94

Analogic for Code Estimation and Detection

by

Xu Sun

Submitted to the Program in Media Arts and Sciences,
School of Architecture and Planning
in partial fulfillment of the requirements for the degree of

Master of Science in Media Arts and Sciences

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2005

Author .....

Program in Media Arts and Sciences,
School of Architecture and Planning
August 5, 2005

Certified by .....
Neil Gershenfeld
Associate Professor
Thesis Supervisor

Accepted by .....
Andrew B. Lippman
Chairman, Departmental Committee on Graduate Students

Analogic for Code Estimation and Detection

by

Xu Sun

Submitted to the Program in Media Arts and Sciences,
School of Architecture and Planning
on August 5, 2005, in partial fulfillment of the
requirements for the degree of
Master of Science in Media Arts and Sciences

Abstract

Analogic is a class of analog statistical signal processing circuits that dynamically solve an associated inference problem by locally propagating probabilities in a message-passing algorithm [29] [15]. In this thesis, we study an exemplary embodiment of analogic called Noise-Locked Loop(NLL) which is a pseudo-random code estimation system. The previous work shows NLL can perform direct-sequence spread-spectrum acquisition and tracking functionality and promises orders-of-magnitude win over digital implementations [29].

Most of the research [30] [2] [3] has been focused on the simulation and implementation of probability representation NLL derived from exact form message-passing algorithms. We propose an approximate message-passing algorithm for NLL in log-likelihood ratio(LLR) representation and have constructed its analogic implementation. The new approximate NLL gives shorter acquisition time comparing to the exact form NLL. The approximate message-passing algorithm makes it possible to construct analogic which is almost temperature independent. This is very useful in the design of robust large-scale analogic networks.

Generalized belief propagation(GBP) has been proposed to improve the computational accuracy of Belief Propagation [31] [32] [33]. The application of GBP to NLL promises significantly improvement of the synchronization performance. However, there is no report on circuit implementation. In this thesis, we propose analogic circuits to implement the basic computations in GBP, which can be used to construct general GBP systems.

Finally we propose a novel current-mode signal restoration circuit which will be important in scaling analogic to large networks.

Thesis Supervisor: Neil Gershenfeld
Title: Associate Professor

Analogic for Code Estimation and Detection

by

Xu Sun

Submitted to the Program in Media Arts and Sciences
in partial fulfillment of the requirements for the degree of
Master of Science in Media Arts and Sciences
at the
MASSACHUSETTS INSTITUTE OF TECHNOLOGY
September 2005

Thesis Advisor.....

Neil Gershenfeld
Associate Professor
Program in Media Arts and Sciences
Massachusetts Institute of Technology

Thesis Reader ....

Rahul Sarpeshkar
Robert J. Shillman Associate Professor
Department of Electrical Engineering and Computer Science
Massachusetts Institute of Technology

Thesis Reader .....

Benjamin Vigoda
Research Scientist
Mitsubishi Electric Research Laboratories (MERL)

Acknowledgments

I realize that I have received invaluable help from so many remarkable people during the thesis study.

Thank you to my advisor Neil Gershenfeld for his great insight to the study of analogic. Thank you for providing us such a free academic atmosphere in Physics and Media Group, where his encouragement to our crazy ideas and tolerance to sometime not-so-good results is always like from a generous father to his kids. Thank you for his directing of Center for Bits and Atoms, which greatly expands our vision and provides me good opportunity to work with excellent people around the campus. Thank you for all his practical support, encouragement, and guidance through out my first two years in MIT, which are also my first two years in this country.

Thank you to my thesis reader Rahul Sarpeshkar! His seeking spirit in the research, everlasting curiosity toward hard problems, great physical intuition, and academic discipline sets up a great pattern to me. Thank you for his excellent knowledge and creativity in analog circuit design, which inspires my interests in this area. Thank you for his constant care for my research. Thank you for giving me the access to his lab where I learned a lot on circuit testing.

Thank you to my thesis reader Benjamin Vigoda! The work in this thesis has largely built on the foundation that he layed in his master and Ph.D. research. Thank you for his willingness to spend so many hours with me whenever I asked him questions. Thanks for his encouragement whenever I made some progress.

Thank you to Prof. Michael Perrott for his excellent simulator CppSim. Great part of the system level simulation in this thesis is done by it. Also thanks to him for sharing his insights on possible applications of noise-locked loop.

Thank you to Yael Maguire for sharing me your knowledge in electronics and giving me many valuable suggestions on my startup. Thank you to Ben Recht for educating me on mathematical optimization and his eagerness to know my progress.Thank you to Jason for the discussion on normalization problem. Thank you to Manu for answering me all the detailed questions on thesis preparation and writing and printing.

Thank you to Amy on helping me fabricate in the shop any thing I want. Thank you to Ara.

I owe a great thank you to Soumyajit Mandal. He answered so many of my questions on circuit design. He generously gave me his own lab bench for my final testing. Also thanks for his proofreading my thesis. Thank youn to Scott Arfin for helping me set up GPIB on my laptop. Thank you to Michael Baker for his suggestions which improved my circuit a lot. Thank you to Piotr Mitros for teaching me analog circuit troubleshooting techniques and spending so many late hours in the lab with me.

Thanks to Linda Peterson and Pat Sokaloff for taking care of us graduate students. Thank you to Susan Murphi-Bottari, Mike Houlihan, Kelly Maenpaa for keeping PHM running smoothly.

Thank you to my Mom and Dad for their love that I can feel it even when I am on the other side of the earth. Thank you to my wife Grace for all her love, care, encouragement during the hard time of research.

Thank you to my Lord Jesus Christ. Thank You for Your life in me.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

A mathematical optimization problem is to minimize an objective function under a set of constraints. Inference problem is a special kind of mathematical optimization problem with constraints including axioms of probability theory. How to construct physical systems to solve the inference problem with very low power, and/or in extremely high speed, and/or with low cost, and/or with very limited physical resources etc is still a open question. Message-passing algorithms [12] [16] [5] are a special method of solving associated inference problem by locally passing messages on a factor graph. The messages can be mapped into physical degrees of freedom such as voltages and currents. The local constraints on the factor graph are the computation units implemented by a class of analog statistical signal processing circuit, which we call analogic [29].

In this thesis, we study analogic in a special test case called Noise-Locked Loop(NLL) [28] [6]. NLL is a pseudo-random code estimation system which can perform direct-sequence spread-spectrum acquisition and tracking functionality and promises orders-of-magnitude win over conventional digital implementation [29]. We now review the previous work on NLL.

1.1 Review of Previous Work

The history of noise-locked loop can be traced back to a nonlinear dynamic system called analog feedback shift register(AFSR) proposed in [6], which was an analog generalization of digital linear feedback shift register(LFSR). The AFSR can entrain to the pseudo-random sequence to achieve synchronization. [28] thoroughly studied different nonlinear functions to find a simpler implementation of AFSR and better performance. The most successful nonlinearity found was a quadratic function which gave very predictable mean acquisition time and can be easily implemented in electronics. This is an important discovery which has interesting relation with the system we built in this thesis.

As mentioned before, analogic is a class of analog statistical signal processing circuit that propagates probabilities in a message-passing algorithm. As an exemplary embodiment of the new approach, an LFSR synchronization system was derived using the message-passing algorithm on a factor graph. This new system was called Noise-Locked Loop(NLL). It is very interesting to notice that NLL can be also viewed as running the noisy sequence through a soft or analog FSR [30]. This connects NLL as a statistical estimator to the AFSR as a nonlinear dynamic system. The synchronization performance of NLL and its comparison with maximum-likelihood estimator were also studied in [29] and [30]. The real electronic implementation was reported in [2] and [3]. In their message-passing algorithm, the messages were represented as the probabilities of binary state variables(called probability representation). In their implementation, each message represented as two probabilities was mapped into two normalized currents in current-mode circuits called "soft-gates" [18].

Belief-propagation algorithms give exact marginalized probability when the associated factor graph has tree structure. However, when the factor graph involves loops, they can only compute approximate marginalized probabilities. The work in [31] [32] [33] has pointed out that belief-propagation achieves local minima of Bethe free energy in any graph. Kikuchi free energy approximation is an improvement to Bethe free energy. By using Kikuchi free energy, a new message-passing algorithm called

Generalized Belief Propagation(GBP) was constructed. GBP is reported to be much more accurate than BP. A novel way of using GBP to improve NLL was proposed in [29] and later in [4]. Their simulation results showed better synchronization performance of improved NLL. But we haven't seen any physical implementation of GBP.

1.2 Motivation

Much work has been done on NLL. But there are more interesting questions that motivate of this thesis.

Firstly, most of the previous work including simulation and implementation of NLL has been focused on studying the probability representation. In this representation, message is probability. And probability is mapped into currents in the current-mode analogic circuits. So each message on the graph needs two wires to carry two currents, which are complementary to each other and they must be normalized to a fixed tail current I_{tail} in each soft-gate. Also each message needs a pair of current-mirrors to interface with soft-gates. It is natural to ask whether we can use other message representation so that messages can be mapped into voltages. In this way we may be able to save the power overhead on current-mirrors.

Secondly, most of the previous work including derivation of NLL from message-passing algorithms and implementation has also focused on the exact form message-passing computation, by which we mean the computation is exactly derived from message-passing algorithms. The voltage mode soft-gates has been used in analog decoders [10]. However the voltage representation unavoidably suffers from temperature variation in the system. It is interesting to ask whether we can do message-passing with some approximate computation, what kind of approximation we can use, and if we can overcome the problem described above. This question motivates our exploration on approximate scalar NLL.

Thirdly, since the NLL is derived for pseudo-random code estimation, the previous work has naturally focused on its pseudo-random code synchronization performance and comparison with existing LFSR synchronization scheme in spread-spectrum systems. Actually in the front end of any direct-sequence spread-spectrum system, even before signal reaches the synchronization device, there is usually some low pass filter that serves to kill the wide-band noise and clean the signal. Higher noise rejection in this filter will greatly improve the later circuit performance. NLL models both the channel noise as additive white Gaussian noise and the dynamics of LFSR. We may ask if we can use NLL to replace the linear filter that NLL gives better noise rejection.

Fourthly, the messages passed on "conventional" NLL are probabilities of one state variable. We call this NLL scalar NLL in the following chapters. The improved NLL with GBP passes messages that are probabilities of a pair of state variables. We call this improved NLL vector NLL. How to implement vector NLL using analogic is a very interesting problem, because the analogic circuits used in vector NLL may also be used in other GBP systems.

Last but not least, the previous research and motivation in this thesis both point to the direction of improving scalar NLL to vector NLL. The circuit structure are inevitably becoming more complex in vector NLL. How to build large analog network is a question that we must ask in next step. This motives another part of the work in this thesis, which is analog signal restoration.

1.3 Contribution

Here we summarize our main results in the thesis.

1. We studied scalar NLL in the approximate log-likelihood ratio(LLR) representation. We call it approximate scalar NLL. The representation itself doesn't

require normalization in each soft-gate. We compared its synchronization performance with the NLL using exact log-likelihood ratio representation in simulation. The result shows approximate NLL outperforms the exact one in terms of acquisition time at low SNR regime. We implemented the approximate scalar NLL in discrete electronics. The measurement results confirmed the simulation. The approximate soft-gate allows us to freely set the scaling factor for conversion from LLR to voltage. In this way, the voltage representation is independent of temperature. This solves the problem in the exact form LLR soft-gates. We started to apply this approximate message-passing algorithm to bigger analog network such as iterative analog decoder. The preliminary results indicate the approximate algorithm work well. The temperature independent property of the voltage representation in the approximate algorithm can be very desirable for large analog network.

2. We investigated the possibility of replacing linear filter in the front end of spread-spectrum system with approximate scalar NLL. Both simulation and experiment suggest that NLL can improve the noise rejection performance in terms of mean squared estimation error. However the improvement is not very significant.

3. We proposed a translinear circuit structure for GBP, which to our best knowledge is the first circuit implementation of GBP. In our circuit, messages are represented as current. The proposed circuit accomplished current production and quotient which can be used as the basic circuit primitive in general GBP systems.

4. We proposed a novel current-mode feedback amplifier with controllable gain, wide linear range, and saturation at both zero current and tail current. The basic structure is winner-take-all(WTA) cell as a high gain current amplifier in the feedforward path and current mirrors on the feedback path. The amplifier can be used in analog signal restoration.

5. We proposed a new geometric approach for all-pole underdamped second-order transfer functions, which will be useful in the design and analysis of filters in NLL.

1.4 Thesis Outline

In chapter 2, we introduce the necessary background information and related topics, including phase-locked loop, spread spectrum system, and message-passing algorithm. The title is termed as "from PLL to NLL", since the NLL can be viewed as a generalization of PLL from periodic signal synchronization to nonperiodic signal synchronization.

Chapter 3 derives in detail the different kinds of NLL from message-passing algorithm. From the simplest scalar NLL to first order Markov structured vector NLL to full trellis processing vector NLL, the performance improves, while algorithm is also more and more complex. For the scalar NLL, we proposed an approximate NLL, which is even simpler than the exact form NLL. We also started to study the application of approximate message-passing in a bigger analogic network.

Chapter 4 is on the circuit implementation of scalar NLL and circuit proposal for vector NLL. We will describe the system design of the scalar NLL with detailed discussion on each components.

Chapter 5 is on current-mode signal restoration which is a very important topic for building big robust analog circuits. We propose a novel current-mode feedback amplifier.

Chapter 6 is a diversion on second-order filter theory. We include it in this thesis because it provides useful geometric techniques for the design and analysis of filters in NLL.

Chapter 7 is summary of the work in the thesis and discussion of future research direction.

Chapter 2

Background Information

In this chapter, we try to ask some basic questions that relate to the background of the thesis. These questions include: what signal does noise-locked loop synchronize to? What is the synchronization usually implemented? What knowledge do we need in order to understand NLL?

The simple answer to the first question is that NLL locks unto certain noise-like signal. This signal pattern is called pseudo-random signal or LFSR sequence or m-sequence, which is one of the basic spreading codes used in spread spectrum systems. More detailed review is given in first section. The second question requires us to briefly review the basic synchronization and tracking techniques used in spread spectrum systems. The third question touches the theme of the thesis which is formulating LFSR synchronization problem as a code estimation problem on the factor graph and use message-passing algorithm to obtain optimal or suboptimal solution. So in the third section we introduce the message-passing algorithm.
Before we answer all these questions, we introduce the phase-locked loop which is well known to be able to lock unto a periodic signal. After we understand PLL, it will be easier for us to understand pseudo-random code synchronization in spread-spectrum system.

2.1 Phase-Locked Loop

The PLL structure was first described by H. de Bellescize in a French journal L'Onde Electrique [14] in 1932. The background was that British engineers developed an alternative to the super heterodyne AM receiver, which was called homodyne or synchrodyne system. PLL was used in the homodyne receiver to automatically keep the phase of local oscillator locked unto that of the incoming carrier. In this way, the demodulation produces the maximum output. After that, the PLL was used in standard broadcast television, stereo FM radio, satellite communication systems. However, phase-locked systems were too complex and costly for use in most consumer applications. In 1970's integrated circuit process technology advanced enough to be able to fabricate monolithic PLL's at very low cost. Since then the Phase-locked loop has become universal in modern communication systems. It has been used in frequency synthesis, frequency modulation and demodulation, and clock data recovery(CDR),etc [14] [27].

The basic PLL is composed of phase detector, loop filter, and voltage-controlled oscillator(VCO). The PLL pull-in process is highly nonlinear in nature. After PLL acquires locking, we can assume linear mode operation of the loop, where VCO has the linear relationship between its output frequency $F_{out}(t)$ and input control voltage $v(t)$ as

$$F_{out}(t) = K_v v(t). \tag{2.1}$$

The output phase can be written as integration of frequency:

$$\Phi_{out}(t) = \int_{-\infty}^{t} 2\pi, F_{out}(\tau)d\tau = \int_{-\infty}^{t} 2\pi K_v v(\tau)d\tau. \tag{2.2}$$

This time-domain relationship can be Laplace-transformed to s-domain as:

$$\Phi_{out}(s) = \frac{K_v}{s} v(s). \tag{2.3}$$

The phase detector compares the phase of the sine or square wave generated by VCO with the phase of the reference signal and produces the phase difference. For the sine wave reference signal as

$$ref(t) = A_c \cos(\omega_0 t + \phi). \tag{2.4}$$

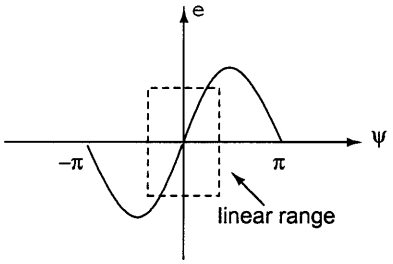

Figure 2-1: Phase detector characteristics.

The VCO output is represented as

$$out(t) = -A_v \sin(\omega_0 t + \theta). \tag{2.5}$$

If the phase detector is an ideal multiplier followed by a low pass filter to remove the double-frequency component, the phase detector output is

$$e(\psi) = K_d \sin(\psi). \tag{2.6}$$

where $\psi = \phi - \theta$ and K_d is a constant. In linear mode operation, $|\psi| \ll 1$, the phase detector characteristics shown in Fig.2-1 is linear to a good approximation. The loop filter exacts the average of the phase difference. When the VCO generated signal locks onto the reference signal, the phase difference is confined within small range. In this small range, the phase detector looks like a linear gain element. When the reference signal phase starts to shift away from the VCO output phase, the loop filter feeds an averaged non-zero phase difference as the control voltage to VCO. If the reference signal phase leads the VCO phase, control voltage increases to drive VCO produce higher frequency output to catch up. If VCO leads, control voltage decreases to produce lower frequency output in order to wait for reference signal. In this way, when the phase of these two signals are locked, their frequencies also become identical. The block diagram of the basic PLL structure with its linear model is shown in the Fig.2-2.

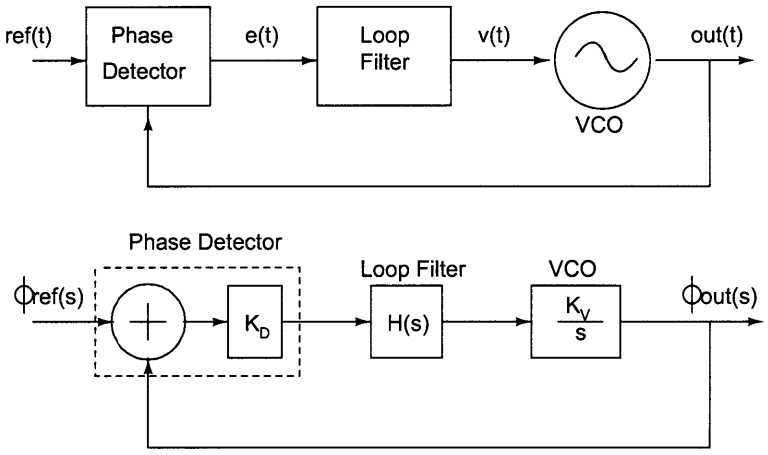

Figure 2-2: Basic PLL structure composed of phase detector, loop filter, and voltage-controlled oscillator. Once acquisition is achieved, the PLL is modelled as a linear feedback system.

2.2 Spread Spectrum Systems

In this section, we briefly review the spread spectrum(SS) system with emphasizing on pseudorandom sequence generation and code synchronization techniques.

Pseudorandom Sequences

The general circuits that are used to generate linear feedback shift-register codes are shift registers with feedback and/or feedforward connections. Consider the logic circuit illustrated in Fig.2-3. The boxes with letter 'D' represent unit delay elements,implemented as shift registers. Circles containing a "+" sign is a modulo-2 adder also called XOR gates. The coefficients besides the lines pointing to the XOR gates are either 1 if there is connection or 0 if no connection. The generating polynomial is defined as:

$$g(D) = g_0 + g_1 D + \ldots + g_{m-1} D^{m-1} + g_m D^m \tag{2.7}$$

Notice this generator might generate many different sequences starting from different initial states. The longest possible sequence has the period of $2^m - 1$, which is called maximum-length sequence or m-sequence. It has been widely used in spread spectrum

systems. Other spreading code can be derived from maximal-length sequence. Two

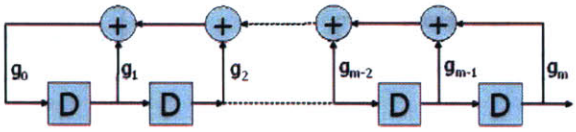

Figure 2-3: Linear feedback shift-register generator.

conditions must be satisfied to achieving the maximal-length sequence: the generating polynomial is primitive and the LFSR registers are initialized in non-zero state. The power spectrum of m-sequence with chip duration T_c and period $T = NT_c$ has the form

$$S_c(f) = \sum_{n=-\infty}^{\infty} P_n \delta(f - nf_0) \tag{2.8}$$

where $P_n = [(N+1)/N^2] sinc^2(n/N)$ and $f_0 = 1/NT_c$. So the power spectrum consists of discrete spectral lines at all harmonics of $1/NT_c$. The envelope of the amplitude of these spectral lines is a sinc function, except the DC value is $1/N^2$. In most spread-spectrum systems the power spectrum of modulated data is continuous, because the carrier is randomly modulated by data and spread code.

In our further study of NLL, we take a simple case of the m-sequence generated by LFSR with polynomial $g(D) = 1 + D + D^4$. The m-sequence generator and its output of one period are shown in Fig.2-4.

Synchronization of PN Sequence

The spread spectrum receiver despreads the received signal with a special PN code sequence, which must be synchronized with the transmitter's spreading waveform.

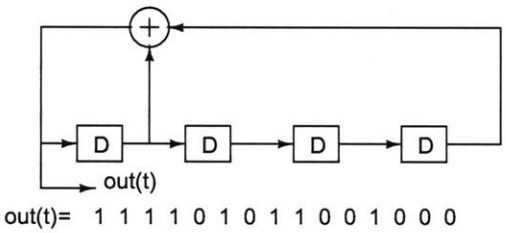

Figure 2-4: An LFSR generates m-sequence with length of 15.

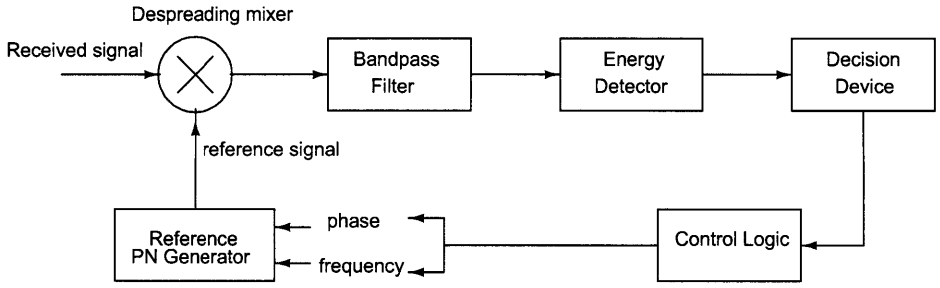

Figure 2-5: Serial-search synchronization system evaluating the phase and frequency of a spreading waveform[22].

As little as one chip offset will produce insufficient signal energy for the reliable data demodulation. There are two parts of synchronization process. One is initial code acquisition. The other is code tracking. There are several ways for the initial code acquisition. One is to use correlators to correlate the received signal with local generated LFSR sequence, and stepped serial-search the phase and frequency uncertainty cells until correct cell is found. The single-dwell search system always integrates over fixed interval or correlation to decide whether the acquisition is achieved or not. Multiple-dwell search system integrates over short correlation window first, if acquisition looks promising increase the window size until acquisition achieves or not. Circuit complexity is increased but acquisition time is reduced. Serial search is by far the most commonly used spread-spectrum synchronization technique. Fig.2-5 is a system block diagram.

Another PN acquisition approach is to use matched filter, which is matched to a segment of the direct sequence spread waveform. In contrast to evaluating each relative code phase as serial search scheme does, matched filter synchronizer freezes the phase of the reference spreading code until a particular waveform comes(identified by the matched-filter impulse response). Once the matched waveform is received, the filter produces a pulse to start the local code generator at the correct phase. In this way, the synchronization time is improved by orders of magnitude. The principal issue limiting the use of matched-filter synchronizers is the performance degradation due to the carrier frequency error and limited coherent integration time[22]. Fig.2-6

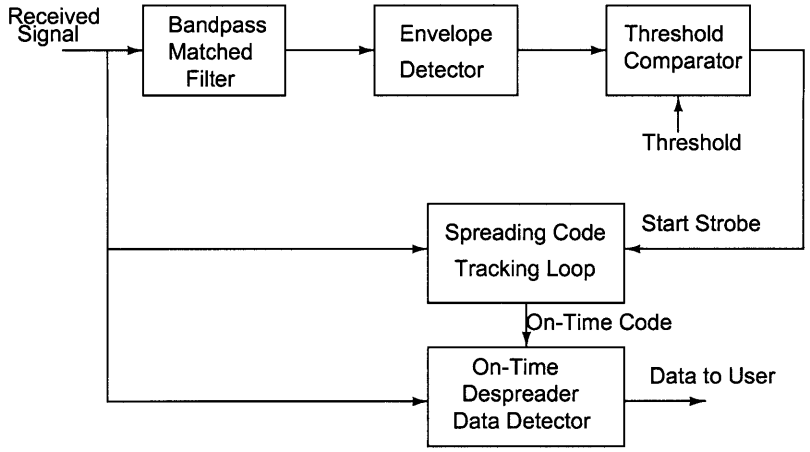

Figure 2-6: Direct-sequence spread spectrum receiver using matched-filter code acquisition[22].

is a block diagram of a direct-sequence spread spectrum receiver using match-filter synchronizer.

