Digital Pulse Processing

by

Martin McCormick

B.Sc., University of Illinois at Urbana-Champaign (2010)

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Science in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2012

Author .....
Department of Electrical Engineering and Computer Science
July 19, 2012

Certified by .....
Alan V. Oppenheim
Ford Professor of Engineering
Thesis Supervisor

Accepted by .....
Professor Leslie A. Kolodziejski
Chair, Department Committee on Graduate Students

Digital Pulse Processing

by

Martin McCormick

Submitted to the Department of Electrical Engineering and Computer Science
on July 19, 2012, in partial fulfillment of the
requirements for the degree of
Master of Science in Electrical Engineering and Computer Science

Abstract

This thesis develops an exact approach for processing pulse signals from an integrate-and-fire system directly in the time-domain. Processing is deterministic and built from simple asynchronous finite-state machines that can perform general piecewise-linear operations. The pulses can then be converted back into an analog or fixed-point digital representation through a filter-based reconstruction. Integrate-and-fire is shown to be equivalent to the first-order sigma-delta modulation used in oversampled noise-shaping converters. The encoder circuits are well known and have simple construction using both current and next-generation technologies. Processing in the pulse-domain provides many benefits including: lower area and power consumption, error tolerance, signal serialization and simple conversion for mixed-signal applications. To study these systems, discrete-event simulation software and an FPGA hardware platform are developed. Many applications of pulse-processing are explored including filtering and signal processing, solving differential equations, optimization, the min-sum / Viterbi algorithm, and the decoding of low-density parity-check codes (LDPC). These applications often match the performance of ideal continuous-time analog systems but only require simple digital hardware.

Keywords: time-encoding, spike processing, neuromorphic engineering, bit-stream, delta-sigma, sigma-delta converters, binary-valued continuous-time, relaxation-oscillators.

Thesis Supervisor: Alan V. Oppenheim
Title: Ford Professor of Engineering

Contents

1 Introduction

Digital signal processing systems normally consist of a series of discrete-time operations applied to signal amplitudes represented by fixed or floating point binary codes. Signal representation is *digital* so that operations can be mapped to boolean functions and built out of switching circuits, while it is *discrete-time* so that storage and processing is possible with a serial von-Neumann architecture. Little exception is found even in fully-parallel ASIC signal processing systems. However, the processing of signals with digital hardware is not limited to this type of design. Instead, this thesis explores an alternative paradigm where analog signals are encoded as a series of digital pulses in continuous-time. These pulses are then processed accurately in the "pulse-domain" using special asynchronous finite-state machines before finally reconstituting an analog output signal.

The idea of pulse processing evokes the biologically-inspired neuromorphic computing systems pioneered by Carver Mead in the 1980's (e.g., [1]). However, what is explored here is different than these systems because most original neural systems forgo the use of exact timing structure in pulse signals. In fact, pulses are not even an explicit part of the processing in common analog VLSI implementations (where a surrogate analog signal indicates instantaneous rate). Later systems that actually employ pulses tend to use sophisticated biological spiking models but focus on increasing neurophysiological realism rather than understanding the significance for computation. For this reason, we break from being wed to biological models and return to a deterministic pulse encoding based on a straightforward generalization of the original integrate-and-fire model. Simple deterministic pulse encodings like this have seen renewed interest in light of the work of Aurel Lazar showing perfect reconstruction of an original analog signal from pulse times[2]. But while simple integrate-and-fire models receive renewed interested, methods for processing their pulses directly in a comprehensive way are not well understood.

A key contribution of this thesis is to reformulation integrate-and-fire into a digital signal processing framework so that a whole suite of pulse processing elements appear naturally. An important step is the recognition of a direct connection between integrate-and-fire and the bit-stream in sigma delta modulators. Sigma delta modulators are employed in sigma delta oversampled noise shaping converters and have contributed significantly to making transitions between the analog and digital domains inexpensive and accurate in modern DSP systems. Normally converters use digital filtering and decimation to convert the one-bit bit-stream of the internal modulator into a fixed-point (PCM) representation, however some work has shown that the bitstream itself is versatile for some classes of computation using simple circuits and without conversion to PCM[3, 4, 5, 6, 7]. Building on the work of these pioneers, this thesis takes the next step of removing the oversampling clock and processing asynchronously. This results in new processing elements that are both easier to analyze, and simpler to construct.

Only certain operations can be performed exactly with simple state machines in the pulse domain. However, there are distinct advantages when the necessary operations are available. Pulses are versatile in mixed-signal applications (demonstrated by the success of sigma delta in converters) and maintain some of the best features of both analog and digital. Pulses still possesses the immunity to noise-floor and transistor variation, modularity, and reuse of a digital format but exhibits the circuit simplicity, single-wire data path and power efficiency attributes of analog circuits. Removing the clock allows true continuous-time computation giving computational benefits when solving differential equations and certain optimization and inference problems. Asynchronous operation also reduces overall power requirements and electromagnetic interference from periodic clock waveforms. Pulses are also more robust to digital errors than a traditional PCM format. Pulse loss, insertion and jitter are the only types of error in a pulse signal and all of these errors impart small distortion on the encoded signal. Applications such as multimedia signal processing, user interfaces and prosthetics, sensor networks, and algorithms for optimization and inference that tolerate small error can have a very significant reduction in the design size.

The organization of the thesis is as follows. In chapter 2, we examine deterministic pulse signals and their connection to random point processes. While there are numerous ways to encode a continuous-time

waveform into a sequence of pulses, a specific generalization of integrate-and-fire is chosen because (1) it is a simple fundamental model (2) an encoder is readily built with basic electrical circuits (3) it admits accurate and robust operations directly in the pulse domain and (4) the original waveform can be reconstructed within fixed bounds only by filtering the pulses. These factors make the encoding desirable for practical pulse-domain processing tasks. The encoding is then generalized by using an equivalent model consisting of a cascade of an integrator, uniform quantizer, and differentiator. This provides both an intuitive model for integrate-and-fire and a new generalization that includes negative pulses. From the pulses of this system, a quantized version of the integral can be reconstructed with an up/down counter. While the quantization incurs a loss of information, it indicates bounded intervals that the encoded signal's integral must lie within. These bounds become a critical tool for analyzing approximate reconstruction and designing pulse-domain processing systems.

In chapter 3, continuous-time digital signal processing (CT-DSP) is introduced as an application for pulse processing. CT-DSP is similar to traditional discrete-time DSP but removes the sampling step and processes digital signals asynchronously. Yannis Tsividis has recently led the investigation into analysis and construction of CT-DSP circuits (see [8]). A number of benefits including the elimination of aliasing and reduction of power consumption are obtained this way. However, the completion logic and handshaking associated with asynchronous logic add complexity to the design especially when fixed-point (PCM) digital representations are used. Instead, by representing signals with digital pulses on single wires, many of the complications are eliminated. We suggest an aggressive simplification by using new pulse-domain circuits that leave out handshaking completely at the expense of occasional random errors. The integrate-and-fire encoding is found to be naturally robust to these errors for many types of processing. The operations for addition, constant multiplication and delay are derived and connections are made to early Δ-domain circuits for the same operations. For each operation, the change to the reconstruction bounds are shown. After the developed pulse-domain operations, a unique method for constructing pulse-domain filters is introduced. A key structure used to implement the filters is a counting network–a network of interconnected balancer components studied in theoretical computer science. Counting networks perform the counting operations in pulse-domain circuits in a distributed fashion, making them particularly robust and efficient. Finally, the general class of memoryless CT-$\Sigma\Delta$ operations are found. This is the set of all operations that can be performed using only an asynchronous finite state machine that passes or blocks pulses. It is shown that these are the positive homogeneous functions of order 1. In particular, this includes piecewise linear operations such as absolute value, min, and max. These functions greatly expand the type of applications possible in the pulse domain.

In chapter 4, the distortion of pulse signals is considered. Distortion is classified as either pulse loss and insertion or pulse timing jitter. The frequency-domain effects of these two types of errors on reconstructed signals are examined for pulse encodings. In addition, the effect of pulse timing jitter on the input to pulse processing circuits is considered. Extremely jittered inputs are modeled as Poisson processes and the asynchronous finite-state machines are interpreted as Markov chains. The state evolution of each is then a continuous-time Markov process from which a solution to the output rate is derived. It is found that many of the pulse-processing circuits still give good approximations of their operation despite the timing jitter.

In chapter 5, applications of pulse-process are built and analyzed. Simulations are performed using custom discrete-event simulator software and FPGA hardware. Applications include a pulse-domain filter bank, a quadratic program solver, the min-sum algorithm, and an LDPC decoder. The decoder, in particular, outperforms the state of the art in fully-parallel digital decoders under a number of metrics.

Finally, chapter 6 briefly examines circuits for implementing pulse modulation and processing. A precise connection between integrate-and-fire and the sigma delta modulator is made. In addition, a variety of potential pulse processing technologies are examined from CMOS to single-electron devices.

2 Pulses as a Signal Representation

While continuous functions and differential equations dominate the classical laws of nature–many fundamental phenomena are better modeled as a series of pulses. Raindrops, for example, produce a series of pulses that can be recorded by marking points in time. A simplified model measures each drop with an equal weight over an infinitesimal instant. The resulting pulse signal is then a temporal point process.

A point process is most often studied as a type of random process. In particular, rain is treated as a Poisson process. With a Poisson process of a given rate, knowing one drop's exact time is inconsequential in guessing the time of any other. The model for a Poisson process consists of a coin-flipping experiment where a heads indicates a pulse. An independent coin flip occurs every dt seconds and the coin is weighted to have a probability of heads $\lambda \cdot dt$. When dt is made infinitesimal, flips are rapid and heads occur at random instants in continuous time. The pulses form a Poisson process with an average rate of λ heads/second while the exact time between any two consecutive pulses follows a probability distribution, $P(\tau) = \lambda e^{-\lambda\tau}$.

In general, Poisson processes are non-homogeneous where the average rain rate–or probability of heads–changes in time. Here, rate is represented by a continuous function, $\lambda(t)$. Non-homogeneous Poisson processes include the incidence of photons on the retina ($\lambda(t)$ corresponds to a changing light intensity), clicks of a Geiger counter measuring radionuclide decay events ($\lambda(t)$ falls as the radionuclides diminish) or electrons passing across some cross-section in a wire ($\lambda(t)$ is the electric current).

In all of these examples, the point process is formed from the combination of many independent pulse-generating sources. Indeed, formally a Poisson process always arises from the superposition of many independent equilibrium renewal processes. If each source has a small rate and is measured without distinction, the merged pulses behave asymptotically like a Poisson process. This is an important theorem attributed to C. Palm and A. Khintchine[9] that explains it's widespread appearance in natural and man-made systems.

However not all pulse phenomena are Poisson processes or random; in many systems, pulses follow a deterministic process. For example, if rain collects together and spills off a ledge forming drips that fall to the ground, the pattern is regular–it is easy to predict the next drop from the last. If the rain falls faster, the drip rate follows but the regularity is maintained. Regular dripping is an example of a relaxation oscillator.

