August 9, 1999

A Nonlinear Dynamic System for Spread Spectrum Code Acquisition

Benjamin Vigoda

In partial completion of S.M. at the MIT Media Laboratory

1 Introduction

1.1 Nonlinear Dynamics for Computation

Nonlinear differential equations and iterated maps can perform any computation. They are Turing equivalent, which means that they can solve the same computational problems as a Turing Machine in about the same amount of time [MOO99]. To make this precise, systems that are in the class called "Turing equivalent" can solve the same computational problems in a time that is a polynomial of the amount of time required by any other machine in this class to solve the same problem [SIP97].

Sometimes, the most difficult part of performing a useful computation, however, is writing the program. Furthermore, engineers often care about a "mere" polynomial speedup, and so we often need to build special purpose computing hardware suited to run a particular program very efficiently. A perfect example of this is high speed digital signal processing (DSP) hardware that is optimized for performing real-time convolutions (often for filtering) on signals.

Although nonlinear dynamic systems will not far exceed conventional Von Neumann machines in general, for example by solving NP complete problems in polynomial time, nonlinear dynamics does provide a novel and useful language for devising new "algorithms" and "computer architectures." Nonlinear dynamics is a relatively new field, though, and its application to computation is still essentially in its infancy. In fact, the system proposed here is one of the first nonlinear dynamic systems engineered to perform a practical computation.

Carver Mead at Caltech has proposed that there is an engineering trade-off between the flexibility of a computing machine architecture to handle many different problems, and its power to solve a specific problem efficiently. For example, the highly parallel architecture of the visual cortex is perfectly organized to excel at edge/feature detection in a visual scene, but it would take a major restructuring at the hardware (wetware) level for the visual cortex to perform any computation other than the one for which it evolved. On the other hand, we have not yet been able to construct a Vonn Neumann architecture machine that can perform the functions of the visual cortex in real time, even though the time order behavior of a silicon transistors is more than five orders of magnitude faster than the time order behavior of the biological neuron. The human nervous system gains incredible computational leverage by having hardware specialized to handle specific tasks. We currently sacrifice this power in the personal computers by requiring that they have an architecture general enough to run any program. We can therefore expect a productive evolution of specialized computing hardware long after we reach the size and time scaling limits of silicon. The work presented here is intended as a step in this direction. Section 2 on page 13 provides an introduction to nonlinear dynamic systems theory to acquaint the reader with the basics of this mathematical language.

1.2 AFSR (Analog Feedback Shift Register)

An ordinary FM or AM radio has a resonant circuit constructed from an inductor and capacitor that resonates with or entrains to a particular incoming electromagnetic wave or "carrier frequency". The resonant circuit is a simple physical mechanism that resonates with one particular incoming signal while rejecting the others. Tuning into a modern digital wireless communication is not so simple. Modern digital wireless uses *Spread Spectrum (SS)* in which a "pseudo-random bit stream" is employed as a "pseudo-carrier." Current state of the art SS receivers use relatively expensive, power-hungry, high speed digital signal processing (DSP) chips to tune into this incoming pseudo-random bit stream. This approach, however, does not scale well for making ubiquitous short-haul wireless communications for application in inexpensive consumer products. As an alternative to using a DSP chip to tune into a SS signal, one could imagine a physical structure analogous to a resonant circuit which would entrain to an incoming pseudo-random bit stream.

In fact, a mathematical system with the necessary dynamics has been proposed by Grinstein and Gershenfeld [GER95]. I have improved upon this mathematical system, the AFSR (Analog Feedback Shift Register), and taken steps toward implementing it in hardware. The AFSR could potentially be a component in a less expensive and more energy efficient front end for SS receivers. The AFSR structures may also have other applications in IrDA transceivers, remotely read auto-I.D. tags, or flat screen technology. AFSR is potentially useful in applications where low cost, low power addressing or channel sharing is desired.

1.3 The Need for Low Power, Low Cost Transceivers that Share the Channel

Demand for bandwidth resources as well as consumer applications will drive the development of low cost, low power short range transceivers. We can make more efficient use of bandwidth if we tile a cellular wireless system using a greater number of shorter range transceivers. We call this *Space Division Multiple Access (SDMA)* to draw a parallel to Time and Code Division Multiple Access (TDMA and CDMA) which will be explained later. In a cellular or multi-hop peer-to-peer network, messages hop from one transceiver to another across a field of transceivers. The basic idea of SDM is if we simultaneously increase the density and decrease the range of the transceivers, the total number of message packets in transit across the network at any given instant can be greater, because each message is occupying less physical space at any given instant in time. To avoid frequent packet collisions, however, each transceiver needs to be able to communicate on a number of different channels. Short range transceivers need to be able to share the available spectrum.

In addition to their use in wireless networks, short range transceivers will soon find consumer applications in the home, automobile, and likely in wearable or “personal area” networks (P.A.N.). If the Internet is to become portable, then short-range transceivers will become increasingly important for transmitting information over the last few meters to the user. Just as with present day cell phones, many of these devices will need to be able to communicate on a shared spectrum to a single base station. Therefore these transceivers require a low cost, low power, low complexity scheme for channel sharing.

1.4 Direct Sequence Spread Spectrum (DS SS) for Simple Channel Sharing

The problem that immediately arises in multi-transceiver radio systems is channel sharing. How can multiple transceivers operate in proximity to one another without causing interference problems? Because short range transceivers are to be portable and/or ubiquitous, they should have a relatively low complexity. We therefore need a relatively simple system for sharing available bandwidth. *Direct Sequence Spread Spectrum (DS SS)* offers an attractive solution to this channel sharing problem.

The advantages of Spread Spectrum (SS) in general and DS SS in particular are manifold and virtually all high performance commercial wireless systems employ DS SS. In addition to providing a simple scheme for channel code allocation, DS SS offers low peak power, excellent resistance to interference including interference from echoes known as *multipath interference*, easily scalable processing gain, excellent bandwidth efficiency when used in a microcell system, and graceful rather than catastrophic bit error rate (BER) degradation as more transceivers share the channel. In addition, improvements for DS SS systems are of interest for their applications in making high resolution timing measurements as in global positioning systems (GPS).

1.5 The Acquisition Problem

As bits leave the transmitter in DS SS, they are multiplied by a string of “pseudo-random bits” known as a *Pseudo-random Number sequence (PN sequence)*. This PN sequence is a string of bits that appears random by statistical standards, but which are actually generated by a deterministic algorithm. The outcome of this multiplication of data bits with the PN sequence is a bit stream which looks still looks like white noise, but which has the data hidden in it. All that is required for a receiver to recover the original data is to have an identical PN sequence generator running in synchrony to the PN sequence generator in the transmitter. The receiver can then subtract out the PN sequence from the received signal thereby leaving the original data. In the process, the receiver ignores all other data signals modulated by other PN sequences, which is why DS SS helps solve the channel sharing problem. A key difficulty lies in getting the PN sequence generator in the receiver to synchronize with or *acquire* the PN sequence coming from the transmitter, and then to maintain synchronization. Achieving this synchronization is called *code acquisition*. Maintaining synchronization is called *tracking*.

State of the art acquisition systems can have long and unpredictable acquisition times which make them unacceptable for multi-hop peer-to-peer networks which make and break connections often as they route traffic. This kind of network will require acquisition times much less than the time it takes to transmit a packet.   Present day acquisition systems are also power-hungry, because they use high speed Digital Signal Processing (DSP). The energy consumption of these processors scales linearly (at best) with the frequency of operations. State of the art acquisition systems are also relatively expensive to manufacture because it is impossible to integrate the high speed 3,5 semiconductor components they require with the baseband system for which standard silicon is adequate. AFSR is a nonlinear dynamic system designed to perform spread spectrum code acquisition. It has the potential to be lower cost by being integrated and lower power by being “adiabatic” compared to the present state of the art SS acquisition systems. One can think of AFSR as a kind of phase locked loop (PLL) which has a “voltage controlled PN sequence generator” instead of a voltage controlled oscillator (VCO). My recent work on AFSR also holds promise for performing tracking as well as acquisition.

2 Nonlinear Dynamics

2.1 Introduction to Nonlinear Dynamics

Never forget that the primary method in the field of nonlinear dynamic systems is simply varying the coefficients of the nonlinear terms in a nonlinear differential equation and examining the behavior of the solutions.

A*nalytical methods* (i.e. using pen and paper) are fine for finding closed form solutions to linear differential equations. Indeed most of the history of physics can be seen as the attempt to fit linear differential equations to natural phenomenon and analytically predict their behavior. This was not so much out of choice, but because the behavior of linear models can be can be predicted using the tools at hand, namely, pen and paper. The methods of analysis used in nonlinear dynamic systems theory, such as varying the coefficients in a nonlinear differential equation and observing the solutions, have been understood to be desirable for quite a long time, but required the advent of modern computers in order to be practical.

An nth order linear differential equation is an equation of the form

$$ {}_N\frac{d^N x}{dt^N} + A_{N-1}\frac{d^{N-1}x}{dt^{N-1}}\dots + A_2\frac{d^2x}{dt^2} + A_1\frac{d^1x}{dt^1} + A_0 x = f(\qquad (2.1.1)$$

The term on the right hand side, $f(t)$, is known as a "forcing term." If $f(t) = 0$, then this is said to be a homogeneous equation. This equation can also be manipulated so that it can be written as a system of linear differential equations in the form

$$\frac{d\vec{x}}{dt} = A\vec{x} \qquad (2.1.2)$$

where $\vec{x}$ is now a vector, $(x_1, x_2, \dots, x_N)$, and ***A*** is now a matrix. A nonlinear differential equation is just a linear differential equation with at least one nonlinear term, a term either of the form, $A_n g\left(\frac{d^n x}{dt^n}\right)$, where g is a polynomial or transcendental function, or of the form $A_n\left(\frac{d^p x}{dt^p}\right)\left(\frac{d^n x}{dt^n}\right)$, where p and n can be the same or different.

Almost all differential equations in physics are no greater than second order, because two phenomenon that occur at vastly different time scales (vastly different orders of derivative) rarely have a strong effect within a single system. This leaves physics with only a handful of linear second order (or fewer) differential equations with which to describe physical phenomenon. In fact we can name all of these handful. They are the Wave (Simple Harmonic Oscillator), Diffusion (Heat), and Laplace equations.

On the other hand, it seems that there can be an enormous number of interesting nonlinear alterations to this handful of linear differential equations. Why has physics been so successful using only these limited linear models? Is the universe really linear? The answer is that physics has indeed confined itself to a subset of physical phenomenon, because without computers it was essentially too difficult to understand the behavior of most nonlinear differential equations. The exception to this is perturbation theory which extends the power of analytical methods by using series approximations as a method for finding approximate solutions to *weakly nonlinear differential equations, linear differential equations containing small nonlinear terms.* In perturbation theory, the coefficient of a nonlinear term is required to be small.

Since we are now asking to find the solution to nonlinear differential equations, we might ask if we are always guaranteed of finding one? In fact, there is always a solution to a linear or a nonlinear differential equation local to a given an *initial condition (I.C.)*. This is stated formally by the local existence and uniqueness theorem (cf. Coddington and Levinson, 1955, Hirsch and Smale, 1974).

Theorem. **(2.1.3)**

Let $(U \subset \Re^n)$ be an open subset of real Euclidean space (or a differentiable manifold M), let $f: (U \to \Re^n)$ be a continuously differentiable (C^1)map and let $x_0 \in U$. Then there is some constant $c > 0$ and a unique solution $\phi(x_0, \ )|((-c, c) \to U)$ satisfying the differential equation $\dot{x} = f(x)$ with initial condition $x(0) = x_0$ [GUC83].

In other words, for any possible differential equation $\dot{x} = f(x)$, a domain U which is a subset of real Euclidean space for which there is a smooth mapping of U to the real numbers, and given some initial condition, x_0, there exists a single solution, ϕ (defined on an interval (-c,c) that maps to U) that passes through x_0. It is important to note, however, that this is a local existence theorem. In particular, for nonlinear differential equations, we are not guaranteed the existence of solutions for all time.

Now that we know that nonlinear differential equations always have a solution, how can we find and understand their behavior? What is needed is a method for examining the solutions of an equation as we modify its coefficients, so that we can understand the effects of increasing the amount of nonlinearity in an equation. Computers with their propensity for numerical solution and graphical representation, make this possible. Even with computers, however, the idea of making a mathematical field about varying any parameter in any system of differential equations is a seemingly vast undertaking. It may seem impossible for there to be a cohesive field of nonlinear dynamics given that there are infinitely many kinds of differential equations and their solutions so varied. Once we examine the solutions to nonlinear differential equations, however, we find that the list of things we want to know in general about how solutions behave as we vary the coefficients of the equation is short. Essentially, we just ask whether the solution settles down or blows up out of control. Refinements to this include how fast it settles down, where to, with which initial conditions, and how often (statistics). Likewise one can ask how fast it blows up, where to, with which initial conditions and how often.

2.2 Lyapunov Methods

Some of the best methods for answering questions about whether the solutions of a differential equation will settle down or blow up come from Lyapunov theory. For a more complete exposition of Lyapunov stability theory, I strongly recommend chapters 3 and 4 of Slotine and Li's "Applied Nonlinear Control [SLO91]." There are actually two methods introduced by Lyapunov, the indirect and the direct method. "The indirect method, or linearization method, states that the stability properties of a nonlinear system in the close vicinity of an equilibrium point are essentially the same as those of its linearized approximation [SLO91]." In other words, if we are interested in solutions of a strongly nonlinear equation after it settles down to equilibrium, then we can essentially turn off the nonlinear terms in the equation, and just solve the linear differential equation.

2.2.1 Hyperbolic Fixed Points and the Indirect Lyapunov Method

A point, $\bar{x}$, is a *hyperbolic or non degenerate fixed point* when $Df(\bar{x})$, the Jacobian, D ,of the linearization, f, of the nonlinear flow around $\bar{x}$, has no eigenvalues with a zero real part. The asymptotic behavior of solutions near a hyperbolic fixed point is determined by the linearization of the system. A non-hyperbolic fixed point is best illustrated by an example from Guckenheimer and Holmes. We can rewrite the equation

$$\ddot{x} + \varepsilon x^2 \dot{x} + x = 0, \tag{2.2.1}$$

as

$$\begin{bmatrix} \dot{x}_1 \\ \dot{x}_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} - \varepsilon \begin{bmatrix} 0 \\ x_1^2 x_2 \end{bmatrix} \tag{2.2.2}$$

using $x_1 = x$ and $x_2 = \dot{x}$. The eigenvalues, $\lambda = \pm i$ have a zero real part, so there is no hyperbolic fixed point for this system. As a result, unless $\varepsilon = 0$, the fixed point $(x_1, x_2) = (0, 0)$ is not a center. If $\varepsilon > 0$ then it is a non hyperbolic or weak attracting spiral sink, and if $\varepsilon < 0$ it is a repelling source.

2.2.2 The Direct Lyapunov Function Method

This indirect method is well and good once we know that there are some equilibrium (stable) points that the system will settle toward, but how did we know about the stable points in the first place? How do we know that the system is stable at all? The direct Lyapunov method is a powerful tool for this kind of analysis. It is also known as the Lyapunov function method, because it involves trying to invent a Lyapunov function, usually "...an energy-like quantity that decreases for |x| sufficiently large, so that $x(t)$ remains bounded for all t and all bounded initial conditions. The method relies on finding a positive definite function $V: U \to \Re$ called the Lyapunov function which decreases along solution curves of the differential equation [GUC83]." It can be difficult to find a Lyapunov function, but "the power of the method comes from its generality: it is applicable to all kinds of control systems, be they time-varying or time-invariant, finite-dimensional or infinite dimensional [SLO91]." The following theorem states how a Lyapunov function can be sufficient to show that a system has a stable point.

Theorem (2.2.3)

Let $\bar{x}$ be a fixed point and $V: W \to \Re$ be a differentiable function defined on some neighborhood $W \subseteq U$ of $\bar{x}$ such that

(i) $V(\bar{x}) = 0$ and $V(x) > 0$ if $x \neq \bar{x}$ ($V(x)$ is positive everywhere, but zero at the fixed point). If

(ii) $\frac{dV(x)}{dx} < 0$ everywhere in U except where $x = \bar{x}$ and

$\dot{V} = \sum_{j=1}^{n} \dot{x}_j \frac{\partial}{\partial x_j} V$, then $\bar{x}$ is a stable fixed point.

(Look for negative diagonals of Jacobian matrix in higher dimensions)

iii) $\frac{dV(x)}{dx} < 0$ everywhere in U except where $x = \bar{x}$, then $\bar{x}$ is an *asymptotically stable* fixed point.

Guckenheimer and Holmes give the following example. Let us apply the direct Lyapunov function method to the nonlinear differential equation

$$m\ddot{x} + k(x + x^3) = 0 \tag{2.2.4}$$

If we let $\dot{x} = y$, then we can rewrite 2.2.4 as a system of first order equations

$$\begin{aligned} \dot{y} &= -\frac{k}{m}(x + x^3) \\ \dot{x} &= y \end{aligned} \tag{2.2.5}$$

The total energy of the system, $E(x, y)$, is the kinetic energy, $\frac{m\dot{x}^2}{2} = \frac{my^2}{2}$ plus the potential energy,

$$-k\int\left(\frac{x}{2} + \frac{x^3}{4}\right)dx = k\left(\frac{x^2}{2} + \frac{x^4}{4}\right). \tag{2.2.6}$$

$E(x, y)$ is indeed a Lyapunov function for the system, since $E(0, 0) = 0$ and $E(x, y) > 0$ for all $(x, y) \neq (0, 0)$ which satisfies part (i) of theorem 2.2.3. Taking the implicit partial derivatives of E, and substituting for $\dot{y}$ using eq 2.2.5, we get

$$\dot{E} = my\dot{y} + k(x + x^3)\dot{x} = -ky(x + x^3) + (x + x^3)y \equiv 0\,. \tag{2.2.7}$$

So our fixed point satisfies part (ii) of theorem 2.2.3. By adding some terms to E, it is possible to simply write down a more powerful Lyapunov function, we'll call V

$$V(x, y) = my\dot{y} + k(x + x^3) + \beta(\dot{x}y + x\dot{y} + \alpha x\dot{x})\,. \tag{2.2.8}$$

The implicit derivative of V

$$\dot{V} = -\beta\frac{k}{m}(x^2 + x^4) - (\alpha m - \beta)y^2 \tag{2.2.9}$$

is always negative if for all x and y, thereby satisfying (iii) of theorem 2.2.3, and assuring that (0,0) is a globally asymptotically stable fixed point.

2.3 Forcing Terms

If we have a forcing term, we can still write the equation as a closed system at the expense of adding some degrees of freedom. For example, a single degree of freedom nonlinear oscillator such as

$$\frac{d^2x}{dt^2} + g\left(x, \frac{dx}{dt}\right) = 0 \tag{2.3.1}$$

when given a forcing term becomes

$$\frac{d^2x}{dt^2} + g\left(x, \frac{dx}{dt}\right) = f(t) \tag{2.3.2}$$

which can be written as a three dimensional system of first order differential equations, with time as a dependent variable

$$\begin{aligned} \frac{dx}{dt} &= y \\ \frac{dy}{dt} &= -g(x, y) + f(\theta) \\ \frac{d\theta}{dt} &= 1. \end{aligned} \tag{2.3.3}$$

It is important to note, however, that “this [increase in the dimensionality] can introduce an uncountably infinite set of new phenomena [GUC83].”

2.4 Continuous Time Flows and Discrete Maps

We will be interested later in moving between continuous time (CT) and discrete time (DT) systems, so we review some of the methods for relating the two.

2.4.1 Flows in Linear Systems

Again following Guckenheimer and Holmes, as we have seen, we can write a linear system as

$$\dot{x} = Ax\,, \tag{2.4.1}$$

where $\boldsymbol{A}$ is an n x n matrix and $x \in \Re^n$. Its solution is can be written in the form $x(x_0, t)$ where x_0 is an initial condition (I.C.). All of the solutions to the system, taken together describe what is called a “flow,” ϕ_t. Calling this solution space a flow is essentially another way of noting that from any point in the solution space, there is a unique trajectory through that point, as was stated earlier in the existence and uniqueness theorem. We write a trajectory starting from a single initial condition x_0 as $\phi_t(x_0)$. For example, the familiar solution to a first order differential equation (even a vector first order differential equation as we have here) is

$$\phi_t(x_0) = x(x_0, t) = e^{tA}x_0\,, \tag{2.4.2}$$

where e^{tA} is the n x n matrix obtained by exponentiating A and defined by the convergent series

$$e^{tA} = \left[I + tA + \frac{t^2}{2!}A^2 + \ldots + \frac{t^n}{n!}A^n\right]. \tag{2.4.3}$$

For t and integer, $e^{tA}x_0$ can be understood as the repeated multiplication of e^A, t times to the initial vector x_0. A useful fact is that repeated multiplication of a matrix with non zero real eigenvalues on any vector eventually converges to the eigenvector of the matrix which has the largest eigenvalue. Therefore $e^{tA}x_0$ will eventually converge to an eigenvector of e^A which will also be an eigenvector of A from eq 2.4.3. We therefore know that a linear system will eventually reach an equilibrium steady state solution. This is why eigenvectors are important in physics; They provide us with global invariants for linear systems. e^A used to create a discrete time version of 2.4.1, as long as our time steps, $t = \tau$ are not too large. This is the essential idea behind Euler and Runge-Kutta methods for simulating the behavior of differential equations which will be used later to analyze our continuous time (CT) AFSR.

2.4.2 Poincare Maps

For a useful introduction to Poincare methods see Guckenheimer and Holmes pp. 16.

2.5 Nonlinear Control Theory

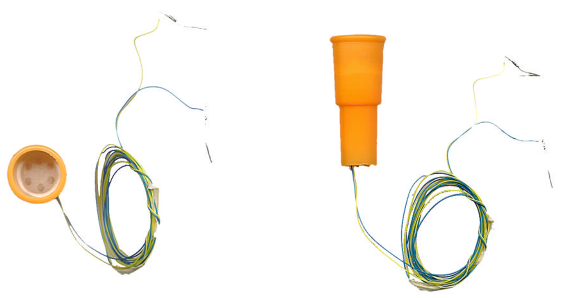


FIGURE 2.5.1 Miniature Gas Turbine Musical Instrument Employing Nonlinear Feedback Control, “Ezoo = Electronic + Kazoo”

2.5.1 Definitinions

Nonlinear control is good for:

1. wide operation range
2. “hard nonlinearities” - sharp thresholds etc. which are not linearizable
3. robust and adaptive controllers to control systems with parameter drift
4. simplicity - right tool for the job of controlling a nonlinear system
5. cheaper - don't have to spend money on linear controllers

Nonlinearity arises in control systems as *unintentional nonlinearity*: undesired centripetal forces or coulomb friction and also as *intentional nonlinearity*: big bang or adaptive controllers.

As discussed above, a *continuous nonlinearity* is a term in a differential equation of form transcendental function of (derivative of) independent variable, independent variable (or its derivative) raised to a power, product of two or more independent variables (includes powers of a single independent variable). By contrast a *discontinuous or “hard” nonlinearity* is created by hysteresis, backlash, stiction, or a threshold.

2.5.2 Equilibrium Points

Linear systems have only one equilibrium point while nonlinear systems have more than one equilibrium point, for example the system

$$\frac{dx}{dt} = -x + x^2 . \tag{2.5.1}$$

has the linearization

$$\frac{dx}{dt} = -x . \tag{2.5.2}$$

which can be solved analytically to find

$$x(t) = x_0 e^{-t} . \tag{2.5.3}$$

We also happen to be able to solve this particular nonlinear equation 2.5.1 analytically to find

$$x(t) = \frac{x_0 e^{-t}}{1 - x_0 + x_0 e^{-t}} . \tag{2.5.4}$$

If I had some diagrams of the solutions to the nonlinear equation and its linearization here, you could observe that the nonlinear equation has multiple fixed points while the linear equation has just one.

2.5.3 Limit Cycles, Bifurcations, Chaos and More

Van der Pol equation

m d2x/dt2 + 2c(x^2 - 1)dx/dt + kx = 0

As coefficient parameter of differential equation changes, the behavior changes qualitatively. At first bifurcation behavior is observed in which the basins of attraction of the system split and new fixed points are introduced. Eventually the system may transition to chaos where strong nonlinearities lead to sensitive dependence on initial conditions.

d2x/dt2 + .1dx/dt + x^5 = 6Sin(t)

Other behaviors of nonlinear control systems include jump resonance, subharmonic generation, asynchronous quenching, and frequency-amplitude dependence of free dependence.

2.6 Nonlinear Dynamics and Computation

The relationship between computers and nonlinear differential equations is actually quite deep. It turns out that computers are required to simulate the behavior of nonlinear differential equations because a nonlinear dynamic system is a computer, stated in a mathematical language which is unfamiliar to most computer scientists. In the language of nonlinear dynamics, computation means giving a nonlinear differential equation some initial conditions and allowing the state to flow to a final solution under the dynamics of the system.

2.6.1 Nonlinear Flows and The Halting Problem

The halting problem in computer science, stated in the language of nonlinear dynamics is the problem: “In general, can we predict if a given flow will pass through a given region of state space in finite time.” It turns out that in general the only way to decide if the nonlinear system will “halt” is to actually simulate the dynamics of the equation.

The definition of a linear update rule is $L(s1) + L(s2) = L(s1 + s2)$. In general, a linear update rule can always be written as a matrix multiplication of some matrix operator, M, on the vector of states. If we start with some initial condition vector, V_I, and repeatedly perform the same matrix operation (update rule) n times on the vector, to derive some final vector of states V_N, we can write this as

$$V_N = M \cdot M \cdot M \cdot M \cdot \ldots M \cdot V_I, \tag{2.6.1}$$

or

$$V_N = M^n \cdot V_I. \tag{2.6.2}$$

As $n \rightarrow \infty$, V_N will evolve to an eigenvector of M. So to find V_N, we didn't really have to do multiple matrix multiplications, we only had to solve the eigenvalue problem for the matrix M. Moreover, if M^n is invertible then the evolution is reversible in time, since

$$V_I = (M^{-1})^n \cdot V_N \qquad . \tag{2.6.3}$$

By contrast, a highly nonlinear update function will not yield to these methods of analysis. In a nonlinear update rule, the update may actually depend on the value of the initial vector to a power. In other words we are not simply adding and subtracting elements of the initial vector with one another to perform update. Sometimes we must multiply an element of the initial vector by itself. This will mean that an update rule itself depends at every time step on the values of the vector states. Even a simple iterative quadratic update rule as we will encounter in the following chapters changes depending on the initial vector in a way that a linear update rule does not. Therefore, we will be forced to simulate the system for n time steps to find V_N, and in general the process will not be reversible.