The third acquisition scheme is recursion aided sequential estimation(RASE) and its extensions. The m-sequence generator described in Section2.2 contains m symbols of the code sequence. If these m symbols can be estimated with sufficient accuracy, then they can be loaded into the identical shift register generator in the receiver to synchronize the system. This approach outperforms all the serial search techniques at medium to high received SNR. For very low SNR and no priori information of the received code phase, RASE has approximately the same mean acquisition time with serial search. If a priori information about received code phase is available, the serial search performs better[22]. The block diagram of RASE synchronizer is shown in Fig.2-7.

The initial code acquisition systems position the phase of the receiver generated spreading waveform within a fraction of a chip of the received spreading waveform phase. Tracking loop takes over the synchronization process at this point and pulls the receiver spreading waveform to the precise phase. The code tracking is accomplished using phase-locked loop very similar to those used for carrier tracking discussed in

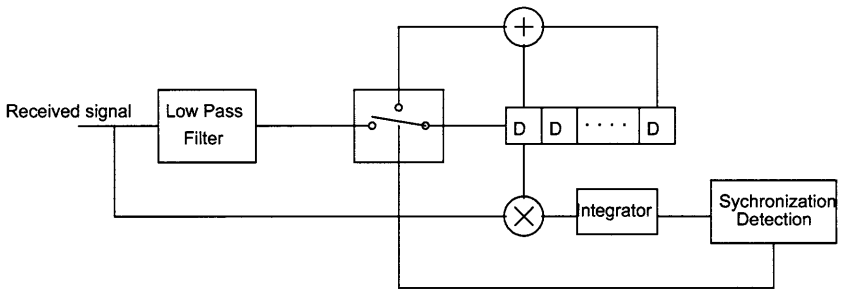

Figure 2-7: RASE synchronization system.

Sec.2.1. The principal difference is in the implementation of phase detector. The carrier tracking PLL often uses simple multiplier as phase detector, whereas code tracking loops usually employ several multipliers and filters and envelope detector[22]. A typical code tracking loop called Delay-Locked Loop(DLL) is shown in Fig.2-8, where the PN sequence is correlated with advanced and delayed signals which goes through loop filter to VCO that drives the speading waveform generator. A similar system to DLL is Tau Dither Loop(DTL), which alternate between doing correlation with advanced or retarded signal. The output of the correlator goes to the filter which is envelope detected. If it came from retarded signal, it will be inverted before it goes into VCO. The advantage of DTL is that there is no need to match two loop filters and correlators perfectly.

2.3 Message-Passing Algorithms

In this section, we introduce message-passing algorithms, which are a key tool used in deriving scalar NLL and vector NLL in the following chapters. Message-passing algorithms marginalize given joint probability distribution $P(u, s, x, y)$ to certain marginal conditional probability distribution such as $P(u|y)$.

An example is channel coding, where a vector of information bits u is mapped to a vector of codeword signals x by adding in redundancy. The encoder state is represented as variable s. After signal x is transmitted across the noisy channel, the

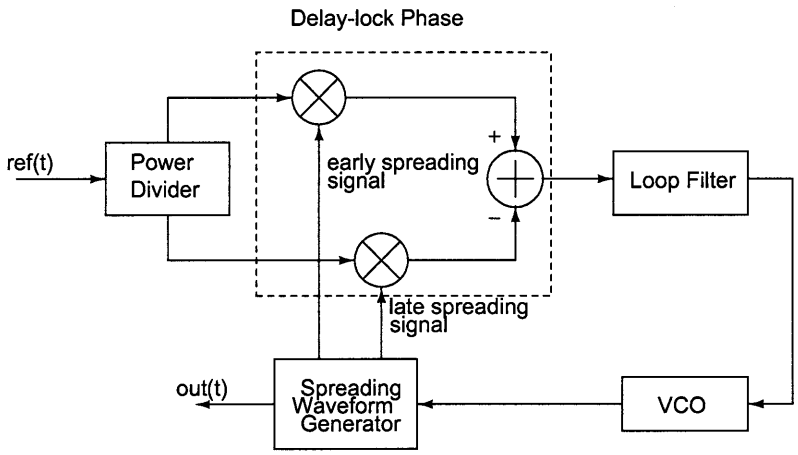

Figure 2-8: Conceptual block diagram of a baseband delay-lock tracking loop.

receiver receives a noise-corrupted version y and makes estimation on the information as $\hat{u}$.

From a probabilistic point of view, we can first specify a probability distribution for the information symbols as $P(u)$. The encoder state is determined from the information symbols using distribution $P(s|u)$. The codeword is generated according to the internal state of encoder and information symbols by $P(x|s,u)$. Finally the received signal y is related to the transmitted codeword by $P(y|x)$. The joint probability distribution is factored as:

$$P(u,s,x,y) = P(u)P(s|u)P(x|u,s)P(y|x).$$

The significant point here is the joint probability distribution over information symbol u, encoder state s, transmitted codeword x, and received signal y has been decomposed into a factorization form according to the probabilistic structure of the model. By taking advantages of such structure, algorithms more efficient than brute-force Bayes' rule can be designed.

The probabilistic structure can be intuitively presented on graphs. Several kinds of graphical model have been extensively researched, such as Bayesian network, Markov random field, and factor graph[19][13][16].

For our channel coding example, the factorized join probability distribution can be drawn in Bayesian network as:

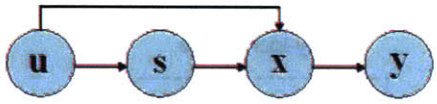

Figure 2-9: Bayesian Network for Channel Coding

In the following we will use a normal factor graph, because factor graph gives an easier way to show the constraint relations between variables. It also provides a straightforward way to present the process of marginalization, which is known as message-passing algorithm. The essence of message-passing on a tree is illustrated in Fig.2-10. The boxes represent the functions or constraints. The edges represent the variables. For example, box f_1 connected with edge x_1 means the f_1 is the function of variable x_1. The messages are passed between boxes and edges. For example, message $\mu_{f_4 \to x_4}$ is passed from function node f_4 to variable x_4. The message represents some marginalized probability distribution. The calculation of the marginal probability for certain variable from the global joint probability distribution of all the variables is obtained by successive local message-passing. The example shown in Fig.2-10 is on calculating marginal probability $p(x_4)$. The process is as following.

First, obtain message $\mu_{f_3 \to x_4}$ as

$$\mu_{f_3 \to x_4} = \sum_{x_1} \sum_{x_2} \sum_{x_3} f_3(x_1, x_2, x_3, x_4) f_1(x_1) f_2(x_2). \tag{2.9}$$

Second, obtain message $\mu_{f_6 \to x_6}$ as

$$\mu_{f_6 \to x_6} = \sum_{x_7} \sum_{x_8} f_6(x_6, x_7, x_8) f_7(x_7). \tag{2.10}$$

Third, we can get message $\mu_{f_4 \to x_4}$ from message $\mu_{f_6 \to x_6}$ as

$$\mu_{f_4 \to x_4} = \sum_{x_5} \sum_{x_6} f_4(x_4, x_5, x_6) f_5(x_5). \tag{2.11}$$

Finally the two messages passing from opposite directions are combined to get the marginal probability $p(x_4)$ as

$$p(x_4) = \mu_{f_4 \to x_4} \cdot \mu_{f_3 \to x_4}. \tag{2.12}$$

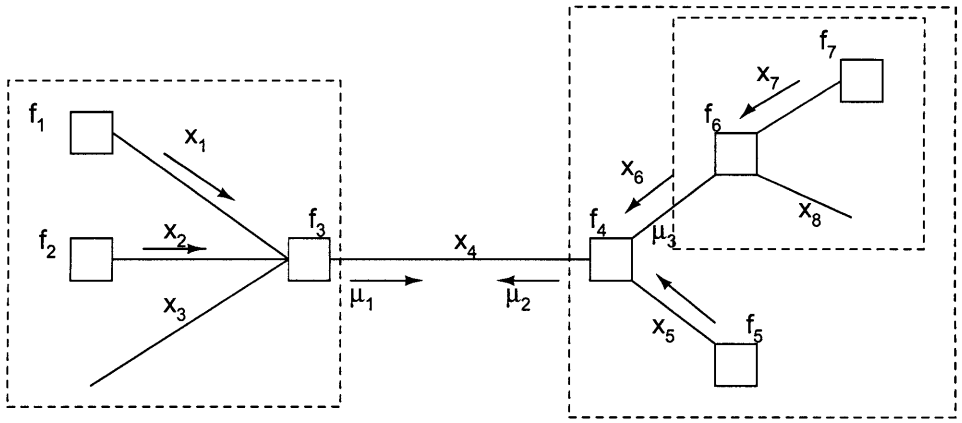

Figure 2-10: Message-passing process illustrated on the tree-like factor graph. Message μ_1 in the figure represents message $\mu_{f_3 \to x_4}$ in the above derivation. Message μ_2 represents $\mu_{f_4 \to x_4}$. μ_3 represents $\mu_{f_6 \to x_6}$[12].

The general sum-product algorithm operates according to this simple rule[12]:

Sum-Product Update Rule:

The message sent from a node v on an edge e is the product of the local function at v(or the unit function if is a variable node) with all messages received at v on edges *other* than e, summarized for the variable associated with e.

Chapter 3

LFSR Synchronization: From Scalar NLL to Vector NLL

In the previous chapter, we reviewed the basic phase-lock technique, basic concepts of spread spectrum, and message-passing algorithms. These provide the background knowledge for this chapter.

As reviewed in Chapter 1, the previous work in [29] and [4] studied scalar NLL and proposed vector NLL. In this chapter, as a good exercise to solidify the ability of communication in the language of message-passing algorithms, we rework out by ourselves the math of scalar NLL and vector NLLs. We connect them together by emphasizing performance improvement from simple suboptimal estimation(scalar NLL) using belief-propagation algorithm to intermediate complexity estimator (vector NLL) using generalized belief-propagation.

The previous work has been mainly focused on exact form message-passing algorithm. By "exact form message-passing", we mean that the basic computations of messages use their exact mathematical form. For example, the parity-check node in log-likelihood ratio representation involves hyperbolic and inverse hyperbolic functions(see Section.??). An interesting question to ask is how robust the message-passing algorithm is when the basic computation is not accurate. More precisely, what is the algorithm performance if the basic computation has been changed to an

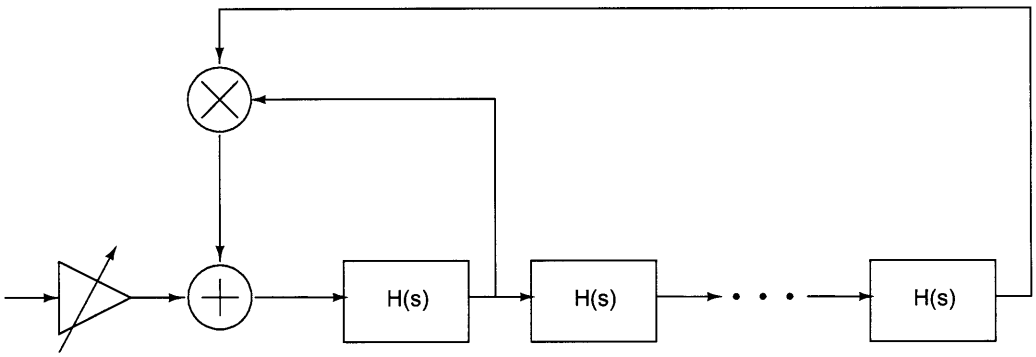

Figure 3-1: Approximate scalar NLL.

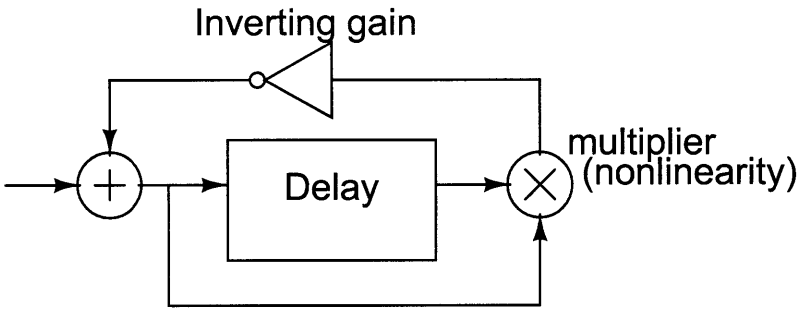

Figure 3-2: Continuous state, continuous time, AFSR[28].

approximate form. The reason to ask this question mainly comes from the considerations on practical implementation of message-passing algorithm in physical systems, especially large-scale systems. But we do not tend to maintain the exact form computation by merely battling with the nonidealities in the circuit through some circuit techniques, which has been studied in [18]. Rather we consider the problem at a higher algorithmic level. We intend to construct some mathematically approximate computation to replace the exact computation in message-passing algorithms. This makes it possible to construct temperature independent or very weakly temperature dependent analogic circuit in a principled way.

It is interesting to point out that the approximate scalar NLL in Fig.3-1 has a similar structure with AFSR in Fig.3-2, which was proposed at the end of Benjamin Vigoda's master thesis [28]. Vigoda originally came up with this configuration in Fig.3-2 from the perspective of nonlinear dynamic system. And he discovered

it mainly by numerical experimenting. Here we study the approximate scalar NLL as a suboptimal estimator derived from message-passing algorithms. The simulation shows the approximate scalar NLL has better synchronization performance than exact form scalar NLL. It is hard to explain by message-passing algorithms. The nonlinear dynamics approach taken in Vigoda's master thesis might provide some hints on the explanation.

Both vector NLL and scalar NLL are non-iterative estimation. We proposed an iterative LFSR synchronization algorithm at the end of this chapter. Then we compare scalar and vector NLL in terms of computational complexity and synchronization performance.

3.1 LFSR Synchronization as Maximum Likelihood Estimation

Consider a sequence of symbols $\mathbf{x}=\{x_k\}$ generated as an m-sequence and transmitted over an additive white Gaussian noise channel(AWGN) with output

$$y_k = x_k + w_k. \tag{3.1}$$

We are interested in the *a posteriori* joint probability mass function for $\mathbf{x}=\{x_k\}$ given the fixed observation $\mathbf{y}=\{y_k\}$:

$$\psi(\mathbf{x}) = p(\mathbf{x}|\mathbf{y}) \propto p(\mathbf{x}) f(\mathbf{y}|\mathbf{x}), \tag{3.2}$$

where $f(\mathbf{y}|\mathbf{x})$ is the priori probability distribution. If the *a priori* distribution $p(\mathbf{x})$ for the transmitted vectors is uniform, then

$$\psi(\mathbf{x}) \propto f(\mathbf{y}|\mathbf{x}). \tag{3.3}$$

The marginal conditional distribution has the form

$$\psi_k(x_k) = p(x_k|\mathbf{y}) \propto \sum_{\sim\{x_k\}} f(\mathbf{y}|\mathbf{x}), \tag{3.4}$$

which marginalizes for each x_k by summing over all variables except x_k. The receiver selects the value of x_k that maximizes each marginal function in order to minimize the probability of error over each symbol. The acquisition of m-sequence is, therefore, through maximum-likelihood(ML) or maximum *a posteriori*(MAP) detection.

The *a posteriori* probability distribution can be represented by a cycle-free trellis through a hidden Markov model. At any given time k, define state variable S_k, the output from transmitter x_k, and the channel observation y_k. The Markov model is hidden because S_k is not observable; only noisy version y_k is observable.

$$
\begin{aligned}
p(\mathbf{x},\mathbf{s}|\mathbf{y}) &\propto p(\mathbf{x},\mathbf{s}) f(\mathbf{y}|\mathbf{x},\mathbf{s}), && (3.5)\\
&= p(\mathbf{x}) p(\mathbf{s}|\mathbf{x}) f(\mathbf{y}|\mathbf{x}), && (3.6)\\
&\propto p(\mathbf{s}|\mathbf{x}) f(\mathbf{y}|\mathbf{x}), && (3.7)\\
&= I(\mathbf{x},\mathbf{s},\mathbf{y}) \prod_{k=0}^{N-1} f(y_k|x_k), && (3.8)\\
&= \prod_{k=0}^{N-1} I_k(s_k, x_k, s_{k+1}) \prod_{k=0}^{N-1} f(y_k|x_k), && (3.9)
\end{aligned}
$$

where assume the *a priori* distribution $p(\mathbf{x})$ is uniform, N is the length of observation, and $I_k(s_k, x_k, s_{k+1})$ is the indication function that constrains the valid configuration of each individual state and transmitted symbol. The normal factor graph for this trellis is in Fig.3-4.

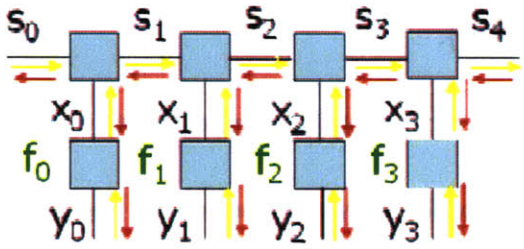

Figure 3-3: A Cycle-free normal factor graph for the joint conditional probability distribution $p(\mathbf{x},\mathbf{s}|\mathbf{y})$.

For each transmitted symbol x_k, the detection gives the *a posteriori* probability as

$$\psi_k(x_k) = p(x_k|\mathbf{y}) \propto p(\mathbf{x},\mathbf{s}|\mathbf{y}). \tag{3.10}$$

Next we explicitly write down the messages and the message-passing procedure.

1. Message initialization:

   (a) messages from y_k to f_k: $\mu_{y_k \to f_k} = \delta(y_k, y'_k)$.

   (b) message at the beginning of observation: $\mu_{s_0 \to I_0} = 1$.

   (c) message at the end of observation: $\mu_{s_N \to I_N} = 1$.

2. Forward message passing through s_k:

$$\mu_{I_{k-1} \to s_k} = \sum_{s_{k-1}} \sum_{x_{k-1}} I(s_k, x_{k-1}, s_{k-1}) \mu_{f_{k-1} \to x_{k-1}} \mu_{I_{k-2} \to s_{k-1}}. \tag{3.11}$$

3. Backward message passing through s_k:

$$\mu_{I_k \to s_k} = \sum_{s_{k+1}} \sum_{x_k} I(s_k, x_k, s_{k+1}) \mu_{f_k \to x_k} \mu_{I_{k+1} \to s_{k+1}}. \tag{3.12}$$

4. Messages passing through x_k:

$$\mu_{x_k \to I_k} = \sum_{y_k} \delta(y_k, y'_k) f(x_k|y_k) = f(x_k|y'_k), \tag{3.13}$$

$$\mu_{I_k \to x_k} = \sum_{s_k} \sum_{s_{k+1}} \mu_{I_{k-1} \to s_k} \mu_{I_{k+1} \to s_{k+1}}. \tag{3.14}$$

5. *A posteriori* probability of x_k:

$$P(x_k|\mathbf{y}) = \mu_{x_k \to I_k} \cdot \mu_{I_k \to x_k}. \tag{3.15}$$

If we are only interested in the ML detection of current state rather than ML detection of the old states by updating the whole history, which is the case for our code acquisition, then the *a posteriori* probability of current symbol x_{N+1} can be easily calculated by the knowledge of formal state $P(x_N|\mathbf{y})$. Forward-backward message-passing is simplified to "forward-only" passing. Note that "forward-only" here means we don't need to calculate the backward messages $\mu_{I_k \to s_k}$. Calculation of $P(x_{N+1}|\mathbf{y})$ must always do $P(x_{N+1}|\mathbf{y}) = \mu_{x_{N+1} \to I_{N+1}} \cdot \mu_{I_{N+1} \to x_{N+1}}$, which is still locally forward-backward message-passing.

3.2 Exact Form Scalar NLL

The trellis processing guarantees the maximum likelihood detection. However, the computation complexity is exponential with r the number of LFSR stages, because the branches at each trellis section is $2^r - 1$. It is desirable to find low-complexity suboptimal algorithm. The hint comes from the fact that there are usually several factor graphs representing one probabilistic structure. For our m-sequence acquisition problem, an alternative factor graph is constructed by considering the generating polynomial of m-sequence. Because there is no need for receiver to update the estimation of all history at each time step, the forward-only message passing is used to achieve the current state estimation. This low-complexity estimator with forward-only message-passing algorithms is called scalar NLL.

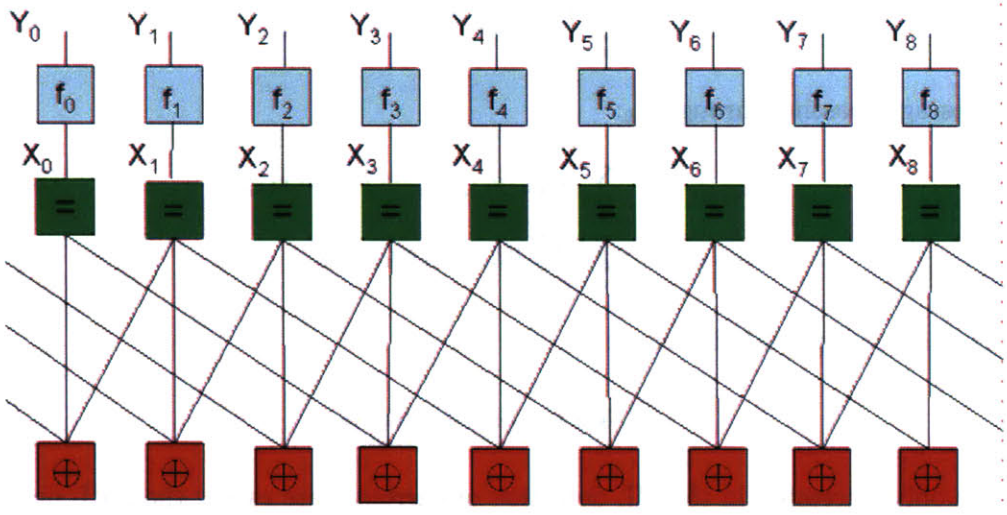

Figure 3-4: A cycle normal factor graph for the joint conditional probability distribution $p(\mathbf{x},\mathbf{s}|\mathbf{y})$.

There are several representations of messages in NLL. Each results different computations [30]. Now we derive the two representations: probability representation and log-likelihood ratio representation. These two are closely related to the development of scalar NLL in following sections. The message computations of soft-Xor and soft-equal in these two representation derived below both use the exact form computation.

3.2.1 Probability Representation

The previous work on scalar NLL has been focused on this representation. Both simulation and implementation [2] [3] has been published. $y(t)$ is the observation of the channel. $x(t)$ is transmitted signal. $f(y(t)|x)$ is the priori probability distribution of two.

1. Mathematical derivation:

   Conversion block:

$$f(y(t)|x=1) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp(-(y(t)-1)^2/2\sigma^2), \tag{3.16}$$

$$f(y(t)|x=-1) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp(-(y(t)+1)^2/2\sigma^2), \tag{3.17}$$

$$p(x=1|y(t)) = \frac{f(y(t)|x=1)}{f(y(t)|x=1)+f(y(t)|x=-1)}, \tag{3.18}$$

$$p(x=-1|y(t)) = \frac{f(y(t)|x=-1)}{f(y(t)|x=1)+f(y(t)|x=-1)}. \tag{3.19}$$

   Soft-Xor gate:

$$p_{xor}(1) = p_{x1}(0)\cdot p_{x4}(0) + p_{x1}(1)\cdot p_{x4}(1), \tag{3.20}$$

$$p_{xor}(0) = p_{x1}(0)\cdot p_{x4}(1) + p_{x1}(1)\cdot p_{x4}(0). \tag{3.21}$$

   Soft-equal gate:

$$p_{eql}(1) = \frac{p_{xor}(1)\cdot p(x=1|y(t))}{p_{xor}(1)\cdot p(x=1|y(t)) + p_{xor}(0)\cdot p(x=-1|y(t))}, \tag{3.22}$$

$$p_{eql}(0) = \frac{p_{xor}(0)\cdot p(x=-1|y(t))}{p_{xor}(1)\cdot p(x=1|y(t)) + p_{xor}(0)\cdot p(x=-1|y(t))}. \tag{3.23}$$

3.2.2 Log-Likelihood Ratio(LLR) Representation

The LLR representation of NLL has been proposed in [30]. The exact form LLR soft-Xor involves hyperbolic tangent function and inverse hyperbolic tangent function as shown in Eq.3.25, which will be approximated in the approximate scalar NLL.