The simplest relaxation oscillator example is the dripping that comes from a leaky faucet. Water molecules slowly flow through a faucet valve and collect at the edge of the tap in a bead of water. The molecules accumulate until their collective weight exceeds the grip of surface tension ($N \approx 2 \times 10^{21}$ molecules), then the complete drop falls and the process begins anew. Here, the formation of pulses is different from the Poisson process. In fact, pulses are formed in the limit of decimation rather than merging. The relaxation oscillator can be modeled as decimation of random input pulses (water molecules flowing in) by a factor N where input pulses are counted and every Nth produces an output droplet. A state machine for this process is shown below.

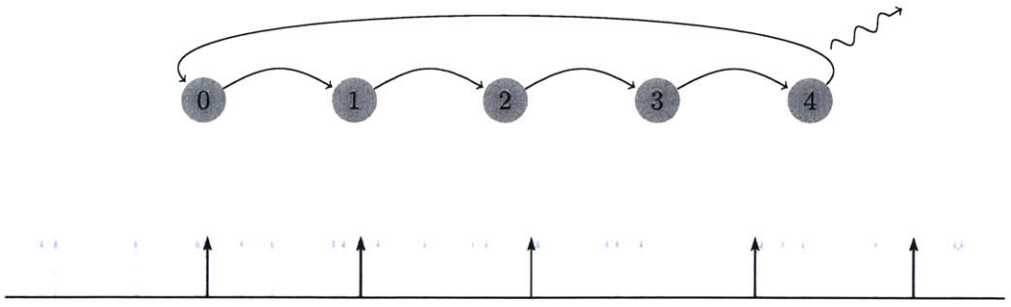


Figure 1: Decimation by 5: input pulses are counted and every fifth produces an output.

In the limit where both the input rate λ and the decimation factor N are large but the ratio is held constant, the output rate is $\mu = \frac{\lambda}{N}$ pulses/second with exactly $1/\mu$ seconds between consecutive pulses[1]. This type of relaxation oscillator has a variety of names from different fields of study. Perhaps the most well-known is "integrate-and-fire" after the model of neuron action potentials attributed to neurophysiology pioneer Louis Lapicque[10]. In the equivalent non-leaky integrate-and-fire model (I&F), a constant input rate signal $x(t) = \lambda$ is integrated until a threshold $\Delta = N \cdot dt$ is reached, at which point an output pulse is generated and the integrator is reset to zero. This process is illustrated in Figure 2.

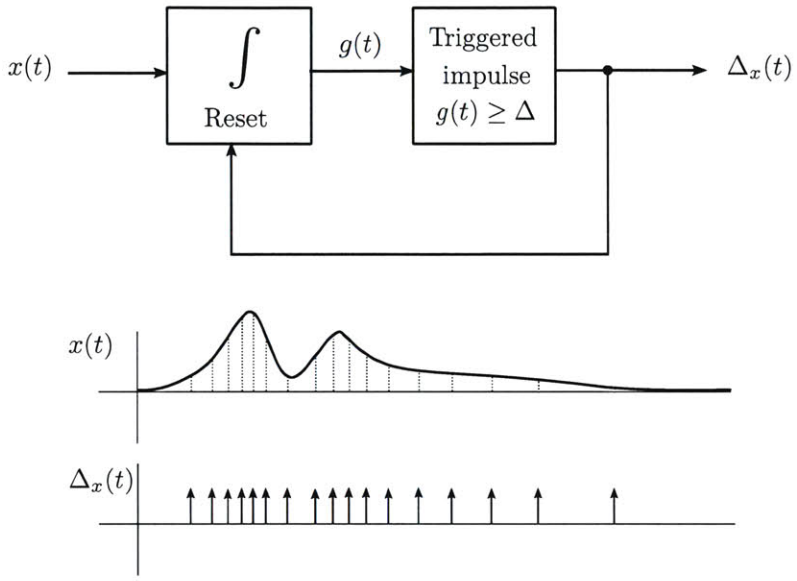
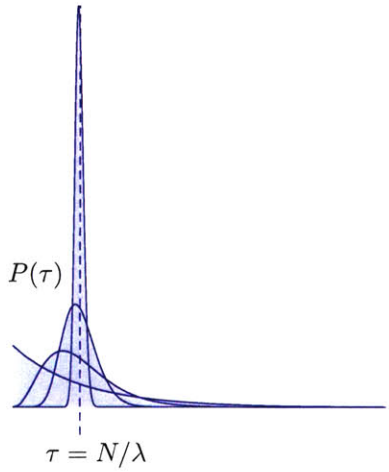


Figure 2: The non-leaky integrate-and-fire model (I&F).

In general, I&F can modulate any non-negative time-varying input $x(t)$. Similar to the Poisson Process, this is clearly a type of pulse-frequency modulation. However, while a Poisson process is inherently random, I&F pulses are formed deterministically from $x(t)$. In particular, the pulses are spaced such that the definite integral of $x(t)$ between consecutive pulse times is always exactly Δ. This integral is concentrated into a Dirac delta of measure Δ to represent an output pulse in $\Delta_x(t)$. The integral of $x(t)$ and $\Delta_x(t)$ are always close–I&F simply delays some of the integral mass by concentrating it at these specific points in time.



[1]Regular pulsing from decimation follows from the law of large numbers. If the original input pulse signal is Poisson at rate λ, the time between pulses τ is an exponentially distributed random variable with $P(\tau) = \lambda e^{-\lambda\tau}$. After decimating by a factor N, the time between pulses is $\tau \sim \text{Erlang}[N, \lambda]$. As N and λ are made large but $\frac{N}{\lambda}$ is held constant, $\tau \sim \mathcal{N}[\frac{N}{\lambda}, \frac{N}{\lambda^2}]$. In the limit, the variance drops to zero and the pulse spacing is concentrated at $\tau = N/\lambda$.

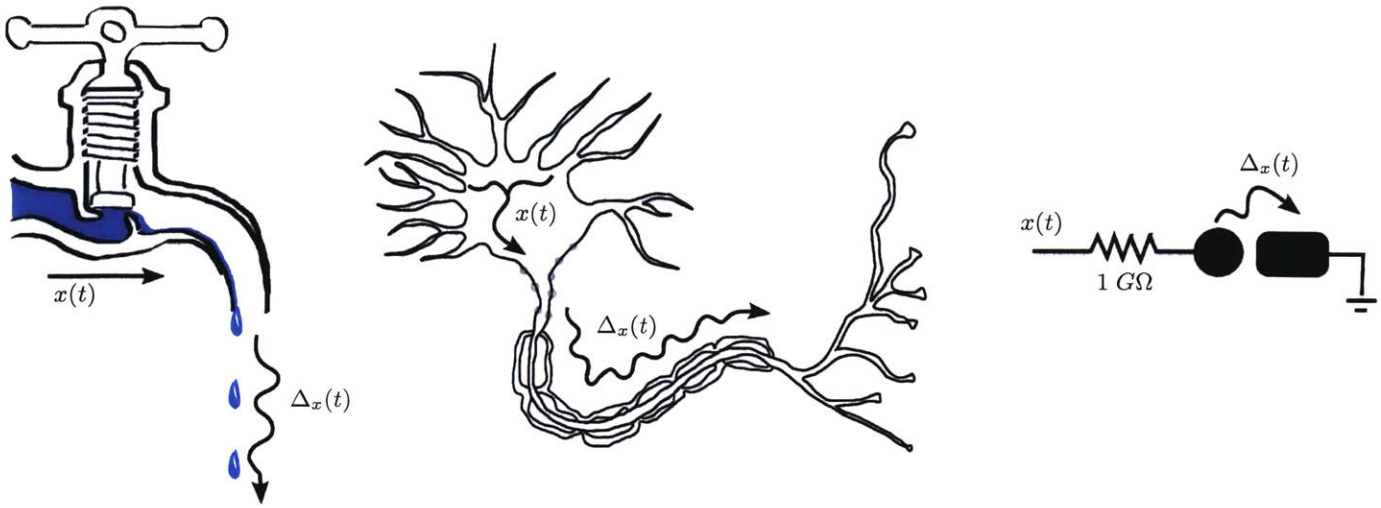


Figure 3: A dripping faucet, biological action potentials and the Coulomb blockade all exhibit integrate-and-fire.

I&F is an excellent model for a variety of natural and man-made systems spanning physiology to physics. The faucet is an I&F system where turning the handle slightly modulates the leak $x(t)$ and changes the drip rate. In this case, the drip sizes remain constant but the the time between drips changes depending on $x(t)$.

Lapicque's model of neuron behavior is also an important example[11]. In a neuron, there is a large difference in concentration of sodium ions (Na^+) and potassium ions (K^+) between the inside and outside of the cell wall maintained by sodium-potassium pumps powered by ATP. Different neurotransmitter stimulus at the dendrites of the neuron allow Na^+ to leak in at rate $x(t)$ and accumulate at a cell region called the axon hillock. These ions raise the voltage potential across the cell's membrane as charge integrates. In the axon hillock, there are many voltage-controlled ion channels that selectively allow Na^+ into the cell or K^+ out of the cell. Once a threshold of about $\Delta = 15\text{mV}$ is met (around 10^{10} ions), a runaway positive feedback loop of Na^+ channels opening permits further sodium influx and raise the voltage to 100mV. However within about 1ms, the K^+ channels open and the Na^+ channels close and the out-flux of K^+ lowers the voltage back down just as rapidly. The complete cycle is an action potential and forms a sharp voltage spike that is repeated down the axon to signal other neurons.

The vibration of the vocal chords (folds) in animals can also be modeled as integrate-and-fire[12]. As intraglottal pressure builds it eventually opens the vocal folds at a certain threshold causing a short burst of air before closing again. This repeats to form regular glottal pulses. The rate of pulses determine pitch.

Integrate-and-fire also occurs at the quantum level with single-electron tunneling. At low temperatures, the Coulomb blockade effect[13] occurs when current flows through a large resistance to an island and encounters a barrier to a lower potential. The barrier behaves as a capacitor that charges over time. After the island voltage exceeds the barrier, a single electron tunnels across. Further tunneling is then blocked until the island is refilled from fractional electron diffusion across the resistor. (Fractional diffusion is interpreted as the electron wavefunction evolving across the resistor). Very regular single-electron tunneling occurs at a rate proportional to the applied $x(t)$ voltage[14]. I&F is also found elsewhere in quantum systems, including photon anti-bunching observed from single-molecule sources[15].

This I&F process is also used in many classical electrical systems. Zero-crossings of frequency modulated (FM) signals and clocked Sigma-Delta ($\Sigma\Delta$) oversampled noise-shaping converters have a direct correspondence to I&F (see Chapter 6). These different viewpoints of the same phenomena are useful for understanding and building pulse modulation and processing elements.