2.6.2 Discretization of State Space

Digital computing in stated in the language of nonlinear dynamics means that we discretize phase space. In other words, we carve up the continuous state space of the system into regions, and assign symbols to each region. For example if our state space is a voltage between 0 and 5V, we might say that all voltages less than 2.5 mean 0 and all voltages above 2.5 mean 1. Starting in a given region of state space corresponds to inputting a particular symbol. Ending in a particular region tells us the answer.

In a digital computer, after each gate, voltages near 0V are immediately mapped to 0V and voltages near 5V are mapped to 5V. For example, we might input the values of 4.9V and 4.85V into a XOR gate. The XOR can be implemented by a nonlinear transfer function which then outputs a value of .2V, say. This value would then be handed over to a comparator. The dynamics of the comparator has two strongly attracting sinks at 0 and 5V and an unstable fixed point source at 2.5V. This can be implemented by a dual supply infinite gain with a threshold at 2.5 volts. The comparator immediately maps the “imperfect” value of .2V to 0V completing the XOR operation.

The comparator therefore helps prevent the accumulation of errors from environmental noise that would arise if we tried to maintain analog values through multiple gate operations. This is generally a prudent thing to do, because it prevents errors from creeping into the computation, but the analog values between 0 and 5 Volts can be thought of as a wasted resource. In fact, some state of the art modern flash memory does make use of four voltage levels instead of two, but this is the exception that proves the rule.

Analog computation attempts to use this under exploited resource. Analog systems can perform useful computation if we choose the right problem for them to solve, and if we construct their dynamics in such a way as to make them fault tolerant. In fact Cris Moore has proved that two or higher dimension iterated analytic maps are Turing equivalent [MOO99]. If iterated maps can be carefully constructed to do many useful computations such as language recognition [MOO98]. If we don’t want to do the work of finding parameters for the iterated map function ourselves, a search algorithm can be used to find the parameters. Such a system, a search algorithm used to set parameters for a nonlinear function which will be iterated on data to perform a computation is known as an artificial recurrent neural network. Recurrent neural networks can “learn” to do language recognition instead of being hand designed.

The disadvantage of digital systems is that we lose the resource of all of the intermediate values that we have thresholded away. Those values could have been useful. Quantum physics tells us that at the most fundamental level, discretization must always occur when we measure a system, and we must measure our system in order to find the result of the computation. Although discretization is both useful and inevitable, however, it is possible to be more creative in how we apply it.

For example, if we hold a frog in the air its legs will flail around in an ever widening orbit in a futile attempt at locomotion. If however we place the frog on a surface with some gravity, the leg encounters the floor during each rotation, which constrains the orbit, and makes it stable. The gravity holding the frog down and the floor pushing up discretize the frog's leg orbits, and in this way are a part of the frogs walking computer. The discretization of phase space imposed by physical world helps the frog compute! We will soon use an identical trick in the dynamics of AFSR so that if we start the AFSR in a random initial condition, the dynamics will proceed along an ever widening orbit until they reach the outer limits of the phase space.

3 Short Range Transceivers

Digital wireless technology is an excellent candidate for applications of nonlinear dynamic computation, because radio systems tend to require high speed iterative computations on complex analog data with few branch points. It is unlikely though possible, that the word processor of the future will have a nonlinear partial differential equation as its engine, although image processing programs may benefit from these kinds of mathematical models. In looking for places to begin applying nonlinear dynamics to computation, we are therefore especially drawn to computations which are repetitive and that do not involve many branch points. By branch points we mean points in an algorithm where it must ask something like "are you sure you want to erase your entire thesis permanently?" The algorithm may ask a question of the user or of another computational module, but a branch point always requires that dynamics qualitatively change depending on the answer to the branch decision. For a nonlinear dynamic system this means there must be a point in the flow which is especially sensitive to external forcing, and from which the dynamics then proceed to two or more basins of state space with different dynamics. We would like to avoid this kind of complexity at first by choosing a problem that does not require branching. A good candidate algorithm for implementation in a nonlinear dynamic system should be a short loop.

Wireless technology is a good place to look for applications of nonlinear dynamic computation. Radio receivers must perform many iterative channel coding operations. Many of these algorithms are short loops without many branch points. In addition, radio designers are used to using analog components in order to perform computations which could be accomplished on a von Neumann architecture if only DSP chips were fast enough. The necessity and benefits of analog components for radio demodulation is firmly established. Finally, cell phones and other portable commercial and consumer wireless systems are very open to innovations which reduce the power consumption and increase the battery life of portable receivers. A nonlinear dynamic system could hope to offer lower power consumption because a flow through continuous state space could be more adiabatic than a typical DSP which performs rapid globally clocked transitions, sometimes even in the absence of data to be processed. Therefore, before we delve deeper into the details of nonlinear dynamics and computation, let us pause to make some observations about the future requirements of wireless transceivers.

3.1 Bandwidth Scaling Argument for Short Range Transceivers

3.1.1 The Growth of Wireless Communications Technology

The explosive growth of digital communications technology is one of the most exciting stories of our age. The information content of the World Wide Web is many millions of billions of bytes and growing. Millions of kilometers of copper and fiber optic cable connect us. Information crosses the globe via satellites and is delivered to our pockets via cellular networks. Digital wireless communications technology is becoming an increasingly important link in the global information infrastructure with the promise of providing cheap, lightweight, remote, reconfigurable, and of course, portable communications. Quite soon, we should be able to contact another person anywhere in the world by simply whispering their name.

Despite the importance and rapid growth of wireless communication technology, a simple scaling argument suggests that the infrastructure remains incomplete. There is a coming demand for low cost short range transceiver technology that is beginning to manifest itself. If we examine the wireless communications infrastructure, on the one hand we see voyager probes beaming electromagnetic signals to ground arrays across hundreds of millions of kilometers of space. On the other end of the scale are atoms communicating "wirelessly" via electromagnetic fields to adjoining atoms in copper or fiber optic cable. In between lies an enormous range of technologies covering virtually every size scale. Today, we routinely receive images from Mars, probe the atomic structure of a silicon wafer, talk to someone on the other side of the globe, and probe signals in the brain. It is still cost prohibitive, however, to deploy a few dozen small consumer objects that communicate wirelessly with one another across distances of meters. The table below clearly illustrates this state of affairs.

3.1.2 More Efficient Use of Bandwidth by Space Division Multiple Access (SDMA)

Distance in meters	Wireless Technology
10^6 and beyond	deep space probes, radio-telescopy
$10^5 - 10^6$	satellite communications
$10^4 - 10^5$	cross continental radio frequency or microwave links
$10^3 - 10^4$	commercial radio, cell phones, pagers, community wireless...
100s	digital household wireless phones, digital “citizen band” radio?
10s	costly wireless LAN’s, IBM promises BlueTooth TM
1s	
$10^{-1} - 1$	
$10^{-3} - 10^{-2}$s	infrared such IRDA for laptop or palm-top communication, Physics and Media’s prior electrostatic fringing field communication (Personal Area Network, PAN), electrostatic sensing...
$10^{-4} - 10^{-5}$	probes record electric fields near a neuron generated by electrical spikes inside the neuron
$10^{-6} - 10^{-7}$	electron tunneling in semiconductor structures
$10^{-8} - 10^{-9}$	atom to atom communication in a wire (mean free path of electron in copper)
10^{-10}	electron to nucleus spin coupling in the atom

TABLE 3.1.3 Wireless Technologies and Distance Scales Covered

The demand for bandwidth will push wireless communication technologies to fill in the empty rows (1’s and 10’s meter scales) in Table 3.1.3 on page 22. We are only given one electromagnetic spectrum. There are only a few ways to increase available bandwidth. First, we can push to higher, currently unused frequencies, second, we can make better use of the bandwidth we have available now by being clever about how we send data via improved compression or channel coding techniques, or third, we can cut up physical space into smaller cells.

We can more efficiently use available bandwidth by cutting up physical space. For example if WBUR, National Public Radio in the Boston Area, transmits on 91.9MHz covering a 10 kilometer radius sphere with large amount of radiated energy at this frequency, other systems cannot use the 91.9MHz band in that volume of space without risk of interference. On the other hand, if we were to fill the same volume of space with many smaller transceiver cells, we could sustain many simultaneous communications crossing this same space hopping from one small cell area to the next. The smaller the cells, the more efficiently we can tile the space. When the volume of these cells approaches atomic length scales, the cells become atoms and the channels become wires, which is the most efficient use of physical space bandwidth resources. Unfortunately, running wires can be expensive and cumbersome. Just imagine if a building could be fitted with a local area network (LAN) simply by replacing ceiling or floor tiles with inexpensive wireless transceiver tiles.

3.2 Applications for Short Range Transceiver Hardware

3.2.1 Client Devices for Low Cost Ubiquitous Wireless

In his 1997 Media Lab S.M. Thesis, Poor predicts that when “the cost of connecting to the digital network drops to a few dollars...” and is small enough to be ubiquitous, “you won’t ever need to set your watch again...tomorrow’s ‘radio’ will be a network appliance... the smoke detector in your basement... if it detects trouble... will start making calls until someone is found to address the problem...and toy dolls will know how to read, talk and listen" by being connected to the resources of the web [POO94].

In addition to firing our imagination with potential applications, Poor also presents a scaling argument involving IP address densities. "Today," he writes, "the density of network connections rarely exceeds one connection per person. But this ratio will change. The number of host computers connected to the internet is estimated to be growing by a factor of ten every three years...Networks will diffuse into objects much smaller than today's personal computer, and a typical room will contain dozens or hundreds of devices connected to networks [POO94]." In the past two years, we can already see the clear beginnings of Poor's predictions being born out.

3.2.2 Multi-hop Peer-to-Peer Networks

One exciting application of short-range transceivers would be multi-hop peer-to-peer networks composed of a large number of rather "dumb" short-range transceivers distributing connectivity throughout a building. One could even imagine this kind of network being spread across a wilderness area for environmental monitoring. In theory, such a network could be very reliable because of the redundancy of the inexpensive transceiver nodes, high bandwidth because of the spatial-multiplexing discussed above, easy to deploy, maintain, and scale because it would be a simple matter to scatter more nodes where they are needed.   Upon closer examination, the matter is unfortunately not so simple. Systems composed of a large number of locally communicating nodes (i.e. flat connected graphs) can exhibit some very counter-intuitive behaviors that can make engineering them quite difficult.

In percolation theory there is a well-known model of groundwater transport through sand or gravel. We can model this as water propagating from a source through an idealized filter, a volume filled with spheres of identical size. If the spheres are very small, i.e. have radius close to zero, they will be closely packed so that volume acts as a solid. Mathematically, this means that the probability of finding water a unit distance from the source is asymptotically equal to zero. As we increase the size of the spheres (and therefore the spaces between the spheres), we might think that we would have a gradually increasing chance of finding water transported from the source. This counter-intuitively turns out not to be the case. In fact there is a characteristic radius, at which a phase transition happens. When the spheres exceed this characteristic radius, there is suddenly a sharply increasing probability of finding water throughout the system.   Before the spheres reach the characteristic radius, practically no water can propagate through the system. This characteristic radius is known as a critical point, a point at which a phase transition in the behavior of the system occurs [HAR77].

A similar analysis of self-organizing multi-hop peer-to-peer networks reveals that unless we can make some clever innovations in how we place nodes and construct the local rules of communication between nodes, there will always be a characteristic network size at which total network bandwidth starts to sharply decrease. The network experiences a catastrophic phase transition when we try to scale it above a characteristic number or density of nodes [POO94]. This is because most proposals for this kind of multi-hop network have generally maintained that messages propagate locally. In other words, a message examines the nodes in its vicinity, and propagates to nodes that are not busy passing other messages. If this assumption is retained, there is generally a critical point in the density of messages, the density of nodes, and the total surface area to volume ratio of the network, when a message more often than not finds itself with nowhere to go. When the majority of messages more often than not collide with an adjacent message, there is a kind of chain reaction whereby the network becomes saturated with collisions, and messages can no longer get through to their destinations.

The assumption of local message propagation would seem essential, because we would like to use short range transceivers for their low cost low, low power consumption, ease of deployment and maintenance, and spatial multiplexing ability. Local message propagation, however, restricts a node to only having information about its neighboring nodes. How can we reconcile the restriction of local message propagation with the need for long range knowledge of the flow patterns in the network, so that the network can order its resources more efficiently? One possibility is to take a lesson from biology. Ants, for example, can only communicate with one another locally by using scent, however they are able to establish surprisingly straight lines of travel over distances which are much greater in scale than the size of the ants themselves.   Feynman was curious about this ability so he made some simple observations and concluded that the ants performed a “scent averaging” operation to make straight lines. For example, when a first ant finds a source of food by chance, it will then wander back to the nest by a quite circuitous route. The next ant to venture toward the food source will follow the fading scent path imperfectly, often missing the more abrupt twists and turns and simply following the general gradient of scent. This second ant is simultaneously laying its own scent trail which is more direct than the first one. After many of these operations, the distance of the path to the food source is minimized [FEY]. The ants essentially use the diffusion equation to establish minimum distance paths. Using this example, we might propose a way for packets traversing a self-organizing wireless peer-to-peer multi-hop network to find a central base station. They could spawn “scent” packets behind them as they travel which diffuse slowly across the local area to aid the next packet in finding the destination by a more direct route.

We still might worry about what would happen if two well established message paths or “ant highways” need to cross in the network. This would seem to be an inevitable problem if the network is roughly two-dimensional. We will need a way for two message paths to pass “over” one another. This means we need to increase the dimensionality of the network topology. A simple way to do this is for the two paths to share the channel by using orthogonal DS SS codes. Increasing the dimensionality of a percolation problem in this way, can drastically change the critical behavior. In this case, it solves our problem of catastrophic network failure. Therefore, for multi-hop peer-to-peer networks to work, our short range transceivers will require channel sharing capabilities.

3.3 Technologies for Short Range Transceiver Hardware

3.3.1 Infrared (IR) Transceivers

The advantage of *infrared (IR)* transceiver technology is that it is ultra low power and small in size due to the simplicity of the hardware. The carrier frequency modulation, amplification, and antenna in an infrared transmitter is simply an LED. The high Q antenna, low noise amplification, and demodulation in an infrared receiver is simply a phototransistor with a band-selective plastic filter cover. The disadvantages of IR are that it is short range (meter scale), directional, and susceptible to interference from ambient light sources such as halogen or fluorescent bulbs or sunlight.

Infrared Data Association (IrDA) is the current protocol standard for high speed infrared communication. It is primarily deployed in Personal Computers (PC’s) and Personal Digital Assistants (PDA), but the Motorola Company and others plan product lines for wireless LAN’s that will have higher power, omnidirectional transmit LED groups. This kind of system would benefit greatly from the same SS channel sharing that benefits radio transceivers. I did some preliminary explorations into “AFSR for SS IrDA.”

3.3.1.1 AFSR for SS IrDA

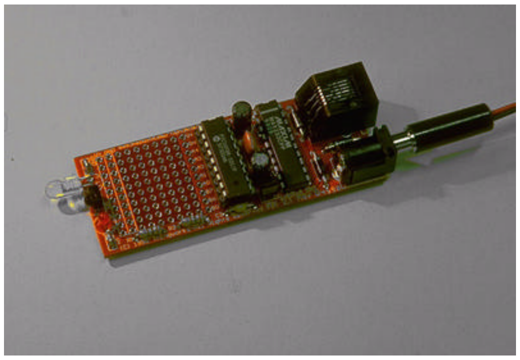


FIGURE 3.3.1 Poor’s IRx Board Used for Infrared AFSR Experimentation [POO99]

I used one of Rob Poor’s IRx boards, shown above in figure 3.3.1, to transmit a PN sequence over an IR transmit channel. I also breadboarded a receiver system that was intended to do AFSR demodulation in software. This system clarified some important issues for implementing AFSR. At first one would think the current software implementation of AFSR (described in Section on page 50) at 10’s or 100’s Kbps would immediately scale for use in IR communication, but the situation is not quite so simple.

In order to maximize transmit range, IrDA needs to maximize its transmit power. To avoid burning out the transmit LED, however, this high power output must have a short duration, so IrDA employs a very low duty cycle. Data is encoded in the relative phase of short bursts of IR light. These bursts are typically one to two orders of magnitude shorter than the delay between bursts.

AFSR like other SS acquisition strategies relies on the SS modulation to be orthogonal to the data modulation so that the data bits do not disturb the acquisition process. For example, in the current conception of AFSR and in many present day commercial SS systems the data bits would be Phase Shift Keyed (PSK) onto the rf carrier, while the PN sequence would be Amplitude Modulated onto the carrier. If we want IrDA to have reasonably good data rates and ranges, then we are restricted to maximum power, short IR bursts. Therefore, we cannot encode any information in amplitude and are left with only phase in which to encode information. If both the PN sequence and data bits are modulated onto burst phase, the data bits will interfere with AFSR acquisition of the PN sequence. This makes it difficult to see how AFSR could be applied to IrDA.

One possible avenue of exploration might be to use *Pulse Code Modulation (PCM) as a modulation scheme which could act as approximately orthogonal to PSK.* PCM uses the number of successive pulses to encode data. If IrDA hardware could be designed to sustain two bursts in quick succession, then data could be modulated as single or double bursts (PCM), while the PN sequence could be PSK encoded.

3.3.2 The TouchTag Reader, a Near-Field Electromagnetic Transceiver

3.3.2.2 Prior Art

Low frequency near-field short range transceivers were actually used in the first pager systems in the 1950's and early 1960's. A loop antenna was wrapped around the building, and would inductively couple at audio frequencies to a coil in the portable pagers carried by the occupants of the building [FEH95]. In the physics and media group at the MIT Media Lab, we have been experimenting with electrostatic near-field coupling both for tomography as well as the transmission and reception of digital data [ZIM95].

Zimmerman realized that signals on the order of 100KHz can propagate across the human body without creating far-field radiation. At 100KHz, the wavelength is

$$\lambda = \frac{c}{f} = \frac{3 \cdot 10^8 \frac{m}{s}}{10^5 \frac{1}{s}} \approx 10^3 m = 1km \qquad (3.3.1)$$

The human body is therefore far too small to be an effective quarter wavelength antenna for these frequencies. By Binary Phase Shift Modulating (BPSK) a 100KHz signal, Zimmerman built the first prototype Personal Area Network (P.A.N.) transceivers for intra-body communication [ZIM96]. I have used this same principle of 100KHz electrostatic coupling to build the TouchTag reader, based on a prototype built by Nivi for his S.M. thesis [NIV97].

3.3.2.3 The TouchTag System (Patent Pending)

.

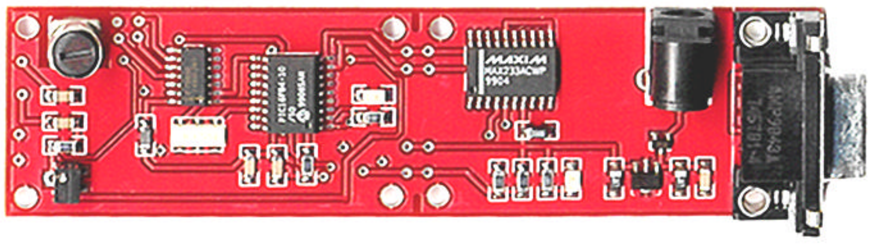


FIGURE 3.3.2 Touch Tag Reader Board, Top View.

The TouchTag reader reads and writes to V4050 and V4100 RFID tag chips which are available from EM Microelectronics Company. They cost a few tens of cents (US) and contain a thousand bits of nonvolatile memory. They also contain a power harvesting circuit which is usually attached to a loop of wire or printed conductor which allows the tag chip to obtain its operating power from surrounding magnetic fields and communicate wirelessly with a nearby tag reader.

A tag chip does not require a battery. It can parasitically power itself when it receives a 125KHz AC signal across its two terminals. Usually, tags communicate with a reader magnetically, inductively coupling via printed coils, to minimize the influence of screening materials. Instead, the TouchTag reader powers and communicates with the tag chip through the human body by coupling to the tag electrostatically. In addition to enabling new applications for RFID tag chips, this is also advantageous for cost because it replaces coils with cheaper flat foil electrodes.

Once the tag is activated, it loads and unloads the AC signal presented across its terminals in order to communicate to the reader. The TouchTag reader uses a “loading mode measurement” on its transmit channel to observe the tag’s pattern of loading and unloading.

3.3.2.4 TouchTag Reader for Near Field Communication

The transmit RFID tag chip in a TouchTag system can be emulated by a simple micro controller like a PIC which can send arbitrary data to the TouchTag reader. This allows the TouchTag reader board to be used to implement a unidirectional Personal Area Network (PAN). For example, Metcalfe for his S.M. thesis at the Media Lab is incorporating the TouchTag reader into a jacket based wearable computer system. A PIC micro controller, emulating an RFID chip, is embedded in a hand-held Global Positioning System (GPS) unit. The GPS unit needs to send data to the jacket computer. When someone holds the GPS unit against their jacket or slides it into their pocket, the PIC inside the it loads/unloads the carrier that the TouchTag reader is emitting across the conductive fabric of the jacket. The TouchTag reader demodulates the loading pattern to receive the bits that the GPS unit is sending to the jacket. Because the TouchTag reader is both emitting the carrier and doing the demodulation, and the “transmitter” is simply loading the carrier, we can do synchronous demodulation, without requiring a carrier acquisition circuit. This works better and is cheaper than previous PAN implementations.

When integrating the TouchTag reader and Post's tauFish capacitive proximity detector into a single architectural system, we found that the tauFish is extremely sensitive to the TouchTag carrier. Future work on PAN should be able to exploit this effect to make better near-field short range transceivers. Such a system could likely have greater bandwidth than IR based systems with comparable hardware simplicity. These signals can propagate through opaque materials. Also, such a system wouldn’t have narrow directionality that restricts the use of IR. This is a promising direction for research.

3.3.2.5 Details of TouchTag Reader Circuit

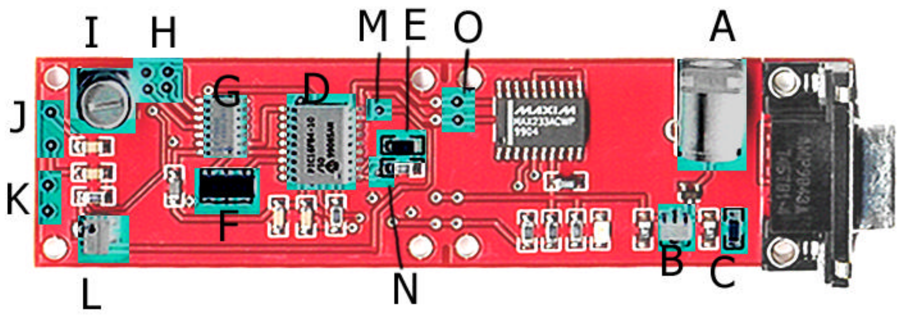


FIGURE 3.3.3 Important Components of the TouchTag Board. (Top)

A schematic of the TouchTag reader is provided in Appendix 1a, Section 12.1 on page 78. The TouchTag board was designed with low cost and therefore low complexity as the primary concern. The demodulation block of the circuit must be able measure an amplitude modulation which is essentially 1-5% of the total carrier magnitude. This is a fairly demanding task for a board that should cost eventually (US) $1 in production.

To power the board, a 9 volt power supply which can supply at least 500mA, is connected to the jack labeled A in figure 3.3.3. The voltage regulator, labeled B, converts this to 5 volts. The capacitors next to the voltage regulator are bypass capacitors to shunt higher frequency noise on the 5 volt power rail to ground. If 5 volts is being successfully supplied, then the red power LED, labeled C, should light.

The PIC microcomputer, labeled D, is the central element in the circuit. It has an 8MHz external oscillator, labeled F. If the PIC has been programmed successfully, and the red power LED indicates that the board has power, then an 8MHz sinusoidal signal should be observed on pins 15 and 16 of the PIC. There is also a yellow debug LED, labeled E, connected to physical pin 2 of the PIC (called PIN_A3 in PIC code). If the PIC has been successfully programmed with code that outputs a 'high' to PIN_A3, then the yellow debug LED should light.

The 74HC383 dual 4 bit counter/divider, labeled G, is connected so as to divide the 8MHz clock. It outputs a square wave signal at 125KHz on pin 10. (A convenient ground access is provided at pads labeled K.) The trace from pin 10 terminates at the lower right pad in a group of 4 pads arranged in a square labeled H, above. For the TouchTag board to transmit, a jumper wire must be connected from the pad connected to pin 10 of the counter/divider to the pad in the upper left of the group of four which is connected to the variable inductor transmit coil labeled I. Before the jumper wire is connected, a 125KHz square wave should be observed on pin 10 of the counter/divider. Once the jumper is connected, the square wave will be deformed by the inductance of the transmit antenna.

The transmit coil, labeled I, must be adjusted so that the resonance of the transmit oscillator is tuned to 125KHz. This will provide the maximum transmit voltage on the transmit electrode which should be connected to the free pads labeled K. To adjust the transmit coil, attach an oscilloscope ground to the ground pads labeled J and oscilloscope signal input to the pads labeled K. A sinusoidal signal should be observed. A small flat head screw driver can be used to turn the black ferrite core of the transmit coil until the output signal is approximately 75-80 volts peak-to-peak. The board is now transmitting. A transmit electrode can be connected to the free pads, K. If this is done properly, the 80 volt sinusoidal signal should be observed on the transmit electrode.

If one lead of an RFID tag is placed on the transmit electrode and the other lead is connected to ground through the body (modeled as a 100pF capacitor and 1K resistor in series) [NIV97], then the tag should start amplitude modulating the transmitted carrier. With some care, this modulation can be observed as a 1-5 volt, 2 or 4KHz (depending on the tag), square wave variation in the amplitude of the 80 volt 125KHz carrier at the transmit electrode.

The demodulation block of the board has two adjusts. It is quite possible that the board is already successfully demodulating the signal from the transmit electrode. There is a pad connected to pin 1 of the PIC (PIN_A2 in PIC code) labeled N above. If the data from the tag is being successfully demodulated, then clean 0-5V bits should be observed at this test point. If these bits are not observed, then it will be necessary to adjust the demodulator.

First the variable resister (pot), labeled L, should be adjusted. This 100K pot controls the voltage divider that reduces the 80 volt peak to peak signal on the transmit electrode to a 5V peak-to-peak signal that can be demodulated. This voltage divider is made up of the pot, R3, and pins 5,6,7 of the quad operational amplifier located on the underside of the TouchTag board labeled P in figure 3.3.4 below. The 100K pot cannot be adjusted too much. It can be adjusted through its full range perhaps 20-50 times before its useful life is expended. To properly adjust this pot one should observe a 0-5 volt 125KHz sinusoidal signal modulated with 100-500mV, 2KHz data bits at the free pad labeled N above, which is connected to pin 1 of the PIC. This measurement should be made while a person is touching the transmit electrode. A ceramic screwdriver can be useful for this adjustment as a metal screwdriver will often cause the signal to be incorrectly loaded by providing a ground return through the screw driver and the adjuster’s body.