1. Mathematical derivation:

   Conversion block:

$$\mu_y(t) = \frac{2y(t)}{\sigma^2}. \tag{3.24}$$

   Soft-Xor gate:

$$\mu_{xor} = 2\tanh^{-1}(\tanh(\frac{\mu_{x1}}{2}) \cdot \tanh(\frac{\mu_{x4}}{2})). \tag{3.25}$$

   Soft-equal gate:

$$\mu_{eql} = \mu_{xor} + \mu_y. \tag{3.26}$$

3.3 Approximate Scalar NLL in Log-Likelihood Ratio(LLR) Representation

The exact form LLR soft-Xor in Eq.(3.25) has been implemented with Gilbert cell and inverse hyperbolic function circuits stacked at output of Gilbert cell [10]. The circuit realized the following function:

$$V_{out} = 2V_T \tanh^{-1}(\tanh(\frac{V_1}{2V_T})\tanh(\frac{V_2}{2V_T})). \tag{3.27}$$

The log-likelihood ratio was represented as V/V_T. Although the circuit faithfully realized the exact mathematical functions, the thermal voltage $V_T = kT/q$ in the scaling factor makes LLR unavoidably subject to temperature variation. This concern has also been pointed out by Lustenberger in his Ph.D. thesis [18]. The conclusion there was negative on voltage-mode log-likelihood ratio implementation. However, if we gave up the insistence on exact form soft-gates, we will have a new view. Here, we put the inverse hyperbolic function circuit in the input instead at output of Gilbert cell, to recompense the hyperbolic function in Gilbert cell, we will have a complete

four-quadrant analog multiplier as

$$V_{out} = \alpha \frac{V_1}{\alpha} \frac{V_2}{\alpha}, \tag{3.28}$$

where LLR is represented by the ratio $\frac{V}{\alpha}$, and α is an arbitrary voltage that we can set.

Three questions we need to answer are how good this approximation comparing with exact form, how good this approximate NLL performs comparing with exact NLL, and whether this approximation method can be extended in other message-passing circuits, especially the large analog network. The following three sections try to answer these questions.

3.3.1 Multiplier Approximation

The multiplier approximation is written here again:

$$2\tanh^{-1}(\tanh(x/2)\tanh(y/2)) \approx \lambda xy, \tag{3.29}$$

where coefficient λ is a free parameter that we choose to obtain better performance of NLL.

First we look at hyperbolic function $\tanh(x)$ and its inverse function $\tanh^{-1}(x)$. The Taylor series expansions are

$$\tanh(x) = x - \frac{1}{3}x^3 + \frac{2}{15}x^5 - \frac{17}{315}x^7 + \ldots \quad \text{for } |x| < \frac{\pi}{2}, \tag{3.30}$$

$$\tanh^{-1}(x) = x + \frac{1}{3}x^3 + \frac{1}{5}x^5 + \frac{1}{7}x^7 + \ldots \quad \text{for } |x| < 1, \tag{3.31}$$

which tells us they can be approximate by the first term x in Taylor expansion, for at least $|x| < 1$. So roughly we have $2\tanh^{-1}(\tanh(x/2)\tanh(y/2)) \approx xy/2$, with $\lambda = 0.5$, $|x| < 2, |y| < 2$.

To more accurately determine the coefficient λ, we cannot rely on the first order Taylor expansion. We need to do linear fitting of the whole compound function. Numerical calculation in Matlab gives the following values of λ in Eq.(3.29) corresponding to $\{x, y\} \in [-2, 2]^2$.

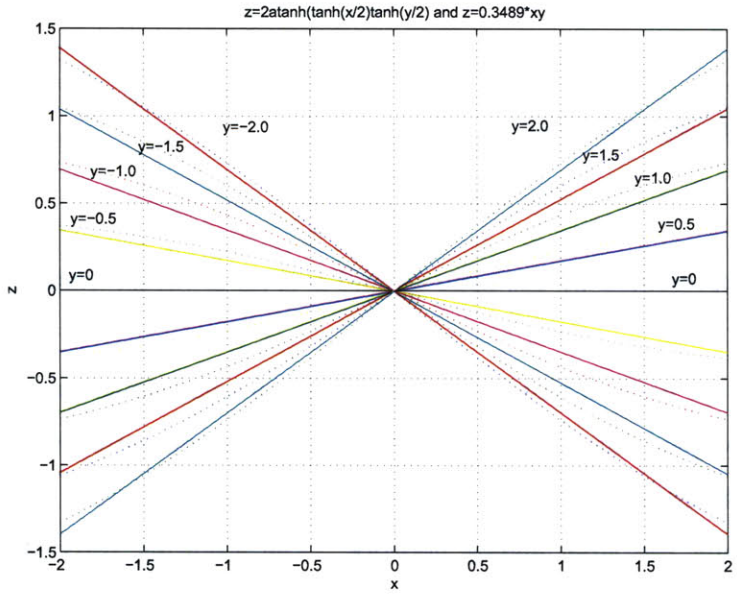

Figure 3-5: Dotted lines are function $2\tanh^{-1}(\tanh(x/2)\tanh(y/2))$ plotted with $x \in [-2, 2]$,$y \in \{\pm 2, \pm 1.5, \pm 1, \pm 0.5, 0\}$. Solid lines are function $0.348 * xy$.

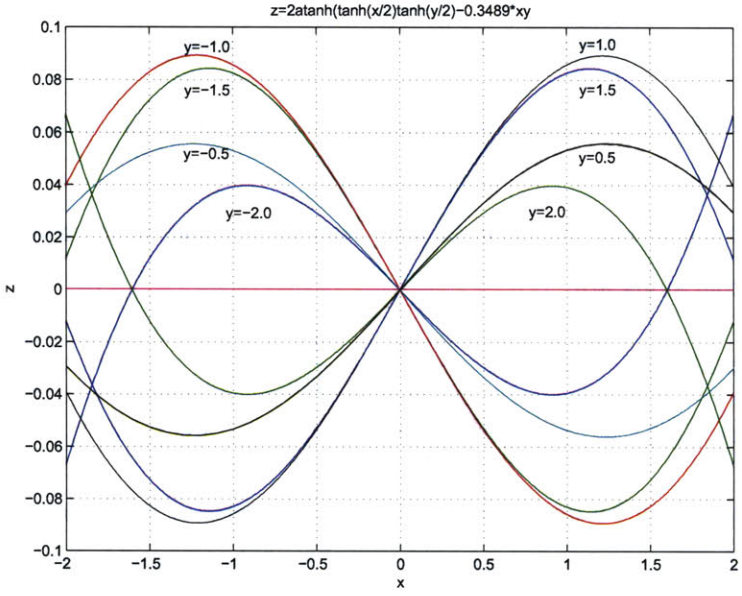

Figure 3-6: $2\tanh^{-1}(\tanh(x/2)\tanh(y/2)) - 0.348 * xy$, plotted for $x \in [-2, 2]$,$y \in \{\pm 2, \pm 1.5, \pm 1, \pm 0.5, 0\}$.

y	λ
± 2	0.3489
$\pm$ 1.6	0.3713
$\pm$ 1.2	0.3907
$\pm$ 0.8	0.4059
$\pm$ 0.4	0.4155
$\pm$ 0.2	0.4180

Table 3.1: Table of λ and y. It shows the variation of λ according to y.

From the above table we can see the real coefficient λ is smaller than 0.5 and it decreases slowly from 0.42 to 0.35 when $|y|$ increases a decade from 0.2 to 2.
This result has a few implications. First it explains why we usually need attenuation after multipliers in the numerical experiments on approximate NLL and approximate analog decoder. Bigger $|y|$ corresponds bigger log-likelihood ratio values. When the estimator is more certain about its decision, it gives higher LLR value, which requires more attenuation in multipliers. This also partly explains our approximate analog decoder needs even bigger attenuation.
Second, the slow variation of λ ensures us that it is reasonable to use a fixed attenuation multiplier in the real implementation. This is what we did in our experiments.

3.3.2 Simulation of Exact Form and Approximate Scalar NLLs

In this section, we will examine the performance of exact form and approximate scalar NLLs by simulation. We first simulate them as pseudo-continuous time systems by over-sampling each chip. The soft-gates are simulated as their mathematical forms. The delay elements are simulated as ideal time delay as assumed in the message-passing algorithm. This simulates the algorithm performance. Then we model the delay element by 8th order Bessel filter and simulate the whole system in CppSim[20][21]. This simulation also models the input output voltage clamping of

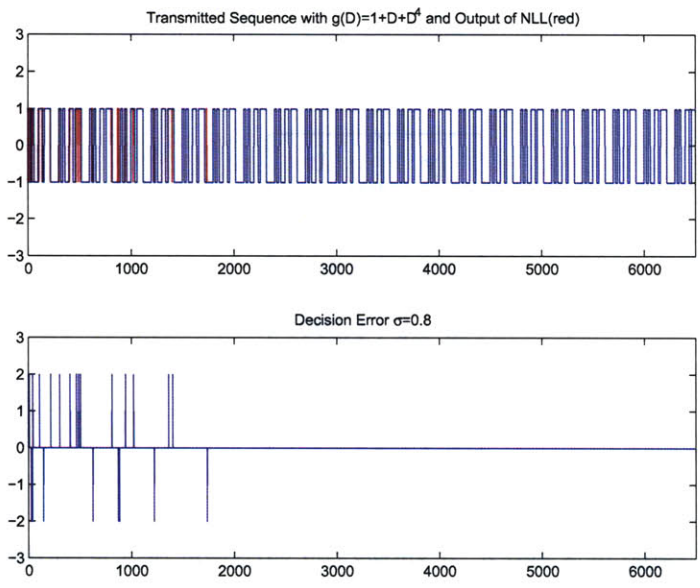

Figure 3-7: Exact form LLR NLL with input $\sigma = 0.8$. The top plot shows the NLL locks unto LFSR sequence after about 50 chips. The error decreases to zero in bottom plot.

the actual multiplier and adder. So it is more close to the actual circuit performance. The detailed discussion of filter design is given in Chapter 4.

Simulation as Pseudo-Continuous Time System

We over-sample each chip by 20 samples in the pseudo-continuous time simulation. The simulation results in Fig.3-7 and Fig.3-8 show the acquisition process of the loops, when the input noise has $\sigma = 0.8$.

We measure the performance of the loop by *probability of synchronization*, which is defined as the probability that NLL has obtained synchronization after k time steps. 600 bits sequence are simulated 1000 times for each noise power level to get this probability.

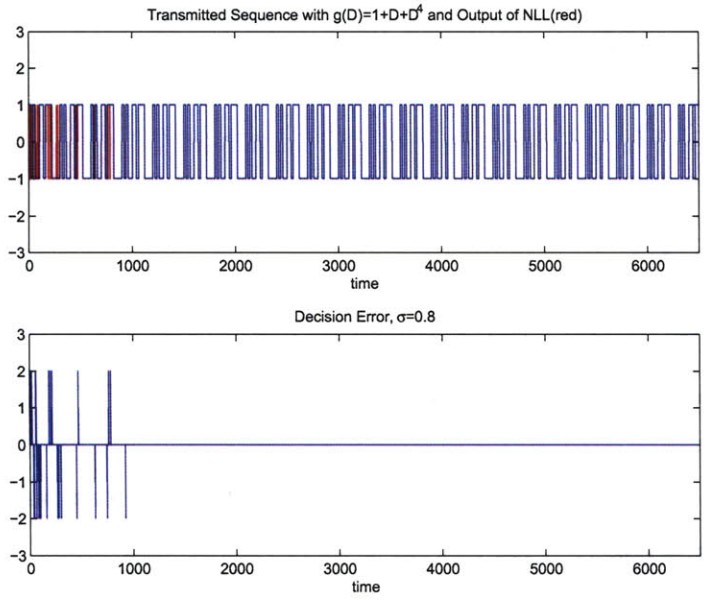

Figure 3-8: Approximate scalar NLL with $\lambda = 0.33$ input $\sigma = 0.8$. The approximate scalar NLL locks unto LFSR sequence at about the same time with exact form NLL.

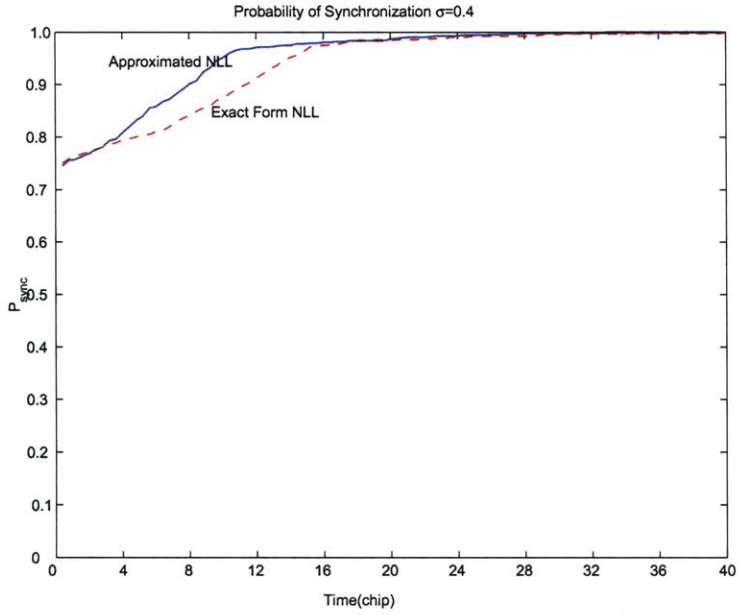

Figure 3-9: The probability of synchronization for both exact form and approximate NLL with noise $\sigma = 0.4$.

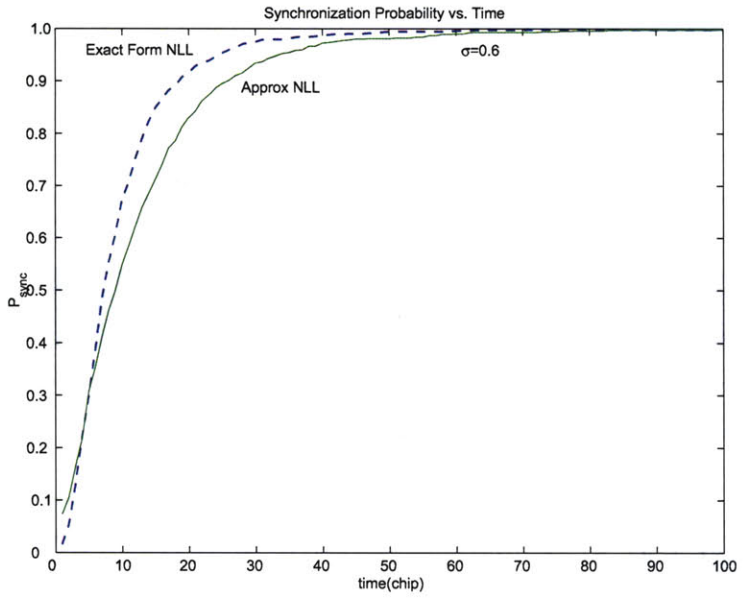

Figure 3-10: The probability of synchronization for both exact form and approximate NLL with noise $\sigma = 0.6$.

The Fig.3-9 and Fig.3-10 show the probability of synchronization of both exact form and approximate NLL with two different noise power $\sigma = 0.4$ and $\sigma = 0.6$. We can see the exact form and approximate NLL almost have the same performance. However, exact form NLL stops synchronizing when σ is greater than 1. Approximate NLL works until σ gets to 1.4 as shown in Fig.3-11.

Simulation as Continuous Time System

In continuous time simulation, we model the delay elements as linear phase low pass filters. The 8th order filter is decomposed into four second-order filter sections cascaded. In this way, numerical precision is maintained, because high order filter simulation is very numerically sensitive. The LFSR signal generator is constructed by using identical analog filters in receiver, multiplier and threshold device. Block diagram in Fig.3-12 is the transmitter structure. Each filter is composed by four second-order filter sections. The generated sequence has typical PN sequence characteristics: sinc envelop with lobes notched at harmonics of chip rate and pulses at harmonics of pe-

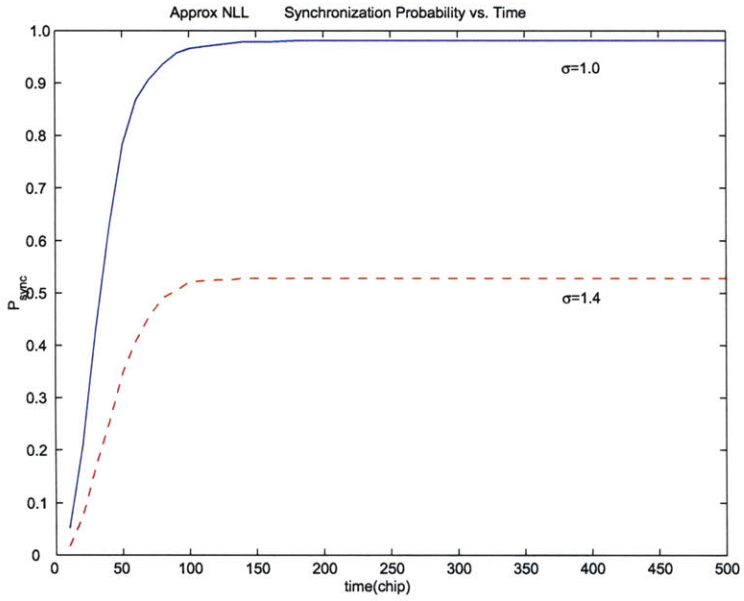

Figure 3-11: The probability of synchronization for approximate scalar NLL with noise $\sigma = 1.0$ and 1.4.

riod frequency as shown in Fig.3-13. The overall system is shown in Fig.3-14. The approximate NLL and exact form NLL are the two loops in the picture. The linear filter is implemented with the same filter in the NLLs.

Simulation time step sets to $0.2\mu s$, which gives $50MHz$ bandwidth. This bandwidth is wide enough for our system operating at $128.8KHz$. Actually we can look at the power spectrum density(PSD) of the signals inside the NLLs. Fig.3-15 shows the PSD of the signal after the delay line. The filters pass the main lobe of LFSR sequence and cuts off the rest spectrum. This indicates the noise bandwidth is 130kHz, which determines the real SNR. We simulated input noise power spectrum density from $0.1 \times 10^{-6}V^2/Hz$ to $6.7 \times 10^{-6}V^2/Hz$, which corresponds to effective SNR from $15.3dB$ to $-2.99dB$. We calculated the performance as mean square error(MSE) defined as $E((x(t) - \hat{x}(t))^2)$, where $E(\cdot)$ takes average, $x(t)$ is clean signal in LFSR transmitter, $\hat{x}(t)$ is the output from NLL or linear filter. Fig.3-16 shows the MSE performance over effective SNR. At low SNR, both NLLs and linear filter tend to make more errors, though NLL is still little better than linear filter. At high SNR,

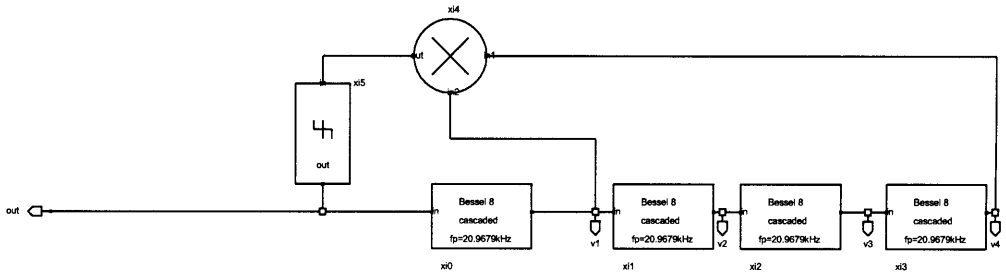

Figure 3-12: LFSR transmitter constructed with analog filters and multipliers. It gives m-sequence with length of 15 .

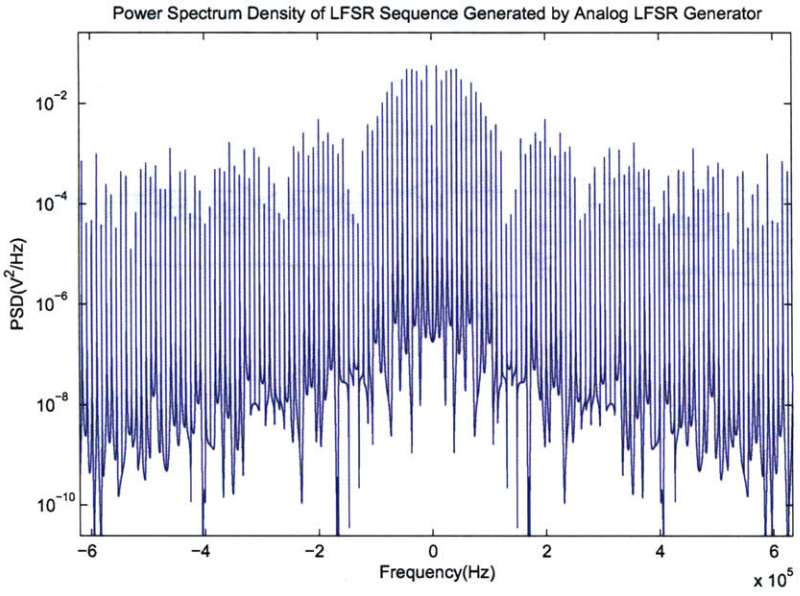

Figure 3-13: LFSR sequence generated by our analog transmitter with chip duration of $7.7\mu s$. The lobes are notched at harmonics of the frequency 128.8KHz. The spectral lines are at harmonics of period frequency 8.59KHz.

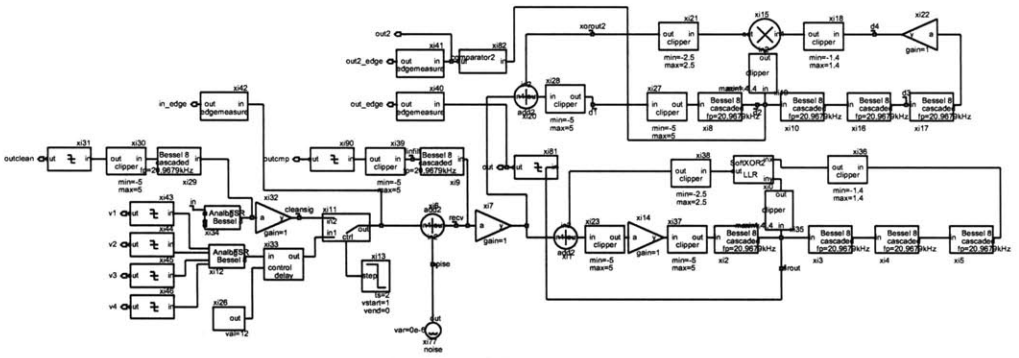

Figure 3-14: Overall system simulated in CppSim. For performance comparison, approximate NLL, exact form NLL and linear filter are constructed. Two loops in the graph are approximate NLL and exact form NLL. Linear filter is implemented with the same filter in NLL to set up same comparison condition.

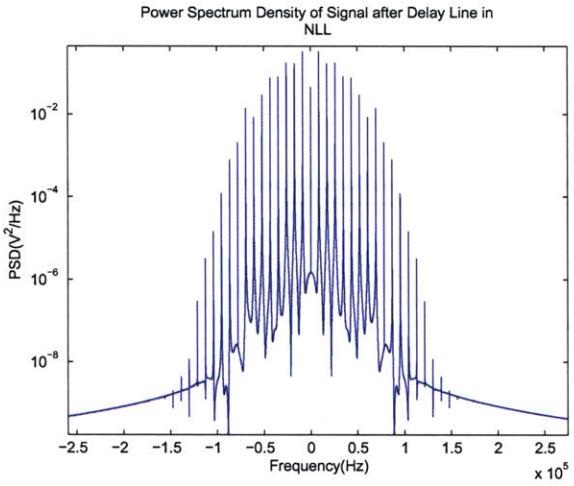

Figure 3-15: The spectrum is truncated by the filter bandwidth. This indicates our noise calculation should only consider in-band noise.

linear filter is sufficient to clean the noise. At SNR range 5dB to 0.5dB, NLL wins over linear filter more. To further evaluate the noise rejection performance of scalar NLL and linear filter, we may calculate their output signal-to-noise ratios and noise figures. In next section, we will discuss vector NLL which promises a significant improvement of scalar NLL.

