines) :	
chip_size	5890.00 X 6842.50 lambda (92.76 X 107.76 mil)
transistors	44364
core_density	.1101E-02 trans/sq-lam (1.1097 gates/sq-mil)

# Post Route:	
CM2CM2	12217
metal	1217816 lambda
metal2	1185132 lambda
unroutes_in_cell_routing	0
unroutes_in_block_routing	0

# Cell Areas:	
size	5890 X 6842.5 lambda
routing_factor	.75
transistors	44364
density	.1101E-02 trans/sq-lam (1.1104 gates/sq-mil)
standard_cells	2739
row_end_cells	80
feed_thru_cells	308

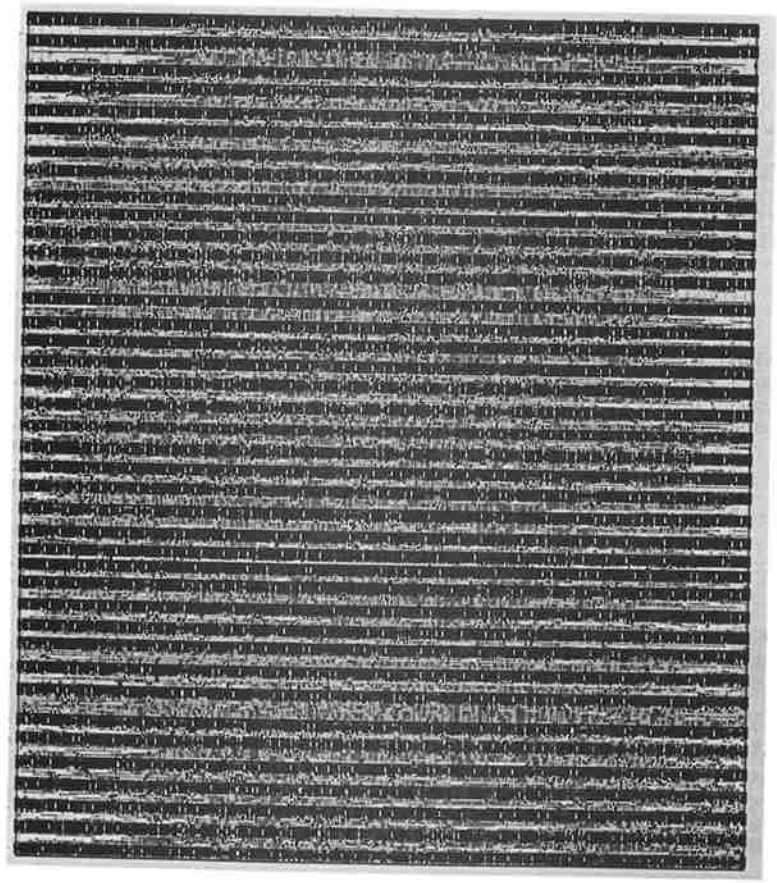


Figure 6.B2. *Layout spectral analyzer: 4 frequency lines. Technology cmn8a3, standard cell library vsc450l.*

B3: Spectral Analyzer ASIC: 6 Lines

```
# Estimation of state variables for:
                          50 Hz, 100 Hz, 150 Hz, 200 Hz, 250 Hz, 300 Hz
input wors                fix<16.15>
output words              fix<16.15>
Sampling Rate: fs         3200 Hz
Clock rate chip           32 * fs

# General Information:
VLSI Tools Version        v8r4.6.4
tool                      Chip Compiler
technology                cmn8a3


# General Information:
VLSI Tools Version        v8r4.6.4
tool                      Chip Compiler
technology                cmn8a3


# Nets And Cells:
nets                      4841
standard_cells            4285
connectors                230

# Total Area (excluding scribe lines) :
chip_size                 7314.00 X 9631.50 lambda (115.18 X 151.68 mil)
transistors               69078
core_density              .9806E-03 trans/sq-lam (.9885 gates/sq-mil)

# Post Route:
CM2CM2                    21517
metal                     2537736 lambda
metal2                    2536111 lambda
unroutes_in_cell_routing  0
unroutes_in_block_routing 0

# Cell Areas:
 size                     7314 X 9631.5 lambda
 routing_factor           .97
 transistors              69078
 density                  .9811E-03 trans/sq-lam (.9890 gates/sq-mil)
 standard_cells           4285
 row_end_cells            100
 feed_thru_cells          375
```