Now that the voltage divider into the demodulation block has been correctly adjusted, the diode peak detector should be successfully AM demodulating the data from the tag. This demodulator consists of R102, C8 and the associated diode. R5 provides some low pass filtering behavior for the peak detector. The output of the diode demodulator enters the follower made up of pins 8, 9, 10 of the quad operational amplifier, labeled P, below. This demodulated signal can be observed on the output (side with only one pin) of the diode labeled R, below. The demodulated signal should appear as a 100-500mV square wave, centered at approximately 2.5 volts. This is test point “demod” on the schematic in Appendix 1a which is Section 12.1 on page 78.

If clean bits are still not observed on the pad labeled N connected to pin 1 of the PIC, then it may be necessary to adjust the pot on the underside of the TouchTag board labeled Q below. This 10K pot controls the threshold level on the comparator circuit that converts the raw 100-500mV signal output from the diode demodulation block into clean 0-5 volt bits for the PIC.

The essential idea of the comparator circuit is to fix the DC offset of the raw demodulated output of the diode demodulation block at exactly 2.5 volts, by AC coupling it to a DC 2.5 volt source. Then the 10K pot, R12, labeled Q below, serves as a fine adjust for the threshold level of the comparator around 2.5 volts. The raw demodulated signal is AC coupled by C9. It is coupled to 2.5 volts instead of ground, forcing it to have a DC offset of 2.5 volts. A clean and stable 2.5 volt reference for the AC coupling is provided by bypass capacitors C11 and C12, and the voltage divider made up of R8, R9, R10, and R11. R7 and C10 form a low pass filter with a 3dB cutoff at 10Hz, so that the 2.5 volt node between R8 and R9 sees only the average value of the raw demodulated signal. This is important, because we are only interested in setting the average DC offset of the raw demodulated signal to 2.5V. We don’t want the raw demodulated signal to be quickly forced to 2.5 volts so that it is completely flattened. We want to preserve the bits, but change the average DC offset. As stated above, R12 provides a ten turn fine adjust on this voltage divider so that the 2.5 volt threshold for the comparator can be finely adjusted. This variable threshold voltage is kept free of noise by bypass capacitors C13 and C14. The comparator circuit has a moderate amount of hysteresis for resilience to noise set by R6.

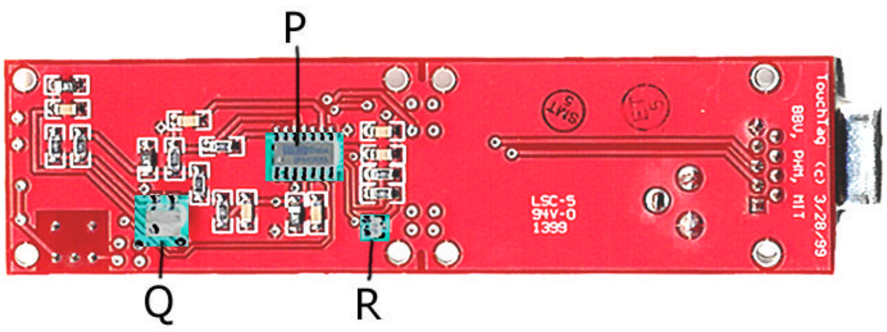


FIGURE 3.3.4 Important Components of TouchTag Board (Bottom)

3.3.3 Configuration of the TouchTag Reader for Various Applications

As my implementation of the TouchTag reader progressed, I realized that I could use it in several new configurations to make different applications possible. TouchTags can be used as a core technology for a variety of physical user interfaces that use human touch to send and receive data. Some of the potential configurations and their applications are listed below:

	Transmit “tag” embedded in portable object	Transmit “tag” worn by person	Transmit “tag” embedded in architectural feature(s)
Reader embedded in portable object(s)	**Configuration D**. Electrostatic near-field communication between objects when a person touches both at once or when they are within ~1 “electrode radius” of one another.”	**Configuration A**. Enables knowing who is touching the system. Data transfer from human to appliance, furniture, or architecture via touch. (Prior work by Nivi on “BodyTags”)	Not clear that there is a way for this configuration to provide proper signal and ground paths for the reader and tag system.
Reader worn by person	**Configuration B**. smart spaces that know who is touching what. (Tagged object must be in contact with a surface which provides electrical ground.)	Person to Person data transfer (PAN)	**Configuration B**. smart spaces that know who is touching what. (Tagged object must be in contact with a surface which provides electrical ground.)
Reader electrode embedded in architectural feature(s)	**Configuration C.** smart space or surface that know what is being touched. (Person provides ground return to floor)	**Configuration A**. Enables knowing who is touching the system. Data transfer from human to appliance, furniture, or architecture via touch. (Prior work by Nivi on “BodyTags”)	**Configuration E**. Electrostatic coupled RFID Tag reader. Packages that talk to the truck or warehouse they are in? (ground return provided without person)

TABLE 3.3.4 Complete Configuration Space for TouchTag Hardware

3.3.4.1 Configuration A

Reader in Architectural Feature or Portable Object. Transmit Tag Worn by Person

The TouchTag reader was originally intended as a system for coupling to RFID tags worn by a person. The idea was that a person could wear a *BodyTag, an RFID tag chip in the insole of their shoe,* so that when they touched a surface or object with an embedded electrode connected to a TouchTag reader, they would be identified to the reader. If the reader were in a laptop or desk top computer, the person could simply touch the keyboard to be authenticated and log on to the system. In general this configuration allows the reader system to know who is touching it. If transmit "tag" in this system is emulated by a micro controller, the send arbitrary data to the object or architectural feature that contains the TouchTag reader.

3.3.4.2 Configuration B

Reader Worn by Person as Wearable Computer. Tags Embedded in Portable Objects or Architectural Space.

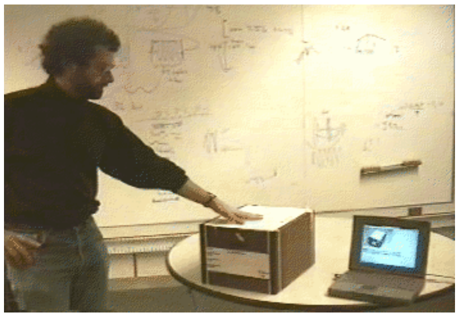


FIGURE 3.3.5 Retrieving information about a package by touching it. The box contains an electrostatically coupled RFID tag. Professor Gershenfeld is wearing a TouchTag reader in his shoe which reads the tag and outputs this information to the laptop. The laptop displays information about the box.

TouchTags can be useful when used in conjunction with a "wearable" computer. When a human wears a computer on their body, we would often like the computer to be able gather information about physical objects in the environment as the human interacts with them. The TouchTag reader we have created, like the tags, is small and cheap. Smaller than a credit card and less than US $20 in parts, it is well suited for use in a wearable system. We have integrated our TouchTag reader into a shoe so that it can be worn by a postal worker. We connect an RFID tag chip between a pair of foil electrodes inside a package. The package must be placed in proximity to a capacitive ground return such as the floor or furniture with a conductive frame. When the user wearing the reader's hand comes into the proximity of the package electrodes by touching the surface of the package, a small displacement current couples through the human body, then through the TouchTag package, and finally down to ground. This powers the tag and causes it to be read by the wearable reader. The end result is that when the worker touches a package, their wearable computer now knows to provide tracking information about that package. This configuration allows the system to answer the question, "who is touching what?" By changing the mode of the reader, the user can also write new information into the TouchTag package by touching it.

This configuration could be useful for checking inventory in factory, warehouse, or retail environment. It could also be used for job training or education. In this scenario, the student would get a "help bubble" in their wearable display about each object they touch. For example at the Boeing airplane factory, a TouchTag system could help guide a person to use a tool correctly or to install a part properly. We can also enable smart spaces, so that a wearable computer can know which objects are being handled by the user. If the wearable is linked to the network, then the room can be aware of people's touch interactions with objects or one another. If the TouchTags were actually embedded in features of an architectural space, a person wearing a TouchTag reader could gather information about the room they are in or the door they are opening.

3.3.4.3 Configuration C

Reader attached to electrode in/on surface of furniture or architectural feature, and tag in a physical icon or "Phicon."

One of the central ideas in modern windowed computer interfaces is the action of clicking on an icon. Recently Ishii et. al. have embedded tag chips within physical objects on a desk and a tag reader in the surface of a desk, to make "physical icons [ULL97]." The TouchTag reader is novel because it only reads a tag when it is touched by the human user, so unlike in previous inductively coupled schemes, a large number of other tagged objects can be in proximity to the reader. And because of the large change in the ground return coupling provided by the body, the relative size of the reader's electrode to the tag can be significantly greater than is possible with the fill-fraction constraint on magnetic tags.

In this configuration, the reader knows what is being touched. TouchTags preserve the intuitive familiarity of pointing to an icon while liberating the icons from the confines of the computer screen. One could have a large stack of *physical icons or phicons* which correspond to actions or records in the computer. These actions or records are only accessed when the user touches a single phicon.

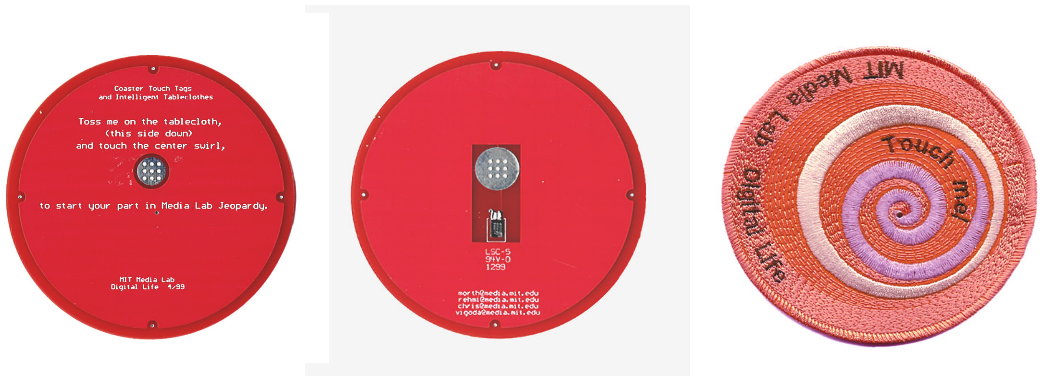

Coaster Touch Tags and Intelligent Tablecloths
Toss me on the tablecloth, (this side down) and touch the center swirl,
to start your part in Media Lab Jeopardy.
MIT Media Lab Digital Life 4/99
LSC-5 9HV-0 1299
north@media.mit.edu
rehmi@media.mit.edu
chris@media.mit.edu
vigoda@media.mit.edu
MIT Media Lab Digital Life
Touch me!

FIGURE 3.3.6 TouchTags used as tabletop login tokens for "Tablecloth Jeopardy Game" TTT Spring 1999. Views from left to right: Bottom, Top (with Swatch RFID tag chip), and "Ebroiderey [POS97]" Top Cover.

These low cost, touch activated, wireless TouchTag phicons are useful for applications such as quickly sorting a large list of information records by preference into several smaller "piles" of records. Imagine a scenario in which your search engine finds 159 red 1998 Suburu station wagons in your price range. Automated filters have done all they can. The final decision is a matter of taste and constraint satisfaction which would take too much time to program the computer to do. Instead, each car record is associated with a TouchTag phicon on your desk. You can perform a kind of physical breadth first search. First you quickly sort the pile into smaller piles of first pick, second pick and so forth. Then you can easily return to the small pile of favorites and sort them further. It makes human sorting of a large list of digital records easier and more efficient [VIG99]. I have been actively collaborating with researchers at SteelCase Company to produce this kind of system as a product.

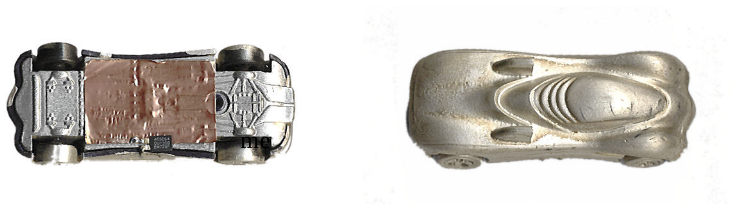


FIGURE 3.3.7 A TouchTag enabled toy car with an RFID tag chip. It could still function if coated with a nonconducting outer surface.

If we put TouchTags in toys, and the reader in a play surface it can provide information about which toy the child is touching. This could be used in interactive play spaces where the computer participates in the play.

3.3.4.4 Configuration D

Reader in Architectural Feature and Tags in Portable Objects

This configuration could enable a truck or warehouse to interrogate packages stored between ceiling and floor, or between two walls. The system could also be set up to interrogate packages as they proceed down a conveyor belt. With our help, this configuration is being actively produced and marketed by the Motorola company.

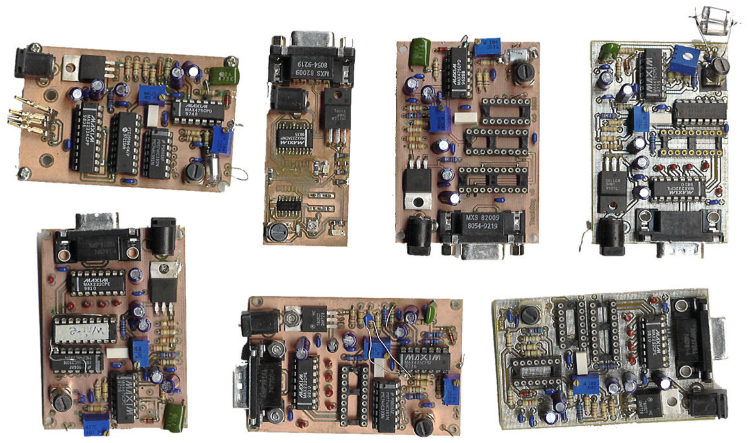


FIGURE 3.3.8 Early Prototypes of the TouchTag Reader Board

3.3.5 Applications and Ethics

“We shall never experience our relationship to the essence of technology so long as we merely conceive and push forward the technological, put up with it, or evade it. Everywhere we remain unfree and chained to technology, whether we passionately affirm or deny it. But we are delivered over to it in the worst possible way when we regard it as something neutral; for this conception of it, to which today we particularly like to do homage, makes us utterly blind to the essence of technology.” - Martin Heidegger from “The Question Concerning Technology”

Table 3.3.4, “Complete Configuration Space for TouchTag Hardware” seemed at first a useful pedagogical conveyance. Constructing this table, however, led me to discover new configurations for the TouchTag reader. This table was therefore not only a useful way to explain the possible configurations of Touch Tags, but also a useful way to discover them. Indeed any equation or logical argument can be a way to make progress in scientific discovery, but I pick on this table precisely because it regards applications or uses of a technology and not only scientific or technological advancement. The application of logical argument to applications or uses of technology is ethically questionable, because it implicitly recommends that all uses that are possible must become actual. By attempting to put in order all possible manifestations of nature in this special case of TouchTag configurations, I have not been content to satisfy my needs as they occur to me. Instead I am envisaging all possible uses of a technology simply because they are logically possible. Heidegger uses the word “enframing” (Gestell) to denote this particularly regimented way of relating to technology. [HEI52]

Heidegger maintains that modern physics, “indeed already as pure theory, sets nature up to exhibit itself as a coherence of forces calculable in advance, it therefore orders its experiments precisely for the purpose of asking whether or how nature reports itself when set up in this way[1].” The creation of a theory of physics is in many ways a creative and beautiful discipline of the mind, a practice which can promote a love and respect for nature as well as a connection to a river of ideas which is greater than any single person. However, the application of the methods of modern physics to the question of the uses of technology, which is to say, to the question of human society is ethically questionable. If I am to use a table like the 3.3.4, then I feel I must state explicitly that it is not a recommendation for uses of this technology, but only "a revealing" of its possible configurations. It is up to us to use our free will to decide whether a particular use of the technology will lead to our betterment. At first, "betterment" may indeed simply mean to us economic profit. But the stronger and more developed our sense of free will becomes, the more prepared we will be to truly understand what is to our true betterment. [HEI52]

1. N. B. Heidegger, in his language, always attempts to chart a middle course through the mind/body problem. He realizes that things occur in the world which seem to be out of human control, but which would not happen without humans. For example, wars. Heidegger is not superstitious. Which is to say he would never intend to impart a true human will to natural occurrences or collective phenomenon. But on the other hand, he does not wish to disrespect the power of events to, as we might say, "have a will of their own." He tries to write in a way which respects the co-will of humans and their environment. This can be frustrating reading for a physicist who is used to understanding our relationship to matter as one of the ideal human mind relating to matter as lawful mechanism. But to the engineer who has ever actually built something and who has therefore confronted the will of matter- Murphy's Law that "anything that can go wrong, will go wrong," Heidegger might not seem so strange.

4 Radios and Modulation

4.1 Introduction to Modulation/Demodulation

Modulation in a transmitter means using data to alter a “carrier” signal. A carrier signal is nothing more than a pattern, usually a repeating pattern. Demodulation in the receiver then involves comparing the carrier altered by the data to a pure carrier pattern. The differences between the unaltered pattern and the pattern altered by the data tell the receiver what data was sent. The main difficulty is in producing a synchronized copy of the pure unaltered pattern in the receiver for comparison to the “modulated” carrier.

But why not just send a data signal? Historically, it was found that we needed a carrier because of the physics of antennas. Modulating a data signal onto a carrier makes it possible to alter the spectrum of the data signal for transmission by radio. In radio transmission it is often desirable to have the transmitted signal to closely surround a single frequency so that the signal will radiate efficiently from a resonant antenna. The antenna is basically a filter in the channel, and modulation can help match the spectrum of a data signal to the filter inherent in a channel.

Modulation is also useful if we want to send multiple data signals through one channel. We can do this by modulating each data signal onto a different carrier. Then it is possible to send multiple data streams through the “ether”, and have them be separable at the receive end by comparing the summed signal to locally generated carriers in the receiver. Another way to send multiple data signals through a single channel would be to have an algorithm for interleaving the data signals into one higher speed signal, and then send this signal on a single carrier, but this won’t work if we want to have multiple independent transmitters and receivers.

Some common modulation schemes are

AM Amplitude Modulation

FM Frequency Modulation

BPSK Binary Phase Shift Keying

QPSK Quadrature Phase Shift Keying

MSK really should be called BFSK Minimum Shift Keying also called Binary Frequency Shift Keying

GMSK – just filtering strategy right? Gaussian Minimum Shift Keying

Feher PSK - FPGA signal generation techniques

4.2 Spread Spectrum

In order to understand the chief advantages of DS SS, it is enough to note the important fact that the frequency spectrum of white noise is approximately flat. A familiar example of white noise is white light, which incorporates an equal mix of every frequency of visible light. Since the information from an SS transmitter looks like gaussian white noise, it also has a flat frequency spectrum within some frequency band. One chief advantage of having the SS signal spread across a band of frequencies is that multiple transmitters and receivers can share this band without having to explicitly negotiate who gets which frequency. SS signals interfere very little with one another allowing many transmitters and receivers to communicate simultaneously on the same frequency band. This is known as *Code Division Multiplexing (CDMA)*. CDMA is the clear choice for applications in which there are many (possibly moving) transceivers communicating simultaneously.

4.2.1 Review of Spread Spectrum

The following information was taken primarily from the Spread Spectrum Communications Handbook

4.2.1.1 Definitions

(from PART 4 Synchronization of Spread Spectrum Systems chapter 1)

judge acquisition strategy by *mean acquisition time*, TACQ, acquisition time variance (2ACQ

also acquisition probability but this harder

coherent means someone have phase lock on rf carrier and are demodulating coherently BPSK for example

non-coherent means we don't need phase lock, because we aren't demodulating before we do acquisition

single dwell time- always integrate over fixed interval of correlation to decide whether or not acquisition has been achieved

multiple dwell time- integrate over short correlation window, if acquisition looks promising increase window size until prove acquisition or not conclusively. - increased circuit complexity but reduced acquisition time

for multiple dwell time system:

fixed or variable integrations time

fixed integration time are either single or multiple dwell

time non-limited serial search strategies take as long as needed to accomplish acquisition-typically used when data modulation is always present

time-limited serial search used when data modulation only commences after acquisition, and the allotted time allows for a high probability of acquisition

returning state is when acquisition is reported to the tracking circuits but it is really a false alarm. If the system is robust the tracker will realize the problem and return control to the acquisition circuits so that they can start searching where they left off.

absorbing false alarm is when in the above situation there is a catastrophic loss of the code synchronization and so there is no returning state

4.2.1.2 Survey of Pseudo Noise Acquisition in DS SS Receivers

from section 1.1

Maximum Likelihood: correlate received signal with every possible internal LFSR sequence and take one that gives highest value

RASE: Rapid Acquisition by Sequential Estimation:p759

load register of receiver with incoming demodulated estimated bits, and try to generate next bits in sequence. Use correlator to see if match is good. If match falls below threshold reload the receiver LFSR.

Advantage: Significantly faster than Sequential Search.

Drawback: 1) Doesn't work with SNR's worse than -15dB. 2) Needs to be coherent detector so can demodulated all incoming PSK bits for correlation.

RARASE: Recursive-Aided RASE p759

Correlates more bits than RASE to decide whether to reload?

7.5 times faster than RASE for 15 bit LFSR.

Sequential Search (also called stepping correlator):
bits come in
internal LFSR generates bits through delay with knob on it.
vary knob linearly and watch correlator
faster knob turning means faster acquisition with worse correlator accuracy

Advantage: (p761 bottom)
chip generators getting faster->
broader spreading->
wider band receiver input filter->
more receiver noise->
worse SNR's for acquisition schemes to deal with->
bad for RASE since doesn't work under -15dB->
therefore serial search more & more common
plus serial search is easy to implement
so that is what we use now

1.2 *Single Dwell Serial PN Acquisition System*

figures 1.6 p764 and 1.7 p 766:
mix incoming with local PN code->
bandpass filter->
square law envelope detector->
integrator->
compare to present threshold.

knob on sequential search usually steps internal PN sequence in half chip increments until acquisition is achieved.

cells are number of knob increments to search entire code- in above case twice the length of the PN sequence

1.3 *The Multiple Dwell Serial PN Acquisition System*

Conceptual Implementation:
An integrator that integrates over a shorter time finishes first and tells whether it thinks there is acquisition or not. If it says yes there is acquisition, then we listen to the next longest integrator which has integrated over the same interval of the shorter time integrator and then some. It finishes its integration after the shorter time integrator, but gives more information. If at any time, an integrator says no lock then, we stop listening to longer and longer integrators. Instead we reset all the integrators, and shift the internal PN code 1 cell.
Actual Implementation:
1 single continuous time integrator whose output is sequentially sampled (but not dumped) at successive time instants.

1.4 A Unified Approach to Serial Search Acquisition with Fixed Dwell Times
boring unless we really care about analyzing these systems

4.2.1.3 A Historical Survey of PN tracking in DS SS Receivers

(from chapter 2)

early late gates - PN correlated with delayed and advanced internal versions of itself to generate timing error correction signal.

2.1 DLL (from Part 4 section 2.1 p. 902)

DLL Delay Locked Loop - PN correlated with advanced and delayed signals which goes through loop filter to voltage controlled clock (VCC) that drives PN generator. When the advance and retard are one half a chip this is called a “one delta loop.”

2.2 TDL

TDL Tau Dither Loop (Time shared loop) - very similar to DLL except alternate between doing correlation with advanced or retarded signal. The output of the correlator goes to the loop filter, is envelope detected and then inverted if it came from the retarded signal before it goes to the VCC that drives the PN generator. The reason for this is so we don't have to match two analog filters and correlators perfectly. Still used now in DSP systems?

2.3 Acquisition (transient) behavior of DLL and TDL

1. What is maximum allowable relative code rate offset (due to doppler, or code instabilities) between received and locally generated PN codes?
2. How long until acquisition?

Nonlinear dynamics starting to happen on p928

2.4 *Mean Time to Loss of Lock*

use some cool Fokker Plank stuff.

2.5 *Double Dither Loop (DDL)*

switch two correlators and loop filters of a DLL so they alternate on the retarded and advanced signals. So balancing of correlators is less crucial.

2.6 *Product of sum and difference DLL*

some math to improve on DLL in a different way than DDL

2.7 *MCTL Modified Code Tracking Loop*

analysis follows that of DLL almost exactly.

2.8 *The Complex Sums Loop* (A Phase Sensing DLL)

Seldom used it is essentially a Sum-Delta DLL for a PN phase modulated onto a carrier.

5.1 *Multiple Access*

good idea of doing CDMA with each new transmitter in the network coming in with its PN code delayed by an exact amount from the last transmitter to join. This way only need to acquire one code, all others are just delayed versions, but still get processing gain! This is done in FH/MFSK, but not reported for CDMA. But to do this have to solve the clock distribution problem.

4.2.1.4 Standard Practice

The CDMA Digital Cellular System - *IS-95* is the last generation industry standard for cell phone communication. It employs QPSK with a data rate of 9.6 Kbps, a 1.25 MHz chip rate, and a 900 MHz carrier. Data is encoded using convolutional coding with Viterbi decoding.

4.2.1.5 SS and microcells

Spread spectrum is good in a microcell situation, because rf falls off as r^4 on surface of earth. That means that with the added protection from SS, neighboring cells can reuse 100% of spectrum so greater channel capacity over multiple cells than narrowband cell network. (p1171) This is why DS CDMA has more channel capacity in a microcell situation. In a microcell context, the capacity of spread spectrum radios is interference-limited, while the capacity of narrowband radios is dimension limited.

4.2.2 Linear Feedback Shift Registers

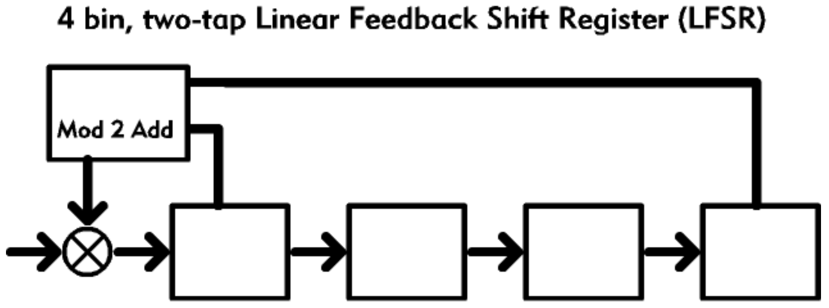


FIGURE 4.2.1 Block DIagram of Linear Feedback Shift Register (LFSR)

Linear Feedback Shift Registers (LFSR) are used in spread spectrum to generate sequences of pseudo random bits. An LFSR is can be thought of as a kind of Fibonacci series generator. For example, a 4 bit, 2 tap LFSR implements the recursion relation,

$$x(t) = [x(t-\tau) \oplus x(t-4\tau)]\mathrm{mod}2$$

where t = time and x = signal.