As shown in the previous simulation, in low SNR regime the approximate scalar NLL performs even better than the NLL with exact form message-passing algorithms. It is interesting to ask if the approximate message-passing algorithm is also useful for bigger analogic systems. Our preliminary study on a memory one, tail-biting analog decoder with 32 soft-gates showed the basic functionality of approximate message-passing algorithm on this network. Our result will be presented in other document.

3.4 First-Order Markov Structured Vector NLL

In last section we derived the vector NLL using full trellis processing, which demands the computation exponentially proportional to the scale of the LFSR sequence. The messages in the factor graph are the joint probability mass function of random vari-

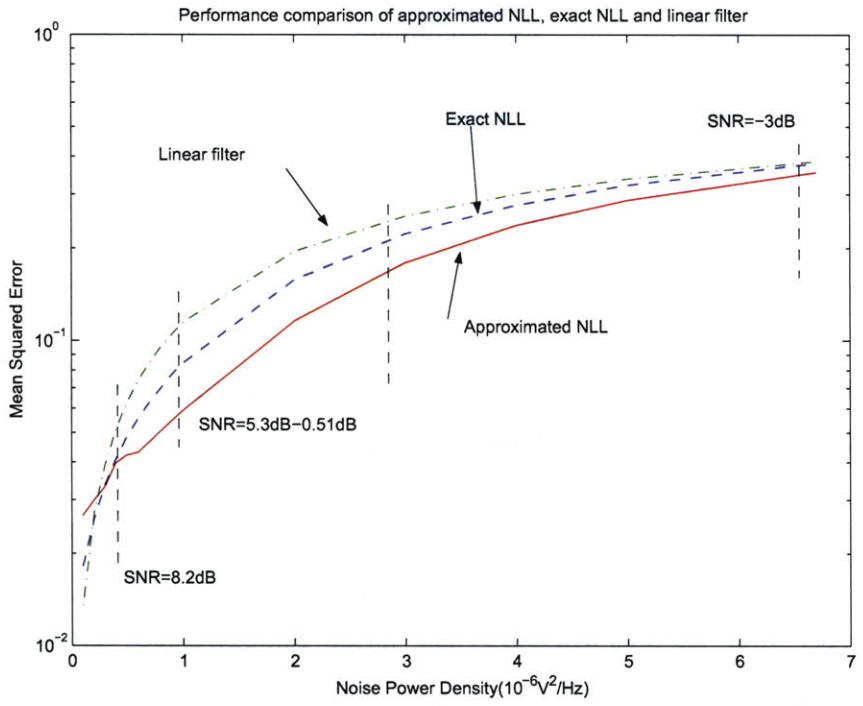

Figure 3-16: The simulation result for mean square error performance for approximate scalar NLL, exact form scalar NLL, and linear filter. At very low noise(SNR$> 8dB$) regime, the three perform very close. At very high noise(SNR$< -3dB$) regime, all of them suffer big MSE. In between, the approximate scalar NLL outperforms the other two, especially when SNR is around 5dB to 0.5dB.

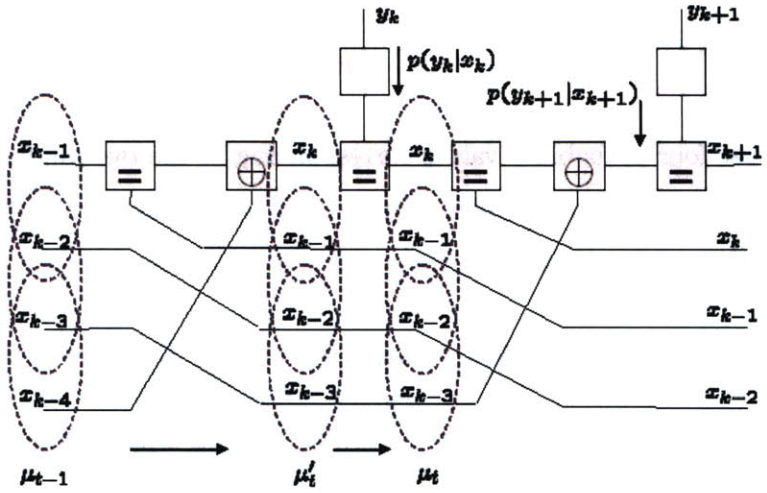

Figure 3-17: Factor Graph of a $g(D) = 1 + D + D^4$ LFSR sequence

ables $(x_k, x_{k-1}, ..., x_{k-4})$. It is desirable to design a less complex algorithm which will give reasonable performance. The light shines in when we approximate the joint probability mass function by imposing more probabilistic structure on the random variables [4]. We will derive the detailed message-passing algorithm according to the factor graph in Fig.3-17.

$(x_k, x_{k-1}, ..., x_{k-4})$ are the estimated state of LFSR sequence. (y_k, y_{k+1}) are the noisy input to the receiver. $(\mu_{t-1}, \mu_t', \mu_t)$ are the messages passing on the graph which correspond to the probability mass functions of the estimated state of LFSR sequence. The subscripts denote the time step. $\mu_t'(x_k, x_{k-1}, ..., x_{k-3})$ is a prediction from the message $\mu_{t-1}(x_{k-1}, ..., x_{k-4})$ and is updated by the likelihood of received information $P(y_k|x_k)$ to generate the message for next step, which is $\mu_t(x_k, ..., x_{k-3})$. As shown in the Fig.3-17, the estimated variables are grouped in pair to imply the first order Markov structure. The joint probability mass function at time step $t-1$ can be written as:

$$\mu_t(x_k, x_{k-1}, ..., x_{k-3}) = \mu_t(x_k)\mu_t(x_{k-1}|x_k)\mu_t(x_{k-2}|x_{k-3}), \quad (3.32)$$

$$= \frac{\mu_t(x_k, x_{k-1})\mu_t(x_{k-1}, x_{k-2})\mu_t(x_{k-2}, x_{k-3})}{\mu_t(x_{k-1})\mu_t(x_{k-2})}. \quad (3.33)$$

Now we first derive the prediction message μ_t' from μ_{t-1}:

$$\mu_t^{'}(x_{k-i}, x_{k-i-1}) = \sum_{\sim\{x_{k-i}, x_{k-i-1}\}} \delta(x_k \oplus x_{k-1} \oplus x_{k-4})\mu_{t-1}(x_{k-1}, ..., x_{k-4}), \tag{3.34}$$

$$= \sum_{\sim\{x_{k-i}, x_{k-i-1}\}} \delta(x_k \oplus x_{k-1} \oplus x_{k-4}) \frac{\mu_{t-1}(x_{k-1}, x_{k-2})\mu_{t-1}(x_{k-2}, x_{k-3})\mu_{t-1}(x_{k-3}, x_{k-4})}{\mu_{t-1}(x_{k-2})\mu_{t-1}(x_{k-3})}, i = 0, 1, 2. \tag{3.35}$$

Each message has four probability values corresponding to the two random variables taking $00, 01, 10, 11$. We give a detailed formula of $\mu_t^{'}(x_k = 0, x_{k-1} = 0)$. Here we use the short notation $\mu_{t-1}(k-2, k-3)_{11}$ for $\mu_{t-1}(x_{k-2} = 1, x_{k-3} = 1)$.

$$\begin{aligned}
\mu_t^{'}(x_k = 0, x_{k-1} = 0) = & \frac{\mu_{t-1}(k-1, k-2)_{00}\mu_{t-1}(k-2, k-3)_{00}\mu_{t-1}(k-3, k-4)_{00}}{\mu_{t-1}(k-2)_0\mu_{t-1}(k-3)_0} \\
& + \frac{\mu_{t-1}(k-1, k-2)_{00}\mu_{t-1}(k-2, k-3)_{01}\mu_{t-1}(k-3, k-4)_{10}}{\mu_{t-1}(k-2)_0\mu_{t-1}(k-3)_1} \\
& + \frac{\mu_{t-1}(k-1, k-2)_{01}\mu_{t-1}(k-2, k-3)_{10}\mu_{t-1}(k-3, k-4)_{00}}{\mu_{t-1}(k-2)_1\mu_{t-1}(k-3)_0} \\
& + \frac{\mu_{t-1}(k-1, k-2)_{01}\mu_{t-1}(k-2, k-3)_{11}\mu_{t-1}(k-3, k-4)_{10}}{\mu_{t-1}(k-2)_1\mu_{t-1}(k-3)_1}.
\end{aligned} \tag{3.36}$$

The other three probability values of $\mu_t^{'}(x_k, x_{k-1})$ can be derived similarly by fulfilling the parity-check constraint.

For the remaining prediction messages, we have

$$\mu_t^{'}(x_{k-1}, x_{k-2}) = \mu_{t-1}(x_{k-1}, x_{k-2}), \tag{3.37}$$

$$\mu_t^{'}(x_{k-2}, x_{k-3}) = \mu_{t-1}(x_{k-2}, x_{k-3}). \tag{3.38}$$

The second step is to obtain μ_t by updating prediction message $\mu_t^{'}$ through the incoming likelihood message $P(y_k|x_k)$.

The equality constraint in the factor graph imposes the following relationship:

$$\mu_t(x_k, x_{k-1}, ..., x_{k-3}) = P(y_k|x_k)\mu_t^{'}(x_k, x_{k-1}, ..., x_{k-3}). \tag{3.39}$$

By marginalization and applying first order Markov property, the messages for three pairs $\{x_k, x_{k-1}\}, \{x_{k-1}, x_{k-2}\}, \{x_{k-2}, x_{k-3}\}$ can be obtained as

$$\mu_t(x_k, x_{k-1}) = \sum_{\sim\{x_k, x_{k-1}\}} P(y_k|x_k)\mu_t^{'}(x_k, x_{k-1}, ..., x_{k-3}), \tag{3.40}$$

$$= P(y_k|x_k)\mu_t^{'}(x_k, x_{k-1}). \tag{3.41}$$

$$\mu_t(x_{k-1}, x_{k-2}) = \sum_{\sim\{x_{k-1}, x_{k-2}\}} P(y_k|x_k)\mu_t^{'}(x_k, x_{k-1}, ..., x_{k-3}), \tag{3.42}$$

$$= \sum_{\sim\{x_{k-1}, x_{k-2}\}} P(y_k|x_k)\frac{\mu_t^{'}(x_k, x_{k-1})\mu_t^{'}(x_{k-1}, x_{k-2})\mu_t^{'}(x_{k-2}, x_{k-3})}{\mu_t^{'}(x_{k-1})\mu_t^{'}(x_{k-2})}, \tag{3.43}$$

$$= \sum_{x_k} P(y_k|x_k)\mu_t^{'}(x_k, x_{k-1})(\sum_{x_{k-3}} \mu_t^{'}(x_{k-2}, x_{k-3}))\frac{\mu_t^{'}(x_{k-1}, x_{k-2})}{\mu_t^{'}(x_{k-1})\mu_t^{'}(x_{k-2})}, \tag{3.44}$$

$$= \frac{\mu_t^{'}(x_{k-1}, x_{k-2})}{\mu_t^{'}(x_{k-1})} \sum_{x_k} P(y_k|x_k)\mu_t^{'}(x_k, x_{k-1}), \tag{3.45}$$

$$= \frac{\mu_t^{'}(x_{k-1}, x_{k-2})}{\mu_t^{'}(x_{k-1})} \sum_{x_k} \mu_t(x_k, x_{k-1}), \tag{3.46}$$

$$= \mu_t^{'}(x_{k-1}, x_{k-2})(\frac{\sum_{x_{k-1}} \mu_t(x_k, x_{k-1})}{\sum_{x_{k-1}} \mu_t^{'}(x_k, x_{k-1})}). \tag{3.47}$$

Here we can see the new message $\mu_t(x_k, x_{k-1})$ is proportional to the corresponding intermediate message $\mu_t^{'}(x_k, x_{k-1})$ scaled by a ratio of updated probability of variable x_{k-1} to that of the old probability.

The new message $\mu_t(x_{k-2}, x_{k-3})$ for the pair $\{x_{k-2}, x_{k-3}\}$ can be obtained in the same way, we neglect the derivation here.

$$\mu_t(x_{k-2}, x_{k-3}) = \mu_t^{'}(x_{k-2}, x_{k-3})(\frac{\sum_{x_{k-1}} \mu_t(x_{k-1}, x_{k-2})}{\sum_{x_{k-1}} \mu_t^{'}(x_{k-1}, x_{k-2})}). \tag{3.48}$$

For clarity and further reference, the intermediate messages and updated new messages are summarized as below:

Intermediate messages:

$$\mu_t^{'}(x_k, x_{k-1}) = \sum_{\sim\{x_k, x_{k-1}\}} \delta(x_k \oplus x_{k-1} \oplus x_{k-4})\frac{\mu_{t-1}(x_{k-1}, x_{k-2})\mu_{t-1}(x_{k-2}, x_{k-3})\mu_{t-1}(x_{k-3}, x_{k-4})}{\sum_{x_{k-1}} \mu_{t-1}(x_{k-1}, x_{k-2}) \sum_{x_{k-2}} \mu_{t-1}(x_{k-2}, x_{k-3})}, \tag{3.49}$$

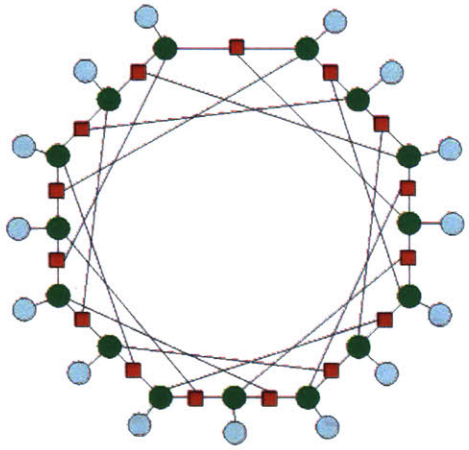

Figure 3-18: A Cyclic factor graph for the joint conditional probability distribution $p(\mathbf{x},\mathbf{s}|\mathbf{y})$.

$$\mu_t^{'}(x_{k-1}, x_{k-2}) = \mu_{t-1}(x_{k-1}, x_{k-2}), \tag{3.50}$$

$$\mu_t^{'}(x_{k-2}, x_{k-3}) = \mu_{t-1}(x_{k-2}, x_{k-3}). \tag{3.51}$$

New messages:

$$\mu_t(x_k, x_{k-1}) = P(y_k|x_k)\mu_t^{'}(x_k, x_{k-1}), \tag{3.52}$$

$$\mu_t(x_{k-1}, x_{k-2}) = \mu_t^{'}(x_{k-1}, x_{k-2})(\frac{\sum_{x_k} \mu_t(x_k, x_{k-1})}{\sum_{x_k} \mu_t^{'}(x_k, x_{k-1})}), \tag{3.53}$$

$$\mu_t(x_{k-2}, x_{k-3}) = \mu_t^{'}(x_{k-2}, x_{k-3})(\frac{\sum_{x_{k-1}} \mu_t(x_{k-1}, x_{k-2})}{\sum_{x_{k-1}} \mu_t^{'}(x_{k-1}, x_{k-2})}). \tag{3.54}$$

3.5 LFSR Synchronization by Iterative Detection

The loopy factor graph for 16 states m-sequence is shown in Figure 3-18.

We can derive the iterative message-passing schedule as following:

- Initialization:

$$LLR(p_i) = log\frac{Pr[v_i = 1]}{Pr[v_i = 0]}. \tag{3.55}$$

- Bit-to-check messages:

$$LLR^{(k)}(q_{ji}) = \sum_{j' \in M(i)\backslash\{j\}} LLR^{(k-1)}(r_{j'i}) + LLR(p_i). \tag{3.56}$$

- Check-to-bit messages:

$$LLR^{(k)} = \min_{i' \in L(j)\backslash\{i\}} (|LLR^{(k)}(q_{ji'})|). \tag{3.57}$$

- Iterate until detection: if $k < N_{it}, k = k + 1$, go back to step 1; otherwise, compute the posteriori information:

$$LLR^{(k)}(q_i) = \sum_{j' \in M(i)} LLR^{(k)}(r_{j'r}) + LLR(p_i). \tag{3.58}$$

Make decision: $\hat{v}_i^{(k)} = 1$, if $LLR^{(k)}(q_i) < 0$, and $\hat{v}_i^{(k)} = 0$ otherwise.

3.6 Comparison of Different Estimation Algorithms

The full trellis is the optimal maximum-likelihood estimation by passing the full joint probability distribution on the graph. The computation complexity thus is exponentially proportional the LFSR scale as 2^m(m is the number of delay elements in the LFSR). Scalar NLL assumes the messages are independent. So it only passes the factorized joint probability distribution. The first order Markov structure is the intermediate stage between the two extremes. It considers the Markov chain relation of messages. Some simulation results of first-order Markov chain structure and full trellis structure are published in [4]and [30]. For a fixed synchronization probability, maximum-likelihood estimation generally wins scalar NLL by 4dB SNR. The first-order Markov chain sits in the middle winning scalar NLL by 2dB SNR. This is an incentive for us to consider the circuit implementation of vector NLL.

Chapter 4

Analogic Implementation of NLLs

In Chapter 3, we approached the LFSR synchronization problem with the tool of message-passing algorithm. We first treated this problem as a statistical estimation problem. In the language of message-passing, we pointed out there is a hierarchical structure of NLLs with different complexity level and performance, ranging from simple scalar NLL to intermediate complex first order Markov NLL to full trellis processing. We took it as an exercise to derive the mathematical expressions for message computation for three different estimators. We discussed the exact form scalar NLL in three different representations. Then we raised the question of approximate message-passing with the motivation from practical implementation. Parity-check constraint(e.g. soft-Xor) and Equality constraint(e.g. soft-equal) are two fundamental computations in any message-passing algorithm including NLL and other analog decoder or receivers. Our approximate soft-Xor computation makes the scaling factor of LLR almost independent of temperature in analogic implementation.

4.1 Fundamental Circuit

In order to understand how analogic circuit works, we review the translinear principle and well-know Gilbert four-quadrant multiplier, which is fundamental building block of analogic circuits.

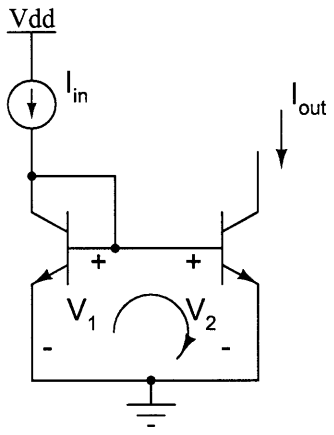

Figure 4-1: Current mirror as a simple current mode translinear circuit with negative feedback creating inverse function.

4.1.1 Static Translinear Principle

The static translinear principle rests on two key results [24]. One is mathematics of logarithms and the other is physics of transistor. The rules for adding and subtracting logarithms are as simple as following:

$$log(ab) = log(a) + log(b), \tag{4.1}$$

$$log(\frac{a}{b}) = log(a) - log(b). \tag{4.2}$$

The physics of transistor provides cheap implementation of nearly-ideal exponential function as following:

$$I = \lambda I_s e^{\eta V/\phi_t}, \tag{4.3}$$

where λ represents scaling in I_s due to transistor geometry scaling. η is a process-dependent constant. For MOS transistor in subthreshold region, η is about 0.7. For bipolar transistor, η is about 1.

We are accustomed to voltage-mode circuit, where we take voltage as the input signal and the exponential properties of the transistor produce current output. Alternative

way is to design current-mode circuit where current is the input signal and we use negative feedback to create the inverse function of exponential — the logarithm. A simple and familiar circuit is shown in in Fig.4-1. Negative feedback creates the logarithm function as: $V_1 = \frac{\phi_t}{\eta} \ln[\frac{I_{in}}{\lambda_1 I_s}]$ and $V_2 = \frac{\phi_t}{\eta} \ln[\frac{I_{out}}{\lambda_2 I_s}]$. The arrow points out the translinear loop, along which the base-emitter voltage drops sum to zero: $V_1 - V_2 = 0$. If the two transistors are matched both in temperature and process, the current density will hold the equality constraint:

$$\frac{I_{in}}{\lambda_1} = \frac{I_{out}}{\lambda_2}$$

This constraint on current densities can be generalized when we connect more base-emitter diodes in the translinear loop. Here instead of common textbook style definition of the translinear principle, we copy a paragraph from the first paper of the famous 1968 JSSC twin-paper by Barrie Gilbert [7]. In this seminal paper, Barrie developed a new wide-band amplifier by marrying the differential amplifier and current mirror, which further bears the fruit of current product-quotient configuration, four quadrature multiplier, and multiple-input matrix multiplication.

"...in a closed loop containing an even number of perfect exponential diode voltages(not necessarily two-terminal devices) arranged in cancelling pairs, the currents are such that the product of the currents in diodes whose voltage polarities are positive with respect to a node in the loop is exactly proportional to the product of the currents in diodes whose voltages are negative respect to that node, the constant of proportionality being the ratio of the product of the saturation currents of the former set of diodes to that of the latter set."
Saturation currents mentioned here are proportional to area which is contained in parameter λ_n in the following formula:

$$\prod_{n \in CW} \frac{I_n}{\lambda_n} = \prod_{m \in CCW} \frac{I_m}{\lambda_m}. \tag{4.4}$$

In the following sections we will see that the current product-quotient configuration mentioned in the same paper [7] by Barrie Gilbert is crucial for implementing

the first order Markov vector NLL.

4.1.2 Gilbert Multiplier Core Circuit

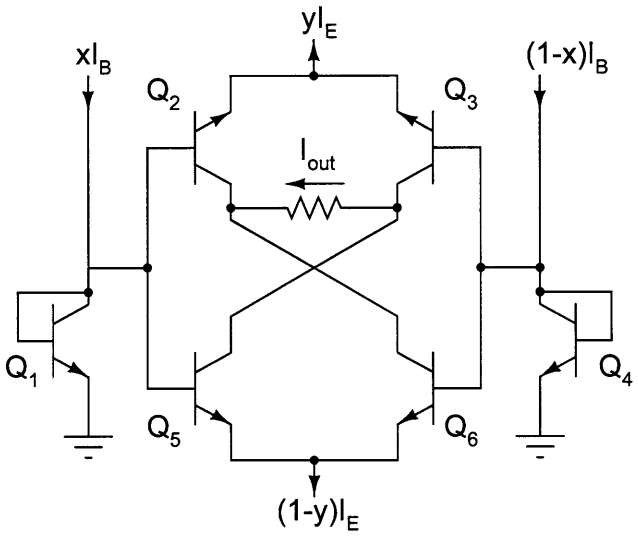

Figure 4-2: Gilbert multiplier core circuit.

In the sister paper[8] in 1968 JSSC, Barrie Gilbert introduced a high-accuracy high-speed two signal four-quadrant multiplier. Fig.4-2 shows a less familiar way to draw Gilbert cell given by Barrie himself. Transistors Q_1, Q_4, Q_5, Q_6 and Q_1, Q_2, Q_3, Q_4 are connected as two wide-band amplifiers described in his first paper[7]. These two amplifiers are then collector-coupled through Q_2, Q_3 and Q_5, Q_6 to form a translinear loop. The translinear principle gives us:

$$I_{Q2} = xyI_E, \tag{4.5}$$

$$I_{Q3} = (1-x)yI_E, \tag{4.6}$$

$$I_{Q5} = x(1-y)I_E, \tag{4.7}$$

$$I_{Q6} = (1-x)(1-y)I_E. \tag{4.8}$$

The differential output is

$$I_{out} = I_{Q2} + I_{Q6} - I_{Q3} - I_{Q5}, \tag{4.9}$$

$$= I_E(2x-1)(2y-1), \tag{4.10}$$

$$= I_E XY. \tag{4.11}$$

where $X = 2x - 1$,$Y = 2y - 1$. The voltage-mode version of Gilbert Cell looks more

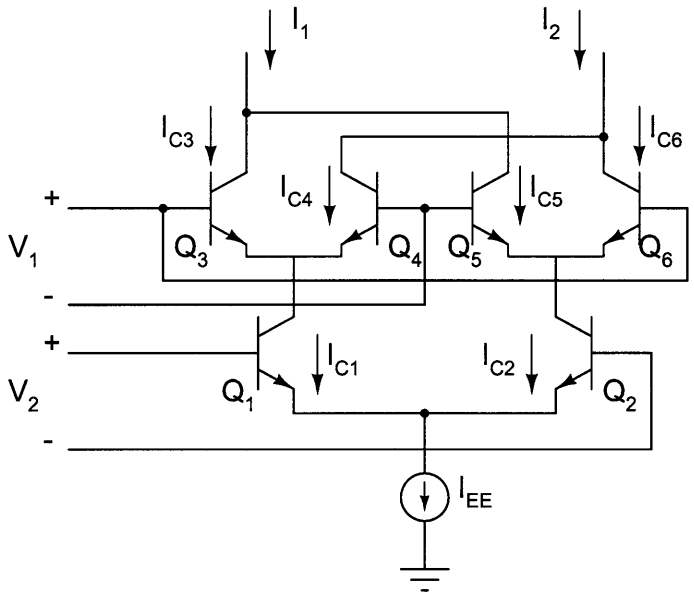

Figure 4-3: Gilbert multiplier core circuit in a more familiar form.

modern in Fig.4-3. Current-mirror connection at input is replaced by voltage input. Differential pair has tanh-like I-V characteristics. The differential output current finally has the form:

$$I_{out} = I_{EE} \tanh(\frac{V_1}{2V_T}) \tanh(\frac{V_2}{2V_T}). \tag{4.12}$$

As we have already derived in earlier chapter, the log-likelihood ratio representation of soft-Xor gate has similar form as 4.12. Actually some implementation of analog decoders uses V/V_t as log-likelihood ratio[10]. The drawback is V/V_t is temperature dependent. Our approximate scalar NLL uses direct multiplication of two voltages, which needs an inverse hyperbolic tangent function to compensate tanh function. The simple circuit in Fig.4-4 accomplishes this task[9]. Assume I_1, I_2 can be decomposed into common-mode current and differential current like $I_1 = I_0 + K_1 V_1$, $I_2 = I_0 - K_1 V_1$.