2.1 Basic Pulse Processing

One of the key objectives of this thesis is to develop ways to process pulse signals directly. In order to develop an intuition for how pulse-processing elements might be constructed, we give a brief example for comparing the pulse rates of two signals. We see that the structure of I&F makes it more amenable to computation. For example, in Figure 4 two Poisson and two I&F signals are generate with constant rate inputs B and A. For I&F, determining the greater input from the pulses is simple: two consecutive pulses without interruption by another indicates that that pulse signal encodes a larger value. Conversely, with Poisson processes long-term averages are required to make even a probabilistic statement about relative rate. The "uninterrupted pulse" method of rate comparison for I&F pulses is particularly convenient because it

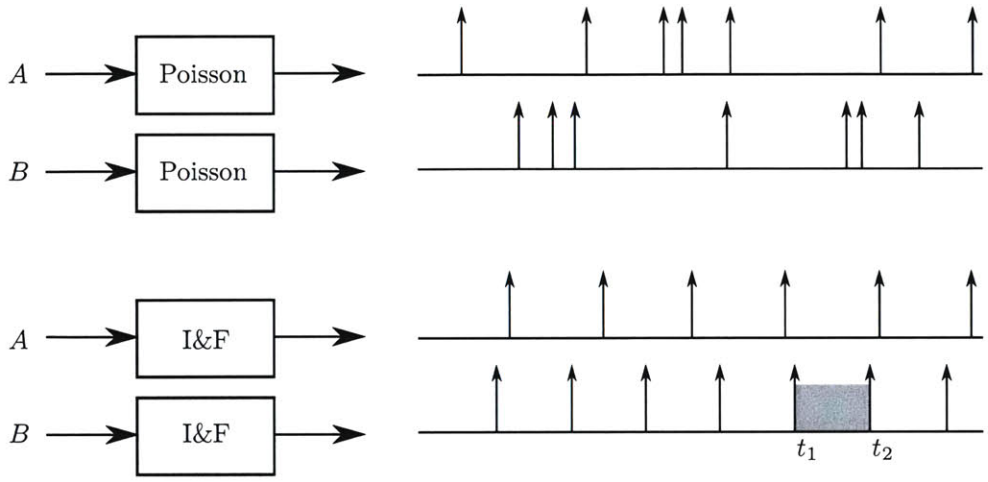


Figure 4: The regular structure of I&F assists in computation. Two consecutive (uninterrupted) pulses at t_1 and t_2 indicate that input $B > A$.

obviates the need for another time source. (while the time between pulses could be measured to determine that B has a shorter period, measuring time requires another more rapid pulse reference–a clock). Instead, detecting uninterrupted pulses only requires a single bit of state indicating which signal pulsed last.

Even when I&F inputs are time-varying, certain information about the encoded signal can be ascertained from the pulse times. Two consecutive pulses indicate that the definite integral of $x(t)$ is exactly Δ over the time interval that they enclose. If consecutive pulses are not interrupted by a pulse from another signal, the definite integral of the other encoded signal is less than Δ over the same time interval (otherwise it would have interrupted with a pulse). This implies that the signal average for one is higher than the other over the interval. This fact hints at a possible pulse-domain max operation for time-varying I&F inputs. Still, it is not immediately clear how this or general operations can be performed exactly in the pulse-domain or whether any guarantees can be made about their performance.

Before tackling the general processing problem, we first generalize I&F to support signed inputs ($x(t) < 0$) by adding an additional negative threshold to the modulator. The output $\Delta_x(t)$ then includes both positive and negative pulses. A formulation of signed I&F is easily interpreted from a digital signal processing perspective. This perspective ultimately leads to a method for constructing a broad class of pulse-processing elements.

2.2 A Signed Generalization of I&F

An equivalent form of I&F is found by noting that the process is quantizing the integral of $x(t)$, with a pulse emitted whenever the integral reaches a new multiple of Δ. Indeed, this is precisely a cascade of integration, uniform quantization and differentiation with respect to time. This process is shown in Figure 5.

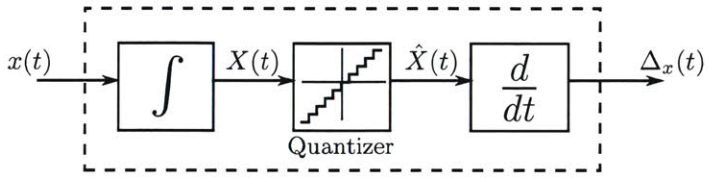


Figure 5: An equivalent representation of I&F, also called here continuous-time ΣΔ (CT-ΣΔ).

The output of the integrator $X(t)$ is a continuous function that may cross up or down through at most one quantizer level at any moment. The quantizer output is time-varying quantized function and the derivative of this function contains a Dirac delta whenever a change in level occurs. To be explicit, we define a function for each stage of the process. The output of the integrator is written as convolution by the Heaviside step function $u(t)$,

$$\begin{aligned} X(t) &= u(t) * x(t) \\ &= \int_{-\infty}^{t} x(t')dt' \end{aligned}$$

The uniform quantizer has levels spaced by Δ. Quantization applies the floor function to an input signal,

$$\begin{aligned} \hat{X}(t) &= Q(X(t)) \\ &= \Delta \lfloor \frac{X(t)}{\Delta} \rfloor \end{aligned}$$

The output of the quantizer $\hat{X}(t)$ transitions to a new value immediately when $X(t)$ enters a new quantization interval. The transition causes the differentiator to produce an output transient. Specifically, a Dirac delta of measure $+\Delta$ is produced when transitioning up a level and a Dirac delta of measure $-\Delta$ is produced when transitioning down a level.

$$\Delta_x(t) = \frac{d}{dt}\hat{X}(t)$$

While I&F normally only supports $x(t) \geq 0$, this formulation clearly extends to $x(t) < 0$. Negative pulses are simply emitted when the integral of $x(t)$ decreases through a multiple of Δ.

From $\Delta_x(t)$ one can reconstruct the quantized integral $\hat{X}(t)$ by integrating the pulses $\hat{X}(t) = u(t) * \Delta_x(t)$ (e.g., with an up/down counter). As before, the definite integral of $x(t)$ over the time period between two consecutive positive pulses is exactly $+\Delta$ (because the integral has increased by one quantization level). However, now there are additional combinations of pulse signs. The definite integral of $x(t)$ between two negative pulses is exactly $-\Delta$ (because the integral of $x(t)$ has decreased by one quantization level) and between pulses of opposite signs the definite integral is exactly 0 (because the integral of $x(t)$ has crossed the same level back again). In general, the quantized integral equals the original integral exactly (i.e., $X(t_0) = \hat{X}(t_0)$) at two times: immediately following positive pulses and immediately preceding negative pulses. Therefore, the definite integral of $x(t)$ over an interval between any two pulses can be computed from the integral of the pulse measures within (and bordering) the interval.

This generalization of I&F can be returned to a feedback form as before by only storing the relative position of the integrator state within a single quantization interval. To show this, the process of quantization is first written as subtraction of a gap signal $g(t)$,

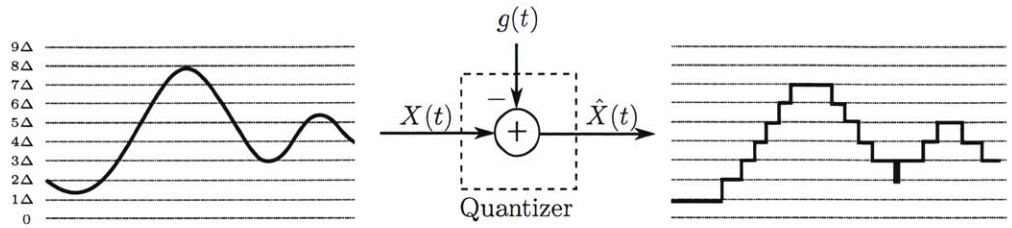


Figure 6: A continuous-time uniform quantizer represented by subtraction of a gap signal, $g(t) \in [0, \Delta)$.

$$\begin{aligned} g(t) &= X(t) - \Delta\lfloor\frac{X(t)}{\Delta}\rfloor \\ &= X(t) \mod \Delta \end{aligned}$$

The gap[1] $g(t)$ is the amount that $X(t)$ exceeds its quantized value–a quantity bounded by the interval $[0, \Delta)$. At any time, t_0, the future output of the I&F modulator depends only on the integrator state at that time, $X(t_0)$, and the future input $\{x(t) \mid t > t_0\}$. However, any offset of $n\Delta$, $n \in \mathbb{Z}$ is an equivalent integrator state value relative to the periodic levels of the quantizer. Therefore, subtracting an integrated count of the output pulses, $\hat{X}(t) = u(t) * \Delta_x(t)$ has no effect on the modulator operation. The resulting difference is the gap $g(t)$ and is all that must be stored to determine pulse times. Only two comparisons of $g(t)$ are needed–whether $g(t) \geq \Delta$ and whether $g(t) < 0$. When $g(t) = \Delta$, a positive output pulse is subtracted setting $g(t)$ to 0. When $g(t) = -\epsilon$, a negative output pulse is subtracted resulting in $g(t) = -\epsilon + \Delta$.

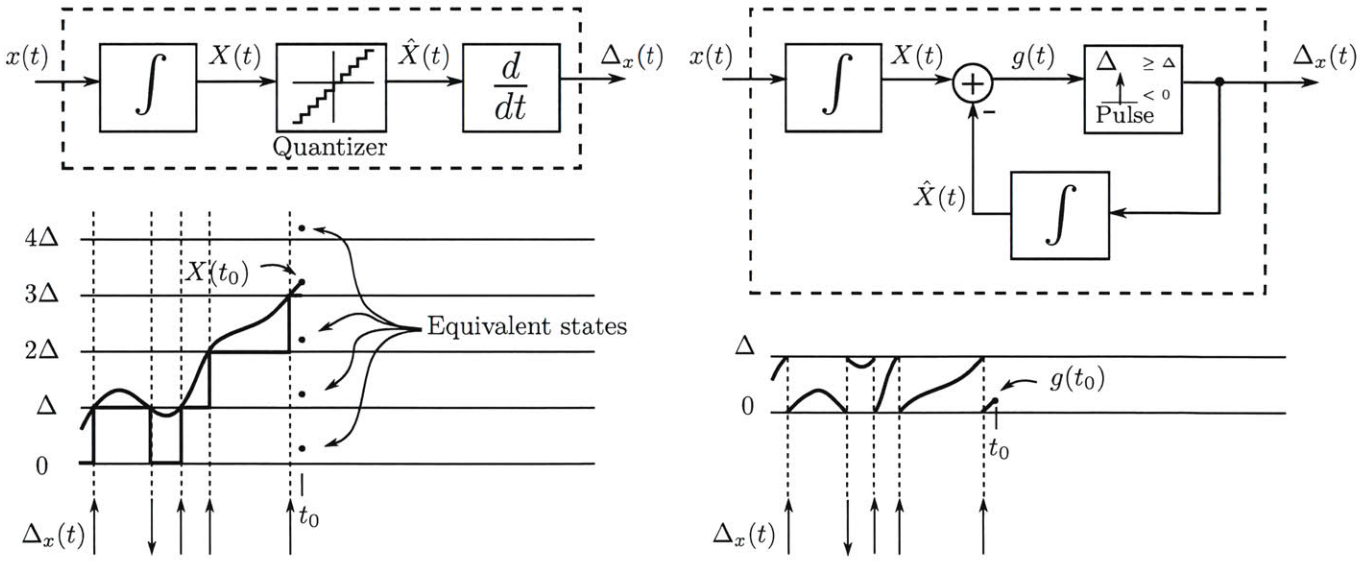


Figure 7: A feedback structure is formed by subtracting integrated output pulses. The difference $g(t)$ remains within $[0, \Delta)$.

[1]The negative gap, $-g(t)$, is commonly called the quantization error.

The feedback structure is simplified by moving the two integrators across the subtraction to form a single integrator. This integrator is reset to 0 when a positive feedback pulse is subtracted or set to $\Delta - \epsilon$ when a negative pulse is subtracted. The feedback loop is identical to the original I&F for inputs $x(t) \geq 0$.