The physical system can be thought of as a “series of registers through which bits are shifted, with taps from a few specified register bins that select the values to be added mod 2, as illustrated above.

TABLE 4.2.3 Maximal Length LFSR Tap Sequences

18 bins would be a typical register length for an LFSR in an actual SS system [MOT97].

Gold codes are created by taking the sum (mod 2) of two shifted maximal length LFSR of different taps configurations but the same size which have the drawback of not being maximal length sequences, but have the advantage of greater mutual orthogonality than standard single LFSR m-sequences.

4.3 Radios in the future

4.3.1 Higher Frequencies

frequency (in Hertz)	**wireless communications technology**
10^{12} - 10^{14}	infrared systems and line-of-site laser transmission
10^{11} s	“pseudo optical” regime
10^{10}	satellites, other high performance microwave systems
10^{9}	commercial microwave: cell phones, wireless LAN’s, pagers...
10^{8}	FM radio...
10^{7}	Television....
10^{6}	AM radio...
10^{5}	Near field electrostatic or inductive coupling since ¼ wavelength far field radiating antenna would need to be enormous. Applications in LAN, PAN, paging

TABLE 4.3.2 Wireless Technologies and Frequencies Covered

5 Prior Art for AFSR

5.1 Chaotic Entrainment for Spread Spectrum Acquisition

The Analog Feedback Shift Register is not the only nonlinear dynamics system that has been proposed for use in noise channel coding applications. Kevin Cuomo, Allan Oppenheim, and Steven Isabelle proposed a scheme for generating and entraining to deterministic noise using nonlinear dynamic system in a report entitled “Spread Spectrum Modulation and Signal Masking Using Synchronized Chaotic Systems [CUO92].” They proposed to use a low dimensional chaotic system to generate noise carrier, and a copy of the same system to entrain to the noise for demodulation in the receiver. As they state themselves, “chaotic dynamic systems are nonlinear deterministic systems which often exhibit erratic and irregular behavior. The signals that evolve in these systems are typically broadband, noise-like and similar in many respects to a stochastic process. Because of these properties chaotic signals potentially provide an important class of signals which can be utilized in various communications, radar and sonar contexts for masking information-bearing waveforms and as modulating waveforms in spread spectrum systems [CUO92].”

The first problem they have is that “...chaotic systems are characterized by 'sensitive dependence on initial conditions' and have at least one positive Lyapunov exponent. Thus long term behavior of chaotic systems is difficult at best since small uncertainties in the initial state of the system will be exponentially amplified... These properties of a nonlinear system would seem to defy synchronization, which, if true, would produce unattractive models for signal processing or communication applications.

To solve this problem and successfully construct the system that can synchronize they must make use of the fact that “chaotic systems which can be decomposed into a drive system and a stable response subsystem will synchronize if they are coupled with a common drive system.” For example it has been “... demonstrated that the chaotic Lorenz and Rossler systems can be decomposed into a drive system and a stable response subsystem which will synchronize when coupled with a common drive signal.” The general theory for constructing these decompositions is found in “Synchronizing Chaotic Circuits,” by T. L. Carroll and L. M. Pecora [CAR91].

The first example they present is the familiar Lorenz system given by

$$\begin{aligned} \dot{x} &= \sigma(y-x) \\ \dot{y} &= rx - y - xz \,. \\ \dot{z} &= xy - bz \end{aligned} \tag{5.1.1}$$

The above equation is used as the drive system with $\sigma = 16$, $b = 4$, and r = 45.92. The stable response system uses duplicates of the states (x, z) called (x', z') so that it is written

$$\begin{aligned} \dot{x}' &= \sigma(y-x') \\ \dot{z}' &= x'y - bz' \end{aligned} \,. \tag{5.1.2}$$

The y(t) signal “serves the purpose of a driving or coupling signal between the drive and response systems. The analytic condition for synchronization is negative real eigenvalues of the Jacobian matrix of the response subsystem which correspond to the conditional Lyapunov exponents. This means that “we should expect $|x - x'|$ and $|z - z'|$ to approach zero exponentially fast.” They go on to show that synchronization does in fact occur in simulation.

Having successfully simulated the synchronization of several continuous time systems and one discrete time system, the authors then “...propose and explore in a preliminary way how synchronized systems can be used for spread spectrum communication and for various signal masking purposes.” They provide block diagrams of hypothetical SS systems employing a chaotic systems. In their scheme, a drive system forces a response subsystem which is filtered by a whitening filter producing a noise-like signal which can be used in place of the transmit LFSR in a conventional DS SS transmitter. The drive system also sends a signal across a separate drive signal channel to the receiver. A duplicate response subsystem at the receiver is to synchronize to this drive signal producing identical noise for demodulation in place of a synchronized receiver LFSR found in a conventional DS SS receiver.

In order separate the drive and spread data channels, they propose to amplitude modulate the spreading signal and phase modulate the drive signal. They amplitude modulate the data (which has a much lower bandwidth) onto the chaotic spreading signal. This scheme is clearly inferior to the AFSR scheme we discuss later, because the data is amplitude modulated while the AFSR system can phase modulate the data which is much more robust to disturbance by noise. If they try phase modulate their data as the AFSR system does, then their drive signal cannot also be phase modulated or synchronization will be disturbed. Their drive signal would therefore have to be amplitude modulated. This would imply that their spreading signal would now have to be phase modulated, thereby destroying the ability of the spreading signal to significantly spread the original carrier. To reiterate, AFSR is superior to the scheme proposed here because AFSR can synchronize to the actual PN sequence and does not require a separate synchronization channel. This leaves room for the data to be encoded in the signal in its own orthogonal modulation scheme. In fact, one of the primary advantages of AFSR above all present SS acquisition schemes is that it does not require a separate synchronization channel and the attendant transceiver complexity that this entails.

There is a second drawback of the scheme proposed here which is more serious. The chaotic systems here are being used to generate white gaussian noise. The fact of the matter is, however as the authors note themselves, the “signal energy is not uniform over a wide range of frequencies.” Chaotic systems do not actually produce perfectly gaussian noise (the spectrum is not really flat) nor truly white noise (there are correlations in the signal). The authors attempt to address this problem by employing a linear time invariant (LTI) Wiener whitening filter to flatten the spectrum and remove correlations. This whitening filter since it is LTI cannot remove all of the long range and structure due to the dynamics. Moreover, a whitening filter could be used on the a simple sinusoidal signal to produce a flat spectrum, but this would also spread all of the thermal noise of this single frequency over the spectrum. The original goal was to produce a deterministic signal that had a flat spectrum. The necessity of the whitening runs counter to this spirit.

To abstract the lesson learned here, we note that positive Lyapunov exponents give rise to the noise like properties of a system (as was stated by the authors on page 1) while negative exponents give rise to the dissipative synchronization properties of the system (as the authors have stated on top of page 5). But the noise-like properties provided by positive Lyapunov exponents are not terrific (as they say on page 22 in their section on whitening filters). The solution to this problem is to still use the negative Lyapunov exponents for dissipative synchronization, but to simultaneously produce the noise by a deterministic process. This is what AFSR does.

5.2 The Beginnings of AFSR

5.2.1 DT Cosine AFSR

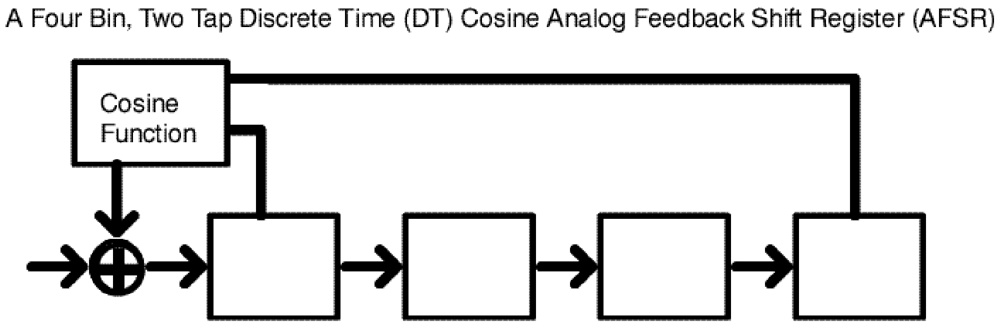

A Four Bin, Two Tap Discrete Time (DT) Cosine Analog Feedback Shift Register (AFSR)
Cosine Function

FIGURE 5.2.1 Schematic Representation of a Four Bin, Two Tap Discrete Time (DT) Cosine Function Analog Feedback Shift Register (AFSR)

AFSR was first proposed by Gershenfeld and Grinstein in a 1995 paper entitled, “Entrainment and Communication with Dissipative Pseudorandom Dynamics [GER95].” The structure of their original AFSR can be most easily understood as an analog LFSR used as the Voltage Controlled Oscillator (VCO) in a Phase Locked Loop (PLL). The general equation for an LFSR as we have seen before is

$$x_n = \sum_{i=1}^{N} a_i x_{n-i} (\mathrm{mod}2) \tag{5.2.1}$$

where $a_i = 1$ sets a tap, of a given delay, n, and $a_i = 0$ sets no tap, and where $a_N = 1$ always from the theory of LFSR’s. In other words this is a delay line of length N, with taps at delays for which $a_i = 1$. The outputs from he taps are summed mod 2 and then shifted into the register. In this system all the x_n always take on discrete values of either 0 or 1.

To create AFSR, the mod 2 function in equation 5.2.1 is modified. “To make an analog version capable of entrainment, we replace the mod2 function by a continuous function that is equal to it for integer arguments, has a slope of magnitude less than one in the vicinity of these integer values, and necessarily then has unstable fixed points between the integer values. This makes the maximal sequence of the LFSR a stable attractor [GER95].” They proposed the difference equation

$$= \frac{1}{2}\left[1 - \cos\left(\pi \sum_{i=1}^{N} a_i x_{n-i}\right)\right] \tag{5.2.2}$$

where ai’s are selected just as in the LFSR above. The mod 2 function of an LFSR and the Cosine function of the original AFSR are shown in Figure 5.2.2 on page 43. Clearly, for x = 0 or 1, the cosine function reduces to mod 2. If we initialize the AFSR with 0 and 1 values in the register bins, its evolution will be identical to that of an LFSR with the same length register, tap configuration, and initial condition. If we were to initialize the AFSR register with random values between 0 and 1, x = [0,1], however, something interesting happens. The values in the AFSR bins will migrate outward toward values of either 0 or 1 until the AFSR is once again following the evolution of the corresponding LFSR. In fact, as the analog values are migrating toward 0 and 1, the AFSR will be simultaneously attempting to follow the evolution of the corresponding LFSR but with degenerate register values. A brief digression to explain the behavior of a simple “quadratic iterated map” or quadratic difference equation will make it clear how this happens.

A quadratic function $y = x^2$ is shown plotted below on the interval $x > 0$. Starting from some initial condition x_0, we can construct a nonlinear iterated map from this function, by following an algorithm whereby we first obtain $y = x_0^2$, and then re-input this y value into the function as x, so that $x_1 = y = x_0^2$, and then repeat. We can write this algorithm more compactly as

$$\begin{aligned} y(t) &= x^2(t) \\ x(t+1) &= y(t) \end{aligned} . \qquad (5.2.3)$$

In Section 2, “Nonlinear Dynamics,” on page 13, we discussed that one of the first questions we are ordinarily interested in about a nonlinear map or flow is which initial conditions will blow up and which will settle to an equilibrium or fixed point. In the case of this “quadratic map,” we can observe two fixed points at x = 0 and x =1. The first is a stable fixed point or sink and any initial condition $x < 1$ will eventually settle to it. On the other hand x=1 is an unstable fixed point or source and any initial condition will move away from it. Therefore, initial conditions $x > 1$ will blow up to ∞ after repeated iteration by the map. If x is exactly 1 (not possible in a physical system due to inevitable noise), x will remain 1 under continued iteration since $1^2 = 1$.

Once we determine the fixed points of a function, it is possible to find if they are stable or unstable mathematically. If the derivative of the function at the fixed point is less than 1 (as for x=0 in the quadratic map), then the fixed point is stable. On the other hand, if the derivative of the function is greater than or equal to one at the fixed point, then the fixed point is unstable.

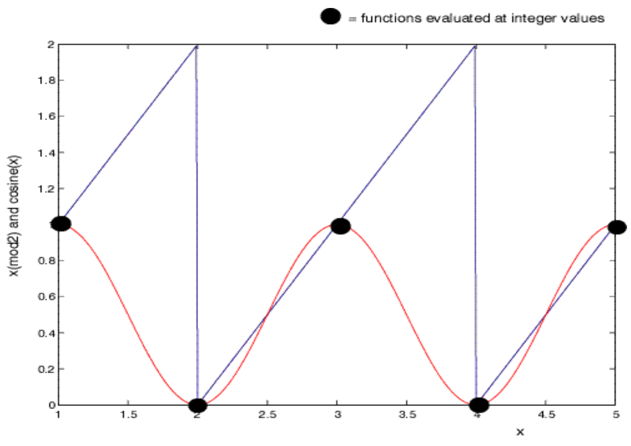


FIGURE 5.2.2 Continuous Valued x (mod 2) and cosine of x on the interval x=[1,5]

Having seen this example, we are now prepared to understand the dynamics of the AFSR iterated nonlinear map. As seen in Figure 5.2.2 on page 43, the cosine function is identical to the mod 2 function for integer values of x. The integer values will be fixed points of the iterated cosine map, and they will be stable since the derivative of the cosine function is zero at x an integer. Furthermore, there will be unstable fixed points at the half integer points where the derivative of the cosine function is greater than 1, so that any initial x under repeated iteration of the cosine map will evolve toward either 0 or 1. However, as we stated before, once we have evolved out to these fixed point extremum, the cosine map is equivalent to the mod 2 map. So if we build an LFSR with a cosine function in place of the mod 2 function, and initialize it to random register values, it will quickly evolve to have binary register values. As stated in the original paper, “an LFSR has an attracting basin” which is the limit cycle of the corresponding LFSR, so that “starting from arbitrary initial conditions that lie in this basin, an AFSR will produce, in the long time limit, ideal pseudorandom noise governed by the well-developed theory of LFSR design [GER95].” We now understand the behavior of the original DT Cosine Map AFSR running as an autonomous system.

Now we must ask, “what happens to this system if it is given an LFSR forcing function?” In other words what happens if it operates non-autonomously? Having answered this question, we will completely understand the DT Cosine AFSR. Miraculously, if we send an output chip at each clock cycle from an LFSR into a corresponding AFSR with randomly initialized register values, the AFSR not only produces a PN sequence identical to the forcing LFSR, but does so in phase with the forcing LFSR. The AFSR values will be nudged to coincide with the LFSR’s transmitted PN sequence. The AFSR can be said to “entrain” to the LFSR. Many plots of this behavior are shown in the following sections. Because the AFSR contains a structure that mirrors the LFSR it is entraining to, in theory it should be able to reproduce the PN sequence of the LFSR even if the LFSR sequence is weak, noisy or completely interrupted. The AFSR, in a sense, exhibits momentum.

The cleverness in creating this mathematical system continues to amaze me the more I learn about it and work to improve it. Unfortunately, however, as we will see later, the DT Cosine AFSR does not perform well with any noise in the LFSR channel, unless we use a trick that called “selective feedback” in the original paper. We will not discuss selective feedback here since by using other nonlinear function than the Cosine function in AFSR, I have been able to discard selective feedback and still improve the performance of AFSR beyond the original system. With a different nonlinearity, AFSR will exhibit robust entraining behavior even when the variance of additive gaussian noise in the LFSR channel is greater than the peak-to-peak value of the incoming LFSR signal without selective feedback. This is the topic of the section on “Analysis and Modeling of CS DT AFSR” on page 45.

5.2.2 Early Attempt at CT Cosine AFSR

In the original AFSR paper, the authors also presented a continuous time (CT) version of AFSR to show that it would be possible to build a physical implementation AFSR. They present the autonomous CT differential equation

$$\frac{dx}{dt} = \varepsilon_1(x - x^3) + A\theta(z(t) - z_c)\cos\left(\pi\frac{1 + x\left(t - \frac{1}{2}\right)}{2}\right)\left[1 - \cos\left(\pi\sum_{i=2}^{N} a_i\frac{1 + x\left[t - \frac{(2i-1)}{2}\right]}{2}\right)\right] \tag{5.2.4}$$

where $\varepsilon_1 > 0$, $z_c < 1$, and A are all positive parameters, and $z(t) = \cos(2\pi t)$ is a forcing function with unit period. “The first term drives $x(t)$ toward fixed points at ±1 with a speed governed by ε_1. Since $\theta(z(t) - z_c) = 1$ for $z > z_c$ (and equals 0 for $z < z_c$), choosing z_c just slightly smaller than 1 makes the second term apply kicks that produce the transitions.”

The important term to note here is $\theta(z(t) - z_c)$. This term produces very low duty cycle square pulses that cause the transitions in the AFSR to happen. There are two major problems with this term. First, if the AFSR is run in non-autonomous mode to entrain to an incoming LFSR signal, an external clock recovery is assumed in the term, $z(t) = \cos(2\pi t)$. This clock recovery would have to track the incoming LFSR signal clock before AFSR starts to do its acquisition work. This is known as a coherent detector, and is quite difficult to build since it may require locking to the clock rate of a single PN sequence which for many SS signals could be is buried 30dB below the noise. Although it is possible that this clock recovery could be successfully accomplished using something like a Phase Locked Loop (PLL), we would much prefer to eventually build the clock recovery into the dynamics of AFSR. If the AFSR is already a kind of PLL, it seems suggestive that we may be able to roll the clock recovery into the AFSR dynamics. We will take up this subject again later with some success.

6 Analysis and Modeling of CS DT AFSR

6.1 Entrainment

Entrainment is a familiar phenomenon in coupled oscillators. The classic story about entrainment is that a visitor to a swiss clock store is amazed to find that every clock in the store is ticking in perfect synchrony. How can the old clock maker possibly pay such close attention to hundreds of clocks in order to maintain them to such a high degree of accuracy? The answer is that the clocks are weakly coupled to one another by the sound which travels through the walls, floor and air of the room. This weak coupling, over time, causes all of the clocks to tick in synchrony. Below is a Figure showing a Runge-Kutta simulation of a damped oscillator being weakly forced by a sinusoidal forcing function. The forcing function has the same frequency as the resonant frequency of the oscillator, but starts out of phase to the oscillator. The red line is the state of the damped oscillator which starts in the 1 state at time = 0. The green line represents the forcing function which starts in state 0 at time = 0. As can be seen, the damped oscillator quickly adjusts its phase to match that of the forcing function.

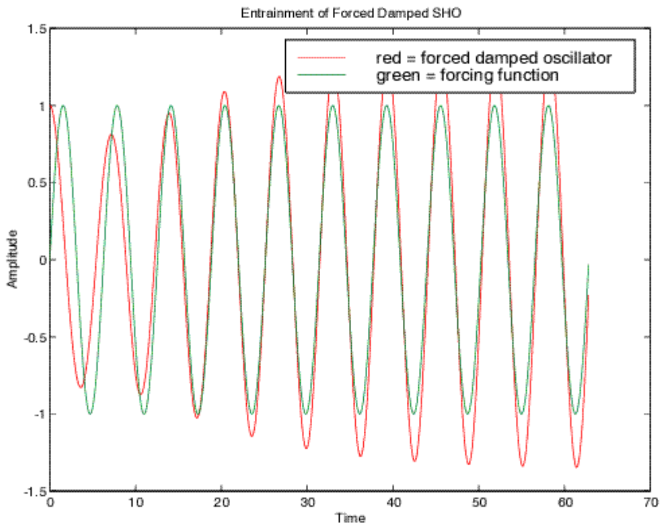


FIGURE 6.1.1 Entrainment of Simple Harmonic Oscillator (SHO)

This behavior can be understood most generally in the language of the state space of the system. The total state space volume of a Hamiltonian system (one with no damping) will stay constant for all time. The topology of the state space may change, but its volume must stay constant, because its energy is conserved. A damped system, however, seeks to lower its energy by removing energy to the environment. It will seek to minimize phase space volume. When two coupled oscillators are out of phase they each require an independent degree of freedom in state space in order to describe their behavior. When they come into phase, however, the dimensionality of the total system is reduced by one, so that the state space volume of the system is smaller. Therefore coupled oscillators with damping (dissipation) will come into phase with one another in order to seek a minimum energy state space configuration.

Similar behavior is exhibited in the simulation of DT AFSR below. The green line is the signal from an LFSR being transmitted through a channel with additive gaussian white noise. The red line is the state of the DT AFSR. Even with very weak coupling between the LFSR and AFSR (the AFSR update is 90% based on its past internal state and only 10% on the incoming LFSR signal (epsilon = .1)), the AFSR still achieves synchronization or acquisition quite quickly. Once it achieves acquisition it exhibits a rectified version of the noisy LFSR forcing function which is actually identical to the original LFSR signal before noise was added.

]

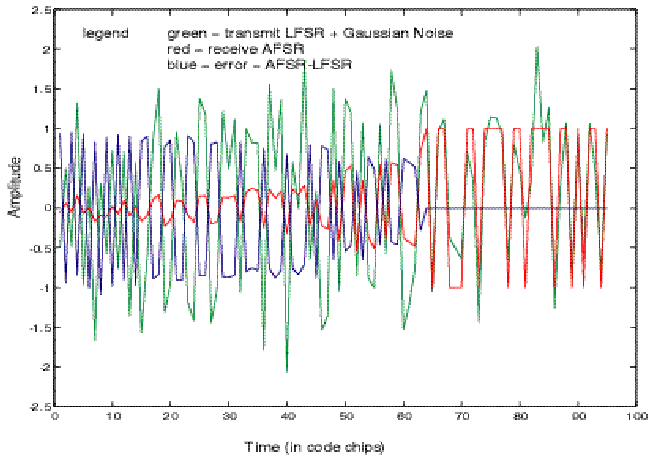


FIGURE 6.1.2 Entrainment of AFSR Weakly Coupled to LFSR with Additive Gaussian Noise (noise variance =.5, register length = 15 bins, epsilon =.1)

6.2 A More Elegant AFSR

6.2.1 Choosing the Nonlinearity for Simpler Implementation and Better Performance

Two taps is sufficient to create a maximal length LFSR for most register lengths. “Linear Feedback Shift Registers” on page 38 contains a complete table of maximal length tap configurations for LFSR’s of increasing register length. For a two tap LFSR, we can represent the cosine function shown in Figure 5.2.2 on page 43 in a new way. In the figures below, the Cosine function for a two tap, DT Cosine AFSR is plotted in three dimensions. The x and y axes are the values from the two taps of the shift register that are input to the function. For a four bin AFSR, for example, the x axis would be the value from the register bin delayed by one chip, and the y axis would be the values from the register bin delayed by 4 chips. The z axis is the output of the cosine function of the two taps as given by equation 5.2.2. The mod 2 map of the equivalent LFSR lies at the vertices of the cube. So (x,y) = (0,0) yields z = 0 since (0+0)mod2 = 0. Similarly (x,y) = (0,1) yields z = 1, (x,y) = (1,0) yields z = 1, and (x,y) = (1,1) yields z = 0. The cosine function smoothly interpolates between these fixed points which are dictated by the mod 2 function.

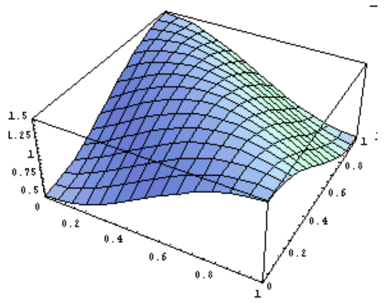


FIGURE 6.2.1 3-D Plot of Original Cosine Map For Two Tap AFSR $x(t+\tau) = \mathrm{Cos}(\pi(x(t-\tau)+x(t-4\tau)))$

How does this system perform? In the chart below, a trial consisted running a 15 bin DT Cosine AFSR for 400 chips or until it performed successful acquisition. Successful acquisition was defined as having produced 2N + 1 error free chips identical to the transmit LSFR, where N is the number of bins in the register. 1000 of these trials were run and a histogram of acquisition times plotted. With gaussian noise on the transmit channel with a variance =.5, the DT Cosine AFSR never successfully performed acquisition within 400 chips. In order to achieve successful acquisition with the Cosine map AFSR it was necessary to remove all noise from the transmit signal. The reason for this is that the map has zero slope everywhere along the line $x = y$ so the map is susceptible to the slightest perturbation due to noise in approaching the (x,y)=(0,0) and (1,1) fixed points. To correct this problem, the map should be adjusted to have a slope along both the $x = y$ and $x = -y$ lines. This is achieved by the absolute value and quadratic map functions I have devised. For all AFSR simulations described here, except when explicitly stated otherwise, epsilon was.6.

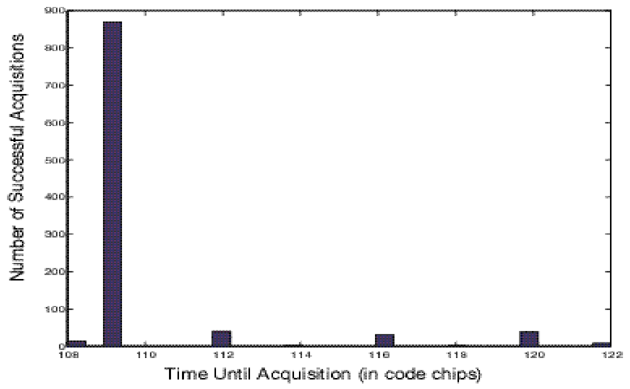


FIGURE 6.2.2 Distribution of Acquisition Times for Cosine Map AFSR

With all noise removed from the channel, the DT Cosine AFSR did perform successful acquisition. The bins in the histogram, Figure 6.2.2, are 1 chip wide. As can be seen from the chart, over 800 out of 1000 trials of the DT Cosine AFSR performed acquisition in 109 chips. Other successful acquisitions were scattered between 108 and 122 chips. The distribution of acquisition times is somewhat erratic. If the DT Cosine AFSR could perform this way in the presence of noise, it would be acceptable performance, however, we would prefer to have a gaussian or other simple and predictable distribution of acquisition times in order to be able make confident predictions about the performance of an AFSR based SS receiver.