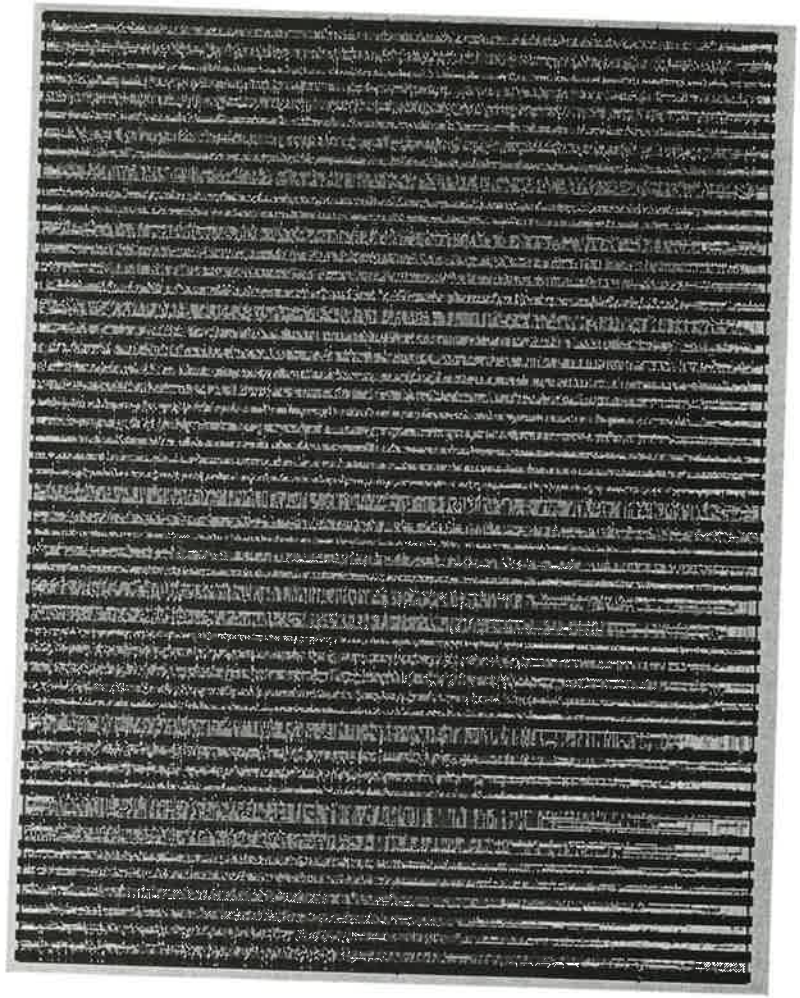


Figure 6.B3. *Layout spectral analyzer: 6 frequency lines. Technology cmn8a3, standard cell library vsc450l.*

B4: Spectral Analyzer ASIC: 10 Lines

```
# Estimation of state variables for:
                        0 Hz, 50 Hz, 100 Hz, 150 Hz, 175 Hz, 188 Hz,
                        200 Hz, 250 Hz, 300 Hz, 350 Hz
input wors              fix<16.15>
output words            fix<16.15>
Sampling Rate: fs       3200 Hz
Clock rate chip         34 * fs

# General Information:
VLSI Tools Version      v8r4.6.4
tool                    Chip Compiler
technology              cmn8a3


# General Information:
VLSI Tools Version      v8r4.6.4
tool                    Chip Compiler
technology              cmn8a3


# Nets And Cells:
nets                    7605
standard_cells          6753
connectors              315

# Total Area (excluding scribe lines) :
chip_size               8482.00 X 13520.50 lambda (133.57 X 212.92 mil)
transistors             109461
core_density            .9545-03 trans/sq-lam (.9622 gates/sq-mil)

# Post Route:
CM2CM2                  35656
metal                   4346624 lambda
metal2                  4491530 lambda
unroutes_in_cell_routing    0
unroutes_in_block_routing   0

# Cell Areas:
 size                   8482 X 13520.5 lambda
 routing_factor         1.03
 transistors            109461
 density                .9548E-03 trans/sq-lam (.9625 gates/sq-mil)
 standard_cells         6753
 row_end_cells          136
 feed_thru_cells        525
```

Figure 6.B4. *Layout spectral analyzer: 10 frequency lines. Technology cmn8a3, standard cell library vsc450l.*