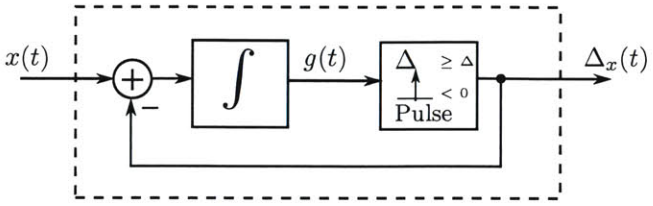


Figure 8: The signed I&F modulation using only a single integrator.

This generalized I&F modulation is the primary pulse encoding used throughout the thesis. The reader who is familiar with $\Sigma\Delta$ converters may recognize a similarity between I&F in Figure 8 and the first-order $\Sigma\Delta$ modulator inside converters. Indeed, there is a direct correspondence between the two. A $\Sigma\Delta$ modulator circuit can be used verbatim to perform the I&F modulation and only produces slight pulse timing jitter (aligning pulses to the clock period). An exact relationship between I&F and $\Sigma\Delta$ is examined in Chapter 6.

Also, some readers may recognize Figure 8 as similar to biphasic integrate-and-fire (e.g., [16]). The only difference is that biphasic I&F uses a lower threshold of $-\Delta$ instead of 0. This mitigates the problem of being sensitive to perturbation when $X(t)$ is close to a quantizer threshold (the modulator may generate a rapid series of pulses with alternating signs as it moves back and forth across the threshold). Biphasic I&F resolves this problem because the feedback pulses reset the integrator to 0 for both the $+\Delta$ threshold and the new $-\Delta$ threshold. Biphasic I&F skips a single output pulse (compared with the I&F described here) whenever the integral changes direction through a threshold (i.e., $x(t)$ changes sign). This is illustrated in Figure 9. Skipped pulses always alternate in sign and are self compensating. In practice, the distortion is negligible and a biphasic I&F circuit can be used for most processing despite the difference.

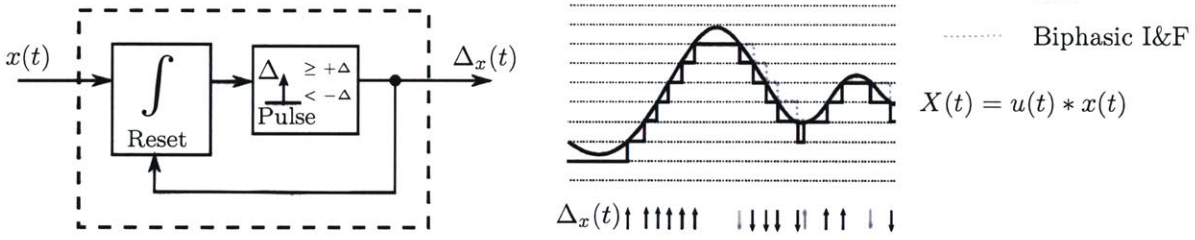


Figure 9: Biphasic I&F is a stable modulator that differs by skipping a pulse during the first of each sign change.

2.3 Methods for Reconstruction

The I&F modulation converts a signal, $x(t)$, into pulses indicating times when the signal's integral, $X(t)$ passes through integer multiples of a unit Δ. Based on these times and directions, can the original signal be reconstructed exactly? To answer this question, we first formally define the pulse signs and times,

$$t_m \in \{\tau \mid X(\tau) = \{\ldots, -2\Delta, -\Delta, 0, \Delta, 2\Delta, \ldots\}\}$$
$$v_m = \Delta \cdot \mathrm{sgn}(x(t_m)) \in \{+\Delta, -\Delta\}$$

The pulse signal is defined by a sum of weighted Dirac delta functions,

$$\Delta_x(t) = \sum_m v_m \cdot \delta(t - t_m)$$

A signal $x(t)$ is *consistent* with an I&F encoding $\Delta_x(t)$ if the following bound holds for all times t,

$$0 \leq u(t) * x(t) - u(t) * \Delta_x(t) < \Delta, \ \forall t \tag{1}$$

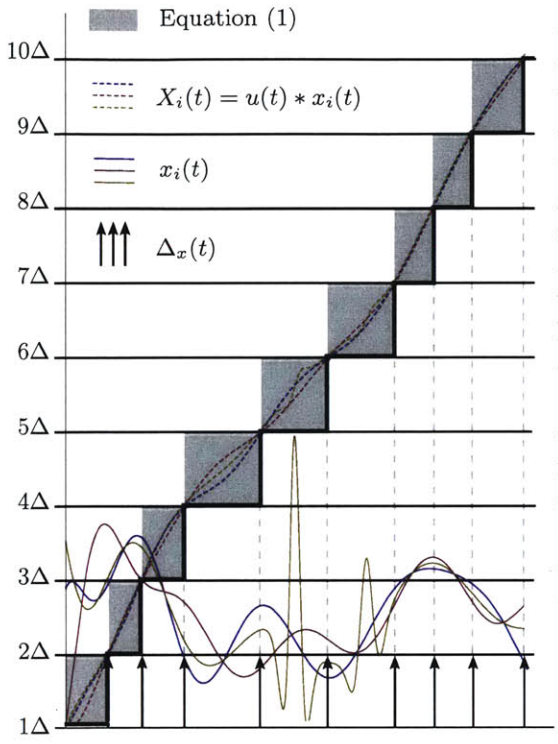


Figure 10: The three signals $x_1(t)$, $x_2(t)$ and $x_3(t)$ have identical $\Delta_x(t)$ but only one is bandlimited.

It is apparent that I&F modulation is a many-to-one mapping due to the information loss from quantization; many $x(t)$ signals are consistent with a given sequence of t_m and v_m. For example, Figure 10 shows three signals with integrals that lie within the bounds defined by a single $\Delta_x(t)$; all three have the same I&F encoding. Consequently, unique recovery of $x(t)$ from $\Delta_x(t)$ is not possible in general. However, if $x(t)$ has limited degrees of freedom, solving a system of equations may lead to an exact reconstruction. For example, for certain bandlimited $x(t)$, there is a system of equations that is well conditioned if the average number of pulses exceeds the Nyquist rate of $x(t)$. In our case, I&F is one of the general time-encoding machines studied by Aurel Lazar[2].

Following Lazar's work, recovery involves constructing a linear system of equations that capture the constraints on possible $x(t)$ imposed by the pulse times and signs of a given $\Delta_x(t)$. In the case of I&F, a subset of the constraints implied by Equation 1 are used. Specifically, we assert that the definite integral of $x(t)$ between two consecutive pulse times t_m and t_{m+1} is a specific value (either $-\Delta$, 0 or $+\Delta$ depending on the two pulse signs),

$$\begin{aligned} q_m &= \int_{t_m}^{t_{m+1}} x(t)dt \\ &= X(t_{m+1}) - X(t_m) \\ &= \begin{cases} +\Delta, & \text{if } v_m = +\Delta \text{ and } v_{m+1} = +\Delta \\ -\Delta, & \text{if } v_m = -\Delta \text{ and } v_{m+1} = -\Delta \\ 0, & \text{if } v_m = +\Delta \text{ and } v_{m+1} = -\Delta \\ 0, & \text{if } v_m = -\Delta \text{ and } v_{m+1} = +\Delta \end{cases} \\ &= (v_m + v_{m+1})/2 \end{aligned}$$

Now if $x(t)$ is bandlimited and has only finitely many non-zero samples, it can be written as,

$$x(t) = \sum_n c_n \cdot \phi(t - s_n)$$

where $s_n = nT$ are uniform sample times, c_n are a series of sample values and $\phi(t) = \sin(2\pi t/T)/(\pi t)$. Constraints are written as $\vec{q} = \mathbf{\Phi}\vec{c}$ where $q_m \in \{-\Delta, 0, +\Delta\}$ and $\Phi_{mn} = \int_{t_m}^{t_{m+1}} \phi(t - s_n)dt$. The elements of $\vec{c}$ can be solved by computing the pseudo-inverse of $\mathbf{\Phi}$ and evaluating $\vec{c} = \mathbf{\Phi}^+\vec{q}$. The solution is unique as long as $m \geq n$ and $\mathbf{\Phi}$ is full rank. The conditioning can be further improved if the s_n are chosen non-uniformly.

Unfortunately, there are some practical disadvantages with this reconstruction method. The pseudo-inverse is computationally expensive and calculating the matrix $\mathbf{\Phi}$ involves non-trivial definite integrals.

If the encoding is biphasic I&F, $q_m = v_{m+1}$.

Furthermore, reconstruction solutions are sensitive to pulse loss, spurious pulse gain or strong timing jitter. Even more importantly, this reconstruction method cannot accept a more general class of I&F pulse signals produced during pulse-domain processing. For these reasons, a simpler form of reconstruction is studied.

2.4 Filter-based Reconstruction

For the purposes of this thesis we propose a simpler (but approximate) I&F reconstruction that uses linear filtering. The pulses in $\Delta_x(t)$ are passed through a lowpass filter so that each pulse is replaced with the filter's impulse response to form a continuous output signal. While the reconstruction is inexact, for certain filters the results are very close and the output error can be bounded to a small interval in the time-domain. To see this, note that the pulse signal $\Delta_x(t)$ is a superposition of $x(t)$ with the gap signal's time-derivative $g'(t)$,

$$\begin{aligned}\Delta_x(t) &= \frac{d}{dt}(X(t) - g(t)) \\ &= x(t) - g'(t)\end{aligned}$$

The quantization gap $g(t)$ exhibits a broadband power spectral density for most inputs. After differentiation, $g'(t) = \frac{d}{dt}g(t)$ has a high frequency power spectral density. An example is pictured in Figure 11.

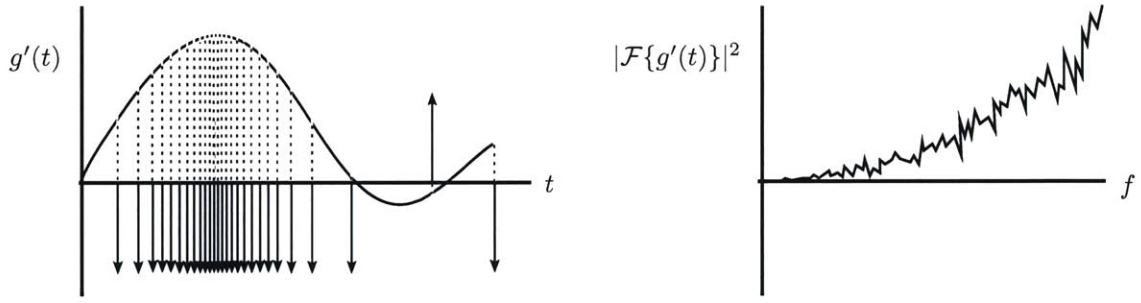


Figure 11: An example of $g'(t) = \frac{d}{dt}g(t)$ and its power spectral density.