I have devised several map functions that improve on the cosine map originally proposed for AFSR. The first is the absolute value map shown below. It should be obvious from the figure, that this map satisfies the extremum imposed by the mod 2 function just as the cosine map did, however it has the advantage of 1) being symmetrical for smoother convergence to the fixed points, 2) having non-zero slopes approaching all of the fixed points, and 3) there being a stronger possibility of being able to implement this function in hardware due to its simplicity.

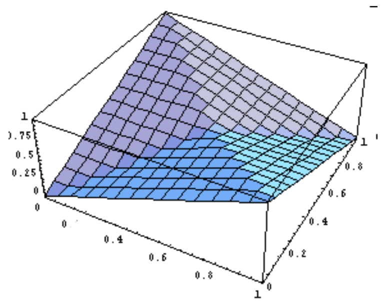


FIGURE 6.2.3 Absolute Value Map $x(t+\tau) = \text{abs}[x(t) - x(t-15\tau)] - \text{abs}[x(t) + x(t-15\tau) - 1]$

An AFSR built with the Absolute Value map is very well behaved, although still not as robust to noise as we might like. Here is a typical acquisition run for a 15 bin DT Absolute Value AFSR with no noise, epsilon=.6, and a gain of.5. The smooth ring up of the AFSR is especially satisfying. The behavior is reminiscent of the familiar ring up of a simple harmonic oscillator.

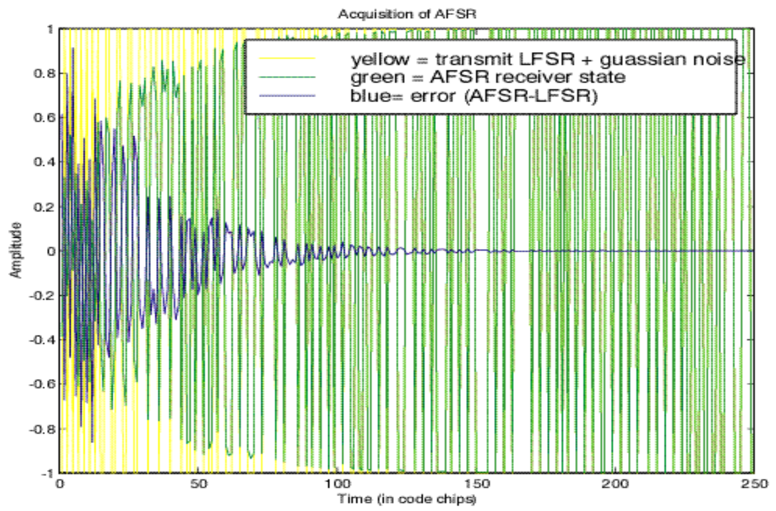


FIGURE 6.2.4 Typical Acquisition Trial for DT Absolute Value AFSR

I ran 1000 acquisition trials like the one shown above except with a noise variance =.5 and gain of 1, and plotted a histogram of the acquisition times. The maximum time allowed for acquisition was 200 chips. If the AFSR failed to acquire successfully in this time, then it returned a -1 for the acquisition time, which accounts for the large peak in the histogram at -1. This peak is comprised of trials that failed to acquire.

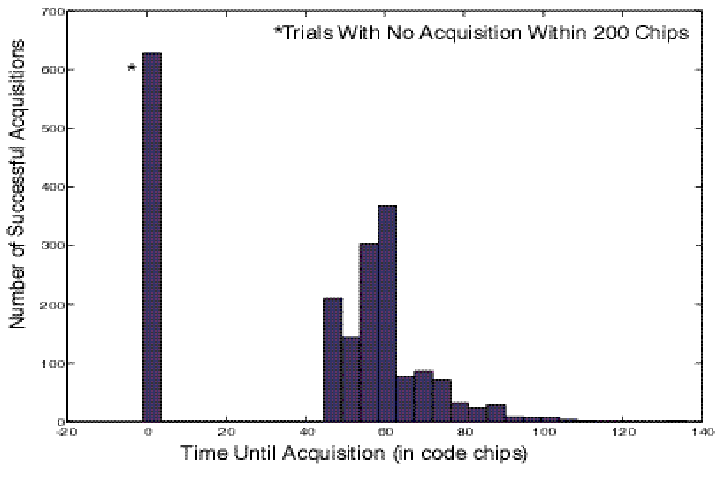


FIGURE 6.2.5 Distribution of Acquisition Times for Absolute Value Map AFSR

As can be seen from Figure 6.2.5, the DT Absolute Value AFSR has a nice distribution of acquisition times. Although I do not have analytical evidence to support this claim, the distribution looks suggestively gaussian. It is at least unimodal and well behaved enough to be able to make strong predictions for the mean and variance of the acquisition time. Also, this behavior was obtained under conditions of significant noise. The peak to peak signal from the

LFSR transmitter had a value of 1 ($\left(\mathrm{eps}\cdot\mathrm{amplitude} = \left(\frac{1}{2}\cdot 2\right) = 1\right)$), while the variance of the gaussian additive noise was.5. On average, the perturbations from the noise were half as large in amplitude as the actual LFSR signal.

The most successful nonlinearity I tried was a square law or "quadratic" function shown below in Figure 6.2.6. Like the other maps, this one satisfies the extremum of the mod 2 map. It has several other advantages as well. The first advantage is that as the map approaches the fixed points, the function grows steeper so that the system accelerates toward acquisition as it draws closer to acquisition. This is desirable, because we would like AFSR to slowly migrate toward the fixed points at first as it explores the phase space trying to acquire the LFSR signal. As the certainty of acquisition rises, however, we would like the AFSR to accelerate toward acquisition. The original AFSR paper proposed to accomplish this by scheduling epsilon, but the quadratic map accomplishes this scheduling by its functional form. This scheduling feature of the functional form also allows us to turn up the gain for the quadratic map. Gain presents a trade-off in previously discussed AFSR maps. Turning up the gain is desirable to robustly push the register states to the fixed points even in the presence of noise. If the gain is too high, however, the AFSR risks not lingering in uncertain values long enough to find the correct phase of the incoming LFSR.   The quadratic map ameliorates this problem by having a shallow in the middle which stays shallow even as the gain is increased. Increases in gain have their main effect at the edges of the map, near the fixed points where the steep slope is most needed.

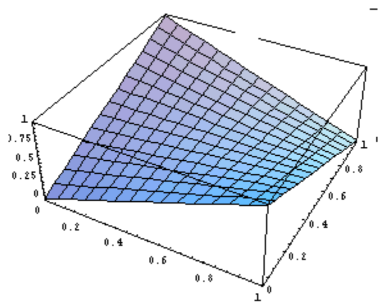


FIGURE 6.2.6 Square Law Map $x(t+\tau) = (x(t)-x(t-4\tau))^2 - (x(t)+x(t-4\tau)-1)^2 + 1$

The second advantage of the quadratic map are the properties of its acquisition time distribution. Below is the distribution of acquisition times for 1000 acquisition trials of a 15 bin DT Quadratic AFSR under exactly the same noise and epsilon conditions as the Absolute Value AFSR trials discussed above. The distribution strongly suggests a gaussian shape which allows us to feel comfortable in making predictions for the mean acquisition time and variance of acquisition times. It has the shortest average acquisition time (approximately 60 for15 bins) and smallest variance in the acquisition time (approximately 10 chips).

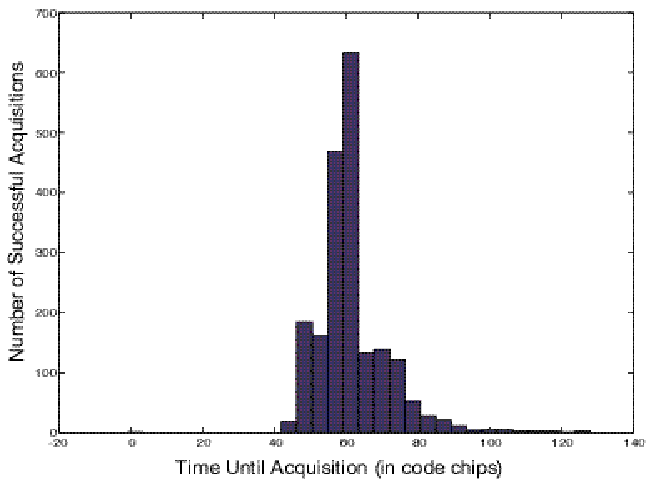


FIGURE 6.2.7 Apparently Gaussian Distribution of Acquisition Times for Quadratic Map AFSR

The third and perhaps most important advantage of the quadratic map is that it immediately suggests a simple implementation with electronic components. Square laws are very common in silicon as the second order approximation of a diode junction's exponential curve. The quadratic function can be computed by a simple double-balanced mixer, which is tremendously helpful in envisioning a compact and inexpensive AFSR system. The advantages of this simple nonlinear function for AFSR are so compelling that the quadratic map will be assumed in all models of AFSR discussed form this point on. An interesting direction for future theoretical work on AFSR will be to understand if it is possible to generalize the quadratic function to an AFSR map with more than two taps. This is a topological question of satisfying the mod 2 boundary conditions with quadratic saddle points in higher dimensions.

6.2.2 Building Fixed Points by Exploiting Gain Clipping

In the previous section, we have delicately avoided a crucial issue. The absolute value and quadratic functions to not have slopes = 0 at the fixed points of (x,y)=(0,0), (1,0),(0,1) and (1,1). Therefore the AFSR register states should continue to blow up to infinity as the dynamics evolve. The actual code kernel to produce the above plots for the DT Absolute Value AFSR was

```
1. afsr(lfsrlen+1) = eps*noisyxmit(t)-(1-eps)*gain*(-abs(afsr(lfsrlen)-afsr(1))+abs(afsr(lfsrlen)+afsr(1)));
2. %%threshold cutoff (rails of amplifier)
3. if  afsr(lfsrlen+1) >= [1]
4.  afsr(lfsrlen+1) = [1];
5. elseif  afsr(lfsrlen+1)<= [-1]
6.  afsr(lfsrlen+1) = [-1];
7. end
```

and similarly the code for kernel for the DT Quadratic AFSR was

```
1. afsr(lfsrlen+1) = eps*noisyxmit(t) - (1-eps)*gain*afsr(lfsrlen)*afsr(1);
2. %%threshold cutoff (rails of amplifier)
3. if  afsr(lfsrlen+1) >= [1]
4.  afsr(lfsrlen+1) = [1];
5. elseif  afsr(lfsrlen+1)<= [-1]
6.  afsr(lfsrlen+1) = [-1];
7. end
```

The innovation here lies in lines 3-7, which simply keep the register values of the AFSR bounded to the interval [-1,1]. Without these bounds the AFSR behavior looks like Figure 6.2.8 below.

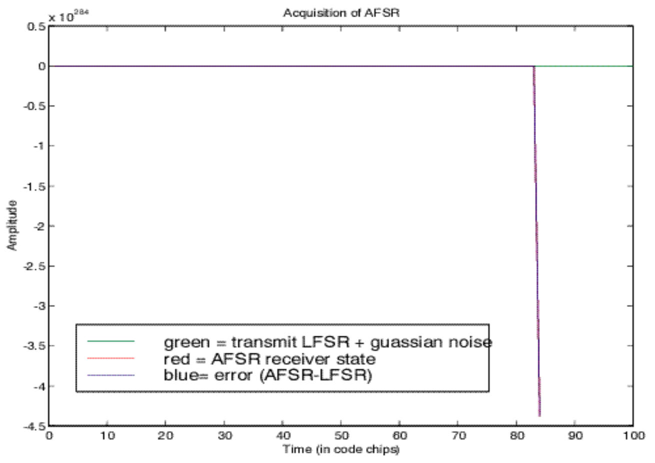


FIGURE 6.2.8 AFSR Without Fixed Points Blows Up to -Infinity.

Without fixed point boundaries the Absolute Value and Quadratic AFSR blow up to ∞ or $-\infty$. With these boundaries imposed by the "hack" in the code above, the quadratic function actually looks like the plot in Figure 6.2.9 below.

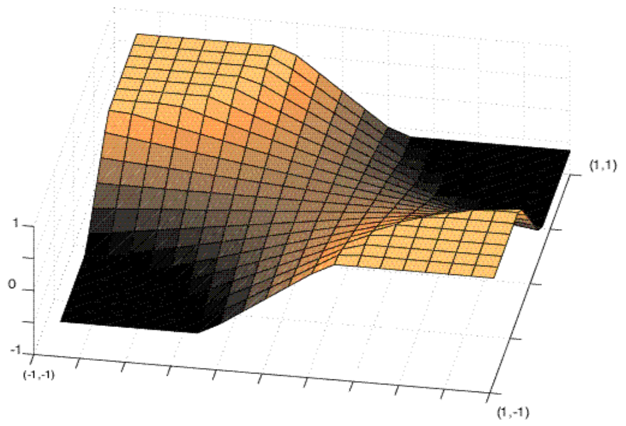


FIGURE 6.2.9 Quadratic Function for AFSR with Gain Clipping Fixed Points at the Extrema

Luckily, this seemingly ad hoc bounding step in the algorithm has a simple physical interpretation; In later sections on the hardware implementation of AFSR we will see that this bounding condition is quite naturally accomplished by the AFSR gain amplifier clipping to its positive and negative rails.   In the language of nonlinear dynamics, we now have stable fixed points at the extrema of the map since the slope beyond the extrema is zero. Of course we already knew that they must be stable since the amplifier won't operate at values beyond its rails!

6.3 Varying Register Length, Epsilon, and Noise

6.3.1 Mean Acquisition Time and Register Length

An important statistic of AFSR performance is the mean time to acquisition for different register lengths. We would like to know if AFSR will perform acquisition for long PN sequences in a reasonable amount of time, since a typical

SS system may employ PN sequences generated by LFSR's with as many as 20 bins. It is important to note that the cross section of the line in Figure 6.3.1 below is the distribution of acquisition times from a chart like Figure 6.2.7 on page 50 above. Therefore, without a unimodal distribution a simple plot like this will not tell us a lot. The Absolute Value and Quadratic AFSR, and to some extent the Cosine AFSR with no noise, do have unimodal distributions of acquisition times though, so we can feel confident in drawing the kind of graph shown below.

.

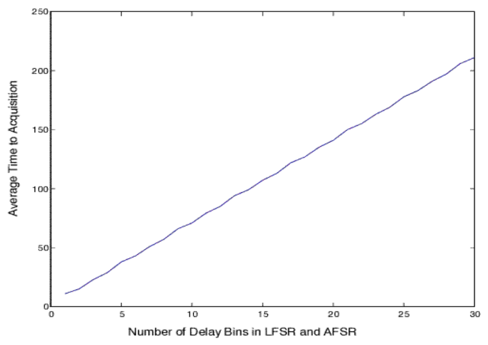


FIGURE 6.3.1 Average Acquisition Time for Cosine Map AFSR vs. LFSR Length (no noise and)

Figures 6.3.1 and 6.3.2 show the average acquisition time of DT Cosine and Quadratic AFSR as register length is increased. I ran 50 acquisition trials for each register length from 2 to 30 with taps on the first and last bins. Some of these LFSR do not produce a maximal length PN sequence, but they still produce a satisfactory PN sequence. Surprisingly, the mean acquisition time scales approximately linearly with the register length. This is good, because it means that AFSR will scale gracefully to perform acquisition for long PN sequences. It is surprising, because one would

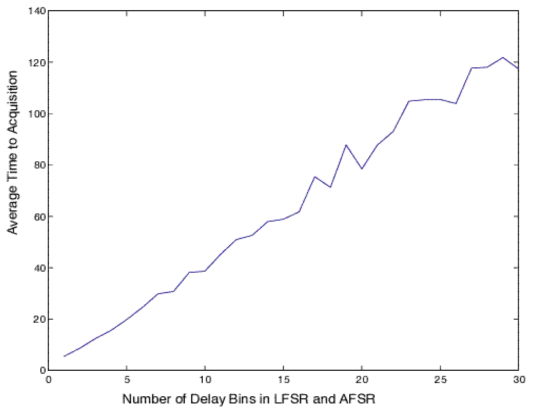


FIGURE 6.3.2 Average Acquisition Time for Quadratic Map AFSR vs. LFSR Length

expect an exponential curve, since as the PN sequences grow exponentially longer, the time spent exploring an exponentially larger configuration space for the proper place in the PN sequence should also increase exponentially. I have verified this graph with epsilon =.1, .3, .6, and with noise variance = 0, .5. This is an issue that requires more careful investigation.

6.3.2 Acquisition Times with Epsilon

Epsilon is the parameter which determines how much the AFSR considers the incoming received (noisy) signal (and disregards its internal state) when updating its state. If epsilon is close to 1, then the AFSR updates almost entirely based on the incoming signal. If epsilon is close to 0, however, the AFSR is nearly autonomous, paying attention mainly to its internal state for update. As is clear from the acquisition time distributions for epsilon=.1 and epsilon .8 shown in the two figures below, Quadratic AFSR maintains a smooth unimodal distribution of acquisition times over a wide range of epsilon values.

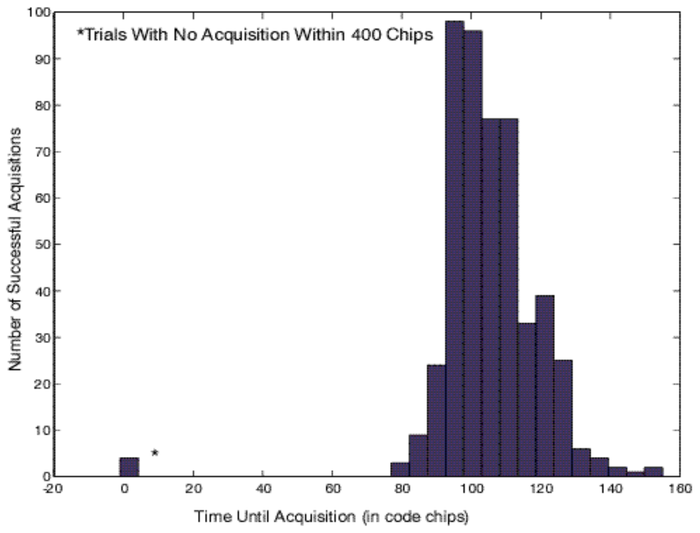


FIGURE 6.3.3 Distribution of Acquisition Times for Small (eps=.1)

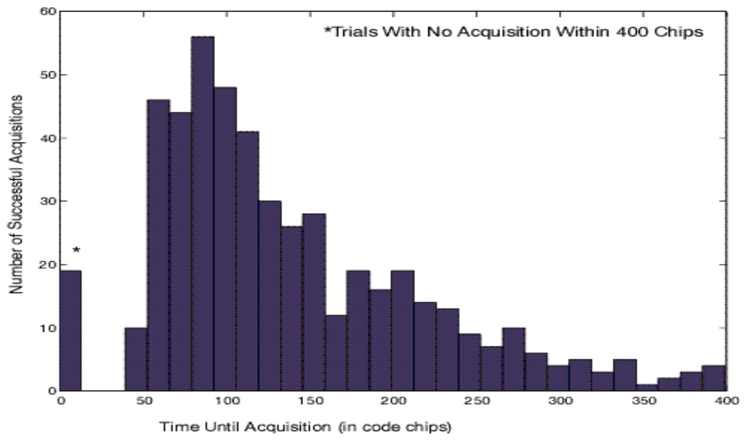


FIGURE 6.3.4 Distribution of Acquisition Times for Large (eps=.8)

We can therefore feel justified in presenting Figure 6.3.5, below which plots mean acquisition time of DT Quadratic AFSR for epsilon values from 0 to 1 with the noise variance = .5. This figure makes it clear that AFSR acquisition time is quite robust across varying values of epsilon. In fact only when epsilon is below.1 or above .8, does performance start to seriously degrade. In the range from epsilon .1 to epsilon .8, although the mean is not changing extremely dramatically (from 100 chips to about 60 chips), the distribution of acquisition time around the mean is changing from the distribution shown in Figure 6.3.3 to the one shown in Figure 6.3.4 above. In other words, for smaller epsilons the distribution of acquisition times is tighter but the mean is larger. For larger epsilons, the distribution of acquisition times has a much longer tail (larger variance), but the overall mean is a bit lower.

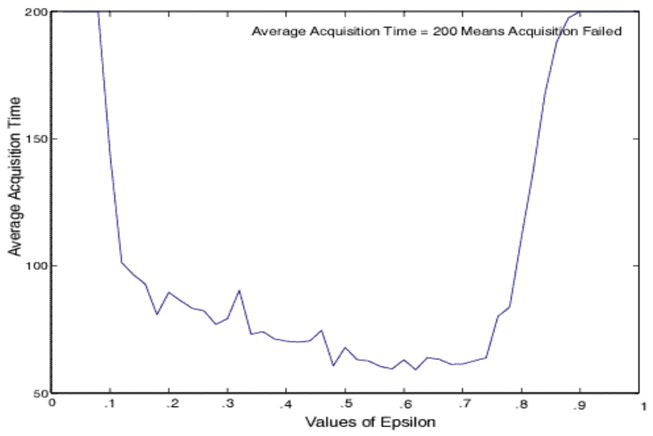


FIGURE 6.3.5 Average Acquisition Time vs. Values of Epsilon, ε

The results in the figure above differ from those in the original AFSR paper, because of the presence of noise. In the original PRL paper of AFSR, there was no noise in the modeling of mean acquisition vs. varying epsilon values. Therefore, a very large epsilon, would essentially feed the values of the LFSR transmitter directly into the AFSR registers without any additive noise resulting in immediate perfect acquisition. In the presence of noise, however, we observe more realistic behavior. It is necessary for the AFSR to listen to its internal state at least a small amount in order to have some sense of system momentum. If epsilon is 1 then the AFSR may as well not have any structure except a buffer, since the dynamics of AFSR are not affecting the update of its state. The test for acquisition success is if the values of the AFSR register are equal to the values of the uncorrupted LFSR register for 2N+1 chips, where N is the number of bins in the LFSR and AFSR. In the presence of noise, an epsilon of 1 means that acquisition never occurs, because LFSR + noise which is being fed directly into the AFSR is never perfectly equal to LFSR without noise that it is being tested against.

6.3.3 Acquisition Times with Noise

Below is a plots that shows the distribution of acquisition times in the presence of additive gaussian white noise of variance 1. In this case epsilon is .6, so the AFSR is successfully acquiring the LFSR signal when the variance of the noise is of equal to magnitude to the received LFSR signal.

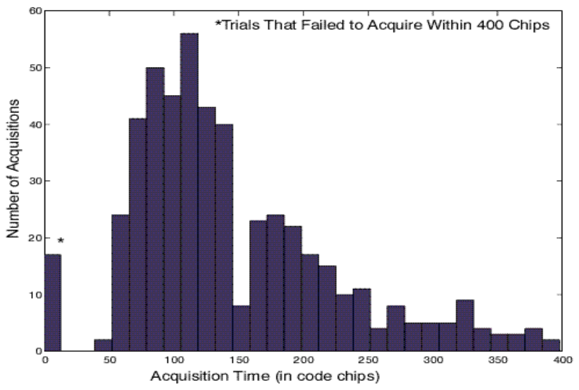
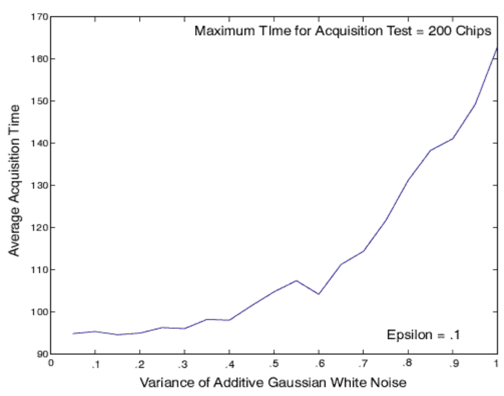


FIGURE 6.3.6 Distribution of Acquisition Times With Extreme Additive Gaussian Noise (noise variance = 1)

Even for this high noise value, the distribution is smooth and unimodal so we can feel safe in simply plotting the mean acquisition time vs. amount of noise. We do this for several different values of epsilon below, in order to see how noise interacts with different values of epsilon.



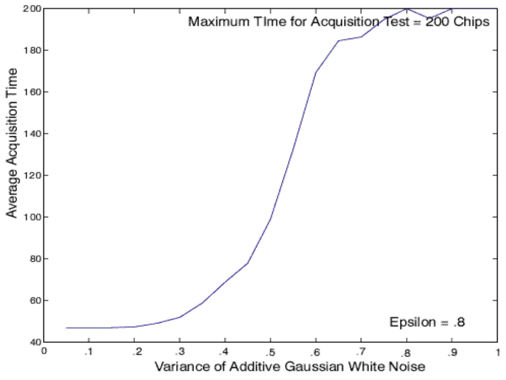


FIGURE 6.3.7 Average Acquisition Time Increases with Increasing Noise (for epsilon= .1 and .8)

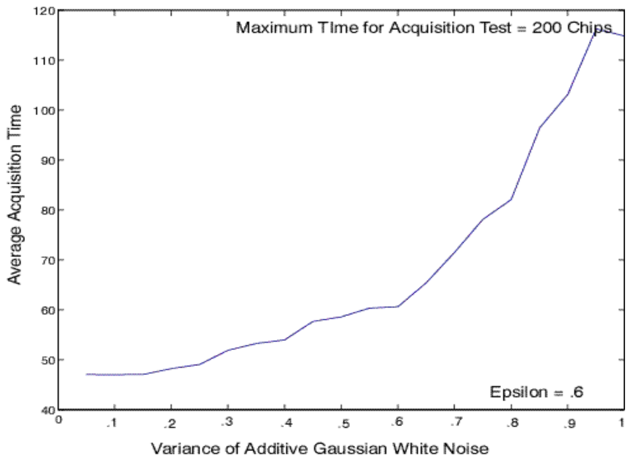


FIGURE 6.3.8 Average Acquisition Time Increases with Increasing Noise (for epsilon = .6)

We also repeat the same plot for DT Absolute Value AFSR with qualitatively very similar same results, except that this AFSR perhaps degrades a bit more sharply as the variance of the noise starts to exceed .5.

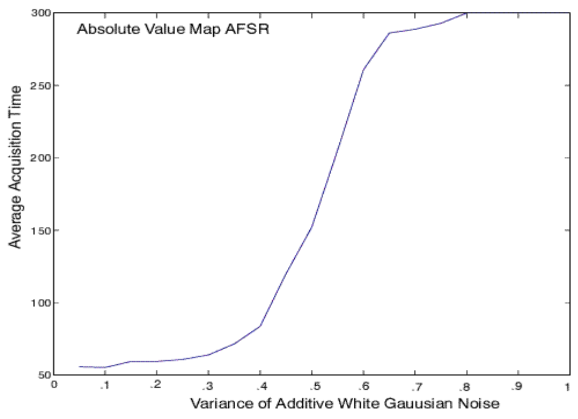


FIGURE 6.3.9 Average Acquisition Time vs. Additive Gaussian Noise for Absolute Value Map AFSR

There is one artifact of the plotting method used in the figures above. If the AFSR does not successfully acquire within 400 chips, then the trial simply returns the maximum acquisition time of 400 chips. This means that as the noise increases, and successful acquisition happens less and less frequently within the allotted 400 chips, the mean acquisition time appears to approach a value of 400. When the noise is so great that no successful acquisition occurs during any trial acquisition run, the mean acquisition time only appears to level off to 400 chips, because that is the maximum acquisition time that the acquisition trial can return. The roll-off at the right hand of Figures 6.3.7, 6.3.8, and 6.3.9 are therefore artifacts and not true phenomenon.