B5: 16-Bit Data-By-Data Multiplier

```
# Data-by-data multiplier:
input wors                  fix<16.15>
output word                 fix<16.15>
Sampling Rate: fs           3200 Hz
Clock rate chip             32 * fs

# General Information:
VLSI Tools Version          v8r4.6.4
tool                        Chip Compiler
technology                  cmn8a3


# General Information:
VLSI Tools Version          v8r4.6.4
tool                        Chip Compiler
technology                  cmn8a3


# Nets And Cells:
nets                        569
standard_cells              518
connectors                  56

# Total Area (excluding scribe lines) :
chip_size                   3218.00 X 1884.50 lambda (50.68 X 29.68 mil)
transistors                 8337
core_density                .1375E-02 trans/sq-lam (1.3858 gates/sq-mil)

# Post Route:
CM2CM2                      1818
metal                       141152 lambda
metal2                      138302 lambda
unroutes_in_cell_routing    0
unroutes_in_block_routing   0

# Cell Areas:
size                        3218 X 1884.5 lambda
routing_factor              .60
transistors                 8337
density                     .1377E-02 trans/sq-lam (1.3882 gates/sq-mil)
standard_cells              518
row_end_cells               24
feed_thru_cells             59
```

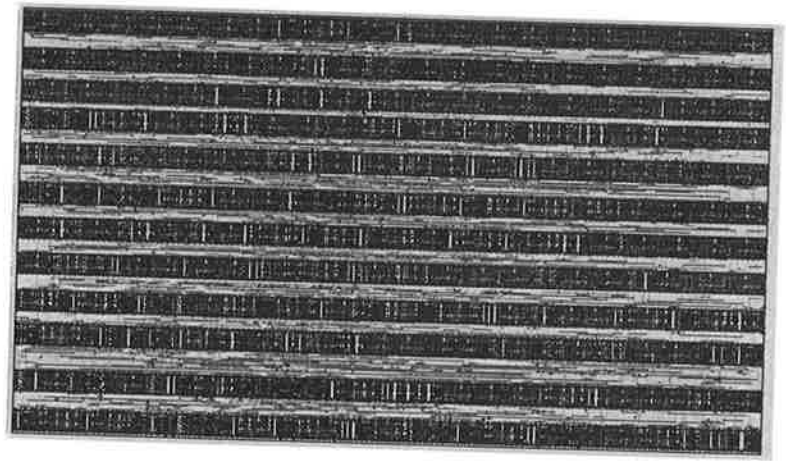


Figure 6.B5. *Layout 16-bit data-by-data multiplier. Technology cmn8a3, standard cell library vsc450l.*

References

[Ami89] G. Amit, U. Shaked : "Minimizing of Roundoff Errors in Digital Realization of Kalman Filters", IEEE Trans. on Accoustics, Speech, and Signal Processing, Vol. ASSP-37, No.12, Dec. 1989, pp.1980-1982.

[Azim90] M. R. Azimi-Sadjadi, S. Bannour : "Two-Dimensional Adaptive Block Kalman Filtering of Radar Imagery", ICASSP 90, pp. 246-249.

[Azim91] M. R.Azimi-Sandjadi, T. Lu, E.M. Nebot : "Parallel and Sequential Block Kalman Filtering and Their Implementations Using Systolic Arrays", IEEE Trans. on Signal Processing, Vol.SP-39, No.1, January 1991 pp. 137-147.

[Bitm86] R.R. Bitmead, A.H. Tsoi, P.J. Parker : "A Kalman Filter Approach to Short-Time Fourier Analysis", IEEE Trans. on Acc. Syst. and Signal Processing, Vol.ASSP-34, No.6, December 1986, pp. 1493-1501.

[Chen87] M.J. Chen, K. Yao : "On Realization and Implementation of Kalman Filtering by Systolic Array", Proc. of the 1987 Conf. on Information Sciences and Systems, John Hopkins Univ., March 1987, pp. 375-380.

[Cour93] B. Courtois : "CAD and Testing of ICs and systems: Where are we going ?", Report requested by CNES, TIMA & CMP, 46 avenue Félix Viallet, 38031 Grenoble Cedex, 1993, France.

[CSEM92] CSEL_LIB Standard Cell Library, Centre Suisse d'Electronique et de Microtechnique SA., 1992 Neuchâtel.

[Falc78] D. Falconer, L. Ljung : "Application of Kalman Estimation to Adaptative Equalization", Trans. of the ASME Journal of Basic Engineering, March 1961, pp. 95-109.