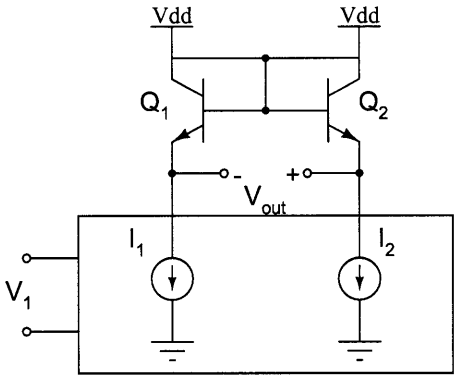

Figure 4-4: Two diodes realize inverse hyperbolic tangent function.

Let $V_0 = I_0/K_1$. Then

$$V_{out} = V_T \ln(\frac{I_0 + K_1 V_1}{I_s}) - V_T \ln(\frac{I_0 - K_1 V_1}{I_s}), \tag{4.13}$$

$$= 2V_T \tanh^{-1}(\frac{V_1}{V_0}). \tag{4.14}$$

Connect inverse tanh block to the input of Gilbert multiplier in Fig.4-3 which can give wide input range multiplier $I_out = I_{EE}(\frac{V_1}{V_0})(\frac{V_2}{V_0})$. Another important feature is now the scaling from LLR to voltage is temperature independent in this equation. The transconductance circuit hidden in the block in Fig.4-4 has transconductance of K in Eq.4.13. In above threshold MOS circuit, temperature dependent term in transconductance involves the carrier mobility. However the mobility has a very weak temperature dependence comparing with kT/q. It is also possible to use some cancellation technique to further make transconductance circuit even more temperature independent. In conclusion, the approximate soft-Xor computation can yield analogic circuit which is more robust than the conventional soft-Xor used in [10].

4.1.3 Analog Linear Phase Filter

In high speed integrated circuit system, people usually use passive delay line[1] or digital feedback structure like DLL to achieve tight timing requirement. In low frequency system such as the scalar NLL running at a few 100K chip/s, passive delay line might require excessive component values, which is usually a hard problem for integrated

circuits. It may be also problematic for discrete components where mismatch might cause severe sensitivity issues especially for high order filters. We choose an integrated switched capacitor delay filter in our implementation, because SC filter gives wider tunable frequency response compared with RC op amp filter. Also because it gives more selections on group delays.

Some earlier work on NLL such as [11][29] proposed to use same but arbitrary continuous time filters in both transmitter and receiver. The transmitted sequence is an arbitrary sequence. In our work, we want to synchronize with LFSR sequence. So NLL needs proper analog delay to set up the same dynamics as the LFSR counterpart. In the work[29], fifth order Chebyshev filter has been used for analog delay. We choose eighth order Bessel filter in our implementation. The reason will be clear after we discuss the characteristics of Bessel filter and comparison with other types of filters.

General Bessel filter has the transfer function as

$$T_n(s) = \frac{B_n(0)}{B_n(s)}, \tag{4.15}$$

where $B_n(s)$ is n-th order Bessel polynomial. Storch was the first one who related the Bessel polynomial to this type of filter through his derivation[26]. His approach [17]published 50 years ago is perhaps the most direct way to see what makes Bessel filter special to be a delay element.

Remember the transfer function for an ideal time delay element with normalized delay T=1 is:

$$T(s) = e^{-sT} = e^{-s}. \tag{4.16}$$

and substitute the hyperbolic function identity $e^s = \cosh(s) + \sinh(s)$ into above equation:

$$T(s) = \frac{1}{\sinh(s) + \cosh(s))} = \frac{1/\sinh(s)}{1 + \coth(s)}. \tag{4.17}$$

If we can write $\coth(s)$ in the form of $A(s)/B(s)$, then the transfer function becomes:

$$T(s) = \frac{B(s)/\sinh(s)}{A(s) + B(s)}. \tag{4.18}$$

Since we know the Taylor expansion of $\sinh(s)$ is

$$\sinh(s) = s + \frac{s^3}{3!} + \frac{s^5}{5!} + \frac{s^7}{7!} + \ldots. \tag{4.19}$$

the key step is to obtain the infinite continued fraction expansion of $\cosh(s)$, which by repeating division of $\sinh(s)$ and $\cosh(s)$ can be obtained as:

$$\coth(s) = \frac{1}{s} + \frac{1}{\frac{3}{s} + \frac{1}{\frac{5}{s} + \frac{1}{\frac{7}{s} + \ldots}}}. \tag{4.20}$$

Here we almost get the filter transfer function. Only thing left is to truncate the continued fraction of coth by n and add the numerator and denominator together for the n-th order filter. For example, for $n = 3$,

$$\coth(s) = \frac{6s^2 + 15}{s^3 + 15s}. \tag{4.21}$$

Adding numerator and denominator gives 3rd order Bessel filter, which was first discovered by Storch in this situation:

$$\mathcal{B}_3 = s^3 + 6s^2 + 15s + 15. \tag{4.22}$$

He also proved that the above process gives maximally flat delay at $w = 0$. The maximally flat delay requires the delay $D_n(w) = -d\theta/dw$ to have:

$$\frac{dD_n(w)}{dw}|_{w=0} = 0 \quad \text{and } D_n(0) = 1. \tag{4.23}$$

The Butterworth response is derived from the requirement that magnitude function be maximally flat at small w. This give less linear phase and peaking in delay values at higher order. This is mainly due to the higher pole Q factors. The Chebyshev filter is derived from the requirement of equal-ripple attenuation in the passband which also gives considerable amount of delay peaking at the cut-off frequency. The elliptical filter gives worse delay because of even higher Q factors. Bessel filter has the maximally flat delay response through the passband. For time domain response, Bessel filter has almost no overshoot for step response, however slower rise time. For our application, maximally flat delay response is most desired. Little overshoot in time domain is also useful. Rising time is not a problem especially when the transmitter and receiver are both implemented with the same filters.

4.2 Implementation of Approximate Scalar NLL

The approximate scalar NLL has been implemented in discrete components with commercial integrated circuits. The soft-XOR gate is approximate as analog multiplier. The soft-equal gate is an adder in log-likelihood ratio representation. These two gates are implemented by Analog Device IC AD835 four-quadrant analog multiplier. Delay elements are implemented by Linear Technology LTC1064-3 8th order linear phase low pass filter. The log-likelihood ratio conversion circuit is actually a variable gain amplifier, which could be implemented by any op amp in a non-invertible gain configuration. The decision is finally made by a threshold device implemented by LM139 comparator. The whole system is modelled and simulated in CppSim as discussed in Chapter 3. Now we discuss each component in detail.

4.2.1 Log-likelihood Ratio Conversion Circuit

We should be clear about how to represent mathematical quantity log-likelihood ratio by a physical quantity voltage in our circuit. As pointed out in the discussion in approximation of soft-xor computation, LLR $\mu(t)$ is encoded as a voltage signal $V(t)$ scaled by a voltage called V_{scale}.

$$\mu(t) = V(t)/V_{scale}. \tag{4.24}$$

In some analog decoder design, the V_{scale} is set by the circuit topology as thermal voltage kT/q. By using analog multiplier approximation, we can set V_{scale} to an arbitrary voltage. We choose $V_{scale} = 1V$ in our circuit. This means when $V(t) = 5V$, the LLR is 5, Prob(1)/Prob(0)=e^5=148. The conversion from noisy signal to log-likelihood ratio in Eq.(3.24) becomes

$$\mu(t) = V(t)/V_{scale} = \frac{2Ay(t)}{\sigma^2}, \tag{4.25}$$

therefore,

$$V(t) = \frac{2V_{scale}A}{\sigma^2}y(t), \tag{4.26}$$

A is the magnitude of transmitted signal in V. σ^2 is noise power in V^2. So the coefficient $2V_{scale}A/\sigma^2$ is a unitless quantity, which can be implemented as the gain of

a non-inverting amplifier with op-amp. It is helpful to group this gain into two parts

$$\text{gain} = 2(\frac{V_{scale}}{A})(\frac{A^2}{\sigma^2}). \tag{4.27}$$

The first part is the ratio of the signal magnitude set in the receiver over that of the transmitter. Usually the transmitted power is known by receiver. So this value is fixed. The next part is the signal-to-noise power ratio, which can be varying all the time. When noise is small, it is a gain. When noise is big, it becomes attenuation. The possible way seems to feedback the estimation error to control the gain, which needs another loop. In our circuit, we fix the gain to 1—directly pass the signal. This is assumed the receiver has no information of the channel noise level, which is a reasonable assumption. Also for practical implementation, especially the high speed data link, the variable gain amplifier must have very low noise figure which is an overhead for power and system complexity.

4.2.2 Soft-Equal Gate and Approximate Soft-Xor Gate

Analog Device provides some nice analog multiplier with summing port. This feature is highly suitable to implement functional block soft-Xor and soft-equal gates. It is interesting to see that each AD835 gives all the fundamental computation that message-passing needs. The Fig.4-5 is the functional block diagram of AD835. AD835

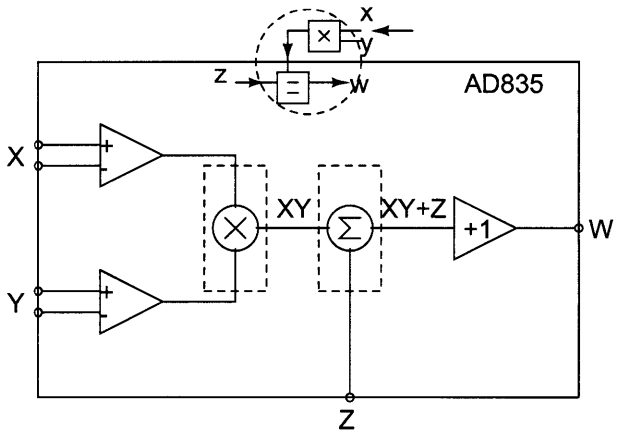

Figure 4-5: Functional block diagram of AD835.

generates the linear product of X and Y voltage inputs with a -3dB output bandwidth of 250MHz. We can assume the gain of the multiplier is flat for our 130kHz circuit. Its differential multiplication inputs(X,Y) and summing input Z are at high impedance. The low impedance output voltage(W) can provide up to $\pm 2.5V$ with $\pm 5V$ power supply. Input differential voltage clips at $\pm 1.4V$.

4.2.3 Analog Delay Line

From the above discussion of analog delay line, we are clear about the choice of analog Bessel filter. We select LTC1064-3 SC 8th order linear phase filter from Linear Technology Inc. Because almost all the high order filters are cascaded from lower order sections like second-order filters. LC Ladders are the best possible filter realization because of their low passband sensitivities to component tolerances. And SC realization of LC ladder structure needs only summing SC integrators. Therefore we guess this LTC1064-3 filter is implemented with SC ladder four second-order filter cascaded structure. We used the filter cutoff frequency of $f_{-3dB} = 66.67KHz$ and clock frequency of 5MHz. However from the data sheet, we cannot get the exact delay time. We shall find it out in order to set right LFSR sequence chip rate. The 8th order Bessel filter transfer function is following:

$$H(s) = \frac{2027025}{s^8 + 36s^7 + 630s^6 + 6930s^5 + 51975s^4 + 270270s^3 + 945945s^2 + 2027025s + 2027025}, \tag{4.28}$$

where the normalized frequency $f_d = 1$ is called delay normalized frequency. However the confusion part of Bessel filter is $1/f_d$ is not equal to the group delay τ_g. Neither does cutoff frequency tell us the group delay. The general relation is:

$$\frac{1}{\tau_d} = (\frac{(2n)!}{2^n n!})^{1/n} f_d. \tag{4.29}$$

For 8th order Bessel filter, the coefficient is 6.14267288. The relation between f_d and f_{-3dB} for 8th order Bessel filter is:

$$f_{-3dB} = 3.1796172375 f_d. \tag{4.30}$$

So $f_{-3dB} = 66.67kHz$ sets $f_d = 20.97KHz$ and group delay of our filter is $7.76\mu s$. Therefore LFSR sequence chip rate 128.8kHz/chip. Fig.4-6 gives the filter frequency response. The plot is linear in frequency axis.

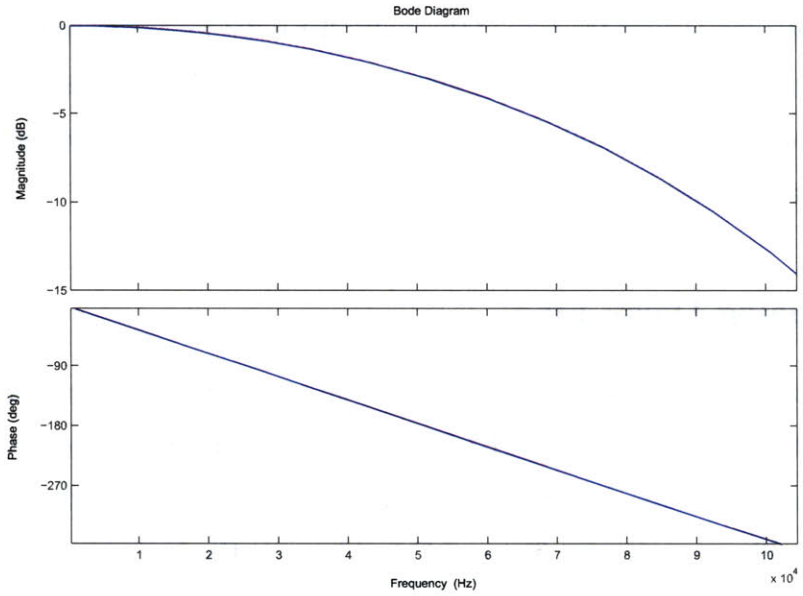

Figure 4-6: LTC1064-3 8th order Bessel filter frequency response with cutoff frequency of 66.67kHz. From the phase plot, we can calculate the group delay to be $7.7\mu s$, which sets the LFSR chip rate of 128.8K chips/s. The magnitude rolls off to about -13dB around 128.8KHz. This cuts off the side lobes of the spectrum of LFSR sequence.

In circuit implementation, the concern about the SC filter is its clock feedthrough and output DC offset. The data sheet specifies the clock feedbthrough is $200\mu V_{RMS}$. However this is measured by buffering the output of filter with a third-order active low-pass filter. In our experiment, we use a simple RC filter as the post filter. R=100Ω and C=10.18nF, which gives cut-off frequency at 156KHz. This cutoff frequence is sufficiently low for filtering out most of the clock feedthrough at 5MHz and also won't effect our system performance. First it is because this is only a first order low pass filter, the roll off is slow comparing to the fast roll-off of 8th order Bessel filter after twice of its cutoff frequency. Also consider the four cascaded filters basically cut off the frequency component of the signal beyond $130kHz$. 100Ω is big enough, because

the filter output impedance is only 2Ω.

The output DC offset for each filter is typically 40mV-60mV. Cascading four filters accumulates the DC offset to about 200mV-250mV. This might affect the multiplier function when the input LFSR signal magnitude is as small as 300mV. A solution is to use AC coupling between the filters. Because the log-likelihood ratio signal is always ground centered, AC coupling won't cause any loss of LLR information. The filter has $22K\Omega$ input impedance, so we selected small resistor value of $1K\Omega$, for the resistor divider. The pole position of the high pass filter in the AC coupling is 17.9kHz, which is proved to be low enough for the filter. Because the high pass filter is only first order, the roll-off at low frequency is very slow. The measured waveform of filters output didn't show DC component loss. A delay filter section is shown in Fig.4-7.

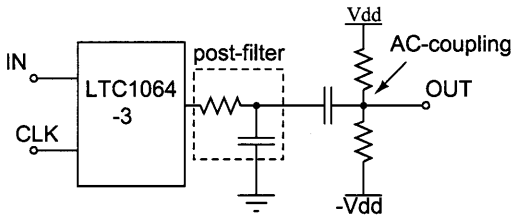

Figure 4-7: Each LTC1064-3 filter connects to a post filter to further kill the clock feedthrough. The AC coupling makes the system insensitive to the filter offset.

4.2.4 Measurement Results

In order to set up same condition for comparison between linear filter and NLL, the linear filter is implemented with the identical LTC1064-3 filter having the same cutoff frequency at 130kHz. We used Agilent E3209 function generators to generate LFSR sequence and white Gaussian noise. The summing of white Gaussian noise and generator LFSR sequence was performed by an AD746 BiFET op amp configured as a summer. The unit gain bandwidth of the AD746 is 13MHz, which is high enough for our system. The spectrum of the white Gaussian noise is flat up to 50MHz, which

is checked by a spectrum analyzer. The data streams were taken by Tektronics TDS 2024 oscilloscope and interfaced to computer through GPIB.

Before we discuss the measurement result, we should be careful about reading the output of the Gaussian noise generator. For example, if the input LFSR has 600mV_{pp} magnitude and noise generator reads 1V noise, naively saying input SNR=$20\log(0.3/1) = -10.45dB$. This is a very low SNR. And we found NLL still works in this SNR. However we shouldn't be too quick to celebrate. 1V noise actually means the noise power density is $1V^2/50MHz = 0.02\mu V^2/Hz$. Since bandwidth of the NLL is limited by the Bessel filter bandwidth, the real noise seen by NLL should be calculated as the in-band noise. For LFSR sequence with chip rate 130k chip/s, most of the signal power dwells in the first lobe notched at 130kHz. f_{-3dB} Bessel filter basically cuts off all the power above 130KHz. Thus $1V^2$ total noise power only accounts as $1V^2 \times 130kHz/50MHz = 2.6 \times 10^{-3}V^2$. The real SNR is 15.56dB.
We compared the noise rejection performance of NLL and linear filter. Fig.4-8 to Fig.4-10 show the measured results at three different noise power levels. When signal-to-noise ratio is 1.4dB, both linear filter and NLL successfully recover the data after thresholding. By observing the oscilloscope for long data stream, we didn't count any bit error for both outputs at this SNR level. When signal-to-noise ratio is reduced to -2.1dB, both NLL and linear filter start to make decision errors as shown in Fig.4-9. The thresholded output of NLL has less glitches than the thresholded output of linear filter, which may be due to that more higher frequency noise has been filtered out in NLL. This can be observed from the output of NLL and linear filter in the first two plots in Fig.4-9. By observing the data stream on the oscilloscope, the bit error rate is about 2% for both NLL and linear filter. Thresholded output of linear filter has more glitches which we didn't count as bit error. When the signal-to-noise ratio drops to -4.6dB, we still can observe the NLL kills more high frequency noise. However, both have about 10% bit error rate. In conclusion, the bit errors in the thresholded output of NLL and linear filter drop from hardly observable(through data stream on oscilloscope) to 10% of total transmitted bits when SNR drops from 1.4dB to -4.6dB.

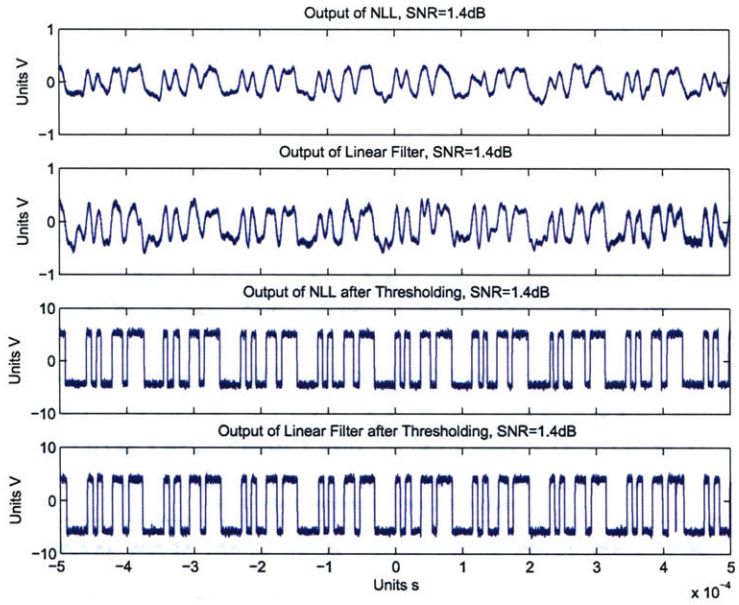

Figure 4-8: Input SNR=1.4dB. The top two plots are output waveform of NLL and linear filter. The bottom two are the thresholded output of NLL and linear filter. We can see both of NLL and linear filter successfully recover the data.

The bit error rate of NLL and linear filter at three different SNR 1.4dB, -2.1dB, and -4.6dB are very close in our manual observation. The NLL output before thresholding(as plotted in the first two subplots in Fig.4-8,Fig.4-9, and Fig.4-10) has less high frequency noise.

4.3 Analogic for First Order Markov Vector NLL

Both simulation and experiment show scalar NLL improve the performance of noise reject over a linear filter. However the improvement is still limited. As discussed in Chapter 3, we should go up to general belief propagation to improve the performance more. The fundamental computation units in GBP are no longer Gilbert multipliers as in scalar LLR NLL. Now we will discuss how to implement this system in circuit.

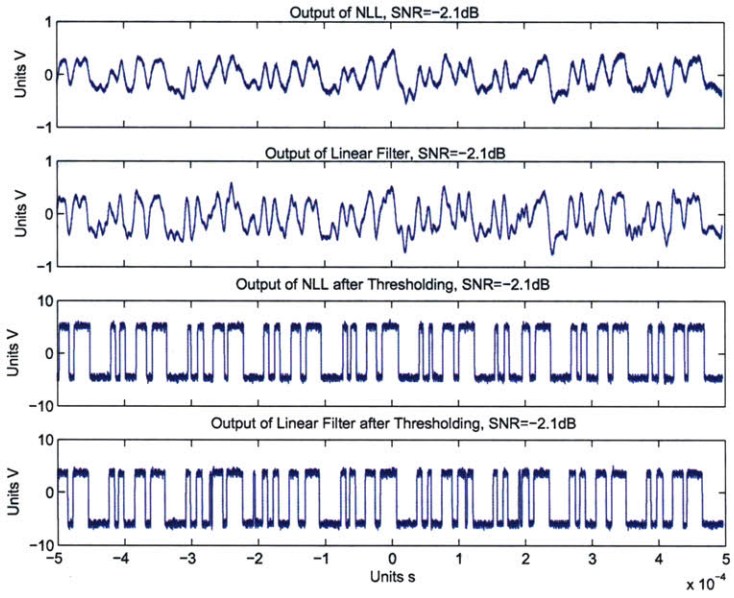

Figure 4-9: Input SNR=-2.1dB.The top two plots are output waveform of NLL and linear filter. The bottom two are the thresholded output of NLL and linear filter. Linear filter output has been more severely disturbed by noise. The thresholded output of linear filter contains glitches which come from the high frequency noise in the filter output.

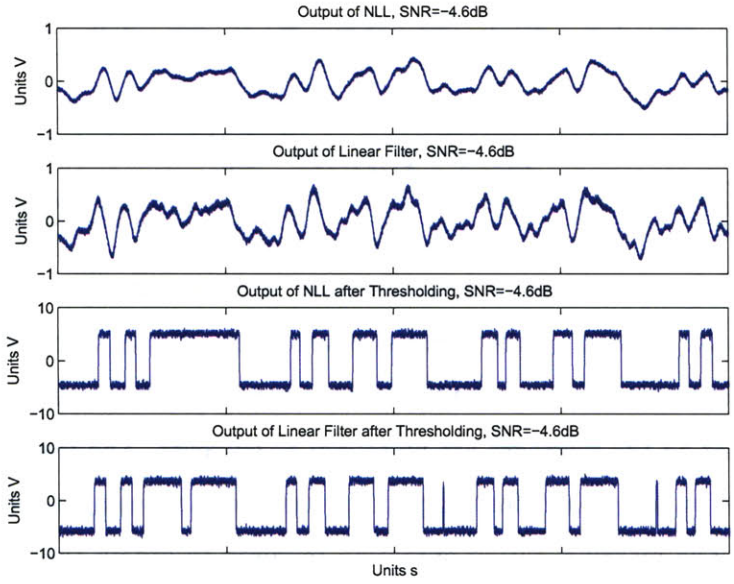

Figure 4-10: Input SNR=-4.6dB. The top two plots are output waveform of NLL and linear filter. The bottom two are the thresholded output of NLL and linear filter. We zoom in to see the bit errors. By manual observation on the oscilloscope, the bit error rates of NLL and linear filter are about 10%.