6.4 Understanding AFSR and Time, DT and CT AFSR.

	Discrete Time (DT) AFSR	Continuous Time (CT) AFSR
Discrete State (DS) AFSR	and Software Radio AFSR	KHz AFSR Hardware Prototype requires external clock recovery CR)
Continuous State (CS) AFSR	Initial Modeling	Radio Frequency (rf) AFSR (requires smooth clocking)

TABLE 6.4.1 AFSR Models and Their Applications

So far we have only considered AFSR as an iterative map in discrete time (DT). This means that a clock must control each shift of the shift register and each calculation of the nonlinear function. For reasons that will be discussed in further depth later, we are extremely interested in removing the need for this external clock and in lifting the DT map AFSR as it has been proposed into a continuous time (CT) system. Also, so far we have been considering AFSR with floating point precision analog values which we refer to as continuous state (CS) AFSR. In a DSP or software radio implementation of AFSR, however, these states will not have such high precision. They will mostly likely be represented with 8 or 16 bits of precision. We call this system discrete state (DS) AFSR and will model its behavior. In all, we discuss four permutations of the AFSR system. The AFSR we have so far considered has been CS DT AFSR in this nomenclature. Soon we will look at prototype AFSR hardware that has been constructed. We consider this system DS because of limitations in the precision of some of the digital components used, and we consider it CT, because it follows the design of the continuous time model originally proposed by Gershenfeld and Grinstein, and can be easily modified to be truly CT once we understand how to do this in the last chapter. Next, we consider the possibility of a software radio AFSR which must be both DS and DT. Finally, we take up the problem of actually creating a CS AFSR in the chapter on rf AFSR. The chart above illustrates this naming scheme for each of the different AFSR implementations.

7 DS CT external CR AFSR- Prototype Hardware

7.1 Goals of Hardware Design

I worked toward implementing a low frequency (10's KHz) version of DS CT AFSR using off-the-shelf components in order to gain more understanding of what will be required to implement AFSR as a physical system. I started with the grand goal of creating a fully functional SS receiver employing AFSR to separate multiple received spread spectrum signals modulated by orthogonal LFSR PN sequences. All of the received signals could be expected to be of comparable power, because a typical wireless cellular system has a power control loop which maintains the received *Signal to Interference Ratio (SIR)* at a fixed level. A good starting point on the way to this fully functioning receiver system, would be to first create an AFSR receiver capable of demodulating a single channel. The channel would involve a high frequency rf carrier, spread by an AM modulated PN sequence, carrying PSK modulated data bits at baseband frequencies. Of course, it is unnecessary to deal with the complexity of the rf carrier electronics, when AM demodulation would present an envelope to the AFSR anyway. So the real starting point is to create a hardware AFSR capable of entraining to a single attenuated and noisy LFSR signal with off-the-shelf components.

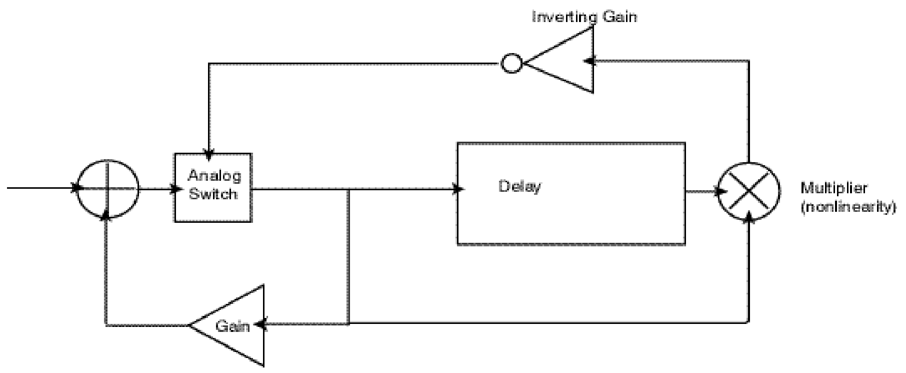


FIGURE 7.1.1

7.2 Implementation Details

The elements in the DS CT (with external CR) AFSR are delay (to implement the shift register taps), a quadratic nonlinear function, gain, and an analog switch (to implement $\theta(z(t) - z_c)$). We will discuss them one by one.

7.2.1 Delay

Probably the most essential element in AFSR is delay. Whether we are building a DS DT AFSR for software radio, a CS CT AFSR our of rf hardware, or a lower frequency hardware system, delay will be needed. The important requirement for the delay element in AFSR is that we need to be able to sample the signal from beginning of the delay line and the end. We need to be able to make a copy of the output of the nonlinear function and delay one copy before recombining the two copies back into the nonlinear function.

At frequencies of 10's Gigachips per second, microstrip lines on a carefully selected dielectric substrate need only be mm's long to implement AFSR delay. At the lower frequencies of my hardware prototype, however, the microstrips would be km long as seen in equation 3.3.1- clearly unfeasible. Instead, I have experimented with constructing an analog delay line by building an $ADC \rightarrow FIFObuffer \rightarrow DAC$ system made from a PIC16C711 with an internal 40KHz 8 bit ADC, a 74ACT2708 Fairchild FIFO buffer, and MAX505 8 bit parallel ADC from Maxim. I have also tried constructing a delay element using an MN3207 “capacitive bucket brigade” chip with an external clock, the MN3102.

I have learned that if I use a FIFO buffer that has an ADC at the beginning of the line and a DAC at the end, or bucket brigade that takes analog samples in time, I need to use a smoothing filter at the output of the delay line in order to mitigate the effects of digitization. The detrimental effects of discretization of state on AFSR are modeled in “Modeling Acquisition Times for Discrete State (DS) DT AFSR” on page 61. It turns out that AFSR is very resilient to discretization of state. In practice and in computer models, 8 bits of ADC is more than sufficient for AFSR to function properly.

It is also important to take at least 16 samples per chip, so that the transitions are accurate. For a 4 bin AFSR, this means we need a 8 bit wide, 64 bit deep FIFO buffer delay line. My FIFO buffer was exactly this large and it worked okay, but the available precision was at the lower limit of acceptability for building an AFSR. Also, the FIFO buffer used all of the spare pins on my ADC PIC to control its input and output functions. I wanted a simpler system with more samples per chip, so I decided to use the bucket brigade chips instead which are 1024 and 2048 samples deep and can sample at up to a few hundred kHz. Using the bucket brigade chips with such plentiful over-sampling seems like a reasonable thing to do in anticipation of having continuous time microstrip lines or thousands of bucket brigade transistors available to eventually build an rf AFSR system.

7.2.2 Nonlinear Function

We would like to use the simplest possible nonlinearity. It is important to note that the fundamental requirement of the nonlinear function is that it must be axis symmetrical. In other words it must perform the operation $g(1, 1) \to 1$ and $g(-1, -1) \to 1$. This means that a simple mixer is not sufficient. Mixers and transistors are sometimes referred to as “nonlinear” circuit elements, because they prevent a circuit from having a simple transfer function the way a circuit built with capacitors, inductors, and resisters does. They are considered nonlinear because they are not simply understood using Fourier analysis and are not completely described by linear differential equations. This is not nonlinear in the sense that we mean it here, however. What we need is slightly more demanding nonlinearity.

A linear function, a line, given by the formula $y = ax + b$, is an odd function; It is point-wise symmetrical about the a point where it cross the y axis. We need an even function which is axis symmetric about the y-axis. This can be considered the simplest possible nonlinear function, a mere polynomial of order two (quadratic) for example will fit the bill. An off-the-shelf analog multiplier or double balanced mixer satisfies this criteria perfectly and even provides the quadratic shape that we desire. It need not bother us at all if the function is not perfectly quadratic. It should simply approximate the saddle point shape of the function in Figure 6.2.6 on page 49. I am using a Burr-Brown MPY634 analog mixer which is good up to a few MHz, and has an extremely accurate quadratic curve.

7.2.3 Gain

Gain is the simplest element in the system. As shown in the block diagram above, we would like one dual supply inverting gain and one non-inverting gain that can be varied from 1 to about 10. Also the amplifiers must be well behaved at the rails. A standard operational amplifier that works as a comparator will be sure to operate well at the rails. The amplifier must also have a sufficiently fast response. I have found the dual supply AD756 from Analog Devices operational amplifiers were more than satisfactory for this purpose.

7.2.4 Analog Switching

In current instantiation of the AFSR system, we need to gate the transitions on each time step. Most of the time the AFSR listens primarily to the external LFSR signal, and a small feedback gain (probably in combination with a low pass filter) causes the steady state value of the AFSR to move toward the rail to which it is nearest. For short times around the clocked transitions, we must gate in the internal signal from the AFSR feedback to cause a transition. At rf frequencies trying to flip a switch for a fraction of the duty cycle of a 1 or 10 MHz chip will be the achilles heel of the system and will probably be the single most formidable obstacle that must be overcome in order to make an rf AFSR a reality. I call this the clock recovery/CT AFSR problem, and we will discuss it at further length when we come to consider the construction of an rf AFSR.

For this lower frequency prototype, however, this gating is not such a big problem. I am using Pericom's PI5A317 solid state SPST analog switch, with -72dB low-off separation at 10MHz, and a more than adequate 10ns switching time. The only disadvantage of these switches is that silicon switches generally operate in one voltage regime, in this case 2-6 Volts, so it will be necessary to shift the DC offset and scale my signal for the switch and then reverse the process afterwards. This kind of compatibility problem between off-the-shelf analog components intended for use in a wide variety of different and esoteric applications has plagued this project. For example, the bucket brigade chips are optimized for use in inexpensive electric guitar effects pedals. It would be wonderful to eventually have integrated components custom made for implementing AFSR.

Beyond the needs of AFSR, it is worth considering whether we can ever expect there to be a standard I/O scheme for analog components for nonlinear dynamic computation just as there is already for digital components. This is actually a difficult problem, because the meaning of adjacent real values output by a nonlinear dynamic system may be specific to the state space configuration of that particular system. What is needed is a scheme for inducing a universal metric across all dynamic system I/O. I have done some preliminary work on this subject in order to understand how to induce a continuous metric on the computational language of finite state machines. The problem is much more difficult, however, for general purpose Turing Machines. Perhaps, the best we may hope to do is induce a standard metric across certain classes of systems.

7.3 Performance of Prototype AFSR Hardware

This system is based on the continuous time system from the original PRL paper on AFSR in as much as we are still using an external clock recovery circuit to gate the transitions with a gating function. We have changed the map from a cosine map to a quadratic map, but this should only improve performance. Therefore the hardware should perform just as the models in the original paper predict. I dropped furiously working toward getting this hardware working, however, when I realized I may have discovered how to make a continuous time (CT) AFSR that dispenses with the need for an analog switch or external clock recovery circuit, and needed time to start modeling this new system. This is the subject of the Section "Towards CT CS AFSR - Radio Frequency (rf)" on page 64.

It should be a fairly simple matter to put the finishing touches on the low frequency hardware prototype. One obstacle that I can foresee is the constant need to use extra operational amplifiers to rescale the signal values to mediate between the I/O specifications of the various key components and the extra noise that this introduces. All of the key components have been assembled and modeling tells us that the system should work. The only other foreseeable obstacle is the effect of imperfections in the length of the delay line on the stability of the system. In the work on CT CS AFSR in the last chapter, however, we will see that delay line imperfections should not propose a serious difficulty.

8 DS DT AFSR - Software Radio

8.1 Modeling Acquisition Times for Discrete State (DS) DT AFSR

AFSR so far has been grounded in a discrete time (DT) iterative map. As we shall soon see, discretizing the analog state of the register bins into quantized levels will not significantly inhibit the proper operation of the Quadratic DT AFSR. This system therefore lends itself to a software radio implementation. As silicon scaling pushes up the clock rate of DSP, there will likely be increasing demand for better algorithms for software radio spreading code demodulation running at chip frequencies. In fact, some state of the art SS base stations currently employ ultra-fast Josephson junction electronics to do software radio on IF and even carrier signals. In applications where low power consumption is not a significant design criteria, software radio has the enormous benefit of being instantly reconfigurable. This reconfigurability means that updating the receiver demodulation architecture to incorporate advances in coding and modulation is a simple matter of updating code instead of replacing hardware. The military already employs ultra fast software radios for eavesdropping. By gathering statistics on the channel, these systems are able to discover the modulation scheme of the communication and demodulate the data. Cell phones are starting to require similar systems so that they can communicate with the various wireless modulation standards employed in different geographical areas. We will therefore attempt to evaluate the prospects of DT Quadratic AFSR as a software radio acquisition algorithm. In this discussion, we will not address the clock recovery problem, which would very likely need to be handled by a separate subsystem in an AFSR based software radio SS receiver.

I have already begun to encounter the effects of discretization of state (DS) in AFSR when I created the $ADC \rightarrow FIFObuffer \rightarrow DAC$ delay element for my prototype AFSR hardware. In theory, this element would only need a one bit ADC in a noiseless world. The AFSR dynamics could operate properly if presented with a perfectly noise free incoming signal DC offset to the threshold of a 1 bit ADC. If the incoming signal were above the threshold, the AFSR would push its internal state (which would have a few more bits of resolution) toward the positive rail, and similarly, if the signal were below the threshold, so that the 1 bit ADC returned a 0 value, the AFSR would push its internal register state toward the negative rail. A push high would send the dynamics outward toward +1 and a push low would send the dynamics out toward -1.

In practice, if we are interested in using AFSR for DS SS CDMA, the other PN sequences will appear as noise on the channel. So even if there were no thermodynamic noise whatsoever, we would still need more than 1 bit of ADC. For software radio DS DT AFSR, there are really two questions: 1) How many bits of resolution does the ADC of the incoming signal require? and 2) How many bits of resolution do the “analog” states of the AFSR register bins actually require? Indeed, we would like AFSR to be able to distinguish between a large positive value and small positive value in both the incoming samples and the internal register states. If we observe a large positive signal, that means we have more confidence that it is not due to noise and consequently we are willing to move the dynamics further toward +1 than if we had observed a small positive signal. Greater resolution allows us to better exploit the shape of the quadratic function.

The answer to the first question appears to be that both in simulation and in my hardware prototype, 8 bits of resolution in the input sampling ADC is more than sufficient. “Average Acquisition Time vs. Number of Bits of Accuracy in Receiver ADC” on page 62 shows the average acquisition time for DS DT Quadratic AFSR for input ADC resolutions of 2 to 32 bits. 50 acquisition trials were run for each level of bit resolution with noise variance =.5, gain = 4, maximum acquisition time = 200, epsilon=.6, and 15 register bins. The figure shows that after we reach 8 bits of resolution, the mean acquisition time drops to around 60 chips and stays there reliably.

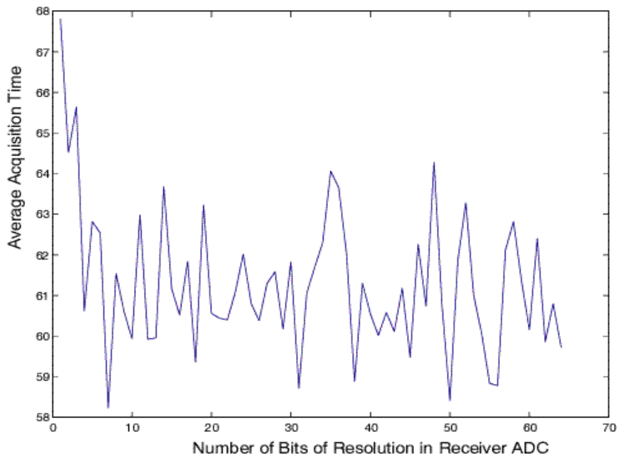


FIGURE 8.1.1 Average Acquisition Time vs. Number of Bits of Accuracy in Receiver ADC

What about our second question? In the above trials, only the AFSR input was discretized. What is the effect of discretizing the internal states of the AFSR throughout the algorithm/signal chain? This a question which will become important if we decide to implement AFSR in a fixed point DSP, and is an avenue for further investigation. We can be confident that AFSR will perform robustly even with truncated precision in the register states. AFSR can operate with so few bits of resolution, because of the resilience of the dynamics.

8.2 Algorithm Complexity

Here is the actual code kernel in Matlab for the discrete time AFSR.

```
1.afsr(lfsrlen+1) = eps*noisyxmit(t) - (1-eps)*gain*afsr(lfsrlen)*afsr(1);
2.%threshold cutoff (rails of amplifier)
3.  if  afsr(lfsrlen+1) >= [1]
4.  afsr(lfsrlen+1) = [1];
5.  elseif  afsr(lfsrlen+1)<= [-1]
6.  afsr(lfsrlen+1) = [-1];
7.  end
```

The most important part of AFSR is contained in line 1 above. Line 1 requires at least one analog to digital conversion (ADC) per chip to get the value of the incoming signal, noisyxmit(t), if we assume an external clock recovery system. The number of bits of accuracy required from the ADC will be determined by the SNR of the incoming signal. The multiply in eps*noisyxmit(t) would most likely be done in hardware before the ADC by an automatic gain control (AGC) which would hope to match the dynamic range of the incoming signal to the dynamic range of the ADC. A slower outer loop to control this hardware eps and the software (1-eps) value will probably be required. This loop might only need to adjust eps once per 100 or 1000 chips, so we can probably disregard its contribution to the algorithmic complexity of AFSR. Proceeding with this assumption, line 1 costs 2 floating point multiplies, and 1 subtraction per chip since (1-eps)*gain can be precomputed by the slower outer loop. The subtraction will generally be dominated by the multiplies.

Lines 3-7 bound the AFSR dynamics by setting a maximum value for the AFSR register values of 1 and -1. Without this cutoff, line 1 would force the register values to move away from 0 at an exponential rate. In a hardware AFSR this bound would be the default situation, because of clipping at the rails of the AFSR gain amplifier. Lines 3-7 require 1 or 2 floating point tests. The average will be 1.5 floating point tests. Depending on the ALU of the AFSR DSP, it may be advantageous to rewrite this as a single test to see whether the absolute value of afsr(lfsrlen+1)<=1, and if not to set the value to 1, meanwhile simply leaving the sign bit of afsr(lfsrlen+1) unmodified. The tests result in one assignation. So the discrete time AFSR software radio has an algorithmic complexity of 1 ADC, 2 multiplies, 1 subtraction, 1 or 1.5 tests, and an assignation per chip.

Information about state-of-the-art SS software radio DSP acquisition algorithm complexity is difficult to collect, because it tends to be proprietary, so it is not entirely clear whether AFSR is advantageous from the viewpoint of algorithmic complexity. Today's state of the art acquisition systems use matched filters on a specially constructed acquisition channel to find the beginning of the PN sequence. The matched filters will require approximately the same accuracy of ADC running at the chip rate as our discrete time AFSR. Importantly, however, these matched filters will also tend to require 10-100 multiplications per chip. These multiplications are likely done in a specialized DSP pipeline so that the clock rate of the DSP is about the same rate as the chip rate. AFSR will require the same clock rate and its own specialized pipeline, so AFSR may or may not be advantageous as an acquisition algorithm for software radio.

9 Towards CT CS AFSR - Radio Frequency (rf)

9.1 The Clock Recovery Problem and CT AFSR

The achilles heel of AFSR as it was originally conceived is that it is fundamentally a discrete time system. Previous proposals for how to make a continuous time AFSR have all required an external clock recovery and tracking loop which would trigger the discrete map iterations. This external clocking poses critical problems for implementing AFSR as an rf system.

The first problem is that the AFSR based receiver will have to be coherent. Most practical SS receivers are non-coherent. Non-coherent detection in this context means that the receiver can acquire the PN sequence independent of clock recovery. Non-coherent receivers accomplish this by having specially constructing codes on dedicated control channels. These codes are constructed so that a matched filter in the receiver can find the beginning of the code. The matched filter can then turn over the beginning of the code sequence to correlators which can search for other PN sequences that are present in the channel. Chip clock synchronization has no bearing on this operation. There have existed coherent detectors in the history of SS receivers. They generally do something like high pass filter the channel to pull out the absolute value of the chip transitions and then try to recover the chip clock with a DLL or Kalman Filter type system. It is possible to revisit this approach, but common practice holds that the added system complexity and sensitivity to noise makes coherent detection undesirable. We therefore dream of creating an AFSR system which includes clock recovery in its dynamics.

The second problem with having an external clock trigger transitions in the AFSR is the triggering itself. This triggering will require a switch operating at chip frequencies, with a duty cycle 2 to 3 orders of magnitude smaller than the chip length. In other words, this switch would have to turn on for 1 ns in order to switch 1 M chip per second chips. Such a switch is not completely out of the question, but the high frequency transients and harmonics that it would likely introduce into an rf system would be extremely problematic. In addition, a 1 GHz switch will negate any power consumption benefits we might have hoped to gain from the adiabaticity we would have hoped to find in a continuous time nonlinear dynamic flow. So we dream of lifting the discrete time AFSR map into a higher dimensional smooth flow which lacks an explicit or implicit clocking. It turns out that there is a system which solves both problems.

It is not a simple matter to lift the discrete time map of AFSR into a continuous time dynamics. There is no formal procedure in nonlinear dynamics for accomplishing this. I hope that the path I took may generalize to some degree. Indeed, if we were to speculate that we are going to make a practice of building systems to perform computations with nonlinear dynamics, such a method could be more widely useful because we will generally be starting from algorithmic or discrete time conceptions of the computation. Be that as it may I have made some headway on this problem.

The crucial first step is to realize that, as a discrete time map, AFSR can be thought of as the Poincare map of some as yet unknown continuous time flow. Basically we are going to try to go the opposite way of a Poincare map analysis, from map to flow instead of from flow to map. The problem we therefore pose is as follows. We have a discrete time nonlinear map, which we want to understand as the poincare map of a some higher dimensional continuous time flow which periodically passes through a Poincare surface by following a periodic orbit, gamma. In other words, we want to invent a continuous time system that when sampled at each chip interval, has the behavior of the AFSR map. This sampling is purely conceptual. We think of the flow periodically passing through the Poincare surface, but the actual dynamics of the flow will indeed be entirely continuous in time.

The first hint how to solve this problem comes from noticing that the discrete time AFSR map has stable fixed points as was discussed in “Building Fixed Points by Exploiting Gain Clipping” on page 50. The derivative at these points is zero due to the clipping as seen in Figure 6.2.9 on page 51. We want these fixed points to be hyperbolic fixed points first of all to be assured of their stability, but more importantly so that we could have reason to hope that the hypothetical orbit, gamma, will approach a regular periodicity as the map settles down to its fixed points.

My first instinct was to imagine a single Poincare surface with a flow orbit traversing it at multiple fixed points as illustrated in Figure 9.1.1 below. The idea was that at some point in the orbit, the flow would cross near to itself, so that in that vicinity, the flow would be sensitive to a forcing function which decide through which fixed point the orbit would pass next.

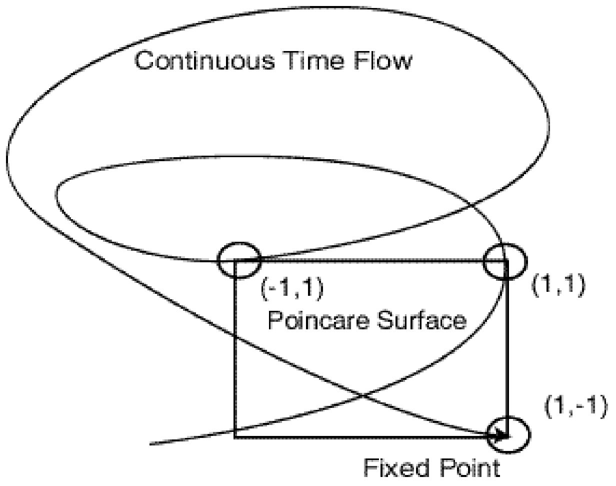


FIGURE 9.1.1 A Graphical Representation of “Pushing the Swing” Through a Poincare Surface

The problem with this approach, which we dubbed “Pushing the Swing” is that at the sensitive point in the dynamics there would also be inordinate sensitivity to noise. This kind of sensitivity will have some of the same drawbacks as the CS CT AFSR with an external clock proposed in the original AFSR paper. Namely that certain specific short periods of time in the dynamics would have favored sensitivity. We would like the sensitivity of the dynamics to be more egalitarian. This means that the orbit must be “unknotted” so that it no longer crosses itself in state space. This is the idea of the successful method that was tried next.

9.2 Successful CS CT AFSR

For simplicity, let us begin with a two bin DT Quadratic AFSR in order to simplify the visualization of the manifolds we will need. The difference equation for this iterative map is

$$x(t+\tau) = -x(t)x(t-\tau)\,. \tag{9.2.1}$$

Iterating this map produces a very simple PN sequence

$$(1, 1) \to (0, 1) \to (1, 0) \to (1, 1)\dots \tag{9.2.2}$$

which is equivalent under GF(2) to

$$(1, 1) \to (-1, 1) \to (1, -1) \to (1, 1)\dots\,. \tag{9.2.3}$$

The evolution of the state of this system in discrete time requires two dimensions to represent and is given in the table below. Let $y(t) = x(t-\tau)$ so that in the table, each column leads to the one to its right by shifting down one, and replacing the top value by the output of the nonlinear function performed on past values with taps.

$x(t)$	1	-1	-1	1	1
$y(t)$	1	1	-1	-1	1
time	0	1τ	2τ	3τ	$4\tau\ldots$

We are looking for a three dimensional flow which passes through each of these 2-tuples in its continuous orbit through time, so that we can lift this two bin DT Quadratic AFSR map into a flow through continuous time as portrayed in Figure 9.2.1 below.