[Gast88] F.M.F. Gaston, G.W. Irwin : "A systolic Square Root Information Kalman Filter", Proc. of International Conf. on Systolic Arrays, 1988, pp. 643-652.

[Gent81] W.M. Gentlemen, H.T. Kung : "Matrix Triangularization by Systolic Arrays", SPIE Vol.298 Real Time Signal Processing VI, 1981.

[Gajy88] D. D. Gajski : "Silicon Compilation", Addison-Wesley Publ. Company, Reading, Massachusetts, 1988.

[Gajy92] D. D. Gajski, Dutt N., A. Wu, S. Lin : "High-level synthesis: Introduction to Chip and system design", Kluwer Academic Publishers, Boston, 1992.

[Hen86] Hen-Geul Yeh : "Kalman Filtering and Systolic Processors", ICASSP 86, pp. 2139-2142.

[Jove86] J.M. Jover, T. Kailath : "A parallel Architecture for Kalman Filter Measurement Update and Parameter Estimation", Automatica, Vol.22, 1986, pp. 43-47.

[Kung85] S.Y. Kung, H.J. Whitehouse, T. Kailath : "VLSI and Modern Signal Processing", Prentice-Hall, Englewood Cliffs, New Jersey 1985, Chapter 21, pp. 375-388.

[Kung87] S.Y. Kung, J.N. Hwang : "Systolic Design for State Space Models: Kalman Filtering and Neural Networks", Proc. of the 26th Conf. on Decision and Control, Dec. 1987, pp. 1461-1476.

[Kung91] S.Y. Kung : "Systolic Array Design for Kalman Filtering", IEEE Trans. on Signal Processing, Vol.SP-39, No.1, January 1991, pp. 171-182.

[Lee88] E.K.B Lee, S. Haykin : "Parallel Implementation of the Tracking Kalman Filter", ICASSP 88, Vol.4, pp. 2092-2095.

[Linc] R.A. Lincoln, K. Yao : "Efficient Systolic Kalman Filter Design by Dependence Graph Mapping", VLSI and Signal Processing, Vol.3, IEEE press.

[Park93] I. Park, K. O'Brien, A. Jerraya : " AMICAL: architectural synthesis based on VHDL", In Synthesis for control dominated circuits", Ed. G. Saucier, Publ. Elsevier, 1993.

[Rao90] P. Rao P., M.A. Bayoumi M.A : "An Efficient VLSI, Implementation of Real-Time Kalman Filter", ICASSP 90, pp. 2353-2356.

[Seitz85] C. Seitz : "Industrial Application of Kalman Filtering", ECCTC-85, pp. 359-362, 1985.

[Shpa89] I. Shpancer, D. Arison : "Kalman Filtering with Zoran's ZR34325, Application Note Zoran, 1989.

[Sung87] T.Y. Sung, Y.H. Hu : "Parallel VLSI Implementation of Kalman Filter", IEEE Trans. Aerospace, Electronics Systems, Vol.AES-23, No.2, pp. 215-224, March 1987.

[Vanh93] J. Vanhoof, K. Van Rompaey, I. Bolsens, G. Goossens, H. De Man : "High-Level synthesis for real-time digital signal processing", Kluwer Academic Publishers, Dordrecht, 1993.

[Yeh88] H.G. Yeh H.G. : "Systolic Implementation on Kalman Filters", IEEE Trans. Accoustics, Speech, Signal Processing, Vol.ASSP-36, No.9, pp. 1514-1517, September 1988.

[Zhan90] J.Y. Zhang, W. Steenaart : "A Very Fast Kalman Filter for Image Restoration", ICASSP 90, pp. 250-253.

7. Conclusion

In the final chapter, certain implementation results will be discussed, as will the deterministic and stochastic design approaches. Several conclusions will be presented pertaining to the two different implementation strategies. Finally, propositions for further improvements and future research possibilities are offered.

7.1. KALMAN FILTERING FOR SPECTRAL ANALYSIS

Time-variation of the sinusoidal frequency is expressed by a stochastic design model which, together with a model of the additive noise source, is the foundation for the design of a Kalman filter.

This thesis emphasizes one very specific approach for estimating sliding Fourier coefficients for a given set of frequencies — the Kalman filter. This method is especially designed for the spectrum analysis of signals with time-varying frequency components and/or when only specific frequency bands are of interest.