Intuitively, applying a lowpass filter to $\Delta_x(t)$ removes much of $-\frac{d}{dt}g(t)$ and restores a lowpass filtered version of $x(t)$. Indeed, for certain filter impulse responses, filter-based reconstruction maintains a time-domain error below a fixed bound. To see this, note that the bounds on $g(t)$ imply,

$$\begin{aligned}u(t) * (x(t) - \Delta_x(t)) &= X(t) - \hat{X}(t) \\ &= g(t) \in [0, \Delta), \quad \forall\, t\end{aligned}$$

This directly implies a bound between the moving averages of $x(t)$ and $\Delta_x(t)$. The moving average is defined as convolution with a rectangular window of width W,

$$w(t) = \frac{u(t) - u(t - W)}{W}$$

The moving average of $x(t)$ and $\Delta_x(t)$ are labeled $x_w(t)$ and $\Delta_w(t)$ respectively,

$$x_w(t) = w(t) * x(t)$$
$$\Delta_w(t) = w(t) * \Delta_x(t)$$

The error between $x_w(t)$ and $\Delta_w(t)$ is,

$$\begin{aligned} e_w(t) &= x_w(t) - \Delta_w(t) \\ &= w(t) * g'(t) \\ &= \frac{g(t) - g(t-W)}{W} \in (-\frac{\Delta}{W}, \frac{\Delta}{W}), \quad \forall\, t,\ W \end{aligned}$$

This error is is also bounded. Therefore, the moving average of $\Delta_x(t)$ is always near the moving average of $x(t)$. The error between them has bounded infinity-norm,

$$\|e_w(t)\|_\infty \leq \frac{\Delta}{W}$$

The bound can be tightened by increasing the width of the filter W or decreasing the threshold Δ. In particular, as $W \to \infty$, the error goes to zero, implying the averages of $x(t)$ and $\Delta_x(t)$ are equal. The tradeoff with moving-average reconstruction is that for larger W the bound applies to an increasingly narrow-band version of $x(t)$ so that only the low frequency components of $x(t)$ are accurately recovered.

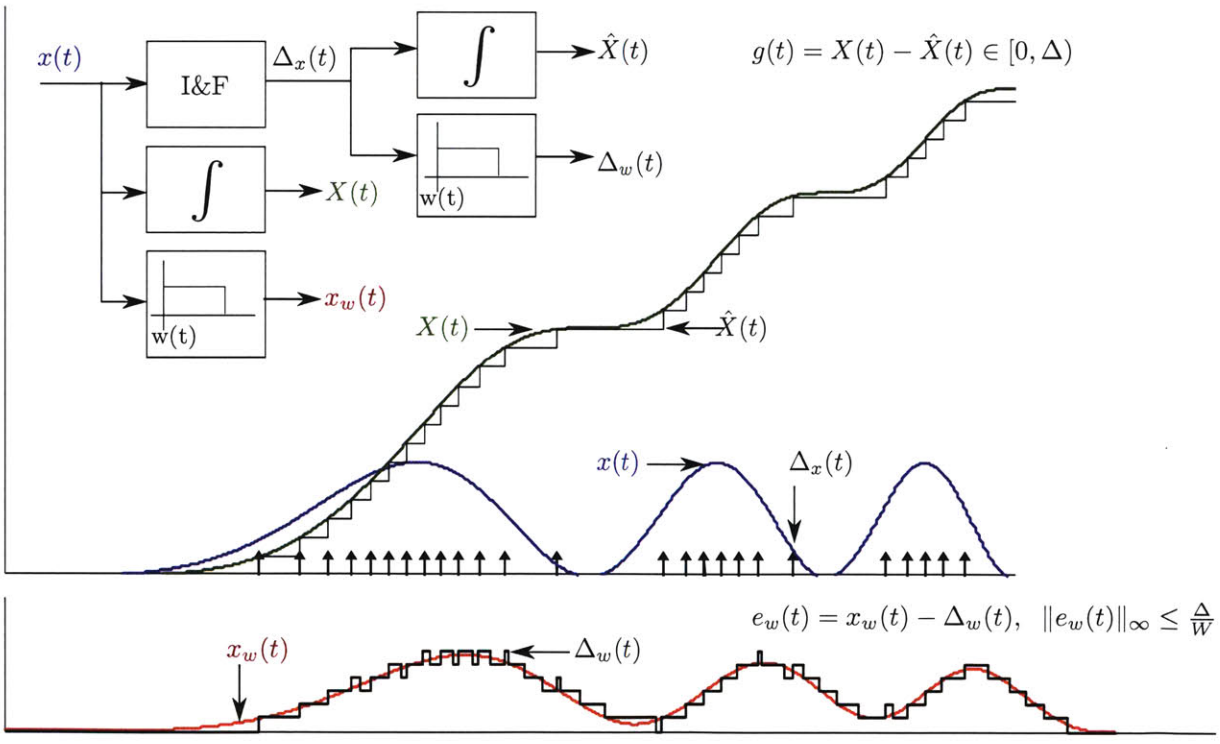


Figure 12: An example of I&F modulation and reconstruction using a moving-average filter.

Moving-average reconstruction is non-ideal due to the undesirable *sinc*-shaped frequency response of the filter. While the time-domain bounds are good, the output $\Delta_w(t)$ is blocky and still contains significant high-frequency error. Fortunately, additional filtering can improve the overall reconstruction frequency response

while maintaining error bounds in the time-domain. If an additional filter with impulse response $h(t)$ is applied,

$$\begin{aligned} x_h(t) &= h(t) * x_w(t) \\ \Delta_h(t) &= h(t) * \Delta_w(t) \end{aligned}$$

The error,

$$\begin{aligned} e_h(t) &= x_h(t) - \Delta_h(t) \\ &= h(t) * e_w(t) \end{aligned}$$

This new error is bounded as well. Convolution of any two absolutely integrable functions obeys Young's inequality,

$$\|h(t) * e_w(t)\|_\infty \leq \|h(t)\|_1 \, \|e_w(t)\|_\infty$$

Therefore,

$$\|e_h(t)\|_\infty \leq \frac{\Delta}{W} \|h(t)\|_1$$

So that $e_h(t)$ has the same bounds as $e_w(t)$ but scaled by the absolute-integral of $h(t)$.

One of the most straightforward reconstruction filters consists of repeatedly computing moving averages. We define $\beta^{(n)}(t)$, called a B-spline of degree n,

$$\begin{aligned} \beta^{(0)}(t) &= w(t) \\ \beta^{(1)}(t) &= w(t) * w(t) \\ &\vdots \\ \beta^{(n)}(t) &= w(t) * \beta^{(n-1)}(t) \end{aligned}$$

Because $\|\beta^{(n)}(t)\|_1 = 1$ for all the splines, reconstruction error from any of them has the same bound,

$$\|\beta^{(n)}(t) * (x(t) - \Delta_x(t))\|_\infty \leq \frac{\Delta}{W}, \quad \forall \, n, \, t, \, W$$

While the strict time-domain bounds remain the same for all orders n, the average error power tends to diminish as the order is increased. In general, higher-order B-splines have better frequency-domain characteristics that approach a Gaussian in time and frequency. An advantage of B-splines is that they are easy to implement in discrete-time. A decimation filter commonly employed by $\Sigma\Delta$ converters is the $sinc^3$ filter. This filter, named for its frequency response, is the B-spline of degree 2 and is implemented as a cascade of 3 counters (integrators) followed by 3 first-differences at sampling rates with progressively greater decimation. Besides B-splines, nearly any low-pass filter does a good job at reconstructing the input from pulses–even if a time-domain bound cannot be given directly.

Compared with the exact reconstruction methods, there are a number of advantages with the filter-based approach. When a filter is used for reconstruction, the I&F pulse encoding is robust to pulse loss or spurious pulse insertion because each pulse contributes the same small weight in the reconstruction. If a pulse is lost at a given time t_0, the error only affects the reconstructed signal when the filter window encloses t_0. As it passes by, the error no longer contributes and the original bounds still hold. In addition, the reconstruction is a linear time-invariant operation and hints that pulse-domain computation can be performed before reconstruction. For example, adding two pulse signals and then reconstructing the result with a filter is equivalent to adding the individual reconstructions due to the linearity of filtering.

2.5 Bounds after Pulse-domain Processing

When processing in the pulse domain, the effective gap $g(t)$ on a resulting pulse signal may exceed the original range $[0, \Delta)$. For example, adding two pulse signals causes the reconstruction's error bounds to add as well. To keep track of the range of values possible for the gap between a signal integral and its pulse encoding integral, a more general upper-bound variable Δ^U and lower-bound variable Δ^L are introduced. A pulse signal $\Delta_y(t)$ is said to satisfy the bounds $\{\Delta^L, \Delta^U\}$ as an I&F encoding of $y(t)$ if,

$$\underbrace{u(t) * (y(t) - \Delta_y(t))}_{g_y(t)} \in [-\Delta^L, \Delta^U), \quad \forall t \tag{2}$$

Likewise a signal $y(t)$ is said to be consistent with $\Delta_y(t)$ under bounds $\{\Delta^L, \Delta^U\}$ if (2) holds. When a pulse signal has these generalized gap bounds, the filter-based reconstruction $\Delta_w(t) = w(t) * \Delta_y(t)$ has accuracy,

$$y_w(t) - \Delta_w(t) \in \left(-\frac{\Delta^L + \Delta^U}{W}, \frac{\Delta^L + \Delta^U}{W}\right), \quad \forall t \tag{3}$$

The output of an I&F modulator with threshold Δ is a special case where the gap bound values are $\Delta^L = 0$ and $\Delta^U = \Delta$. These bounds $\{\Delta^L, \Delta^U\}$ will change as the pulse signal undergoes various operations. The bounds of a given operation's output are determined by the type of processing operations performed and the bounds of the input pulse signals. Certain operations will expand the gap while others will contract it. In this thesis, every time a new pulse-domain operation is defined the exact operation being performed along with the output bounds as a function of the input bounds are given in a boxed form. For example, the process of merging multiple pulse signals is stated as,

$$\boxed{y(t) = \sum_i x_i(t) \qquad \Delta_y^L = \sum_i \Delta_{x_i}^L \qquad \Delta_y^U = \sum_i \Delta_{x_i}^U}$$

It is particularly important to keep track of these bounds because input bounds to a pulse processing operation often determine the number of states in the state machine. In particular, general piecewise linear operations vary their design depending on the input bounds.

Occasionally, additional random noise or processing error may remove or insert a single pulse in $\Delta_y(t)$ that results in temporarily violating the bounds. These 'exception points' will not be included in $\{\Delta^L, \Delta^U\}$ directly because they are temporary, are often rare, and complicate the analysis. In any case, the potential for exception point errors from processing will always be identified. In the end, the actual reconstruction error is proportional to both the bounds and the number of spurious exception points in a given window.

3 Digital Signal Processing with Pulses

By interpreting the integrate-and-fire modulator as the cascade of an integrator, uniform quantizer, and a differentiator, many pulse-domain operations are easily derived by applying traditional DSP operations on the encoded integral. A key difference from DSP is that the operations occur immediately in continuous-time rather than at a regular sampling intervals. While this may at first appear to be dramatically different than discrete-time processing, the analysis of continuous-time DSP operations is straightforward. Before developing the pulse operations, a short discussion illustrating the differences between discrete-time and continuous-time DSP is given. The following explanation is based heavily on the work of Yannis Tsividis [8] who has pioneered the modern study of continuous-time digital signal processing (CT-DSP) systems. After this review, pulse-domain operations for addition, constant multiplication and delay operations are developed, followed by a general set of piecewise linear operations.