4.3.1 Circuits for Intermediate Message Computation

First we look at the intermediate message in Eq.(4.31). For clarity, we copy the equation here again.

$$\mu_t^{'}(x_k, x_{k-1}) = \sum_{\sim\{x_k, x_{k-1}\}} \delta(x_k \oplus x_{k-1} \oplus x_{k-4}) \frac{\mu_{t-1}(x_{k-1}, x_{k-2})\mu_{t-1}(x_{k-2}, x_{k-3})\mu_{t-1}(x_{k-3}, x_{k-4})}{\sum_{x_{k-1}} \mu_{t-1}(x_{k-1}, x_{k-2}) \sum_{x_{k-2}} \mu_{t-1}(x_{k-2}, x_{k-3})}. \tag{4.31}$$

Its first value is further expanded in Eq.(4.32). We also reproduce the equation here for clarity. $\mu_{t-1}(x_{k-2} = 1, x_{k-3} = 1)$.

$$\begin{aligned}
\mu_t^{'}(x_k = 0, x_{k-1} = 0) = & \frac{\mu_{t-1}(k-1, k-2)_{00}\mu_{t-1}(k-2, k-3)_{00}\mu_{t-1}(k-3, k-4)_{00}}{\mu_{t-1}(k-2)_0\mu_{t-1}(k-3)_0} \\
& + \frac{\mu_{t-1}(k-1, k-2)_{00}\mu_{t-1}(k-2, k-3)_{01}\mu_{t-1}(k-3, k-4)_{10}}{\mu_{t-1}(k-2)_0\mu_{t-1}(k-3)_1} \\
& + \frac{\mu_{t-1}(k-1, k-2)_{01}\mu_{t-1}(k-2, k-3)_{10}\mu_{t-1}(k-3, k-4)_{00}}{\mu_{t-1}(k-2)_1\mu_{t-1}(k-3)_0} \\
& + \frac{\mu_{t-1}(k-1, k-2)_{01}\mu_{t-1}(k-2, k-3)_{11}\mu_{t-1}(k-3, k-4)_{10}}{\mu_{t-1}(k-2)_1\mu_{t-1}(k-3)_1}.
\end{aligned} \tag{4.32}$$

The basic computation here is multiplication of three variables and division by the result of multiplication of the other two variables. These desperate awkward-looking computation block actually can be solved with little effort by static translinear circuits. Consider all the messages are presented as currents. The translinear circuit in Fig.4-11 is the building block for computing intermediate messages, which has only six transistors. The marginalization operation required to obtain the single variable messages such as $\mu_{t-1}(k-2)_0$ can be easily done by current summing with a junction of wires.

The two equations Eq.(4.33) and Eq.(4.34) for computing updated messages are listed here again.

$$\mu_t(x_{k-1}, x_{k-2}) = \mu_t^{'}(x_{k-1}, x_{k-2})\left(\frac{\sum_{x_k} \mu_t(x_k, x_{k-1})}{\sum_{x_k} \mu_t^{'}(x_k, x_{k-1})}\right). \tag{4.33}$$

$$\mu_t(x_{k-2}, x_{k-3}) = \mu_t^{'}(x_{k-2}, x_{k-3})\left(\frac{\sum_{x_{k-1}} \mu_t(x_{k-1}, x_{k-2})}{\sum_{x_{k-1}} \mu_t^{'}(x_{k-1}, x_{k-2})}\right). \tag{4.34}$$

In the light of translinear principle, these computations can be realized in an even simpler circuit in Fig.4-12 with four transistors. Again the marginalization of two

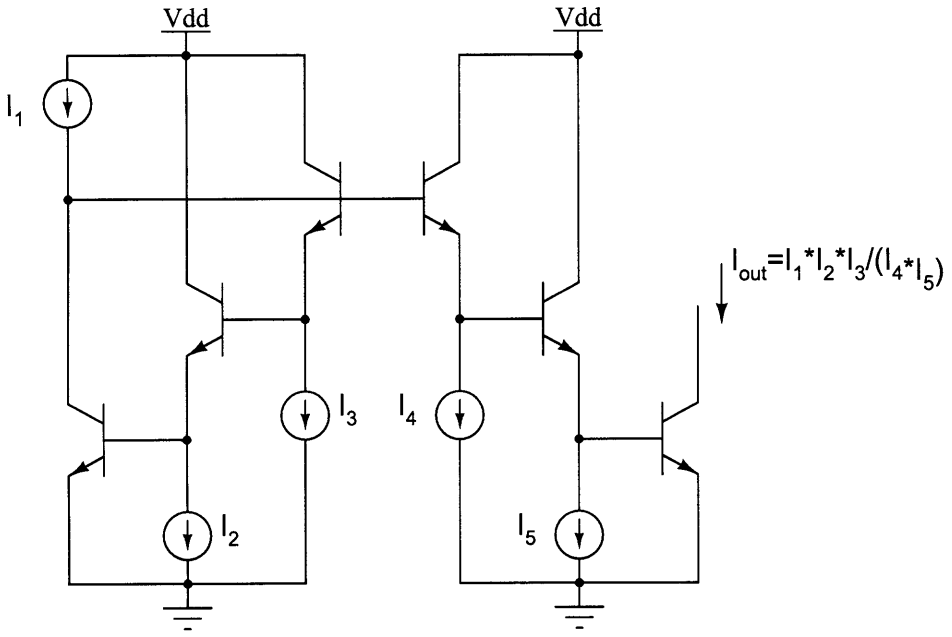

Figure 4-11: Translinear block for intermediate message computation.

variable messages such as $\sum_{x_k} \mu_t(x_k, x_{k-1})$ can be implemented as directly connecting two currents using KCL.

4.3.2 Analogic Computational Blocks

The computational block for the intermediate message is composed by the circuit block of Fig.4-11 and current adders which are simply junctions of wires. One computation block is shown in Fig.4-13 which has 12 input messages(actually 10 of them are used) and computes the message $\mu_t'(k, k-1)_{00}$. The T_1 blocks are the translinear circuit shown in Fig.4-11. In the same principle, only by re-arranging the connection between input currents to the translinear circuit blocks, the other three computation blocks for messages $\mu_t'(k, k-1)_{00,01,10,11}$ can be easily built. Therefore, we demonstrated the principle to construct basic computation blocks for generalized belief propagation.

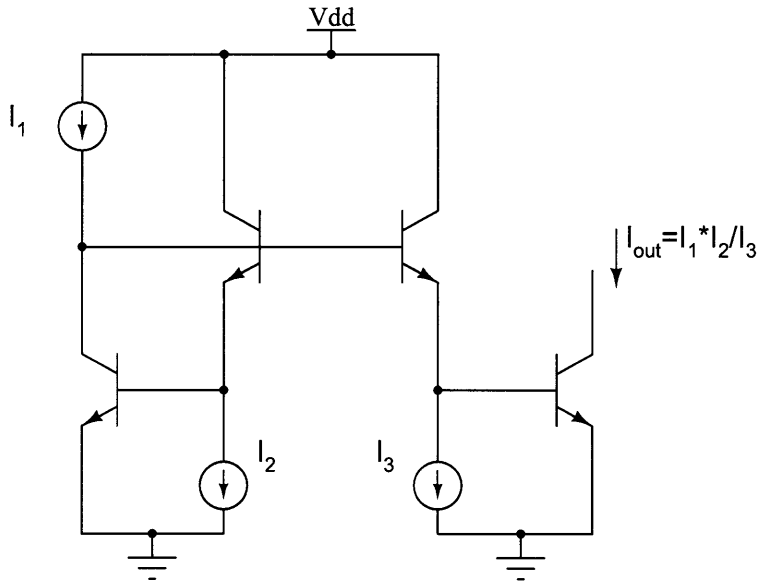

Figure 4-12: Translinear block for updated message computation.

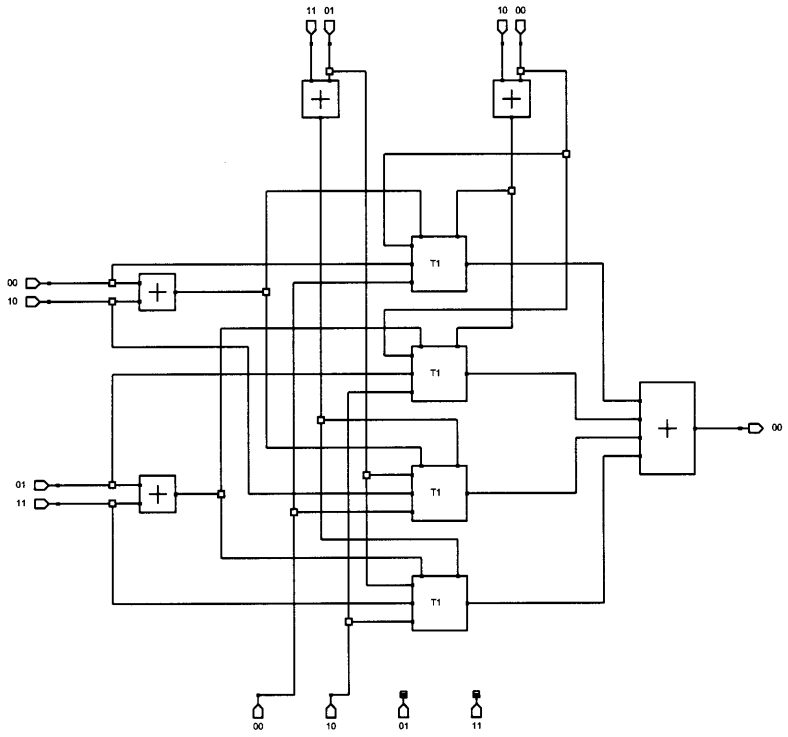

Figure 4-13: Computational block for message $\mu_t'(k, k-1)_{00}$. The blocks marked with "T1" are the tranlinear circuit in Fig.4-11, which computes the multiplication of three messages and divide by the fourth message. The blocks marked with "+" are current summation, which actually are simply junctions of wires. The top four inputs are messages $\mu_{t-1}(k-1, k-2)_{00,01,10,11}$. The left four inputs are messages $\mu_{t-1}(k-2, k-3)_{00,01,10,11}$. The bottom four inputs are messages $\mu_{t-1}(k-2, k-3)_{00,01,10,11}$. Only two of them are used.

Chapter 5

Scaling to Large Complex System and Current-Mode Signal Restoration

In this chapter, we seemingly discuss an irrelevant topic — current-mode signal restoration. But actually it is on the way of further development of the ideas in the earlier chapters: from scalar NLL to vector NLL.

5.1 Scaling to Complex Systems

Until now, we discussed LFSR synchronization as an estimation problem solved by message-passing algorithm. We discussed the implementation of the simple scalar NLL as analogic circuits and the circuit structure proposed for vector NLL, which is a bigger analogic network. We see the goal is not only scalar NLL but more powerful analogic circuits which naturally raise the question of how to scale to systems of large complexity. A thorough and deep study of this problem has been published in [25], where authors outlined a hybrid analog-digital scheme with three important features that enable it to scale to large systems: collectively computation by moderate precision analog units, discrete signal restoration and state machine.

In the light of the hybrid computation, we can look at scalar NLL from a new point of view. We have already seen scalar NLL is composed of two kinds of circuits. One is analog computation blocks(soft gates and approximate soft gates) which do the computation for estimation. The other kind of circuit is the switch-capacitor filter circuits which actually determine the dimensions and dynamics of this nonlinear system. Although the switch-capacitor filters are driven by clock, the clock is *not* an intrinsic part of the system. It is the filters as delay elements that drive the NLL to evolve through a sequence of states. And time in the algorithm is the delay time as a physical time. We can even implement the analog filters array by digital delay line through A/D and D/A converters to interact with analog processing part. It may be too complex to do this for simple scalar NLL. However, it might be beneficial for vector NLL because digital circuits can precisely control the state transition. In addition, A/D and D/A converters restores the analog signals periodically.

5.2 Current-Mode Signal Restoration

As discussed above, frequent discrete signal restoration is crucial for large-scale complex computation system. Here we present a novel current-mode signal restoration circuit using negative current mode feedback to achieve very good linearity and purely current input and output capability. The basic building block for both circuits is winner-take-all circuit. So we first walk through the mechanism of winner-take-all circuit. Then we introduce the controllable gain current mode amplifier with negative feedback linearization.

5.2.1 Winner-take-all Circuit

Winner-take-all(WTA) circuit is a very interesting circuit that cleverly utilizes collective feedback to parallel compute the max function of arbitrary number of input variables. We take a two-input WTA as example to analyze how it works. For more detailed analysis, please refer to [24].

First, when $I_{in1} = I_{in2}$, the drain voltages of M_1 and M_2 are equal. Thus M_3 and M_4

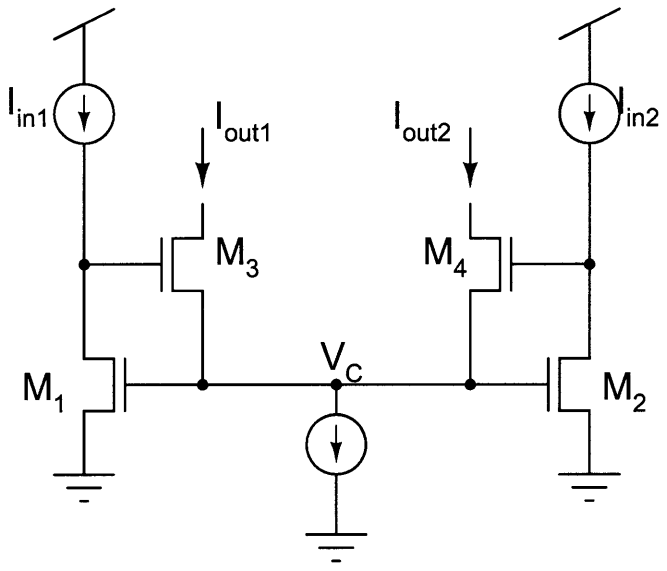

Figure 5-1: Winner-take-all cell.

have the same gate-to-source voltage. The differential pair splits the tail bias current equally. Now suppose I_{in1} increases by Δi. The drain voltage of M_1 increases which causes more current flow into node V_c. This current is divided between g_{mb} of M_3 and the effective impedance of cascaded pair M_2 and M_4. The Fig.5.2.1 shows the local negative feedback loop in the pair M_2 and M_4, which boosts the original conductance of M_2 up to $\frac{1}{Z_{eff}} = (1 + \kappa g_{m2} r_{o2})(g_{m4} + g_{mb4})$, where $\kappa = 1/(1 + g_m/g_{mb})$.

Fig.5-3 shows the small signal analysis and overall block diagram. The transfer function can be easily written out as:

$$\frac{i_{out}}{\Delta i} = \frac{1}{2} g_{m3} r_{o1}. \tag{5.1}$$

The analysis above also applies to the WTA which has $N (N > 2)$ legs. The current gain is accordingly:

$$\frac{i_{out}}{\Delta i} = \frac{N-1}{N} g_{m3} r_{o1}. \tag{5.2}$$

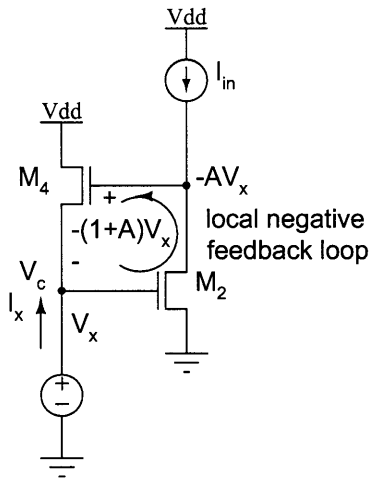

Figure 5-2: Local negative feedback increases conductance

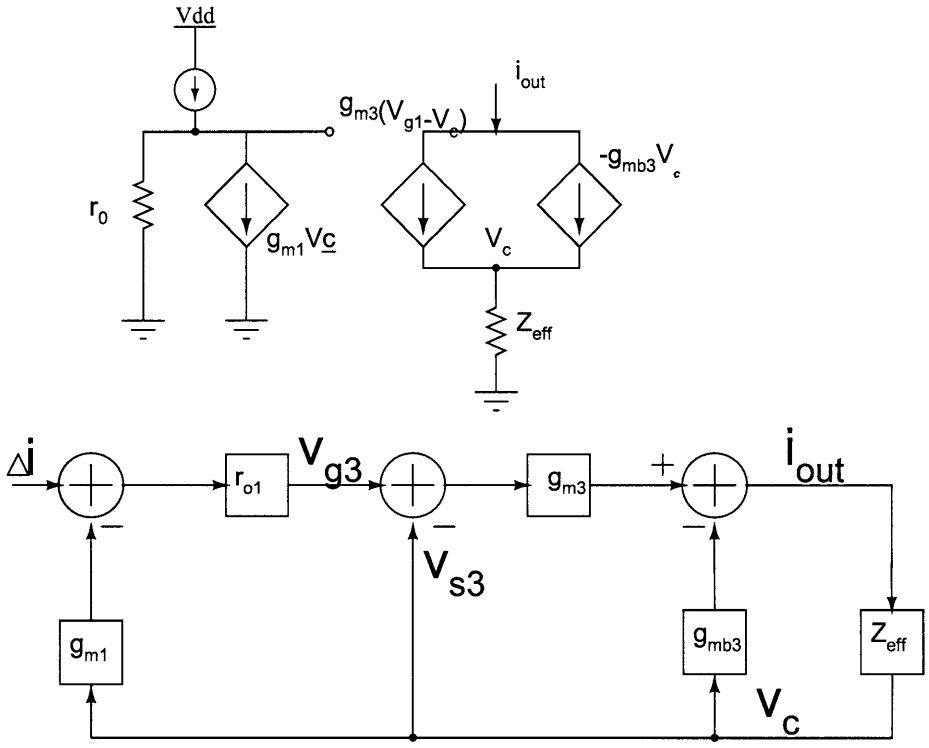

Figure 5-3: Block diagram of small signal analysis of WTA.

5.2.2 Controllable Gain Current-Mode Amplifier

The analysis of WTA shows that WTA has very high current gain $g_m r_o$. In order to implement a controllable gain current-mode amplifier, the first idea in our mind is to reduce this high current gain in a controllable way. The first circuit we designed according to this idea is to use controllable resistor in parallel to M_1 that r_o is shunted. This is an open-loop control. A better way is do feedback control. Instead of fighting with WTA's high gain as what we did in the transistor shunting circuit, the high current gain of WTA actually provides the good feedback performance similar to the convential operational amplifier provides versatile feedback configurations. So here we will first discuss the second circuit, then compare performance with the first one at the last of this chapter. Instead of fighting with the WTA's high gain as in the shunting resistor circuit, we can rather take WTA as a current-mode amplifier with more than 20dB gain. From this perspective, feedback is a natural way to control the gain. The general current mode circuit feedback diagram is drawn in Figure 5-4. We use WTA as the current-mode amplifier with gain derived in (5.1). The current

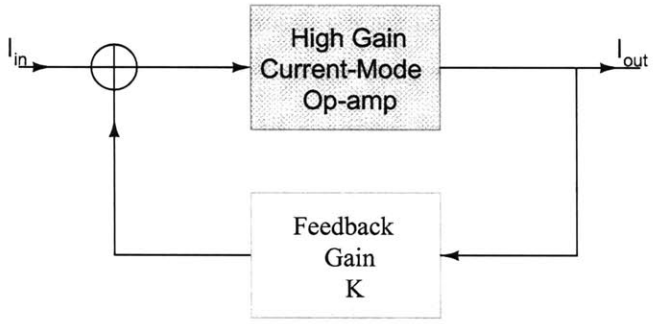

Figure 5-4: Feedback block diagram-1.

feedback is cheaply done by current-mirrors as in Figure 5-5. Use Black formula to get the closed-loop gain:

$$\frac{I_{out}}{I_{in}} = \frac{g_m R_o/2}{1 + K g_m R_o/2} \approx \frac{1}{K}. \tag{5.3}$$

There are several ways to control the gain. The geometry of the current mirror sets the DC gain. The well-terminal of PMOS in the current mirror can also control the

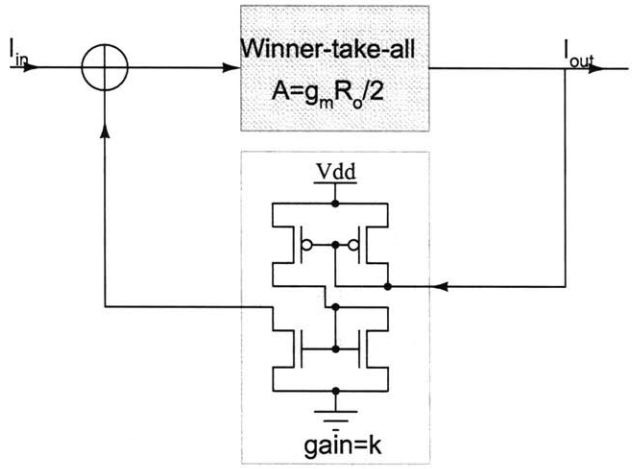

Figure 5-5: WTA in feedforward path and current mirror in feedback path.

current gain. More flexible way is to use simple digital-to-analog converter to switch the aspect ratio of current mirror. In our chip, we used the D/A converter approach.

Now consider the WTA cell again. There are two current inputs and two current outputs. The combinations of these terminals in feedback diagram Fig.5-5 will give us two types, three possible negative feedback configurations. The first type uses single feedback path, which has two configurations: feedback positive output negatively to input using PMOS and NMOS current-mirrors(we call it N-feedback); feedback negative output positively to input only using PMOS current-mirror(we call it P-feedback). The second type is dual feedback of both N-feedback and P-feedback. Now we discuss them one by one.

Controllable Gain Current-Mode Amplifier with N-Feedback

Fig.5-6 shows the circuit structure of the N-feedback amplifier. WTA block in the circuit is the same as in Fig.5-1. The DC analysis in SPICE shows that the amplifier output I_{out1} saturates at two rail currents: 0nA and I_{bias} set by tail current in WTA. In between, when input I_{in1} increases bigger than I_{in2}, output I_{out1} starts to linearly ramp up, until it hits the upper rail when $I_{in1} = I_{bias}/K + I_{in2}$, where K is current

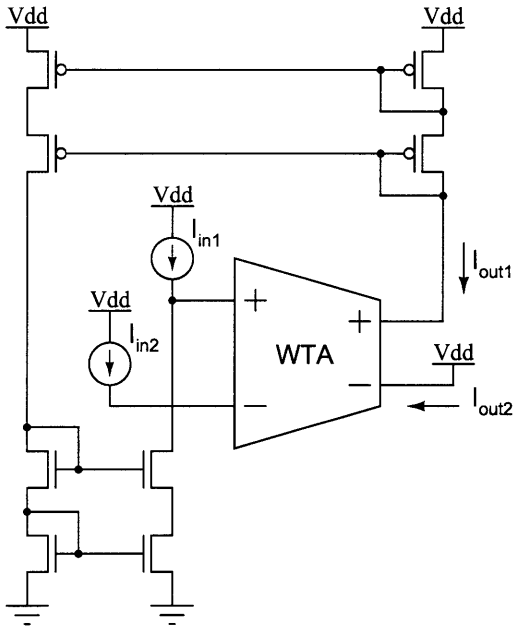

Figure 5-6: N-feedback configuration.

gain. As mentioned before, the current gain can be controlled by well-voltage of current-mirror. The following simulation(Fig.5-8) changes the well-voltage from 4.6V to 5V. The current gain changes from 25 to 2.17. The disadvantage of this is the gain changes exponentially. Also the balance point(the value of input current when two outputs are equal) changes exponentially with gain.

Controllable Gain Current-Mode Amplifier with P-Feedback

The P-feedback configuration also gives good linear range from rail to rail. When I_{in1} is greater than I_{in2}, output of I_{out1} is at rail I_{bias}. When I_{in1} decreases below I_{in2}, I_{out1} starts to linearly ramp down until it hits the bottom rail. So the upper bound of linear range of I_{in1} is set by I_{in2}. Compare with N-feedback, where the lower bound of linear range is set by I_{in2}. In Fig.5-10 we plotted output current I_{out1} as solid curve and input current I_{in1} in dotted line. We can clearly see that during the linear range, the input current due to the high current gain of WTA barely changes. Feedback sets the closed-loop gain.