Using the Kalman method, one can select the range of the frequency axis and the distance that separates the frequencies. A special feature is that the sharpness, or in other words, the shape of the frequency response (noise rejection), can be parametrized individually for each frequency.

The discussion centered on the estimation of the spectral components of quasi-stationary sinusoidal signals measured in noise. More specifically, time-frequency analysis and the complexity of the algorithmic operations were studied.

A Dynamic Signal Model is preferred to a Static Signal Model because signals measured in noise are stochastic by nature. A stochastic design (e.g. Kalman theory) incorporates the dynamics of the signal model and therefore offers advantages over a deterministic design approach (e.g. Fourier analysis).

7.1.1. Achieved Results

Two stochastic signal models were applied to the Kalman filter, the Random Walk Signal Model and the Oscillator Signal Model. Both were analyzed in the time and frequency domains. A detailed study of windowing functions was undertaken to prepare for the comparison of the stochastic approach to the well- known Discrete Fourier Transform.

The important characteristics of the Kalman estimator are its tracking speed and its frequency response. The two depend on the covariance matrices or, more specifically, on the design parameter. The Kalman estimator is tracking on true values more slowly than DFT windowing, but its frequency response is completely tunable.

Furthermore, there are no constraints on the range of frequencies or the increment separating them. At each estimated frequency, a design parameter, `q/r`, shapes the selectivity of the main-lobe and the amount of noise rejection (roll-off rate of the side-lobe): A small design parameter increases selectivity and side-lobe rejection, but with a slower tracking speed, necessitating a stationary signal. The selectivity of the frequency response is also affected by the relative position of its neighboring frequencies. Thus, increased selectivity can be achieved for higher values of `q/r`.

Under the assumption of stationary random process, the error covariance matrices and the Kalman Gain can be computed in advance because the error covariances tend to stationary functions. This means that the matrix multiplications and the matrix inversions are transferred from real-time to off-line calculation. This reduces the number of equations needed to determine the on-line estimates from five to two, thus exorbitantly abbreviating the real-time computational complexity of the steady-state Kalman estimator. New algorithms based on the steady-state Kalman filters were then proposed.

There is no noticeable difference between the frequency characteristic of each steady-state Kalman estimator and its on-line algorithm. However, the steady-state Kalman estimator has a faster tracking speed than its on-line algorithm.

The most significant improvement that has been achieved with the implementation of the RWSM is an 11% decrease in computation, as compared to the steady-state OSM. In both models, the number of arithmetic operations is linearly related to the number of frequency lines — the filter order is linearly dependent upon the number of analyzed frequencies.

Using the steady-state Kalman estimator, each input sample creates an estimate of the output. This iterative calculation procedure can be seen as short-time spectrum analysis. The time needed to track correct estimates is comparable to the window function used in classical short-time Fourier transform.

The number of arithmetic operations gained by using the steady-state RWSM, instead of Radix 2 and split-radix FFT, two very efficient Fast Fourier Transform algorithms, is remarkable. The gain variation is the difference between the linear dependence of the steady-state Kalman estimator and the logarithmic dependence of the FFT algorithms. The gain over the split-radix FFT increased from 4 to 10.8 for 16 to 1024 estimated frequencies.

Because the steady-state Kalman estimator uses an iterative calculation method to estimate the Fourier coefficients, no data block has to be stored and thus memory occupation remains low. This is an advantage over the block-data processing methods of the FFT approaches. On the other hand, steady-state Kalman estimators require a sine and a cosine table. If the ratio of the lowest estimated frequency to the sampling frequency is small, the tables could be large, engaging more space.

7.1.2. Limitations

FFT algorithms are best suited for applications that require high tracking speeds, where noise rejection is secondary, and when the distance between neighboring frequencies is fixed. FFT algorithms determine the entire spectrum and results are available only after calculation of the complete data block.

The steady-state Kalman estimator offers better quality estimations, especially for signals measured in noise. Because the stationarity of the sinusoid is related to the tracking-time, the Kalman filter is well-suited for quasi-stationary signals. An important advantage of the Kalman filter is that estimates are available at each sample — for any set of frequencies anywhere along the frequency axis.

Today's CAD design environments do not offer automated synthesis. To enter this domain, a deep knowledge of the proposed Kalman approach is necessary because a user would have to derive a CAD design environment himself. To understand the theory and implementation of the Kalman filtering approach is very complicated.