3.1 Discrete-time DSP (DT-DSP)

Traditional implementations of digital signal processing in fully-parallel ASICs are synchronous. This is because of the way boolean functions are implemented with switches. The gates and wires in a circuit have finite delay so that when an input to the boolean function changes, different bits in the resulting output code may take varied amounts of time to transition to their new value. The meaningless intermediate codes are glitches that are passed on to the next circuit. If there is ever a feedback loop, these glitches will accumulate so that digital signals become meaningless. The consequence of synchronous design methodologies for signal processing systems is the need for discretization of the signal in time through sampling.

Normally, a fixed-point discrete-time DSP system contains three stages: A/D conversion, synchronous digital processing and D/A conversion. We will consider the mathematical formation of the three stages and then investigate how the removal of sampling modifies the process. A/D conversion consists of quantization and sampling. As before, if we assume a uniform quantizer with levels spaced by Δ, quantization consists of applying the following function to an input signal,

$$\begin{aligned}\hat{x}(t) &= Q(x(t)) \\ &= \Delta \lfloor \frac{x(t)}{\Delta} \rfloor\end{aligned}$$

Again, the difference between the quantized and unquantized signal is called the gap,

$$g_x(t) = x(t) - \Delta \lfloor \frac{x(t)}{\Delta} \rfloor$$

The quantization function is a memoryless nonlinearity so that the frequency spectrum of $g_x(t)$ has broadband frequency components. For example, if $x(t)$ is a single sinusoid, $g_x(t)$ contains harmonics at integer multiples of the frequency.

While the output of the quantizer may transition at any instant, the sampling step only measures new values at sample times. Sampling is equivalent to multiplication of the signal by a regular impulse train $s(t)$ with sampling period T seconds,

$$s(t) = \sum_{n=-\infty}^{\infty} \delta(t - n\,T)$$

Multiplication by $s(t)$ only captures the quantized signal at multiples of the sampling period T. The values at these times can be represented by $\hat{x}[n] = \hat{x}(nT)$. Multiplication by $s(t)$ results in,

$$\begin{aligned}x_s(t) &= \hat{x}(t) \cdot s(t) \\ &= \sum_{n=-\infty}^{\infty} \hat{x}[n] \cdot \delta(t - n\,T)\end{aligned}$$

The weighted periodic pulse train $x_s(t)$ is entirely described by the quantized pulse weights $\hat{x}[n]$ that can be represented as a sequence of binary codes and processed synchronously by a DSP.

A frequency-domain representation of sampling indicates that periodicity is introduced in the signal's frequency components. The Fourier transform of a periodic pulse train $s(t)$ is also a periodic pulse train,

$$S(f) = \frac{1}{T} \sum_{m=-\infty}^{\infty} \delta(f - \frac{m}{T})$$

The period of this pulse train is the sampling frequency, $\frac{1}{T}$. Multiplying by $s(t)$ produces a signal with Fourier-transform consisting of a superposition of shifted copies of the original signal's Fourier transform, i.e. aliasing,

$$\hat{X}_S(f) = \frac{1}{T} \sum_{m=-\infty}^{\infty} \hat{X}(f - m\frac{1}{T})$$

If $\hat{x}(t)$ is band-limited to half of the sampling frequency, these copies will not overlap and $\hat{X}_S(f) = \hat{X}(f)$ over an interval $f \in [-\frac{1}{2T}, \frac{1}{2T}]$. For this reason, most systems include an analog anti-aliasing filter at the beginning of the conversion process to remove noise and other interference at higher frequencies. However, despite an anti-aliasing filter, the subsequent quantizer non-linearity always adds an infinite-bandwidth signal $g_x(t)$. After sampling, all of the broadband components of $g_x(t)$ alias into the interval $f \in [-\frac{1}{2T}, \frac{1}{2T}]$. The compounded power spectrum is fairly flat over the interval and is often approximated as white[17]. The quantization error power can be reduced by decreasing the quantization interval Δ or oversampling the input signal. Oversampling spreads the aliased quantization spectrum thinner over a larger bandwidth and allows more to be removed by filtering. For example, if $g_x[n]$ is assumed white, doubling the sampling rate allows a low-pass filter to reduce in-band SQNR by about $6dB$.

Once a sequence of codes $\hat{x}[n]$ is available from A/D conversion, a synchronous digital signal processor can manipulate them arbitrarily. An important class of DSP computation is linear time-invariant (LTI) filtering–convolving the input sequence of numbers $\hat{x}[n]$ by some desired impulse response sequence $h[n]$ to form $y[n] = \hat{x}[n] * h[n]$. The resulting sequence $y[x]$ is then converted back to a periodic output pulse train. This process is exactly equivalent to continuous-time convolution of the input pulse train $\hat{x}_s(t)$ by a fixed periodic pulse train with programmable weights,

$$h_s(t) = \sum_n h[n] \cdot \delta(t - nT)$$

The periodic pulse signal $h_s(t)$ is the response of the DT-DSP to a single input pulse $\delta(t)$. By linearity and time-invariance of the processing, the response to an input $x_s(t)$ is,

$$\begin{aligned} y_s(t) &= \sum_n y[n] \cdot \delta(t - nT) \\ &= x_s(t) * h_s(t) \end{aligned}$$

The digital LTI filter effectively applies a filter with impulse response $h_s(t)$. The effective frequency response of the DT-DSP is then,

$$H_s(f) = \sum_n h[n] \cdot e^{-j2\pi fnT}$$

This frequency response is periodic in the clock frequency $\frac{1}{T}$. With the selection of weights $h[n]$, the DT-DSP chain can give any desired frequency response over the interval $f \in [-\frac{1}{2T}, \frac{1}{2T}]$. The signal $y_s(t)$ is then passed through an analog anti-imaging filter with bandwidth $\frac{1}{2T}$. The complete process is illustrated in Figure 13.

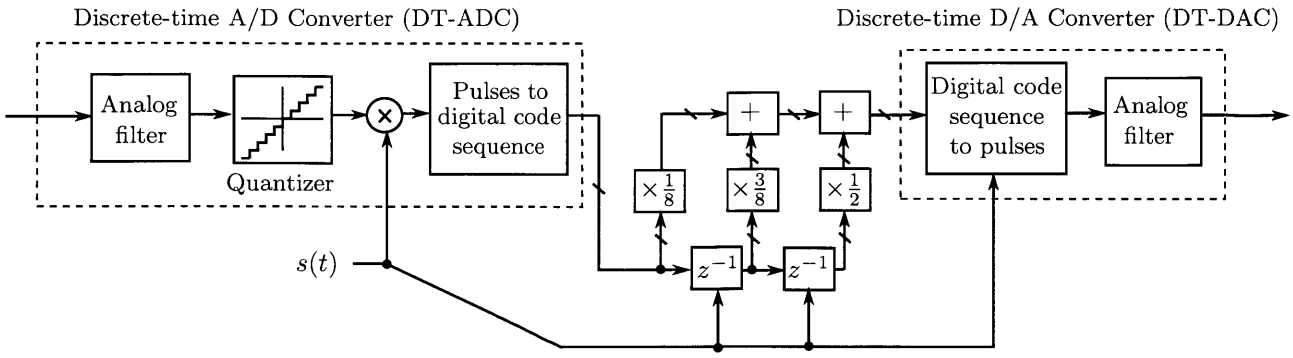


Figure 13: A depiction of discrete-time digital filtering with $h[n] = \{\frac{1}{8}, \frac{3}{8}, \frac{1}{2}\}$.

3.2 Continuous-time DSP (CT-DSP)

While not usually considered, the sampling process in DSP can be eliminated and a quantized input signal can be processed in continuous-time. Continuous-time digital signal processing (CT-DSP) is a name introduced by Yannis Tsividis for this process. Ostensibly this involves a quantized signal represented by codes that change value at arbitrary times. When the A/D quantizer changes levels, the new code is immediately processed asynchronously by the signal processing system. This processing is indeed feasible in a fully-parallel ASIC if the issue of erroneous glitches can be resolved.

In an idealized CT-DSP model, everything operates like a typical discrete-time processing circuit except that addition and constant multiplication circuits update their digital output codes immediately in response to a changing digital input code. Changes in codes do not suffer from glitches or timing variation between bits. A controlled digital delay of an entire code is also used to build LTI filters. If these controlled delays are all a fixed time T, the CT-DSP impulse response can be exactly the same as a DT-DSP. However a CT-DSP can also be generalized to include arbitrary delays. Now, the response may be a general aperiodic pulse signal,

$$h_{CT}(t) = \sum_n h_n \cdot \delta(t - t_n)$$

Both the weights h_n and the delays t_n can be chosen arbitrarily. The resulting frequency response $H_{CT}(f)$ is no longer periodic in the clock frequency,

$$H_{CT}(f) = \sum_n h_n \cdot e^{-j2\pi f t_n}$$

The extra freedom with the t_n's allows more flexibility in the filter design. Filters can be constructed with fewer taps and can give a desired response over any band with the proper selection of delays.

In principle, a CT-DSP can be constructed using asynchronous circuits. Asynchronous digital design has remained less popular than clocked approaches but has found success in certain applications where benefits outweigh the increased design complexity. Quasi-delay insensitive (QDI) circuits are relatively common and make only slight timing assumptions. In such circuits, typically two or four phase handshakes and completion logic are used where modules assert their readiness to accept or send data, and acknowledge full reception of the data. This introduces extra delay for messages to be sent back and forth and also excess circuitry–often doubling the number of transistors in a design[18]. However, this can still be worthwhile for the benefits of asynchronous operation. The removal of a clock can reduce power consumption by about 70%. A CT-DSP system only performs switching when the quantized input changes, making the dynamic power consumption low and a function of the input signal. Without discretization of time, a CT-DSP also generates less in-band quantizer distortion and eliminates the disadvantages of sampling.

Yannis Tsividis et. al. have recently led the investigation into CT-DSP systems. Authors have proposed an A/D converter that maps an analog input into a continuous-time digital representation and processes the information without a clock. Computation is done with clever asynchronous circuit techniques and handshaking between modules. For example, a continuous-time FIR and wave-digital filters using tunable continuous-time digital delay elements was demonstrated in papers by Schell, Bruckman and Li [8, 19, 20, 21]. Observed benefits of CT-DSP include,

- Reduced in-band quantization noise and alias-free nonlinear signal processing.
- Circuit power consumption that depends on the activity of the signal.
- Extra degrees of freedom in delay values that can be optimized to give more efficient filters than discrete-time counterparts.
- Fast response to input variations without needing to wait for a clock edge.

These benefits are all desirable, but CT-DSP techniques have still not become widely employed. One concern with CT-DSP designs is the complexity of the continuous-time A/D and D/A converters, handshaking overhead and continuous-time digital delay elements. This excess complexity increases the design area and power requirements. Indeed, the overall power consumption of the design in [19] is comparable to low power general-purpose DSPs from Texas Instruments with more functionality. Much of the complexity stems from a need to protect from the detrimental effect of digital errors. If a bit is flipped in the data path, it generally produces a large or even unbounded distortion in the processed output. This necessitates careful handshaking and arbitration circuits. An alternative is to represent digital signals differently than in traditional DSP using pulse encodings. WIth the I&F encoding an even more radical approach to CT-DSP is possible where handshaking is completely eliminated and the conversion process is simplified.