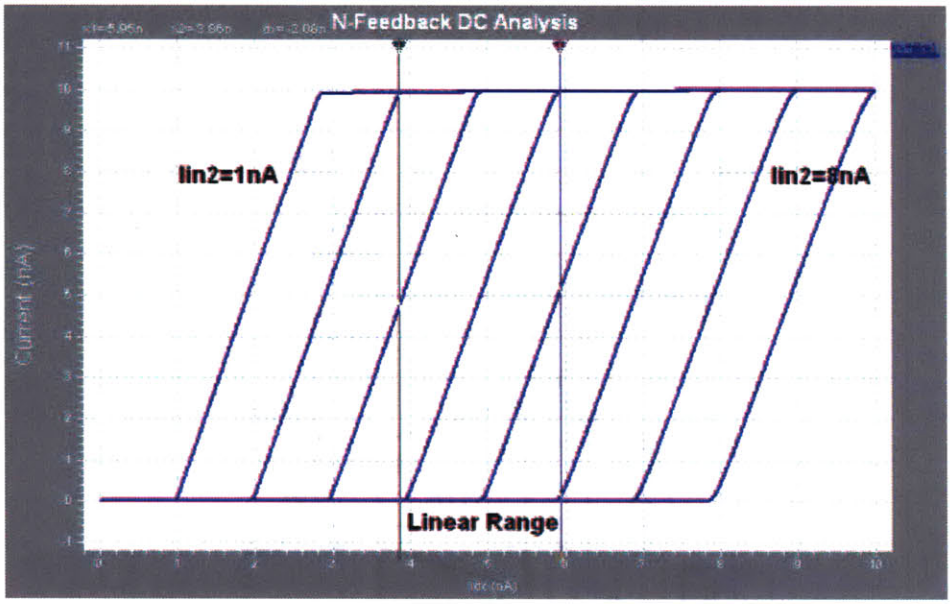

Figure 5-7: I_{out1} of N-feedback when sweeping I_{in1} at different I_{in2}.

Controllable Gain Current-Mode Amplifier with Dual Feedback

The linear range is set by I_{in2} and current gain in N and P-feedback configurations. However, the *balance point* of the linear range(where $I_{out1} = I_{out2}$) is determined by both I_{in2} and current gain in both N and P-feedback as shown in Fig.5-8. What we desire is when $I_{in1} = I_{in2}$, the output I_{out1} is equal to I_{out2}, no matter what I_{in2} and the current gain are. This property is very useful when the amplifier is used as a fully differential amplifier. The dual feedback configuration achieves this property.

The stability of the feedback circuit is investigated by root-locus approach. The result shows the system is stable with the feedback gain from 0 to 10^5.

In conclusion, we proposed a novel current-mode feedback signal restoration circuit. It has a wide linear range and uses purely current input and output. It is a very interesting direction to apply this circuit in time-domain spike computation.

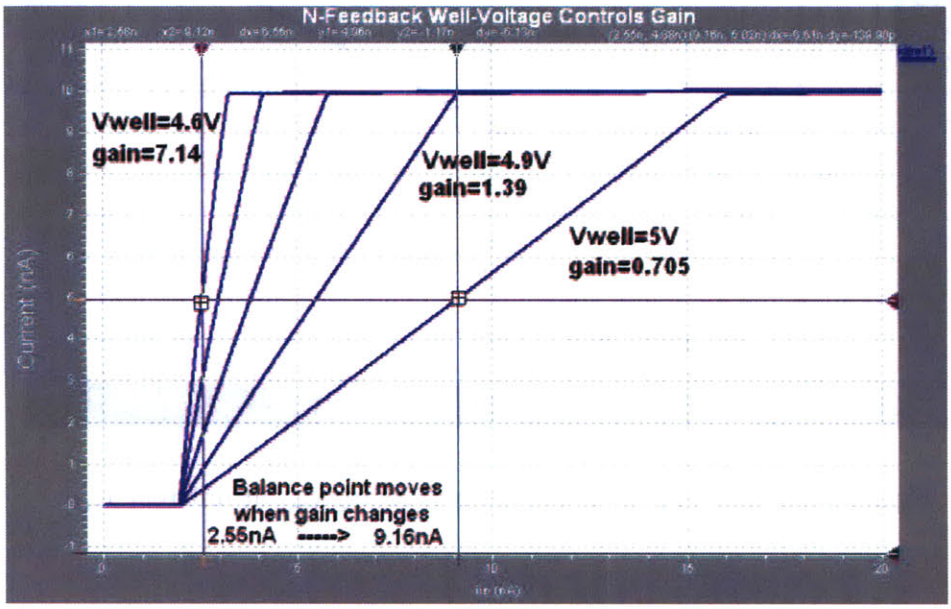

Figure 5-8: Control current gain by changing the well voltage of PMOS current-mirror. Middle point also moves.

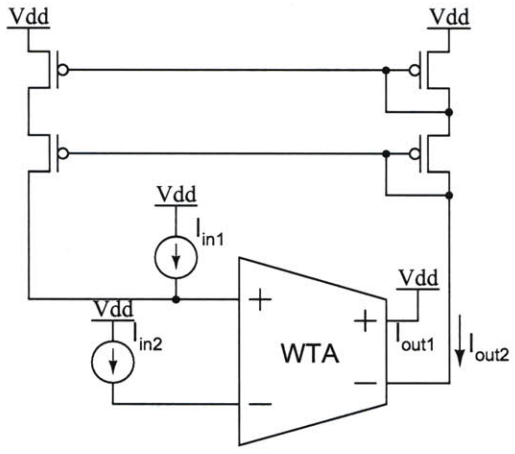

Figure 5-9: P-feedback configuration.

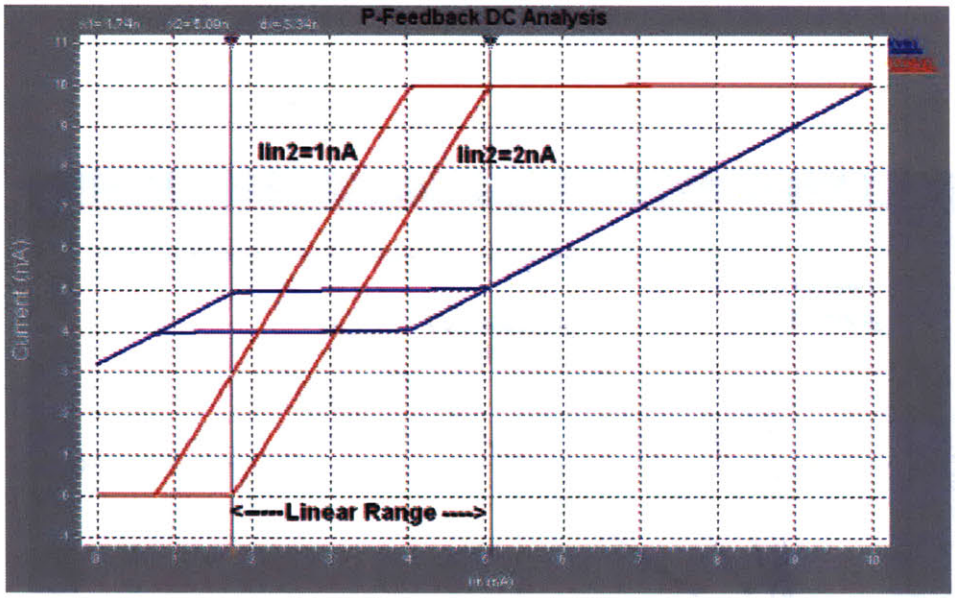

Figure 5-10: I_{out1} of P-feedback when sweeping I_{in1} at different I_{in2}.

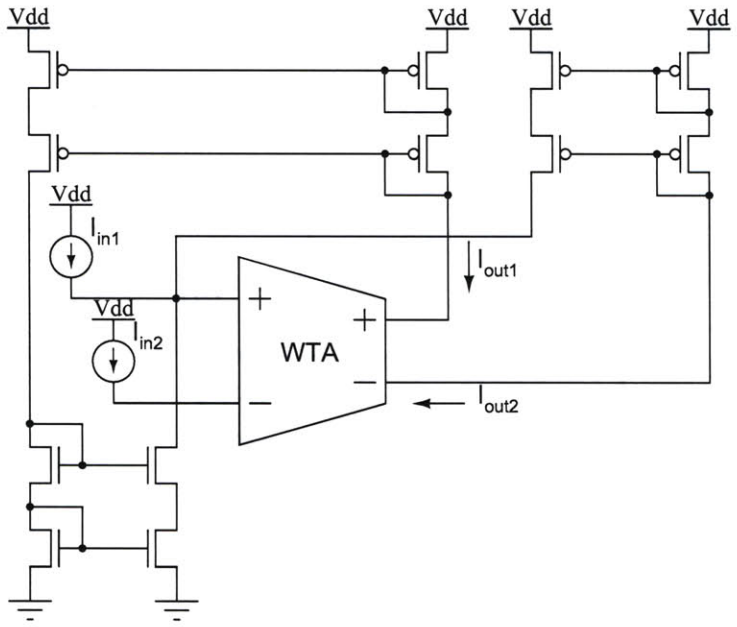

Figure 5-11: Dual feedback configuration.

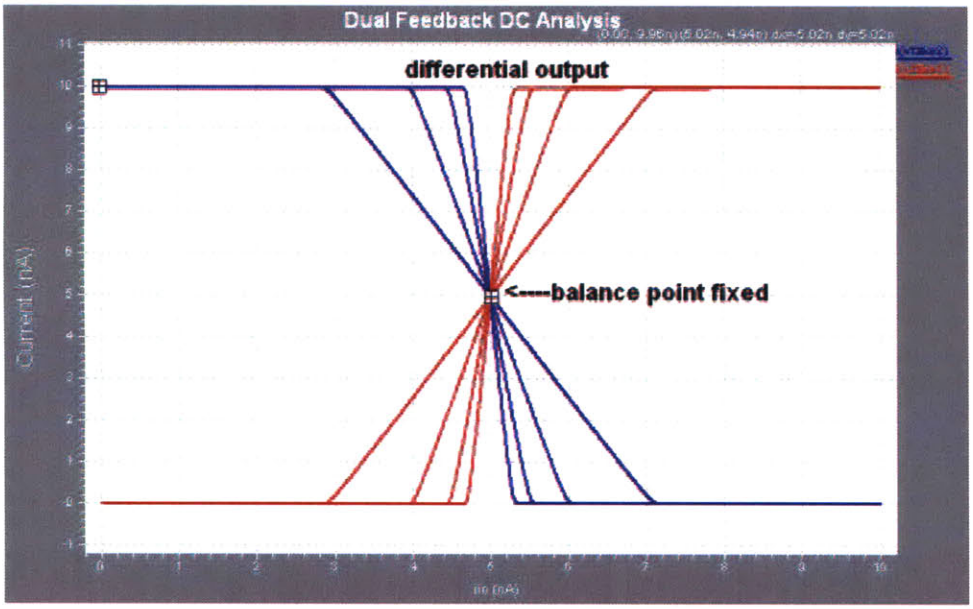

Figure 5-12: Dual feedback. Balance point is set only by I_{in2}.

Chapter 6

A New Geometric Approach for All-Pole Underdamped Second-Order Transfer Functions

This chapter is on a new geometric approach for all-pole underdamped second-order transfer functions, which provides useful geometric techniques for future research on cascaded filters in noise-locked loop.

The common s-plane geometry of second-order all-pole transfer functions interprets the gain as the product of two lengths and phase as the sum of two angles. This approach makes the overall frequency response less intuitive to understand. Using the geometric transformation proposed in a geometry called "one-pole" geometry [23], gain depends on one length, and the phase corresponds to one angle. This makes many important quantities such as the gain peaking and the corner frequency obvious. Other quantities like group delay and bandwidth also have simple geometric interpretations. In our new approach, gain as well as phase is represented as one angle which is directly constructed from the original two-pole plot without any geometric transformation. Comparing with the "one-pole" geometry, the new geometry is simpler. Yet how to represent group delay in the new geometry is still an open question. We first briefly review the usual geometry and "one-pole" geometry. Then we introduce our new geometric approach.

6.1 Usual Geometry

The transfer function of second-order system with two poles is[23]

$$H(s) = \frac{1}{1 + \frac{\tau s}{Q} + \tau^2 s^2}, \tag{6.1}$$

where H is the transfer function, s is the s-plane variable, τ is the time constant, Q is the quality factor. This transfer function form is normalized to have unit gain at DC. We only discuss the underdamped($Q > 0.5$) second-order system which has two complex poles given as

$$p_{1,2} = -\frac{1}{2Q} \pm j\sqrt{1 - (\frac{1}{2Q})^2}. \tag{6.2}$$

The magnitude of the transfer function with $s = jw$ has the form

$$|H(j\omega)| = \frac{1}{|j\omega\tau - p_1| \cdot |j\omega\tau - p_2|}, \tag{6.3}$$

$$= \frac{1}{\sqrt{(1 - \omega^2\tau^2)^2 + \omega^2\tau^2/Q^2}}. \tag{6.4}$$

The phase of the transfer function has the form

$$\text{Arg } H(j\omega) = -\arctan(\frac{\omega\tau/Q}{1 - \omega^2\tau^2}). \tag{6.5}$$

The Eq.6.3 gives the standard geometric interpretation of gain of the transfer function shown in Fig.6-1. The phase can be obtained by algebraic sum of two angles θ_1 and θ_2.

6.2 "One-Pole" Geometry

Eq.6.4 gives the hint on the x-space geometry proposed in [23]. First we define two new variables as

$$x = \omega^2\tau^2, \tag{6.6}$$

$$\theta = \arcsin(1 - \frac{1}{2Q^2}). \tag{6.7}$$

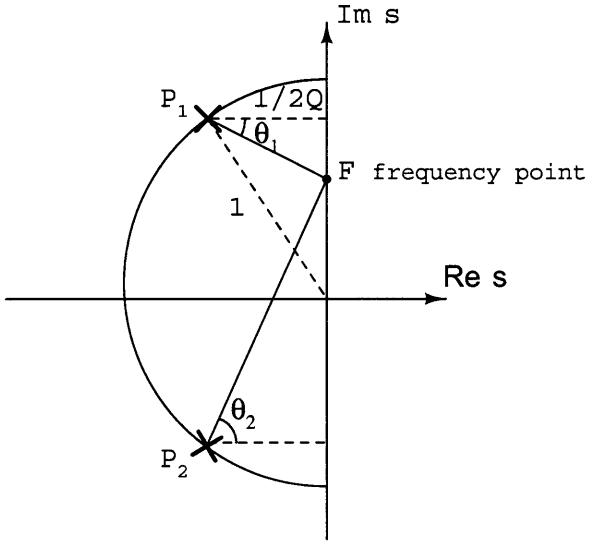

Figure 6-1: The usual geometry of the underdamped second-order transfer function. Assuming $d_1 = |P_1F|$ and $d_2 = |P_2F|$, the gain at the frequency point F is $1/d_1d_2$. The phase at F is $\theta_1 + \theta_2$. Note θ_2 is always positive(F takes positive frequency), θ_1 can be negative as the case drawn in the figure.

Eq.6.3-Eq.6.5 can be simplified as

$$|H(x)| = \frac{1}{\sqrt{(x - \sin\theta)^2 + \cos^2\theta}}, \tag{6.8}$$

$$\text{Arg } H(x) = -\arctan(\frac{\sqrt{x}/Q}{1-x}). \tag{6.9}$$

Consider the fact that the position of one pole determines the position of the other pole. We should be able to "compress" two poles as in usual geometry Fig.6-1 to a one "pole" geometry without losing any information on the system frequency response. Two new variables in Eq.6.6 and Eq.6.7 actually define the geometric transformation from normal frequency space to x-space: frequency point F is transformed from $(0, j\omega\tau)$ to $(x, 0)$, two poles are represented by one point $(\sin\theta, \cos\theta)$ called Q-point[23]. The gain in the new geometry is the reciprocal of the distance d from point x to Q-point. The phase needs more manipulations. Draw an arc of radius d centered at point $(x, 0)$ to intersect with line $x = 1$ at point $(1, \frac{\sqrt{x}/Q}{1-x})$ called *phase point*. The *phase vector* is the vector from $(x, 0)$ to the phase point. The angle ψ

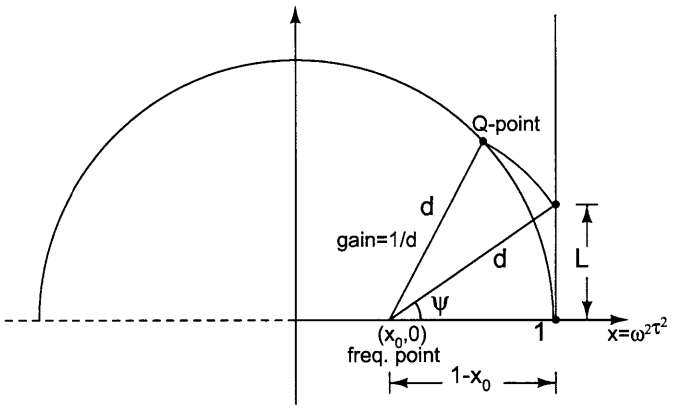

Figure 6-2: The new geometry proposed in [23]. Frequency point is transformed to $(x_0, 0)$. Poles are "compressed" to one Q-point. Gain is $1/d$. The length $L = \sqrt{x_0}/Q$. Phase is $\arctan(L/(1-x_0))$.

between x axis and phase vector is the negative of the phase of the transfer function. A pictorial representation of the new geometry is shown in Fig.6-2. The discussion on group delay and an alternative construction of phase can be found in [23].

6.3 A New Geometry

The x-space geometry compresses the "redundant" pole information by geometric transformation. The important quantities in frequency response such as peaking frequency, peaking gain, $-90°$ frequency and gain, etc. can be directly read out from Fig.6-2[23]. An interesting question to ask is if we can achieve similar amount of geometric intuition without any geometric transformation. The answer to this question leads to the simpler geometry where one-"pole" is uncompressed back to original two poles for calculating gain, and mirrored to four-"pole" for phase. In this way, phase and gain are represented by two angles in the usual pole plot like Fig.6-1. And important quantities listed above can be obtained geometrically.

The Geometry of Phase

The frequency point F can be either within the circle or outside the circle. We first discuss the case when F is inside the circle as shown in Fig.6-3. From the usual geometry discussed in the first section, the phase of the transfer function is the negative of the sum of two angles $\theta_1 + \theta_2$, equivalently, $|\theta_1| - \theta_2$, since $\theta_1 < 0$. Mirror the original two poles P_1 and P_2 to the right half plane to get P_{1m} and P_{2m}. m in the subscript denotes "mirror". Recognize the angle relation:

$$\theta_2 = \angle FGP_2 + \phi, \tag{6.10}$$

$$\angle FGP_2 = \theta_1. \tag{6.11}$$

Therefore, the negative of the phase of the transfer function at frequency point F is angle ϕ. The supplement angle ψ is easier to use. So we represent the phase at frequency point F by one angle $\psi - \pi$. This relation holds for the case when frequency point F is outside the circle as shown in Fig.6-4. The difference is $\theta_1 > 0$, so the negative of the phase of the transfer function is $\theta_1 + \theta_2$. It is readily to see

$$\text{phase } = -\theta_1 - \theta_2 = \psi - \pi. \tag{6.12}$$

In conclusion, *the phase of the transfer function, regardless of the position of frequency point, is* $\angle P_2FP_{1m} - \pi$. We now see some special frequency points and their phases.

- At DC($\omega = 0$), $\angle P_2FP_{1m} = \pi$. Therefore, the phase is 0.

- At frequency point E in Fig.6-3,$\angle P_2FP_{1m} = \pi/2$. This is because the segment $|P_2P_{1m}|$ is the diameter of the circle, passing the center of the circle. Point E is the $-90°$ frequency point of the transfer function.

- At very high frequency, $\angle P_2FP_{1m}$ approaches 0. Therefore, high frequency phase shift is $-180°$.

The Geometry of Gain

The first key observation to new geometry of gain is that the area of triangle P_1FP_2 is invariant of the frequency point as shown in Fig.6-5. This can be easily seen as

$$\begin{aligned} \text{Area of } \Delta P_1FP_2 &= \frac{1}{2}|P_1H| \cdot |P_1P_2|, & (6.13) \\ &= \frac{1}{2Q}\sqrt{1-(\frac{1}{2Q})^2}. & (6.14) \end{aligned}$$

The second key observation is the area of ΔP_1FP_2 can be also written as

$$\begin{aligned} \text{Area of } \Delta P_1FP_2 &= \frac{1}{2}d_1d_2\sin\theta, & (6.15) \\ &= \frac{1}{2}\frac{\sin\theta}{\text{gain}}. & (6.16) \end{aligned}$$

where $d_1 = |P_1F|$, $d_2 = |P_2F|$. Therefore the gain at frequency point F has the form

$$\begin{aligned} \text{gain} &= \frac{\sin\theta}{\frac{1}{Q}\sqrt{1-(\frac{1}{2Q})^2}}, & (6.17) \\ &\propto \sin\theta. & (6.18) \end{aligned}$$

We finally use one angle and the Q value to represent the gain at any frequency point. Now we check some important frequency points.

- DC($\omega = 0$) gain is 1. This can be seen as $\cos\theta/2 = 1/2Q$,$\sin\theta/2 = \sqrt{1-(\frac{1}{2Q})^2}$, and triangular equality $\sin\theta = 2\sin\theta/2\cos\theta/2$. The numerator is cancelled by the denominator in Eq.6.17.

- Peaking appears at frequency point F_2 shown in Fig.6-6. This happens when the circle with diameter P_1P_2 can intersect with $j\omega$ axis. This requires $Q \geq 0.707$, which is the critical Q value for peaking. Another observation available is peaking frequency is always lower than $-90°$ frequency.

- At very high frequency, angle θgoes to 0. Therefore, gain goes to 0.

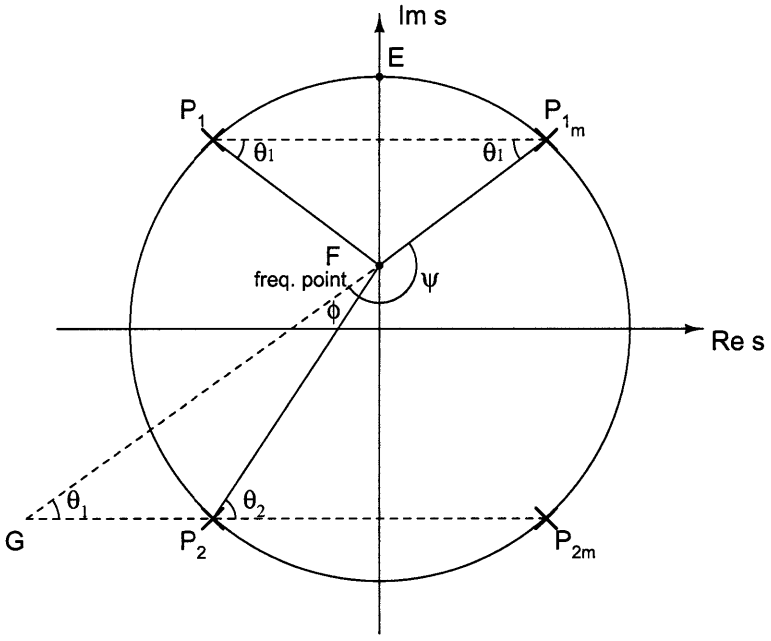

Figure 6-3: The phase of the transfer function is $|\theta_1| - \theta_2$, which has already been pointed out in section on usual geometry. By mirroring the two poles to right half plane, the phase can be represented by one angle.