This work includes a complete design and analysis environment inside Mathwork's MATLAB tool. The tool that has been developed, Design and Analysis of Spectrum Analysis based on Kalman filters (*daSak*©), is a general purpose CAD environment run on DOS, Macintosh, and UNIX computers. A complete design environment that produces on-line and steady-state OSM and RWSM for spectral analysis has been written and integrated with the domain-specific knowledge. Thus, users of *daSak*© need not master Kalman filtering theory. The program is simple to use, performs the design and analysis, and presents graphical results on the computer screen. The derived results can be re-used for further implementation with DSPs and ASICs.

This thesis is a collection of ideas, algorithms, CAD-tools, and results which simplify real-time implementation of sliding spectral analysis. Due to its low complexity, the exhibited concept is very interesting for system integration.

7.2. IMPLEMENTATION

7.2.1. Achieved Results on DSPs

Two steady-state Kalman algorithms and a Bergland FFT have been implemented on a Motorola DSP56001 signal processor. It has been shown that the steady-state Kalman approaches require more instruction cycles for all $M \leq 8$, with M depicting the number of estimated frequency lines. However, for all $M \geq 16$, the Kalman techniques are faster.

The FFT approaches feature several major drawbacks. The frequency resolution depends on the number of points and on the applied window function. In order to increase the frequency resolution, a large N is needed. This is complicated by the fact that N is limited to the set, 2^z, for all integers z. Furthermore, the complete spectrum is automatically calculated, even if only a certain frequency band is of interest. Finally, the algorithms' complicated addressing of data and coefficients, based on block data processing, requires that the entire data block be stored.

These drawbacks are irrelevant for the Kalman approaches. Only the desired frequencies have to be implemented and thus, for applications where only a band of frequencies needs to be analyzed, Kalman is executed more efficiently. Also, the data and coefficient addressing is unambiguous, with no pre-address calculations required.

The steady-state Random Walk Signal Model was used for an industrial application — a Digital Power Network Signal Analysis System implemented on the DSP56001. The DSP performs the estimation of the Fourier coefficients for the power network signals and determines the power, the distortion, and the ripple control signals. The DSP board, interfaced to the PC screen, communicates the various findings.

It has been shown that the steady-state Kalman approach is appropriate for real-time implementations. The regular structure of the signal flow graph and the sequential addressing of coefficients and variables lead to a very efficient assembly code. There is no need to program in nested loops, which invariably increases the processing time.

7.2.2. ASIC Results

The ASIC design is based on silicon compilers and VHDL-logic synthesis. The target silicon compiler, Mistral 1™, has been chosen for bit-serial hard-wired DSP-architecture synthesis.

The steady-state OSM algorithm has been selected for silicon implementation. A modified steady-state OSM signal flow graph, which further reduces the number of arithmetic operations and offers very few memory requirements, has been discovered.

The modified steady-state OSM is used for four integrated circuits that have been designed to the layout level. It offers uncomplicated arithmetic, leading to efficient ASIC integrations.

Real-time power spectral analysis can be performed at very high sampling rates. Results have shown that, in practice, the maximum sampling rate is limited by the standard cell library, the place and route efficiency, and the target technology.

The implementation has shown an exponential dependency of the implementation complexity. The reasons for this is that the error signal must be pre-buffered using pre-buffering strategy. Furthermore, routing becomes more difficult because many rows have to be crossed via feed-trues, in standard cells. As a result, the routing channels are enlarged.

Comparing DSP and ASIC implementations, the results have shown that the maximum achievable sampling rate is much higher for Mistral 1™ ASICs and they consume much less power. Finally, estimations of the chip area show the main drawback of hard-wired bit-serial implementations. The signal flow graph is mapped directly onto the hardware. As a result the Chip area increases as the algorithm becomes more complex. Thus, applications that require more than 64 lines have to be time-multiplexed.

7.3. FUTURE IMPROVEMENTS

In this thesis, two signal models have been examined, but there are parts that have not been covered. It would be very interesting to investigate the time-frequency domains of other state-space signal models and to compare their computational complexities and performances in noise.

Research on analytical descriptions of tracking speed suggests that tracking speed could be expressed as a function of the design parameter. Such an increase in value could be used for faster estimations during synthesis inside *daSak*©.

Studies connected with the performance in finite arithmetic implementations would be interesting to derive. Analytical descriptions of the quantization type and overflow characteristics could be contrasted with DCT performances.