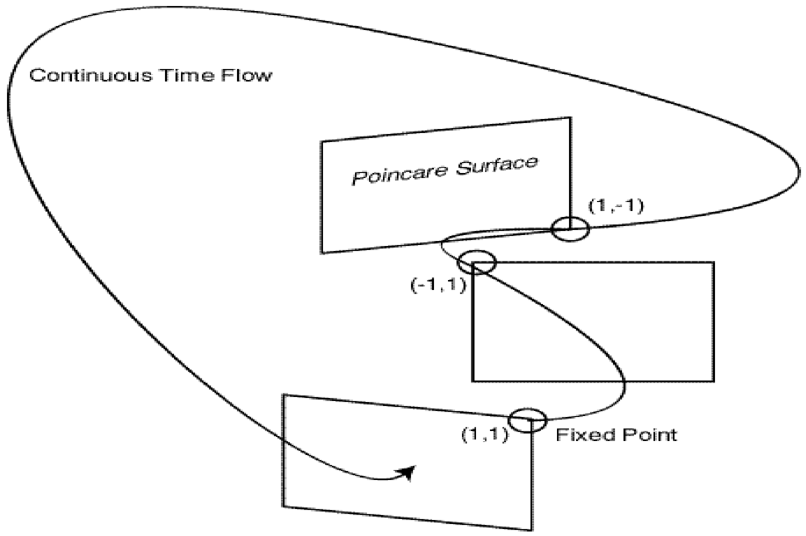


FIGURE 9.2.1 Lifting 2 Bin Discrete Time AFSR Into Continuous Time Flow

We can see from the table above that during a single chip period we would somehow like for (9.2.4)

$$\begin{aligned} y(t) &\to x(t-\tau) \\ x(t) &\to (-x(t-\tau)y(t-\tau)) \end{aligned} \quad (9.2.5)$$

in a smooth way. Now that we have posed the problem in this way, we can see this can be accomplished by writing the following dynamics

$$\begin{aligned} \frac{dy}{dt} &= A[x(t-\tau)-y(t)] \\ \frac{dx}{dt} &= B[-x(t-\tau)y(t-\tau)-x(t)] \end{aligned} . \quad (9.2.6)$$

Once an iterative map has been represented in the proper number of dimensions, this last step is the heart of the method for converting it to a continuous flow. I simulated these continuous dynamics using Euler's method with the following Matlab code (xgain = ygain = 1).

```
y(t) = y(t-1) + ygain*(x(t-100)-y(t-1));
x(t) = x(t-1) + xgain*(-x(t-100)*y(t-100)-x(t-1));
%%threshold cutoff (rails of amplifier)
if  y(t) >= [1]
    y(t) = [1];
elseif  y(t) <= [-1]
    y(t) = [-1];
end
```

Depending on the value of the gain, this code produces the behavior shown in Figures9.2.2. When the gain is high, the dynamics are indistinguishable from the behavior of a 15 bin LFSR.

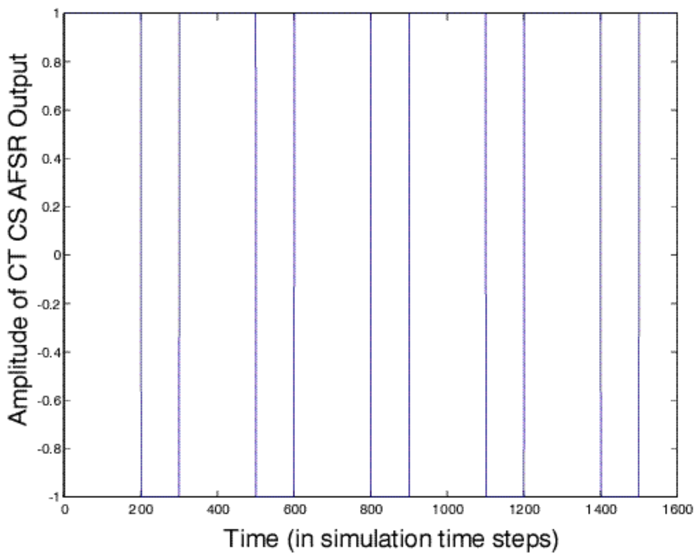


FIGURE 9.2.2

However, when the gain is not so high, we can see the autonomous continuous dynamics at work. With a gain of .5 as in Figure 9.2.3 or a gain of .1 as in Figure 9.2.4, the dynamics are dissipative but still attempt to produce a PN sequence until the system runs out of energy.

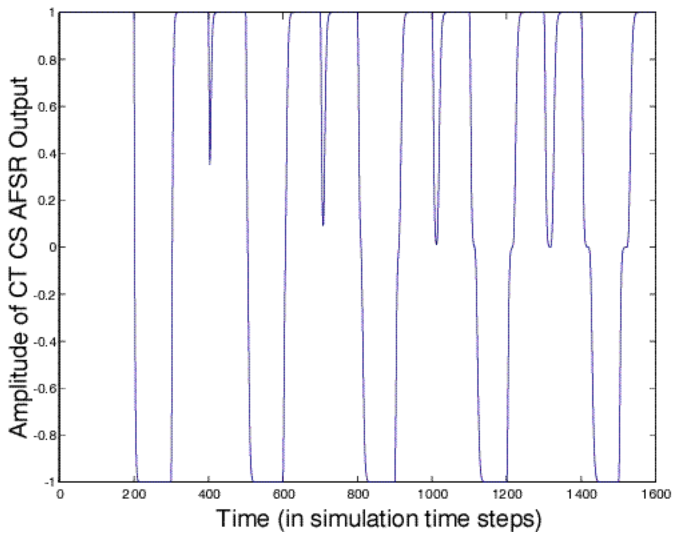


FIGURE 9.2.3 CT Quadratic AFSR with xgain=ygain=.5.

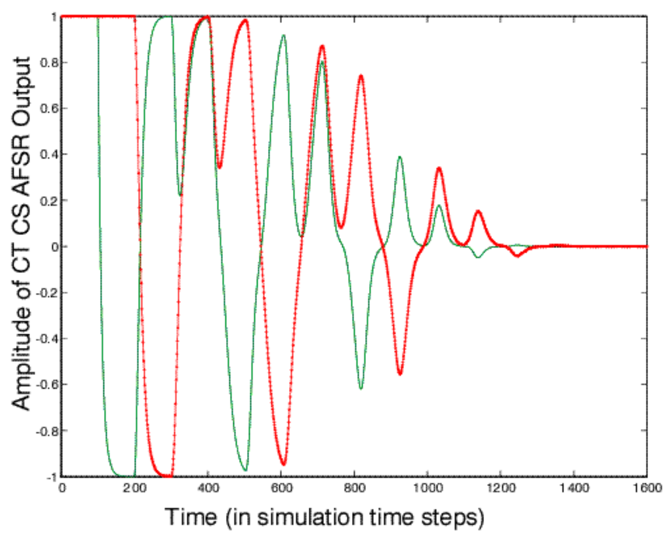


FIGURE 9.2.4 CT Quadratic AFSR with xgain=ygain =.1.

For purposes of optimizing the simulation code, x and y can be decoupled so that we write the following, this time for a 15 bin CT Quadratic AFSR

```
y(t) = y(t-1) + gain*(-y(t-1500)*y(t-100)-y(t-1));
%%threshold cutoff (rails of amplifier)
if  y(t) >= [1]
    y(t) = [1];
elseif  y(t) <= [-1]
    y(t) = [-1];
end
```

The output from this 15 bin CT Quadratic AFSR is shown in Figures 9.2.6 and 9.2.5 below.

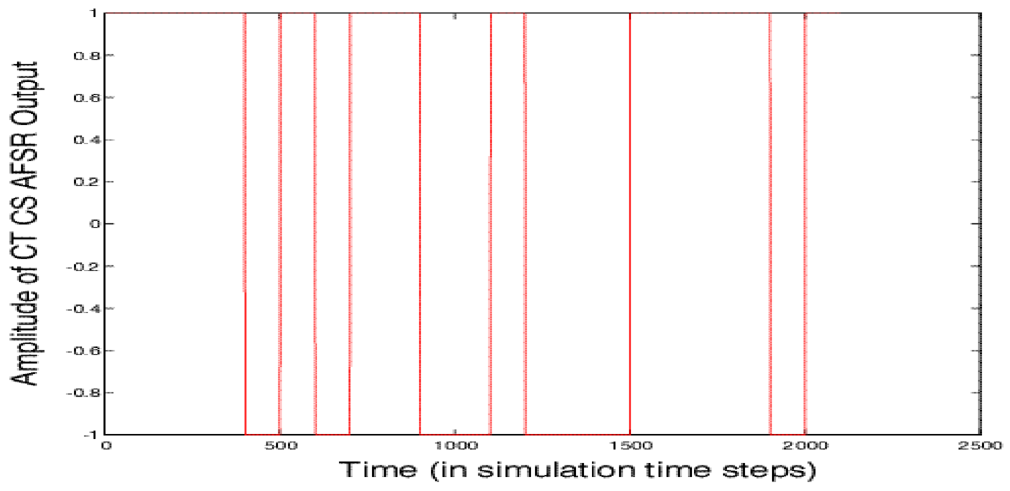


FIGURE 9.2.5 CS CT AFSR Running Autonomously with Gain = 2, noise variance = 0.

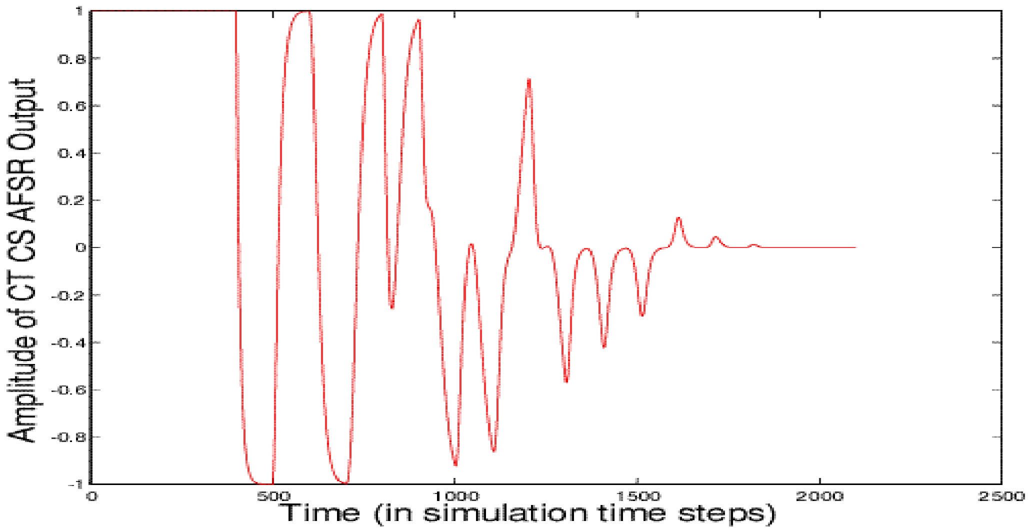


FIGURE 9.2.6 CS CT AFSR Running Autonomously with Gain = .5, noise variance = 0.

This system will be very simple to implement with actual hardware. All that is required is the delay element, double-balanced mixer (multiplier), and dual supply gain. One might even hope to print such a system since it requires very few transistors. The schematic for the system is represented in Figure 9.2.7 below.

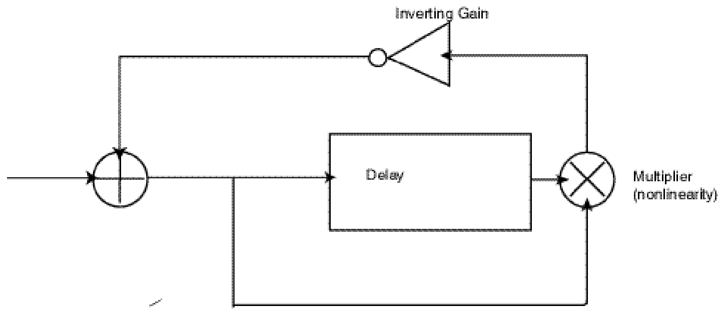


FIGURE 9.2.7 Block Diagram of CS C AFSR

Having gone through the previous analysis to arrive at the schematic above, we might observe that without the delay element this system is similar in nature to the familiar ring oscillator shown in Figure 9.2.8 below. The ring oscillator in practice requires 3 inverting gains in series to buffer the output of a gain from its own input. For conceptually, however, one inverter would be enough. In our system, the delay line and multiplier provide the necessary buffer.

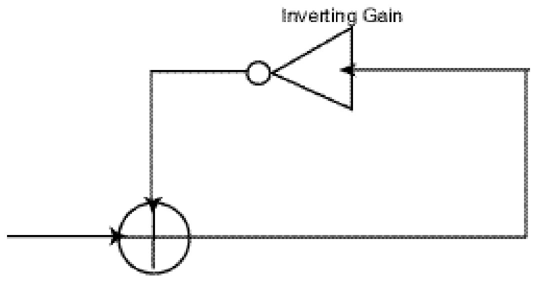


FIGURE 9.2.8 Familiar Ring Oscillator Made from Single Inverting Gain

9.3 AFSR and Power Consumption of SS Acquisition

An interesting result from early prototypes of multi-hop peer-to-peer radio networks, is that the receivers tend to use more power than the transmitter. Ordinarily, we are accustomed to transmitters using the majority of the power in a transceiver, because of the cost of radiating rf energy. In wireless networks where the transmission range is on the order of meters, however, we have found that the demodulation and decoding operations in GHz receivers actually cause the receiver to use more energy than the transmitter. There is a simple physical explanation for this.

On the one hand, let us consider the energy used to emit electromagnetic radiation at a given frequency, say 1GHz. Radiated energy goes as $E\alpha\nu$ where ν is frequency of the radiated electromagnetic wave. On the other hand let us consider the energy used in a DSP chip to do software radio demodulation of the signal carried on that electromagnetic wave. A software radio will generally have to do 10-100 operations per data bit. Today a 1GHz signal typically carries data at order 100Kbps, therefore requiring (conservatively) a 1Mflop software radio. Higher carrier frequencies enabled by advances in high speed silicon for rf will always be followed by increasing data rates enabled by the same silicon scaling and driven by the insatiable human hunger for bandwidth. For the next 20 years we can safely predict that the frequency of operation of a software radio, f, will be roughly related to the carrier frequency, ν, of the radio by the relation

$$f = \beta\nu \tag{9.3.1}$$

where $\beta \approx \frac{1}{1000}$.

The energy consumption of the DSP does not scale linearly with the frequency of operation, f, however. Forcing a state transition in a transistor can be simplistically modeled as charging or discharging a small capacitance across a small resistance. The charge on a capacitor is related to the voltage applied via the capacitance C by the relation

$$C = \frac{Q}{V}. \tag{9.3.2}$$

The current across a capacitor is therefore

$$C\frac{dV}{dt} = \frac{dQ}{dt} = I. \tag{9.3.3}$$

We conservatively approximate the transitions on the transistors as a nice adiabatic sinusoidal voltage of period $w = 2\pi f$, so that $V = \exp(iwt)$. The current across the capacitor is therefore

$$I = C\frac{dV}{dt} = iwCe^{iwt}. \tag{9.3.4}$$

Since the power dissipated in a resistance is

$$P = -\frac{dW}{dt} = I^2R, \tag{9.3.5}$$

the power dissipated in the transistor can be approximated as

$$P = (iwCe^{iwt})^2R = \alpha f^2C^2. \tag{9.3.6}$$

So power dissipation in the DSP goes as the square of the frequency of operation. Even if we attempt to counteract this effect by reducing the capacitance of the transistors quadratically by reducing the feature size to keep pace with the increase in frequency, the number of transistors tends to increase as we can fit quadratically more onto the same size die so again the power consumption of the DSP goes up quadratically.

Since we have not specified the scale factors here, we cannot make an exact prediction, but we can make the general prediction that if the volume of space that a transceiver is expected to fill with E-M energy is kept constant, the quadratic power consumption curve of the DSP due to silicon scaling will eventually overtake the power consumption of the transmitter which is linear.   For meter range transceivers operating at 100's MHz, this crossover has already occurred, so that the DSP software radio requires more power than the transmit power amplifier.

A primary advantage of AFSR over DSP software radio is that the AFSR dynamics perform an approximately adiabatic flow through state space until acquisition is achieved, at which point the system is performing transitions at exactly the chip rate. The AFSR also has an extremely small parts count of only a few transistors. The power dissipation of AFSR can therefore be expected to be less than that of the equivalent software radio acquisition system.

10 The Future of AFSR and Beyond

10.1 Integrating rf AFSR for Low Cost

Much of the cost of the radio hardware comes from problems with integrating the higher frequency carrier and spread spectrum demodulation hardware with the baseband decoding computer onto a single chip. At frequencies of Gigahertz, high speed DSP for spread spectrum demodulation is not only extremely difficult to integrate, but also consumes a lot of power. If some of the same signal processing could be achieved with an integrated AFSR which is adiabatic as it searches for acquisition and only needs to transition at the chip rate during tracking, it could potentially be less expensive and consume less power.

10.2 AFSR Coding for Complex Channel Requirements

It would be nice to extend AFSR to be able to perform acquisition for Gold Codes and other advanced SS codes. It is not at all clear how to do this. We would also like to be able to sculpt the frequency spectrum of the PN sequence of an AFSR in order to be able to match the spectrum to the transfer function of a given channel. Further modeling would be required to determine if AFSR would likely entrain to a filtered version of the LFSR signal. An alternative to post-filtering would be to follow Feher's idea of using radio frequency digital waveform synthesizers (made from FPGAs) to sculpt the spreading chips to have a different harmonic structure and therefore to present a different overall spectrum. The AFSR would need to be made resilient to these differently shaped chips. This actually seems possible, since these chip signal would be a diffeomorphism of the standard LFSR signal, there should be a smooth transformation of the AFSR dynamics suited to deal with this signal. Another important problem for portable wireless is dealing with doppler shift in a fading channel. This would require that the orbit of AFSR be able to frequency shift to some degree. This seems like a promising application for AFSR since doppler shift is a difficult problem for conventional DSP SS receivers with their more time sensitive matched filter techniques.

10.3 Integrated Hybrid Analog-Digital Signal Processing

In this work, we have proposed an analog system for computing a DSP coding problem. It is interesting to consider the possibility of building integrated signal processing chips in which would have specialized components of the form $\mathrm{DAC} \rightarrow \mathrm{NonlinearDynamics} \rightarrow \mathrm{ADC}$ integrated into the signal chain. We are used to thinking of ADC and DAC as expensive specialized components, but this is primarily because they are most often used to interface to the outside world. That means that they must generally have impedance, power, and noise specifications to deal with external loads. It is not inconceivable to imagine that integrated ADC and DAC for conversions between integrated analog and digital segments on a chip could be made with smaller parts counts, and extremely high speed and accuracy. I don't know of any reason why on-chip ADC and DAC scaling shouldn't follow silicon scaling closely. We tend to shy away from mixed signal integration, because conventionally it is necessary to shield the analog ground from the noisy digital ground. As we have seen from modeling, however, nonlinear dynamic systems can be quite robust to noise, and can offer the advantage of performing a computation in a more efficient and robust way. Hybrid integrated analog nonlinear dynamics and digital signal processing may therefore offer some promise for the future.

10.4 AFSR for Energy Distribution

Artificial satellites can deploy solar panels to collect a virtually unlimited crop of clean solar energy. The problem is that this energy is only useful to the satellite because there is no easy way to get it down to the earth's surface for our use. Proposals have been made for sending a focused beam from a maser in the satellite to ground. This has the obvious problem that the beam can fry people and things like ants under a magnifying glass. In principle, if you could spread the frequency of the beam, there would be less energy at any given frequency, so that the quantity of energy available for absorption at any given frequency would not be so large and the beam of the same power would cook us much slower. A high power pseudo-optical AFSR on the ground could conceivably be used to perform the extremely high power, high speed demodulation that would be required to recover the original beam. In a metaphorical sense, the AFSR would be acting like a maxwell's demon to recover energy from a maximum entropy channel. This is what is known as a crazy idea. What I have learned at MIT is that all seemingly insane technological proposals, if they are in principle feasible, have an application somewhere.

For example, flat panel displays cost a lot to make because you have to run a row and column conductor to every pixel and at each intersection, there must be a few control transistors connected to the address lines to activate the pixel. This means it is necessary to manufacture a large panel of transistors with very few transistors that don't work. The number of transistors in a flat panel display scales with the area of the panel, so making the display bigger is very difficult because it means manufacturing many more perfect transistors. With the probabilities of transistor yield as they are, making a flat panel display twice as large requires it to cost 10 times as much. A laptop screen twice the size of the current standard would cost on the order of $10,000. However, if we could embed a different AFSR at every pixel, we could simply beam rf energy at the display with the appropriate PN sequence. The appropriate AFSR would be excited by the incoming signal, but the other AFSR with orthogonal PN sequences would not be excited. We could therefore activate a pixel of our choice with no connections and no multiplexing. We could just send in a beam of energy with the destination address built into it.

10.5 AFSR and Auto ID Tags for a Penny

In a few years printed transistors will be commercially available. It is interesting to consider how we could fashion low cost I.D. tags by printing a few transistors. It is possible that an AFSR could be fashioned from a few transistors and a printed delay line. If this were possible, we could use the energy distribution method above. A tag reader would issue different PN sequences on the channel in series or in parallel. If an AFSR RFID tag was excited by a sequence from the receiver it would appear to the reader as a singularity in the "PN-spectrum" just as a LC tag appears as a resonance in the frequency spectrum.

Another idea would be to augment present resonant LC, magnetostrictor, or j-wire tags so that they exhibit new interesting time behavior due to a nonlinearity imposed by an AFSR structure. Even if only simple transient modifications of the time behavior are accomplished this way, it is still a very useful thing to do. Suppose we make 5 different frequency LC tags. We can store 5 bits or 25 unique ID's. Suppose, however, that each LC tag can be modified to exhibit 3 independent types of transient time behavior. We now have 15 properties present or not. So we now have 215 unique ID's. The moral of the story is that we don't need much from the nonlinear time behavior in order to push penny tags significantly closer to being able to store 30 bits for a billion unique I.D's.

The general most case of looking for nonlinear time series behavior would be to make an embedding tag reader. The tag would be a low dimensional nonlinear dynamic system which would be excited by energy from the reader. We would need to be able to vary the topology of the embedding space by an inexpensive, reliable parameter variation in the tag system. However, this is the most general case of how to pull out information about a nonlinear dynamic system. If we built a CS CT AFSR tag, which either modulated or reradiated energy from the tag reader, then the time series sampled in a carefully chosen lag space would present a PN sequence to the reader. Just as in a spread spectrum system, we would expect to be able to identify each tag that is present, and read multiple tags at once without interference since they would be sharing the channel.

10.6 AFSR and Biological Neurons

Whenever we send a signal, we must encode it in some modulation scheme. Essentially this means that the data is encoded as deviations from some regularity. In an old fashioned radio the regularity is a sine wave. In a SS radio, the regularity is deterministic pseudo-random bits. Either way, the data is encoded as deviation from regularity and the receiver must be able to generate a copy of the regularity in order to detect the deviations. In an old fashioned radio, a phase locked loop is used to lock onto the sinusoidal carrier even in the presence of data modulation and noise. In an SS system we can hope that AFSR will be used as the equivalent of a PLL for pseudo-random noise.

In section, we talked about AFSR for IrDA. The fundamental design constraint for IrDA is the need to maximize pulse amplitude for maximal transmit range and immunity to interference. This maximum amplitude constraint quickly leads to the need to minimize the duration of a pulse and the duty cycle of transmitted signal, so that the transmit hardware can recover from the strain of sending maximum power bursts. If we examine the design trade-offs for a neuron, the is problem is virtually identical. The neuron needs to maximize burst power in order to deal with a noisy environment. Both IrDA and neurons use pulse trains with some combination of pulse code modulation and phase keyed modulation to encode data in temporal patterns of bursts.

In IrDA the regularity is the burst clock. The receiver uses a delay locked loop to acquire this regularity. The transmitter can then send data by phase keying. Instead of sending a burst on every clock cycle, the transmitter encodes a ‘1’ by sending a burst in between clocks. It has been shown that the encoding scheme of the neuron is not this simple. In fact, the coding scheme of the neuron is still poorly understood. We can be sure, however, that once it is found i t will involve some regularity and some scheme for encoding information by defying this regularity. If a relatively small and simple structure like a single neuron or even a single dendrite needs to be able to synchronize to a complex regularity in order to decode received signals, making use of whatever computing resources are available including analog space, then we might guess that we would find structures akin in spirit to the AFSR within our very own minds.

11 Conclusions

11.1 Summary of Results

This work presented the TouchTag reader (patent pending) for near-field electrostatic communication and the AFSR system for spread spectrum code acquisition. We recognize the coming demand for short range transceivers and propose some enabling technologies including AFSR for IrDA and the TouchTag reader board. Then we examined the construction, operation and application of the TouchTag reader in depth.

The original Discrete Time (DT) AFSR was shown to entrain to a PN sequence. This mathematical system was then modified to improve its ability to entrain more quickly, more reliably, and in the presence of significant thermodynamic noise. The best form for the nonlinear function in AFSR was found to be a quadratic map, because of its short average acquisition time, small variance in acquisition time, and because of its ability to schedule the dependence on the received PN sequence, making use of its functional form to accelerate to lock as the confidence of successful acquisition increased. Inspired by the motions of a frog's leg, a method for exploiting gain clipping in hardware or a simple modification to the AFSR algorithm in software was proposed as a simple way to impose fixed points on the quadratic map thereby bounding the dynamics.

We proceeded to a more extensively model the AFSR system, examining how the average acquisition time varied with the register length of the shift register, with the variance of additive gaussian white noise in the channel, and the value of epsilon which determines the system's dependence during update on the incoming signal vs. its internal state.

A nomenclature for classifying the different AFSR systems discussed here was presented as a mnemonic. Four AFSR systems were proposed. The original discrete iterated map in continuous state space is called Continuous State Discrete Time (CS DT) AFSR. A software radio version of AFSR is proposed called Discrete State Discrete Time (DS DT) AFSR. A low frequency prototype AFSR system which is nearing completion with continuous dynamics but external clocking to force discrete transitions and some ADC discretization is called the Discrete State Continuous Time (DS CT w/ ext. clock) AFSR. Finally, a design for an rf implementation is dreamt of and found which truly operates in continuous time called the CS CT AFSR.

The key components in the hardware I have built are examined. They are FIFO buffer and bucket brigade delay, a y-axis symmetric nonlinear function constructed from a double balanced mixer, and a simple op amp gain. Analog switching is problematic but necessary in the design of the originally proposed DS CT (w/ ext clock) AFSR, but discarded in the new design for CS CT AFSR. Resilience of AFSR to discretization of the state space in a digital delay line or software radio implementation was modeled and confirmed. Also the algorithmic complexity of a software radio AFSR was shown to be approximately comparable to state of the art spread spectrum acquisition systems.

We then examined the clock recovery problem in more detail and finally resolved it lifting the DT AFSR iterative map into a higher dimensional flow to create CS CT AFSR. We use Euler's method to model this new system in autonomous mode for different values of the feedback inverting gain and the register length. The model performed as hoped. This system can be understood as a ring oscillator modified with a delay element to produce deterministic noise sequence from a nonlinear dynamic flow. Given the prospect of a working rf AFSR, the power consumption scaling is compared to DSP based acquisition systems.