3.3 CT-DSP with Deltas

The basic DSP operations for I&F pulse processing are developed directly from continuous-time Delta-modulation (CT-Δ) processing. CT-Δ modulation simply indicates positive or negative changes in the output of a uniform quantizer. A quantized input signal takes a step up or down by Δ whenever the input enters a new quantization interval. The derivative of the quantizer output is a train of Dirac deltas–a positive Dirac delta, $\Delta\delta(t)$, when the quantizer input crosses upward to a new level and a negative Dirac delta, $-\Delta\delta(t)$, when it crosses downward to a level. The fundamentals of this process were first described in [22] and [23]. In CT-DSP, these deltas completely encode the evolution of the quantized signal. Positive/negative deltas can be conveyed in the system by digital pulses or binary toggles on two separate lines. Further, reconstruction from deltas consists of only an asynchronous counter. By incrementing for each positive delta and decrementing for each negative delta, the full binary value of the quantized signal is maintained.

As a mental model, operations performed in the Δ-domain can be developed directly from equivalent fixed-point operations. In general, each operation can be implemented trivially in three steps. (1) input deltas are accumulated with an up/down counter (2) the desired fixed-point operation is computed asynchronously on the counter value code and (3) output pulses are emitted when the computed value transitions up or down by Δ. This conceptual approach trivially performs the desired Δ-domain, however, it will be shown in the next section that these three steps combine and reduce significantly for each of the basic LTI operations: addition, constant multiplication and delay.

A simplified depiction of the overall process is shown in Figure 14. CT-Δ modulation produces digital pulses indicating upward or downward level crossings of the input signal. These pulses are processed asynchronously by special digital modules, perhaps each performing the three-step process. After processing, an up/down counter reconstitutes a binary coded signal. The evolution of the code defines a continuous-time quantized output signal. Lastly, an optional filter removes out-of-band quantization error (but not images).

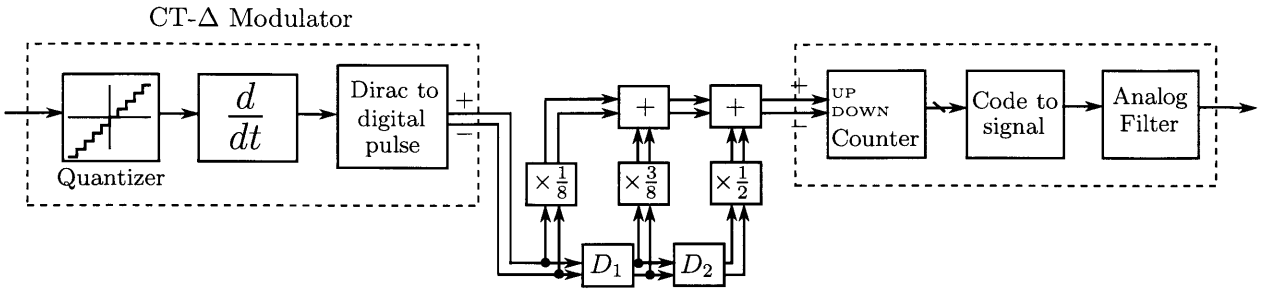


Figure 14: CT-Δ overview. Digital pulses are passed on two wires–one for either sign of Dirac delta–and processed in continuous-time by special pulse-domain processing elements.

There are some disadvantages to the encoding. With CT-Δ, any loss of a delta causes the final reconstruction counter to be off in value from that point on. This generate a large DC offset and all the errors accumulate over time. If low-frequency noise is unacceptable, pulse collision must be avoided at any cost.

A surprisingly simple but effective alternative is to move the counter from the final reconstruction to the very beginning of the system, as an input integrator. The circuits for adding, multiplication and delaying stay the same–it is merely CT-Δ-domain processing on the integral of the original signal instead. During the final reconstruction phase, the pulses can be low-pass filtered directly without a counter as the signal was already "pre-integrated". This new variation is shown in Figure 15.

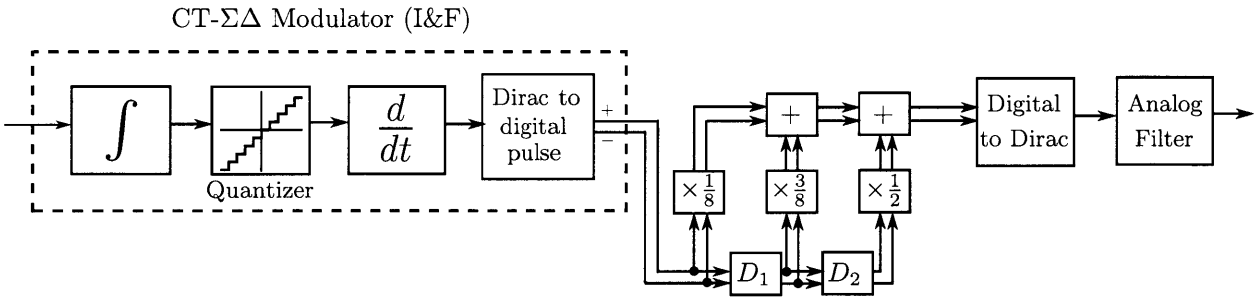


Figure 15: I&F modulation is formed by preceding the Δ modulator with an integrator. Pre-integrating allows for the removal of the up/down counter in final reconstruction while giving the added benefit of noise-shaping.

One immediate advantage is that errors (e.g. from discarded pulses in collisions) will no longer accumulate in a reconstruction counter; the output distortion from any transient pulse loss or gain is minimal. For the same reason, initialization of counters and other state in the processing elements is not as critical as with CT-Δ processing; initial state has a transient effect on the reconstruction output. Benefit extends to all processing error, including multiplier roundoff. The error component of each processing element is differentiated and the final low-pass filter removes significantly more error power.

3.4 Linear Time-invariant Processing

The three linear time-invariant (LTI) operations–addition, constant multiplication, and delay–are straightforward in the CT-Δ and I&F pulse domain. Because these operations are LTI, CT-Δ and I&F processing elements are identical. Modules that perform these operations take in positive or negative pulses and output

positive or negative pulses. Processing may modify the pulse times and signs so that they encode a different signal representing a processed waveform.

3.4.1 Addition

A pulse-domain adder takes in two input pulse signals and produces one output pulse signal. As with any Δ-domain operation, addition consists didactically, of two up/down counters that reconstitute two binary codes that are then added followed by differentiation to convert back to output pulses. Here it is apparent

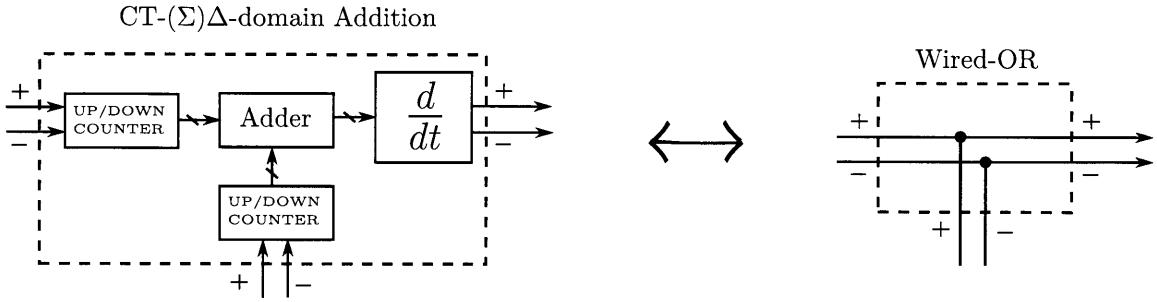


Figure 16: A CT-$(\Sigma)\Delta$-domain addition reduces to merging or superposition of inputs into a single stream of indistinguishable output pulses

that any positive input pulse will simply produce a positive output pulse and any negative input will produce a negative output. It is then equivalent to directly merge the input pulses to form the output. Assuming pulses do not arrive at exactly the same time to cause a collision, merging consists of a simple wired-OR (e.g., a tri-state bus). The result of merging input signals $\Delta_{x_1}(t) + \Delta_{x_2}(t) + \ldots$ is,

$$\Delta_y(t) = \Delta_{x_1}(t) + \Delta_{x_2}(t) + \ldots$$

This output still obeys a bound of the form in Equation 2 with respect to the signal,

$$y(t) = x_1(t) + x_2(t) + \ldots$$

The output gap bounds $\{\Delta_y^L, \Delta_y^U\}$ are the sum of the constituent input bounds. The bounds add by the rules of interval arithmetic.

$$\begin{aligned} g_y(t) &= u(t) * (y(t) - \Delta_y(t)) \\ &= \sum_i g_{x_i}(t) \implies -\sum_i \Delta_{x_i}^L \leq g_y(t) < \sum_i \Delta_{x_i}^U \end{aligned}$$

Therefore, the merged pulses in $\Delta_y(t)$ satisfy Equation 2 with the following parameters,

$$\boxed{\begin{array}{c} y(t) = \sum_i x_i(t) \qquad \Delta_y^L = \sum_i \Delta_{x_i}^L \qquad \Delta_y^U = \sum_i \Delta_{x_i}^U \\ \text{*Exception point errors occur during collisions} \end{array}} \tag{4}$$

It may seem disconcerting that the bounds widen after addition. Merging involves a loss of information and the resulting pulses are indistinguishable. Indeed, in the limit of many independent merged signals, a completely unstructured Poisson process is formed. We have already seen that it is more difficult to perform

operations on such a pulse signal. Fortunately, this problem is unique to addition and if followed by certain other operations, the bounds can be reduced again.

Another obvious concern is that in practice, pulses from two or more inputs can occur simultaneously or overlap in time if they have finite width. If single-level logic is being used, the receiver detects only a single pulse. Collisions can be managed in different ways of increasing circuit complexity. The simplest is that they can be ignored in exchange for pulse loss. This works well in signal processing when using I&F as the encoding. However, with CT-Δ modulation there can be severe distortion. Otherwise, queuing methods can be employed. Here extra pulses involved in collision are stored so that loss is exchanged for timing jitter. A hybrid-analog scheme involves accumulating merged pulses as charge stored on a capacitor. Charge is then removed serially by the receiving module. Alternatively, input pulses can be converted to discrete-time by sampling a latch on each input. Clocked bit-stream processing techniques[24, 25, 26, 27, 4] simply queue pulses using one or more bits of state. A further alternative is to eliminate the possibility of collision by system construction. In some cases, it is possible to design the arrangement of operations so that two pulses are guaranteed to not occur on both inputs simultaneously.

3.4.2 Constant Multiplication

CT-(Σ)Δ-domain multiplication can be derived similarly to addition. Any constant multiplier value α within $|\alpha| \leq 1$ can be implemented. However, rational values of the form $\alpha = \frac{N}{2^K}$ have a simple construction with a single binary accumulator. Again, a didactic construction begins by reconstituting a binary code from input pulses. The resulting counter value is multiplied by $\frac{N}{2^K}$ and then quantized with the quantization interval Δ (imparting roundoff error as with any fixed-point DSP multiplication). Finally, any change in the quantized value generates an output pulses.