ASIC design results have shown that the implementation complexity diverges from a linear to an exponential dependency. The signal flow graph of a complete set of frequencies has been mapped onto the Mistral 1™ architecture. To split up the problem inside the delay management, divide the algorithm into blocks as shown in Figure 7.1. Each block estimates only one frequency line and can be mapped onto Mistral 1 individually. The error signal should also be mapped separately. With this method, a modular block implementation is achieved.

The synchronization and scheduling of signals between blocks can be solved by inserting fixed delay potentials on specific signals during Mistral 1™ architecture synthesis. This method produces the preferred linear dependency for the number of required hardware operators. Furthermore, the chip clock rate can be controlled via fixed delay potentials.

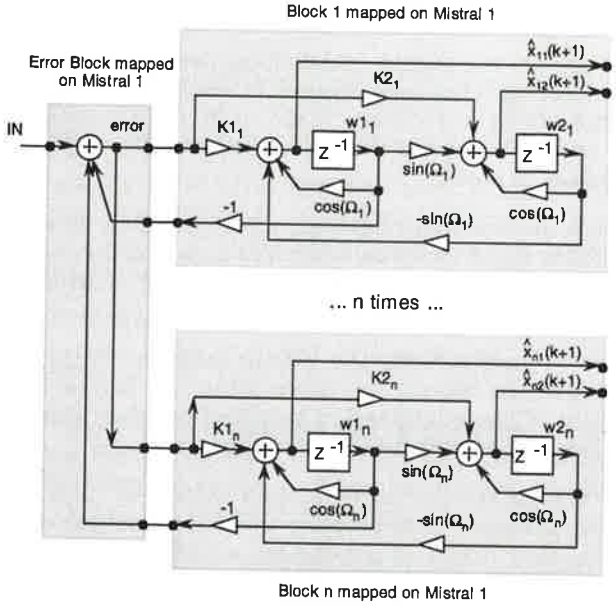


Figure 7.1. *Block implementation: Modified steady-state OSM.*

The place and route optimization is divided into blocks and thus performs much better in density. The remaining non-linear contribution from the place and route operation is reduced. The only non-linear contribution that remains is the pre-buffering of the separately realized error signal. This signal must be buffered and will only affect the achieved linear dependency of the implementation complexity.

This block implementation strategy could substantiate the estimated results presented in Chapter 6.

Abbreviations

AC	Alternating Current
AFG	Architecture Flow Graph
ASIC	Application Specific Integrated Circuit
CAD	Computer Aided Design
CD	Compact Disk
CFG	Cell Flow Graph
CSD	Canonical Signed Digit
daSak	Design and Analysis of Spectrum Analysis based on Kalman filters
DC	Direct Current
DFL	Data Flow Language
DFT	Descrete Fourier Transform
DfT	Design for Test
DRC	Design Rule Check
ESFG	Expanded Signal Flow Graph
WT	Wavelet Transform
DPNSAS	Digital Power Network Signal Analysis System
DSP	Digital Signal Processing (Or Digital Signal Processor)
FFT	Fast Fourier Transform
FIR	Finite Impulse Response (filter)
GPS	Global Positioning Systems
HDTV	High Definition Television
IC	Integrated Circuit
IIR	Infinite Impulse Response (filter)
ISDN	Integrated System Digital Network
KFCE	Fourier Coefficient Estimator
LSB	Least Significant Bit
MSB	Most Significant Bit
Natel-D	Digital mobile telephones
OSM	Oscillator Signal Model

OSFG	Optimized Signal Flow Graph
PCU	Power Computing Unit
QMF	Quadrature Mirror Filter
RTPG	Random Test Pattern Generation
RMS	Root Mean-Square
RLS	Recursive Least Square (method)
RWSM	Random Walk Signal Model
RTL	Register Transfer Level
SSM	Static Signal Model
TF	Time Frequency plane
VDD	Power Supply Voltage
VSS	Ground Voltage
VLSI	Very Large Scale Integration
VHDL	Very high-speed IC Hardware Description Language
WT	Wavelet Transform

Notations

$H(\omega)$	Transfer Function
a_n, b_n	Fourier Coefficients.
$p(t)$	Instantaneous Power
P	Active Power
P_q	Reactive Power
P_s	Apparent Power
$u(t)$	Instantaneous Voltage
$i(t)$	Instantaneous Current
I_{RMS}	Effective- or Root-Mean-Square Current
U_{RMS}	Effective- or Root-Mean-Square Voltage
q/r	Design Parameter