11.2 Contributions

I have attempted to push each section in this thesis until some interesting nugget of wisdom was exposed. I can only hope that the rigor of the main body of the work will support what must in the end be speculations about an uncertain technological future. I now unsentimentally list the gems to reward the “reader” for skipping to the end without actually reading this document.

First we notice that with the impending demand for short range transceivers, there is a need for low cost low power channel sharing for these transceivers that is critical to many of their applications. The Human Touch Division Multiple Access (HTDMA) of TouchTags has its limited domain of application to this problem, but we think the general problem needs to be solved with direct sequence spread spectrum which quickly leads us to the problem of how to do acquisition and a proposed answer to this problem which is the AFSR system.

The AFSR system I have finally proposed is composed of only a few transistors and a delay element operating with continuous time dynamics. This continuous time dynamics was constructed by following a novel method for lifting a discrete time map into a continuous flow which may have other applications in the field of nonlinear dynamic computation. The dynamics are bounded by exploiting the clipping of the AFSR’s gain component.

Along the way, I had to address the problem of how to encode data and PN sequence in a channel so that AFSR acquisition would not be disturbed by the data bits. I propose to solve this problem for an IrDA AFSR by encoding the data with PCM and PSK encoding the PN sequence. In a radio system we would PSK encode the data and AM modulate the PN sequence as is current SS practice. This separation of data and spreading code onto orthogonal channels is a small price to pay considering that nearly every other SS acquisition system besides AFSR requires additional synchronization and control channels in addition to the PSK data and AM SS code. These systems also tend to perform only acquisition and then hand the rough phase over to a tracking loop which maintains tight loop synchronization. If fading or bursting noise destroys tracking, the entire process has to be repeated from the beginning of acquisition. In contrast, the AFSR dynamics seamlessly integrate acquisition and tracking into a single functionality. The AFSR acquisition time has a nearly gaussian distribution even in the presence of significant noise, so strong predictions can be made about system performance. In addition, the power consumption of AFSR is predicted to scale much better than conventional SS code acquisition schemes.

Finally, we make flamboyant proposals for future work including printed AFSR for tags and low cost addressing of power distribution, hybrid analog-digital signal processors incorporating nonlinear dynamics systems for performing special purpose computations, and the possibility that we may have billions of AFSR-like structures in our brains.

I hope that this work may make a contribution to the science of performing computations in novel ways and understanding computation as a powerful language for describing the natural world. I also hope that we can establish a scientific culture in which it is okay to discuss the ethics surrounding what we learn and make.

12 Acknowledgements

Thank you to Sal Mastroianni for helping me to have my questions answered at Motorola Company, and to Andrew Peterson for answering many of them. Thank you to Cris Moore for sharing his knowledge of nonlinear iterative maps and flows. Thank you to Goeff Grinstein for answering emails at 3am with great humor and of course for agreeing to read this thesis. Thank you to Joe Jacobson for agreeing to be my reader. Thank you to Rehmi Post and Matt Reynolds for being willing to answer questions about electronics shouted toward their work benches even in the face of loud and unfashionable music. Thank you to Bernd Schoner for sharing his knowledge of nonlinear time series modeling, and for his abiding love of theory even though he knows in his heart that it is completely useless. Also to Yael Maguire for knowing the answer to every question and always being willing to take the time to explain, for his Frame Maker thesis format, and for maintaining a large number of outlandish yet strangely optimistic opinions on all kinds of subjects, for his good karma, and for always doing his dishes. Thank you to my family for their love and support through all of my exploits. Deepest thanks to Neil Gershenfeld for being a true mentor. Thank you to the sponsors of Things That Think Consortium and the MIT Media Laboratory for supporting this research.

12.1 Appendix 1a: TouchTag Schematic

FIGURE 12.1.1 TouchTag Reader Schematic

12.2 Appendix 1b:TouchTag Reader Code

12.2.1 Code to read EM microelectronics v4100 RFID tags

```
//This is the simplest tag reader code to understand how to write effective code for the TouchTag reader.

//////////////////////////////////////////////////////////////////////////
////                TouchTag Electrostatic Tag reader/writer         ////
////                                                                  ////
////                    Reads data from v4100 RFID tag.              ////
////                                                                  ////
////                                                                  ////
//////////////////////////////////////////////////////////////////////////


#include <CTYPE.H>
//#include <STDIO.H>
#include <16F84.H>
#use delay(clock=8000000)

#use rs232(baud=19200, xmit=PIN_B2, rcv=PIN_B3)
#define ENABLE_PIN PIN_B4

#define DEBUG_PIN PIN_A1

#ifndef TAG_PIN
#define TAG_PIN  PIN_A2
#bit    TAG_PIN_BIT = 6.0
#endif

#defineLED_PIN PIN_A3
byte data[5];
byte parity[11];
int half_bit_length = 250; /* in us */
int trans_time = 30;

void
look_for_trans() {
    while(input(TAG_PIN)) ;       /* if it's high, wait for a low, if it's low. just go on */
    delay_us(trans_time);        /* account for fall time */
    while(!input(TAG_PIN));       /* wait for signal to go high */
    delay_us(trans_time);         /* account for rise time */
    /* end on a high */
}

void
look_for_translow() {
    while(!input(TAG_PIN)) ;       /* if it's low, wait for a high. if it's high, just go on */
    delay_us(trans_time);        /* account for rise time */
    while(input(TAG_PIN));       /* wait for signal to go low */
    //delay_us(trans_time);         /* account for fall time */
    /* end on a low */
}

boolean look_for_header() {
    byte i;
    boolean header;

    look_for_translow();
    delay_us(trans_time);


    /* check for 5 ones in a row*/
    header = TRUE;
    i = 1;
    while ((i<=9) && (header==TRUE)) {
    i = i+1;
    delay_us(half_bit_length);
    if (!input(TAG_PIN)) {
    delay_us(half_bit_length);
    //if (input(TAG_PIN)) {
    //just keep going (you will wait a half bit length at beginning of cycle)
    //}
    if (!input(TAG_PIN)){
    header = FALSE;
    }

    }
    else {
    header = FALSE;
    }
    }
    return(header);
}

read_tag_bits() {
    boolean tag_bit, parity_bit;
    byte i, j;
    for (j=0;j<=9;++j) {
    for (i=1; i<=4; ++i) {
    tag_bit = input(TAG_PIN);
    /* if we read a low, we know this is the */
    /* beginning of a one bit so put this in data */
    while(input(TAG_PIN) == tag_bit);
    shift_right(data, 5, !tag_bit);
    delay_us(half_bit_length);
    delay_us(trans_time);
    delay_us(trans_time);
    }

    parity_bit = input(TAG_PIN);
    while(input(TAG_PIN) == parity_bit);
    parity[j] = !parity_bit;
    delay_us(half_bit_length);
    delay_us(trans_time);
    //output_high(DEBUG_PIN);
    //delay_us(half_bit_length);
    //delay_us(half_bit_length);
    //output_low(DEBUG_PIN);
    }
}



byte calculate_column_parity_byte() {
    byte i, bit;
    byte mask = 1;
    byte par;
    byte temp = 0;
    for (bit=1; bit <= 8; ++bit) {
    par = 0;
    for (i=0; i<=3; ++i) {
    /* add up column number, bit, of data[0] thru data[3] */
    par += ((data[i] & mask)?1:0);
    }
    /* rightmost data[0-3] column parity will be rightmost in temp */
    shift_right(&temp, 1, (par & 1));
    shift_left(&mask, 1, 0);
    }
    return(temp);
}

/* only works for nibble from 0-9 */
boolean check_parity(nibble) {
    byte bit, tmp;
    byte mask;
    byte par = 0;


    if (!(nibble&1)) { /*if nibble is even*/
    mask = 1;
    tmp = nibble/2;
    }
    if (nibble&1) {/*if nibble is odd*/
    mask = 16;
    tmp = (nibble-1)/2;

    }
    for (bit=1; bit <= 4; ++bit) {
    par += ((data[tmp] & mask)?1:0);
    shift_left(&mask, 1, 0);
    }
    return((par&1)==parity[nibble]);
    //printf("nibble=%U ", nibble);
    //printf("calc=%U ", (par&1));
    //printf("read=%U ", parity[nibble]);
    //printf("\n\r");

}


main() {
    boolean header;
    boolean clean_bits;
    boolean parity_good;
    byte i;

    while (TRUE) {
    /* power up */
    output_high(LED_PIN);
    output_low(ENABLE_PIN);

    while(!look_for_header());
output_high(DEBUG_PIN);
    read_tag_bits();
output_low(DEBUG_PIN);

    parity_good = TRUE;
    for (i=0; i<=9; ++i) {
    if (!(check_parity(i))) {
    parity_good = FALSE;
    }
    }

    if (parity_good) {
    printf("%X", data[0]);
    printf("%X", data[1]);
    printf("%X", data[2]);
    printf("%X", data[3]);
    printf("%X", data[4]);
    printf("\r\n");
    }

    }
}
```

12.2.2 Code to write to v4050

```
//////////////////////////////////////////////////////////////////////////
////                      Electrostatic Tag reader/writer             ////
////                                                                  ////
////                         Write to v4050 RFID tags.                 ////
////                                                                  ////
////                                                                  ////
//////////////////////////////////////////////////////////////////////////

/* try writing to tags too. */

#include <CTYPE.H>
//#include <STDIO.H>
#include <16F84.H>
#use delay(clock=8000000)
#use rs232(baud=19200, xmit=PIN_B2, rcv=PIN_B3)
```

```
#ifndef TAG_PIN
#define TAG_PIN  PIN_A2
#bit    TAG_PIN_BIT = 6.0
#endif
#definePOW_PIN PIN_B4
#define DEBUG_PIN PIN_B1
#define LED_PIN PIN_A3

byte data[4];
byte parity[6];
int half_bit_length = 250; /* in us */
int trans_time = 30;

void
look_for_trans() {
    while(input(TAG_PIN)) ;       /* if it's high, wait for a low, if it's low. just go on */
    delay_us(trans_time);    /* account for fall time */
    while(!input(TAG_PIN));       /* wait for signal to go high */
    delay_us(trans_time);         /* account for rise time */
    /* end on a high */
}

void
look_for_transdown() {
    while(!input(TAG_PIN)) ;       /* if it's low, wait for a high, if it's high then just go on */
    delay_us(trans_time);    /* account for fall time */
    while(input(TAG_PIN));       /* wait for signal to go low */
    delay_us(trans_time);         /* account for rise time */
    /* end on a high */
}

/* look for transitions that are no less than half_bit_length apart */
/* if see a clean bit then return true */
/* if you see transitions too close together return FALSE */
boolean look_for_clean_bit() {
    byte i,j;
    boolean clean_bit;
    int num_checks;
    look_for_trans(); /* TAG_PIN is now high */
    /* for example if half_bit_length = 250us, wait for 10us and */
    /* check if transition occured already. do this num_checks times.*/
    num_checks = 10;
    for(i=1; i<= num_checks; ++i) {
    delay_us(5);
    if (input(TAG_PIN)) {
    clean_bit = TRUE;
    }
    if (!input(TAG_PIN)) {
    clean_bit = FALSE;
    }
    }
    while(input(TAG_PIN));      /* wait for signal to go low */
    delay_us(trans_time);       /* account for fall time */
    /* now check for clean low half of bit */
    for(i=1; i<= num_checks; ++i) {
    delay_us(5);
    if (input(!TAG_PIN)) {
    clean_bit = TRUE;
    }
    if (input(TAG_PIN)) {
    clean_bit = FALSE;
    }
    }
    return(clean_bit);
}

/* need to use absolute delays and not while's because */
/* auto thresholder on the tag board gets confused when you turn */
/* off the power altogether */
write_zero_bit() {
    int pow_cycle = 20;  /* 7/4 cycles at 125 KHz is about 13.7 us */
    /* the thing is, the PIC timing isn't perfect, it takes */
    /* some time for it to enter and exit the execution of each command */
    /* the tag needs a longer power on time to give it a chance to */
    /* to power up.  meanwhile, it doesn't really need a totally full */
    /* length power off time in order to notice an RM */
    /* so the fudge factor makes the power off shorter than the power on */
    /* while keeping a bit exactly 2*half_bit_length long. */
    int fudge = 35;
    /* ouput short power up burst to tag */
    output_high(POW_PIN);
    delay_us(pow_cycle);
    /* now turn off power entirely */
    output_low(POW_PIN);
    delay_us(half_bit_length - pow_cycle - fudge);
    output_high(POW_PIN);
    delay_us(half_bit_length + 5);
}

loop_write_zero_bit() {
    int pow_cycle = 20;  /* 7/4 cycles at 125 KHz is about 13.7 us */
    int fudge = 35;
    output_high(POW_PIN);
    delay_us(pow_cycle);
    output_low(POW_PIN);
    delay_us(half_bit_length - pow_cycle - fudge);
    output_high(POW_PIN);
    delay_us(half_bit_length + 5 - 15);
}

/* need to use absolute delays and not while's because */
/* auto thresholder on the tag board gets confused when a
/* a preceeding zero bit turns off the power altogether */
write_one_bit() {
    output_high(POW_PIN);
    delay_us(half_bit_length);
    //delay_us(trans_time);
    //while(input(TAG_PIN));
    delay_us(half_bit_length);
}

loop_write_one_bit() {
    output_high(POW_PIN);
    delay_us(half_bit_length);
    //delay_us(trans_time);
    //while(input(TAG_PIN));
    delay_us(half_bit_length - 15);
}

boolean look_for_liw() {
    byte i;
    boolean liw = FALSE;
    while(!liw) {
    look_for_transdown();
    /* signal is now low */
    delay_us(trans_time); /* account for fall time */
    liw = TRUE;
    /* check that signal stays low for 3 half_bit_lengths*/
    for(i=1;i<=3;++i) {
    delay_us(half_bit_length);
    if (input(TAG_PIN)) {
    liw = FALSE;
    }
    }
    }
    if (liw) {
    return(TRUE);
    }
    else {
    return(FALSE);
    }
}

read_tag_bits() {
    boolean tag_bit;
    byte i;
    for (i=1; i<=32; ++i) {
    tag_bit = input(TAG_PIN);
    /* if we read a low, we know this is the */
    /* beginning of a one bit so put this in tag_byte */
    while(input(TAG_PIN) == tag_bit);
    shift_right(data, 4, !tag_bit);
    delay_us(half_bit_length);
    delay_us(trans_time);
    }
}

setup_write_to_tag() {
    byte i;
    /* write two "0" bits to tag */
    write_zero_bit();
    write_zero_bit();
}

byte calculate_parity_bit(byte temp) {
    byte bit;
    byte mask = 1;
    byte par = 0;
    for (bit=1; bit <= 8; ++bit) {
    par += ((temp & mask)?1:0);
    shift_left(&mask, 1, 0);
    }
    return((par & 1));
}

byte calculate_column_parity_byte() {
    byte i, bit;
    byte mask = 1;
    byte par;
    byte temp = 0;
    for (bit=1; bit <= 8; ++bit) {
    par = 0;
    for (i=0; i<=3; ++i) {
    /* add up column number, bit, of data[0] thru data[3] */
    par += ((data[i] & mask)?1:0);
    }
    /* rightmost data[0-3] column parity will be rightmost in temp */
    shift_right(&temp, 1, (par & 1));
    shift_left(&mask, 1, 0);
    }
    return(temp);
}

void calculate_parities() {
    parity[0] = calculate_parity_bit(data[0]);
    parity[1] = calculate_parity_bit(data[1]);
    parity[2] = calculate_parity_bit(data[2]);
    parity[3] = calculate_parity_bit(data[3]);
    parity[4] = calculate_column_parity_byte();
    //printf("parity 0 = %X \n\r", parity[0]);
    //printf("parity 1 = %X \n\r", parity[1]);
    //printf("parity 2 = %X \n\r", parity[2]);
    //printf("parity 3 = %X \n\r", parity[3]);
    //printf("parity 4 = %X \n\r", parity[4]);
}

write_byte(byte temp) {
    byte mask = 128;
    byte bit;
    for (bit=0; bit<=7; ++bit) {
    if (temp & mask) {
    loop_write_one_bit();
    }
    else {
    loop_write_zero_bit();
    }
    shift_right(&mask, 1, 0);
    }
}

login_command() {
    write_byte(0x01);
    write_one_bit();/* parity bit */
}

write_password_command() {
    write_byte(0x11);
    write_zero_bit();/* parity bit */
}

write_word_command() {
    write_byte(0x12);
    write_zero_bit();/* parity bit */
}

selective_read_mode_command() {
    write_byte(0x0A);
    write_zero_bit();/* parity bit */
}

reset_command() {
    write_byte(0x80);
    write_one_bit();/* parity bit */
}

password() {
    write_byte(0x00);/* byte 1 of psswd */
    write_zero_bit();/* parity */
    write_byte(0x00);
    write_zero_bit();
    write_byte(0x00);
    write_zero_bit();
    write_byte(0x00);
    write_zero_bit();

    write_byte(0x00); /* column parity byte */
    write_zero_bit();/* last zero bit */
}

write_words(byte add) {
    byte i;
    write_word_command();
    write_byte(add);/* send 9-bit address */
    if(parity[5]) {/* send address parity   */
    write_one_bit();
    }
    else {
    write_zero_bit();
    }
    for (i=0; i<=3; ++i) {
    write_byte(data[i]);
    if(1 & parity[i]) {
    write_one_bit();
    }
    if(!(1 & parity[i])) {
    write_zero_bit();
    }
    }
    write_byte(0x00);
    write_zero_bit();
}

boolean look_for_ack_low() {
    boolean acklow = TRUE;
    delay_us(half_bit_length);/* check that signal stays */
    if (input(TAG_PIN)) {/* low for 3 half_bit_lengths*/
    acklow = FALSE;
    }
    delay_us(half_bit_length);
    if (input(TAG_PIN)) {
    acklow = FALSE;
    }
    delay_us(half_bit_length/2);
    if (input(TAG_PIN)) {
    acklow = FALSE;
    }
    return(acklow);
}

boolean look_for_ack() {
    boolean ack = TRUE;
    delay_us(half_bit_length); /* wait through 1st bit of hypothetical ACK */
    delay_us(half_bit_length);
    delay_us(trans_time); /* account for rise time, should be low now */

    ack = look_for_ack_low();
    while(input(TAG_PIN));/* wait for end of first low */
    delay_us(trans_time); /* account for rise time, should be high now */
    if (!input(TAG_PIN)) {/* now check for short high */
    ack = FALSE;
    }
    return(ack);
}

main() {
    boolean liw, liw2, ack, ack2;
    boolean clean_bits;
    byte i;
    byte address = 05;
    byte ID = 0xDD;
    for (i = 0; i <= 4; ++i) {
    data[i] = ID;
    }
    while (TRUE) {
//  for (address= 0x03; address <= 0x20; ++address) {
    printf("address = %X \n\r", address);
    /* power up tag */
    output_high(POW_PIN);
    output_low(DEBUG_PIN);
    /* loop until you see clean highs on 16 bits in a row */
    clean_bits = FALSE; //just to enter the loop
    while(!clean_bits) {
    clean_bits = TRUE;//innocent til proven guilty
    for(i=1;i<=16;++i) {
    if (!look_for_clean_bit()) {
    clean_bits = FALSE;
    }
    }
    }

    /* program tag */
    calculate_parities();
    parity[5] = calculate_parity_bit(address);
    look_for_liw();
    while(!input(TAG_PIN));/* wait for end of low */
    delay_us(3*trans_time);  /* account for rise time */
    setup_write_to_tag();/* send RM */
    write_words(address);
    output_high(POW_PIN);
    delay_us(half_bit_length);/* wait for tpp */
    delay_us(half_bit_length);
//  ack = look_for_ack();
output_high(DEBUG_PIN);
delay_us(50);
output_low(DEBUG_PIN);
//  if (ack) {
//  printf("SUCCESS writing address %u \n\r", address);
//  }
//  else {
//  printf("FAILURE writing address %u \n\r", address);
//  }


/* test tag to see if can go into receive mode */
//  reset_command();
//  delay_us(half_bit_length);
//  delay_us(half_bit_length);
//  delay_us(trans_time);  /* account for fall time */
//output_high(DEBUG_PIN);
//  ack = look_for_ack();
//output_low(DEBUG_PIN);
//  if (ack) {
//  printf("tag says: ACK!!!");
//  }
//  else {
//  printf("NO ACK!");
//  }

//  }
    }
}
```

Bibliography

[ISH97] Ishii, H. and Ullmer, B. *Tangible Bits: Towards Seamless Interfaces between People, Bits and Atoms.* Proceedings of CHI 1997, ACM Press, Atlanta GA, May 1997.

[CAM99] Campagnolo, M. L., Moore, C., Costa, J. F. *Iteration, Inequalities, and Differentiability in Analog Computers*. Submitted to the Journal of Complexity, 1999.

[CAR91] Carroll, T. L., Pecora, L. M. *Synchronizing Chaotic Circuits*. IEEE Transactions on Circuits and Systems, 38(4):453-56, 1991.

[CUO92] Cuomo, Kevin M., Oppenheim, Alan V., Isabelle, Steven H. *Spread Spectrum Modulation and Signal Masking Using Synchronized Chaotic Systems*. Research Laboratory of Electronics Technical Report No. 570, MIT, Cambridge, MA, February 1992.

[DIX94] Dixon, Robert, C. *Spread Spectrum Systems with Commercial Applications, 3rd Ed.* John Wiley and Sons, New York, 1994.

[FEH95] Feher, Kamilo. *Wireless Digital Communications, Modulation & Spread Spectrum Applications*. Prentice Hall, New Jersey. 1995.

[GER95] Gershenfeld, N. and Grinstein, G. *Entrainment and Communication with Dissipative Pseudorandom Dynamics*. Volume 74, Number 25, Physical Review Letters, The American Physical Society, 19 June 1995.

[GER98] Gershenfeld, Neil. *The Nature of Mathematical Modeling*. MIT Press, Cambridge, MA, 1998.

[GER98b] Gershenfeld, Neil. *When Things Start to Think*. Henry Holt and Company, New York, NY, 1998.

[GUC83] Guckenheimer, J. and Holmes, P. *Nonlinear Oscillations, Dynamical Systems, and Bifurcations of Vector Fields.* Springer-Verlag, New York, NY, 1983.

[HAR77] Harr, Milton. *Mechanics of Particulate Media: A Probabilistic Approach.* McGraw Hill, 1977.

[HEI52] Heidegger, M. *The Question Concerning Technology*. trans. William Lovitt. Harper and Row, New York, NY, 1952.

[FEY] Hutchings, E. (Ed.), Leighton, R., Feynman, R. P., Hibbs, A. *"Surely You're Joking, Mr. Feynman!": Adventures of a Curious Character.* W.W. Norton & Company, 1997.

[ULL97] Ishii, Hiroshi and Ullmer, Brygg. *Tangible Bits: Towards Seamless Interfaces between People, Bits and Atoms*. ACM, Conference on Human Factors in Computing Systems, Atlanta, Georgia, March, 1997.

[MOO99] Koiran, Pascal and Moore, Cristopher. Closed-form Analytic Maps in One and Two Dimensions Can Simulate Universal Turing Machines. Theoretical Computer Science 210, (Special Issue on Real Numbers) (1999) 217-223.

[LAK99] Lakdawala, Porus and Moore, Cris. *Queues, Stacks, and Transcendentality at the Transition to Chaos*. To appear in Physica D, 1999.

[MOO98] Moore, Cris. *Dynamical Recognizers: Real Time Language Recognition by Analog Computers*. Theoretical Computer Science 201 (1998) 99-136.

[MOO98b] Moore, Cris. *Finite-Dimensional Analog Computers: Flows, Maps and Recurrent Neural Networks*. First International Conference on Unconventional Models of Computation in Auckland, New Zealand, 1998

[MOT99] *Multiplexing and Channel Coding (FDD)*. 3rd Generation Partnership Project (3GPP), Technical Specification Group (TSG), Radio Access Network (RAN), Working Group 1 (WG1), TS 25.212 v2.0.0 and TS 25.213 v2.1.0 (1999-06).

[NIV97] Nivi, B. Passive Wearable Electrostatic Tags. Thesis for Master's Degree in Department of Electrical Engineering and Computer Science, MIT, Sptember, 1997.

[POO94] Poor, R. D. *Hyphos: A Self-Organizing, Wireless Network*. Thesis for Master of Science for Program in Media Arts and Sciences, School of Architecture and Planning, Massachusetts Institute of Technology. June 1997.

[POO99] Poor, R. D., *Guided Tour of the IRx Board*. Information available at www.media.mit.edu/~r/projects/picsem/irx2_1/

[POS97] Post, E. R., Orth, M. *Smart Fabric or Washable Computing.* Digest of Papers, pp. 167-168, First IEEE International Symposium on Wearable Computers, Cambridge, Massachusetts, October 13-14, 1997.

[POZ93] Pozar, David, M. *Microwave Engineering.* Addison-Wesley, Reading, MA, 1993.

[SIM94] Simon, Marvin, K., Omura, Jim, K., Scholtz, Robert A., Levitt, Barry K. *Spread Spectrum Communications Handbook*. McGraw-Hill, New York, 1994.

[SIP97] Sipser, Michael. *Introduction to the Theory of Computation*. PWS Publishing Co., Boston, MA, 1997.

[SLO91] Slotine, Jean-Jacques, E. and Li, Weiping. *Applied Nonlinear Control*. Prentice-Hall, New Jersey, 1991.

[STR94] Strogatz, Steven H. *Nonlinear Dynamics and Chaos.* Addison-Wesley, Reading, MA, 1994.

[VIG99] Vigoda, B., and Gershenfeld, N. *TouchTags: Using Touch to Retrieve Information Stored in a Physical Object. Extended Abstracts*, pp. 264-265, Proceedings of CHI 1999, ACM Press, Atlanta, GA, 1999.

[MOT97] Wideband CDMA concept group. *Wideband DS-CDMA Concept Group- W-CDMA System Description, Draft 0.1b*. ETSI SMG2, UMTS Alpha Group Meeting, Stockholm, Sweden, September 1997.

[ZIM96] Zimmerman, T. G. *Personal Area Networks: Near-field Intrabody Communication*. IBM Systems Journal, Vol 35, Nos 3&4, (1996) 1-9.

[ZIM95] Zimmerman, T., Smith, J.R., Paradiso, J. A., Allport, D., Gershenfeld, N. *Applying Electric Field Sensing to Human-Computer Interfaces*. Proceedings of CHI '95, ACM Press, Denver CO, May 1995. Available at www.acm.org/sigchi/chi95/Electronic/documnts/papers/tgz_bdy.htm