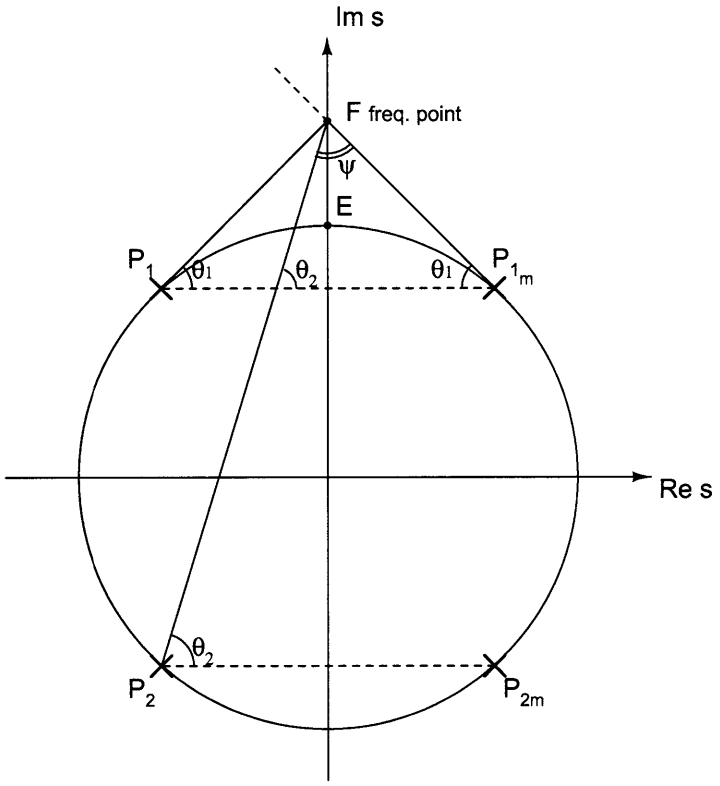

Figure 6-4: When the frequency point is outside the circle, the phase written in two angles is $-\theta_1 - \theta_2$. It can be also represented as one angle.

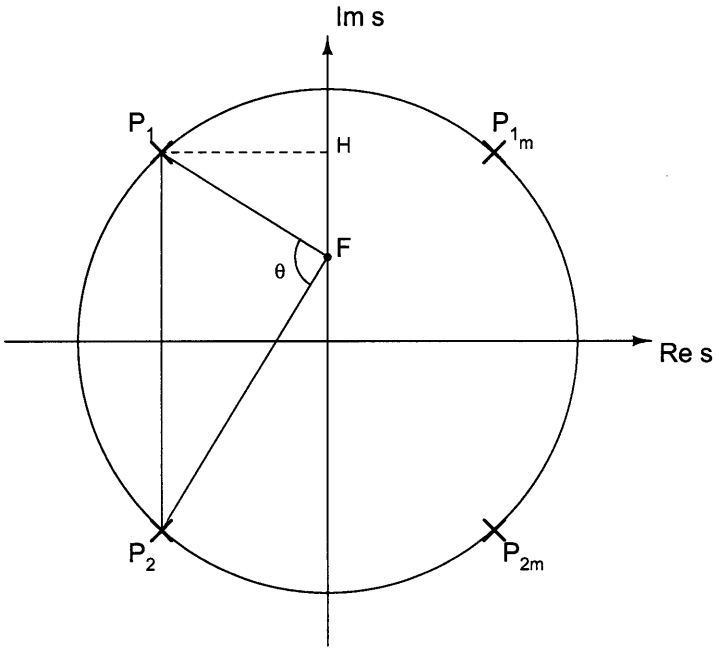

Figure 6-5: The key observation is that the area of triangle $\Delta P_1 F P_2$ only depends on quality factor Q of the second-order system, invariant under frequency change.

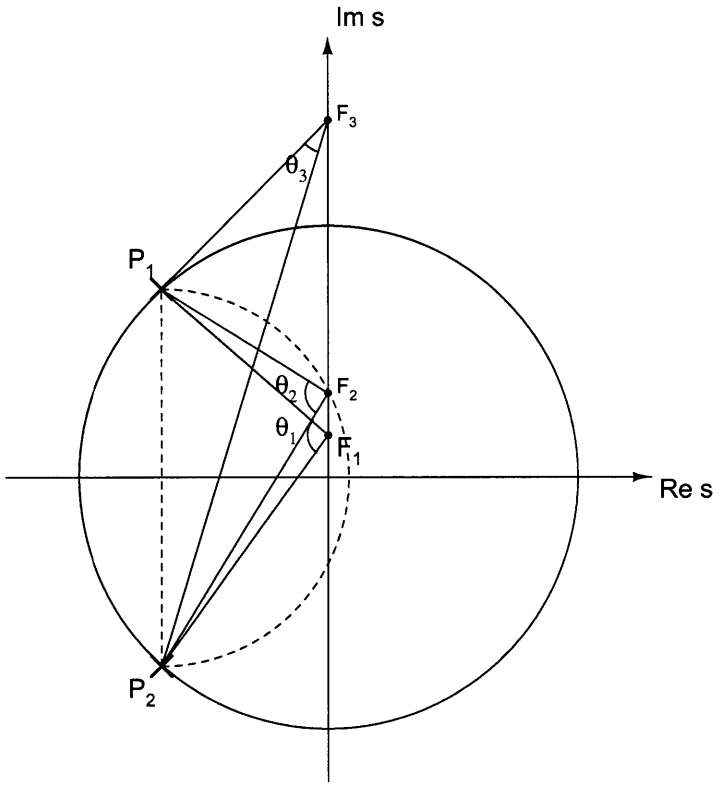

Figure 6-6: The gain peaking happens at the point F_2, where semi-circle with diameter P_1P_2 intersects with the $j\omega$ axis. Only θ_2 achieves 90°. So there is only one peaking frequency.

Chapter 7

Conclusion

7.1 Summary of the Works

Modelling, simulation, and implementation of approximate scalar NLL in discrete electronics

We first formed the LFSR synchronization problem as a least square estimation problem with goal and constraints. Then we used message-passing algorithms running on the factor graph to solve the estimation problem, which resulted the estimator called noise-locked loop(NLL). We focused on the approximate scalar NLL. Pseudo-continuous time simulation has done in Matlab. Continuous-time modelling and simulation of the circuit has been done in CppSim. The circuit of approximate scalar NLL was implemented in discrete electronics.

Simulation and comparison of the performance of scalar NLL and linear filter

In any spread spectrum receiver, right after the signal is collected by antenna, a linear filter is used to select the frequency-spread signal and pass it to CDR(clock and data recovery) circuit. We explored the possibility of using NLL to replace this linear filter in hope of getting better noise rejection and providing cleaner signal to CDR. We compared their performance in terms of mean square error of filtered signal with

transmitted signal. The result shows that scalar NLL improves the noise rejection performance of a linear filter. However the amount of improvement is limited and its effect to later CDR circuit needs to be explored more.

Approximate message-passing in big analog network

As best as we know, the previous work on circuit implementation of message-passing algorithm, like analog decoders, has been focused on realizing exact message-passing computation in circuits. However we approximate the LLR soft-xor by multiplication. The circuit implementation becomes temperature independent. We discovered the approximate scalar NLL has faster acquisition time and better noise rejection ability than the exact form NLL in low SNR regime. And some preliminary result shows the approximate analog decoder using soft-xor approximation has almost the same performance as the conventional analog decoder using exact form computation. This raises an interesting direction for further exploration.

Circuit primitives for vector NLL and general belief propagation

The scalar message-passing has two fundamental computation units: soft-Xor(parity-check node) and soft-equal(equality node). They are actually implemented by one fundamental circuit: Gilbert cell(thanks Barrie!). And they use only one fundamental principle: translinear principle. All the analog decoders are built on these. Interestingly enough, our proposed circuits for generalized belief propagation(GBP)[31] [29] [4] are simply the current-mode translinear circuits. As far as we know, we are the first to propose the circuit structure for generalized belief propagation. Thanks to Barrie Gilbert again. The fundamental circuit structure we use for GBP has been published in his 1968 JSSC papers almost forty years ago.

Current-mode signal restoration circuit for hybrid signal processing

Along the line from belief propagation to generalized belief propagation, and from scalar NLL to vector NLL, building large robust analog network is the goal. Inspired by the idea of hybrid state machine published in [25], we investigated the signal

restoration circuit, which is very useful to keep the fidelity of analog signal in large system. The circuit we came up with is a purely current-mode feedback structure that has never been used before(to our best knowledge). It utilizes the high current-gain of winner-take-all cell to generate linear range as wide as two rails. The current gain is controllable by simple DAC. The circuit will be fabricated with MOSIS 0.5um technology.

Another geometric approach for all-pole under-damped second order transfer function

This is a geometric approach to understand the behavior of second order transfer function. Mirror the two poles to right half plane, connect the frequency point to one left half plane pole and to the mirrored image of the other pole. The angle expanded by these two lines are the phase of the transfer function at this frequency. The triangle composed of frequency point and two left half plane poles has invariant area no matter when the frequency point is. Gain is the sin of the angle facing to two poles. This geometric approach will be useful for design and analysis on the filters in noise-locked loop.

7.2 Future Work and Outlook

While this thesis has accomplished the work mentioned above, its function is not to terminate the research but rather to raise more interesting questions which direct the future research.

Approximate message-passing in large analog networks

It is very interesting to investigate how message-passing algorithm can tolerate the nonlinearity. The question is related to non-normalized probability theory and statistical estimation which has much room to explore. In practice, it may greatly improve the robustness of large analog network.

Construct circuits to do generalized belief propagation

We recognized the current-mode quotient and product circuit as the exact circuit we need for GBP. However due to time limit, we didn't fully simulate the circuits. We relied on the fact that such a simple translinear circuit would certainly work. Further study requires careful simulation of the larger system like vector NLL and optimization of circuit structure and performance in terms of low power or high speed. Also find other good applications of GBP will stimulate more interests on analogic implementation.

Low power high speed NLL

To design low complexity, low power NLL will make it more useful for communication systems. The scalar NLL now uses active filters which consumes a lot of power. If we can implement analog delay as passive filter but also maintain performance, the power will dramatically reduced. This is not hard for high speed circuit. The multiplier has about ten transistors in core plus four for nonlinear compensation. Summing can also be implemented by Gilbert cell plus inverse hyperbolic function, which needs about eight transistors. So the soft-Xor and soft-equal computation core needs about 25-30 transistors. If we can use only few transistors mixer, even diode mixer for soft-Xor, the circuit structure will be greatly simplified. This requires careful study of the impact of all the mixer nonidealities to message-passing algorithm.

Appendix A

Appendix

A.1 Continuous-Time Simulation Code in CppSim

A.1.1 Module Description Code

```
%
% White Gaussian Noise
%
module: noise
parameters: double var
inputs:
outputs:  double out
static_variables:
classes: Rand randg("gauss")                                    10
code: out = sqrt(var/Ts)*randg.inp();

%
% First Biquad Section of 8th Order Bessel Low Pass Filter
%
module: myBesselbiquad1
```

```
parameters: double fp
inputs: double in
outputs: double out
classes: Filter filt("48.4320186525887","48.4320186525887
                      +5.677967896/(2*pi*fp)*s
                      +1/(2*pi*fp)^2*s^2","fp,Ts",fp,Ts);
code:
filt.inp(in);
out=filt.out;

%
% Second Biquad Section of 8th Order Bessel Low Pass Filter
%
module: myBesselbiquad2
parameters: double fp
inputs: double in
outputs: double out
classes: Filter filt("38.56925327532649","38.56925327532649
                      +8.736578434/(2*pi*fp)*s
                      +1/(2*pi*fp)^2*s^2","fp,Ts",fp,Ts);
code:
filt.inp(in);
out=filt.out;

%
% Third Biquad Section of 8th Order Bessel Low Pass Filter
%
module: myBesselbiquad3
parameters: double fp
inputs: double in
```

```
outputs: double out
classes: Filter filt("33.93474008457527","33.93474008457527
                    +10.40968158/(2*pi*fp)*s
                    +1/(2*pi*fp)^2*s^2","fp,Ts",fp,Ts);
code:
filt.inp(in);
out=filt.out;

%
% Fourth Biquad Section of 8th Order Bessel Low Pass Filter
%
module: myBesselbiquad4
parameters: double fp
inputs: double in
outputs: double out
classes: Filter filt("31.97722525877369","31.9772252587736
                    +11.17577209/(2*pi*fp)*s
                    +1/(2*pi*fp)^2*s^2","fp,Ts",fp,Ts);
code:
filt.inp(in);
out=filt.out;

%
% Exact Form LLR Soft-Xor
%
module: mysoftXOR2
parameters:
inputs: double inx, double iny
outputs: double out
classes:
```

code: out=2*atanh(tanh(inx/2)*tanh(iny/2));

%

% Approximate LLR Soft-Xor

%

module: multiplier

parameters:

inputs: double in1, double in2

outputs: double out

classes:

static_variables:

code: out = in1 * in2;

%

% Exact Form LLR Soft-Equal

%

module: add2

parameters:

inputs: double in1 double in2

outputs: double out

static_variables:

code: out = in1+in2;

%

% Clamp

%

module: clipper

parameters: double min, double max

inputs: double in

outputs: double out

```
classes:
code:
if (in<=min)
        out=min;
else
{if (in>=max)
        out=max;
else
        out=in;
}
```

A.1.2 Postprocessing Code in Matlab

```
%
% CppSim generates transcient analysis data in 'test.tr0'
% Signals are loaded into Matlab by 'evalsig()'command
%
cppsim;
x = loadsig('test.tr0');
lssig(x);
t = evalsig(x,'TIME');
in = evalsig(x,'in');
xorout=evalsig(x,'xorout');
xorout2=evalsig(x,'xorout2');
out=evalsig(x,'out');
out2=evalsig(x,'out2');
outclean=evalsig(x,'outclean');
recv=evalsig(x,'recv');
noise=evalsig(x,'noise');
d1=evalsig(x,'d1');
```

```
d2=evalsig(x,'d2');
d3=evalsig(x,'d3');
d4=evalsig(x,'d4');
v1=evalsig(x,'v1');
v2=evalsig(x,'v2');
v3=evalsig(x,'v3');
v4=evalsig(x,'v4');
outcmp=evalsig(x,'outcmp');
cleansig=evalsig(x,'cleansig');
out_edge=evalsig(x,'out_edge');
out2_edge=evalsig(x,'out2_edge');
in_edge=evalsig(x,'in_edge');
linfilt=evalsig(x,'linfilt');

%
% FFT
%
N=length(out2);
Fs=(5e6);
f=[-0.5*N:0.5*N-1]*Fs/N;
Out2=fftshift(fft(out2)/N);
Xorout2=fftshift(fft(xorout2)/N);
Xorout=fftshift(fft(xorout)/N);
Linfilt=fftshift(fft(linfilt)/N);
D1=fftshift(fft(d1)/N);
D2=fftshift(fft(d2)/N);
D3=fftshift(fft(d3)/N);
D4=fftshift(fft(d4)/N);
Outcmp=fftshift(fft(outcmp)/N);
Cleansig=fftshift(fft(cleansig)/N);
```

```
Noise=fftshift(fft(noise)/N);
Recv=fftshift(fft(recv)/N);
In=fftshift(fft(in)/N);

%
% Calculation of Power from PSD
%
powerclean=0;
for i=length(Cleansig)/2:(length(Cleansig)/2+length(Cleansig)*1.3e5/5e6)
    powerclean=powerclean+(abs(Cleansig(i)))^2;
end
powerclean

powerrecv=0;
for i=length(Recv)/2:(length(Recv)/2+length(Recv)*1.3e5/5e6)
    powerrecv=powerrecv+(abs(Recv(i)))^2;
end
powerrecv

powerd4=0;
for i=length(D4)/2:(length(D4)/2+length(D4)*2.6e5/5e6)
    powerd4=powerd4+(abs(D4(i)))^2;
end
powerd4

%
% Calculation of MSE
%
mse_approx = mean((in-d2/1.7).^2)
mse_exact = mean((in-xorout/1.45).^2)
```

```
mse_linfilt = mean((in-linfilt/2.1).^2)

%                                                              80
% Print PSD for Clean Signal and Signal After Delay Line
%
figure(1)
    semilogy(f,abs(D4).^2);
figure(2)
     semilogy(f,abs(Cleansig).^2);
```

A.2 Pseudo-Continuous Time Simulation Code in Matlab

```
%
% Pseudo-continuous time LLR NLL
%

bins=4; % g(D)=1+D+D^4 LFSR generator
time = 9000; % 9000 bits
smp = 20; %sampling rate
timect = time*smp; %total sampling points
outputfromLFSR = zeros(1,time);                                10
outputfromLFSR = LFSR(time, bins, 1, bins);
outputfromLFSRct = zeros(1,timect);
%mapper 1-> -1, 0->1
for i=1:time
    if outputfromLFSR(i)==1
        outputfromLFSR(i)=-1;
```

```
    else outputfromLFSR(i)=1;
    end
    outputfromLFSRct(smp*(i-1)+1:smp*i)=outputfromLFSR(i);
end

VDD=5; % modelling the power rail in real circuit
sigma=0.8; % noise standard deviation
input = outputfromLFSRct+sigma*randn(1,timect);
Ldelay1 = zeros(1,timect); % output of first delay filter
Ldelayn = zeros(1,timect); % output of the last delay filter
out = zeros(1,timect);
outbitslicer=zeros(1,timect);
out2 = zeros(1,timect);
out3 = zeros(1,timect);
La = zeros(1,timect); % received signal
Lb = zeros(1,timect); % output of soft-xor
Lxx = zeros(1,timect); % output of soft-equal
for i=1:timect
    La(i) = input(i);
if input(i)>=0
    outbitslicer(i)=1;
else
    outbitslicer(i)=-1;
end

%
% parity-check node in LLR
% four cases are simulated:
% approximate and exact NLL with clipping to model real multiplier;
% approximate and exact NLL without clipping.
```

```
% each case is simulated separately in real simulation
%

%Case 1: approximate NLL with clipping of output                    50
  Lb(i) = max(min(Ldelay1(i)*Ldelayn(i)/3,VDD/2),-1*VDD/2);

%Case 2: exact NLL with clipping at output
  Lb(i) = max(min(2*atanh(tanh(Ldelay1(i)/2)*\\
          tanh(Ldelayn(i)/2)),VDD/2),-1*VDD/2);

%Case 3:  exact NLL without clipping at output
  Lb(i) = 2*atanh(tanh(Ldelay1(i)/2)*tanh(Ldelayn(i)/2));

%Case 4:  approximate NLL without clipping                          60
  Lb(i) = Ldelay1(i)*Ldelayn(i)/3;

%
% equality-node in LLR
%
% Case 1: soft-equal with clipping
Lxx(i) = max(min(La(i) + Lb(i),VDD),-1*VDD);
% Case 2: soft-equal without clipping
Lxx(i) = La(i) + Lb(i);
                                                                   70
%
% dynamics of system
%
if (i>=smp)
    if (Lxx(i-smp+1)>=1.4)    % the clipping at input of multiplier
       Ldelay1(i+1)=1.4;
```

```
      elseif (Lxx(i-smp+1)<=-1.4)
            Ldelay1(i+1)=-1.4;
      else
        Ldelay1(i+1) = Lxx(i-smp+1);
      end
end
if (i>=smp*bins)
    if (Lxx(i-smp+1)>=1.4)
        Ldelayn(i+1)=1.4;
    elseif (Lxx(i-smp+1)<=-1.4)
            Ldelayn(i+1)=-1.4;
    else
        Ldelayn(i+1) = 1*Lxx(i-smp*bins+1);
    end
end
%
% A posteriori probability in LLR estimation of current state
%
    out2(i) = Lxx(i);
%
% hard-decision rule: if log(p(0)/p(1))>0, then out=1
%
    if Lxx(i) > 0
        out(i)=1;
    else
       out(i)=-1;
    end
end
```

Bibliography

[1] Behnan Analui and Ali Hajimiri. Statistical analysis of integrated passive delay lines. *IEEE Custom Integrated Circuits Conference*, pages 107–110, Sep. 2003.

[2] Justin Dauwels, Matthias Frey, Tobias Koch, Hans-Andrea Loeliger, Patrik Merkli, and Benjamin Vigoda. Synchronization of a pseudo-noise signal using an analog circuit. *Technical Report No. 200401,Swiss Federal Institute of Technology, Zürich*, January, 2004.

[3] Justin Dauwels, Matthias Frey, Tobias Koch, Hans-Andrea Loeliger, Patrik Merkli, and Benjamin Vigoda. An analog circuit that locks onto a pseudo-noise signal. *Technical Report No. 200403,Swiss Federal Institute of Technology, Zürich*, June, 2004.

[4] Justin Dauwels, Hans-Andrea Loeliger, Patrick Merkli, and Maja Ostojic. On structured-summary propagation, lfsr synchronization, and low-complexity trellis decoding. *Proc. 41st Allerton Conf. on Communication, Control, and Computing*, pages 459–467, Oct.1-3 2003.

[5] Brendan J. Frey. *Graphical Models for Machine Learning and Digital Communication.* The MIT Press, 1998.

[6] N. Gershenfeld and G. Grinstein. Entrainment and communication with dissipative pseudorandom dynamics. *Physical Review Letters*, 70(25):107–110, Sep. 1995.

[7] Barrie Gilbert. A new wide-band amplifier technique. *IEEE Journal of Solid-State Circuits*, pages 353–365, December 1968.

[8] Barrie Gilbert. A precise four-quadrant multiplier with subnanosecond respond. *IEEE Journal of Solid-State Circuits*, pages 365–373, December 1968.

[9] Paul R. Gray, Paul J. Hurst, Stephen H. Lewis, and Robert G. Meyer. *Analysis and Design of Analog Integrated Circuits.* John Wiley Sons, Inc., 2001.

[10] J. Hagenauer, Moerz M., and Offer E. Recent progress in decoding with analog vlsi networks. *34th Conference on Information Sciences and Systems (CISS),Princeton University, NJ, USA*, March 2000.

[11] Tobias Koch. Continuous-time synchronization. *Semester Project,Swiss Federal Institute of Technology, Zürich*, June, Winter,2002/2003.

[12] F.R. Kschischang, B.J. Frey, and H.-A. Loeliger. Factor graphs and the sum-prodcut algorithm. *IEEE Trans. Inform. Theory*, 47:498–519, Feb. 2001.

[13] S.L. Lauritzen. *Graphical Models.* Oxford University Press, New York NY, 1996.

[14] Thomas Lee. *The Design of CMOS Radio-Frequency Integrated Circuits.* Cambridge University Press, 1998.

[15] H.-A. Loeliger, F. Lustenberger, M. Helfenstein, and F. Tarköy. Probability propagation and decoding in analog vlsi. *IEEE Trans. Inform. Theory*, 47:837–843, Feburary 2001.

[16] Hans-Andrea Loeliger. Introduction to factor graph. *IEEE Signal Processing Mag.*, January 2004.

[17] L.Storch. Synthesis of constant-time-delay ladder networks using bessel polynomials. *Proc.IRE*, 42:1666–1675, 1954.

[18] Felix Lustenberger. *On the Design of Analog VLSI Iterative Decoders.* PhD thesis, Swiss Federal Institute of Technology, Zürich, 2000.

[19] J. Pearl. *Probabilistic Reasoning in Intelligent Systems.* Morgan Kaufmann, San Mateo CA., 1988.

[20] M. Perrott. Cppsim behavioral simulation package. 2002.

[21] M. Perrott. Fast and accurate behavioral simulation of fractional-n synthesizers and other pll/dll circuits. *Design Automation Conference (DAC)*, pages 498–503, 2002.

[22] Roger L. Peterson, Rodger E. Ziemer, and David E. Borth. *Introduction to Spread Spectrum Communications.* Prentice Hall, 1995.

[23] Rahul Sarpeshkar. *Efficient Precise Computation with Noisy Components: Extrapolating From a Electronic Cochlea to the Brain.* PhD thesis, California Institute of Technology, 1997.

[24] Rahul Sarpeshkar. Current-mode static and dynamic circuits. *Lecture Note of Low-Power Analog VLSI*, Oct 2004.

[25] Rahul Sarpeshkar and Micah O'Halloran. Scalable hybrid computation with spikes. *Neural Computation*, pages 2003–2038, 2002.

[26] Rolf Schaumann and Mac E. Van Valkenburg. *Design of Analog Filters.* Oxford University Press, 2003.

[27] Donald R. Stephens. *Phase-Locked Loops For Wireless Communication: Digital and Analog Implmentations.* Kluwer Academic Publishers, 1998.

[28] Benjamin Vigoda. A nonlinear dynamic system for spread spectrum code acquisition. Master's thesis, MIT, August 1999.

[29] Benjamin Vigoda. *Analog Logic: Continuous-Time Analog Circuits for Statistical Signal Processing.* PhD thesis, Massachusetts Institute of Technology, June 2003.

[30] Benjamin Vigoda, Justin Dauwels, Neil Gershenfeld, and Hans-Andrea Loeliger. Low-complexity lfsr synchronization by forward-only message passing. *Submitted to IEEE Trans. on Information Theory*, June, 2003.

[31] Jonathan S. Yedidia, William T. Freeman, and Yairs Weiss. Generalized belief propagation. *MERL Technical Report TR-2000-26*, Jun. 2000.

[32] Jonathan S. Yedidia, William T. Freeman, and Yairs Weiss. Bethe free energies, kikuchi approximations, and belief propagation algorithms. *MERL Technical Report TR-2001-16*, 2001.

[33] Jonathan S. Yedidia, William T. Freeman, and Yairs Weiss. Characterizing belief propagation and its generalizations. *MERL Technical Report TR-2001-15*, 2001.