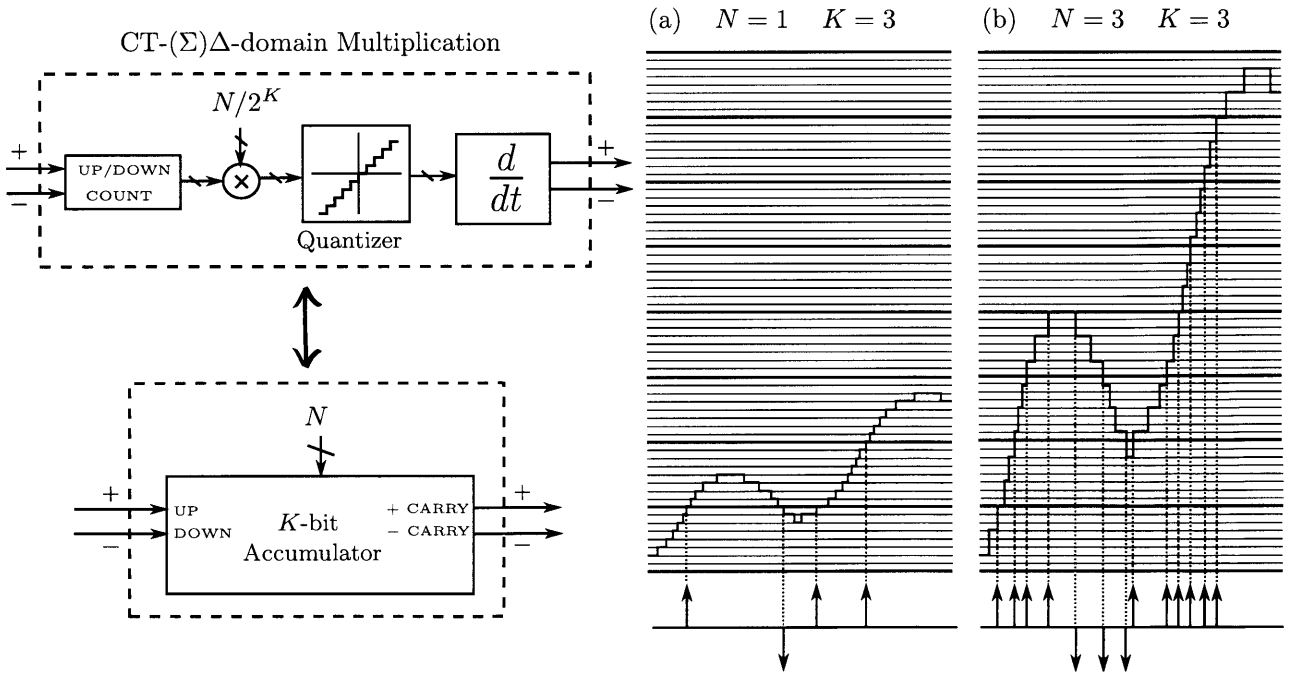


Figure 17: Pulse-domain multiplication by $\frac{1}{8}$ and $\frac{3}{8}$ are shown in (a) and (b) respectively. An output pulse is generated at multiples of 8 (heavier lines). The accumulator in (a) steps by $N = 1$ while (b) steps by $N = 3$.

This process can again be simplified. By commutativity, the multiplication by N can be performed before the counter. Now we increment/decrement an accumulator by N in response to positive/negative input pulses. Further, the multiplication by $\frac{1}{2^K}$ with quantization is equivalent to truncation of the lower K bits. The quantized value changes (and a pulse is emitted) only when the accumulator carries to the $K+1$ bit. The full operation only requires a K-bit accumulator with the overflow/underflow generating output pulses. A $+\Delta$ output pulse occurs with an overflow and a $-\Delta$ output pulse occurs with an underflow. This is illustrated in Figure 17. The first known description of a similar process is by Lockhart[28].

Of course, the multiplier denominator is not required to be a power of 2. In general the overflow/underflow can occurs with any period. In the case of multiplication by $\frac{1}{D}$ where D is any integer, the process is a type of decimation by D. This can be represented in an asynchronous finite state machine as shown in Figure 18. It is apparent that this state machine is a signed version of the one used to describe I&F in

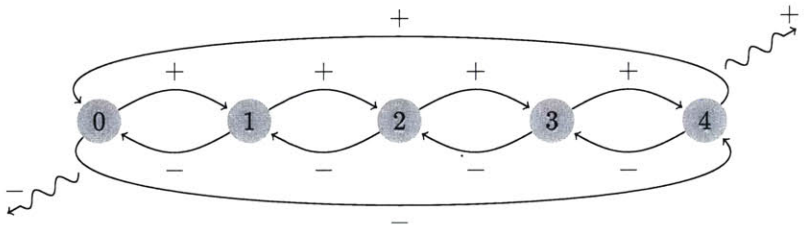


Figure 18: Pulse-domain multiplication by $\frac{1}{5}$. The notation indicates a positive output pulses occurs when a positive input pulse causes a state transition from 4 to 0. A negative output occurs when a negative input pulse causes a state transition from 0 to 4.

Chapter 2. In fact, pulse-domain multiplication by $\frac{1}{D}$ is equivalent to scaling input Dirac delta measures by $\frac{1}{D}$ and then performing (signed) I&F. With I&F, only when a full Δ measure has accumulated is an output pulse produced (this is necessary because pulses indicating Dirac deltas of measure $\frac{\Delta}{D}$ cannot be represented without requiring new types of digital pulses). This interpretation suggests that an I&F modulator and multiplication by a constant are equivalent and both delay input Dirac delta measures so that they align to produce a full Δ. The distortion from this is simply timing jitter that produces high frequency error.

If pulses indicating Dirac deltas of measure $\frac{\Delta}{D}$ could be represented, the multiplication would be performed exactly and both the encoded signal and the bounds $g_y(t)$ would be scaled down by precisely $\frac{1}{D}$. However, the I&F process expands the bounds from this slightly. The output gap is,

$$\begin{aligned} g_y(t) &= u(t) * \left(\frac{x(t)}{D} - \Delta_y(t) \right) \\ &= \frac{g_x(t)}{D} + g_d(t) \end{aligned}$$

where $g_d(t)$ is an additional new gap caused by the I&F timing jitter (quantizer error). $g_d(t)$ holds the Dirac deltas queued before an output is emitted. For multiplication by $\frac{1}{D}$, this ranges from 0 to $(D-1)$ pulses. Each has measure $\frac{\Delta}{D}$ so that $g_d(t) \in [0, \frac{D-1}{D}]$ and the overall bounds for this operation are,

$$\boxed{y(t) = \frac{x(t)}{D} \qquad \Delta_y^L = \frac{\Delta_x^L}{D} \qquad \Delta_y^U = \frac{\Delta_x^U - \Delta}{D} + \Delta} \tag{5}$$

Notice that if the input pulse signal to the constant multiplier has bound $\Delta_x^U > 1$, the output bounds will be reduced for any $D > 1$. This is important to take advantage of by ordering operations properly. For example, in averaging two signals, performing addition first followed by multiplication by $\frac{1}{2}$ will minimize the overall output bounds.

More generally, multiplication by any real constant $0 \leq \alpha \leq 1$ is equivalent to scaling Dirac deltas by α followed by I&F modulation. The result is a type of irregular decimation. For example, if $\alpha = \frac{3}{5}$, positive input pulses give an output pattern of: drop, pass, drop, pass, pass, (repeated). When α is a rational number of the form $\frac{N}{D}$, the pattern is periodic with period no longer than D. For irrational α, the pattern is aperiodic (and may be difficult to implement with a finite state machine). The bounds for general α follow as before but now $g_d(t)$ ranges from 0 to a full Δ,

$$\boxed{y(t) = \alpha x(t) \qquad \Delta_y^L = \alpha \Delta_x^L \qquad \Delta_y^U = \alpha \Delta_x^U + \Delta} \tag{6}$$

In the special case of multiplication by -1, positive output pulses are simply exchanged with negative pulses and vice-versa. For this case, the output bounds are equal to the input bounds but swapped,

$$\boxed{y(t) = -x(t) \qquad \Delta_y^L = \Delta_x^U \qquad \Delta_y^U = \Delta_x^L} \tag{7}$$

3.4.3 Delays

Pulse-domain delays are simply continuous-time digital pulse delays. Generally, digital delays are easier to construct compared with analog delays. Digital delay line memory was more common in early computers (e.g., mercury or magnetostrictive delays) and allowed simultaneous propagation of many pulses. On integrated circuits, short delays can be built from chains of inverters or microstrip lines. Asynchronous CMOS delay elements described in [29, 30] offer low power delay tunable from nanoseconds to milliseconds. Here, variability in thresholds over time due to temperature variation is a real concern (a special exception is in vivo implants which maintain excellent temperature stability). A master-slave delay locked loop with a clocked reference can be employed to compensate for threshold variation in time by setting an analog control signal. The reference clock does not need to be distributed, saving power. Additional control logic that bypasses delays can be incorporated to compensate for mismatch.

Clocked shift register delay lines may be a better method for delay in some cases due to area reduction. Conversion from pulses to a clocked bit-stream is done with a latch that is then passed to a shift register. While this requires a clock, there are still savings in the rest of the circuit and the delays are very precise. If a delay line is built out of a cascade of discrete elements, often only one pulse may occupy an element at a time. It is important to guarantee that input pulses are spaced by at least the period of the shortest single delay element so as to not interfere with one another. With correct processing, the minimum time between consecutive pulses can be deterministically bounded. For the output of an I&F modulator, this depends only on the maximum encoded signal amplitude.

The bounds for the delayed output are the same as the input,

$$\boxed{y(t) = x(t - t_{delay}) \qquad \Delta_y^L = \Delta_x^L \qquad \Delta_y^U = \Delta_x^U} \tag{8}$$

3.4.4 Filters

Adders, multipliers and delay modules can be connected together to build filters. A direct approach is to employ standard DSP filter designs but substitute the pulse processing structures for these three operations. On the other hand, the extra freedom given with continuous delays admits unique filter design techniques. At the very least, delay optimization can relax the values of the constant multipliers to round rationals that are easier to implement. For example, delay time optimization for continuous-time filters was examined by Brückmann[31]. Simplified continuous-time wave digital filters have also been explored[20].

Another approach to filter design with continuous delays is to use standard sparse filter design tools on a very fine discrete-time grid. In the resulting sparse filter, the spacing of nonzero values determines the necessary delays. Optimization can generally improve filter performance relative to discrete-time filters and relax the overall size and complexity of the filtering structure. It is important to use filter structures that

minimize addition collisions (if they are not handled properly). Empirically, most filter structures appear to perform well unless a single pulse is distributed and then later merged with a relative delay of zero. This can occur in lattice filters and some wave digital filter designs.

A completely different approach is to construct a filter consisting of only delays and adders without constant multipliers at all. In this case, the frequency response has the following form and depends only on the tap signs $v_l \in \{+\Delta, -\Delta\}$ and times t_l,

$$H(f) = \sum_l v_l \cdot e^{-j2\pi f t_l}$$

While a number of methods can be used to optimize t_l and v_l to give a desired frequency response, one simple approach is to design an FIR filter using the I&F modulation[2] of the desired impulse response $h(t)$. Intuitively, if $\Delta_h(t)$ is the I&F modulation of $h(t)$, the Fourier transform of $\Delta_h(t)$ is close to $h(t)$ at low frequencies. If a filter with impulse response $\Delta_h(t)$ is used, the undesired power spectrum at higher frequencies is removed anyway by a reconstruction filter. In fact, it is shown that this filter also has precise time-domain error bounds with respect to the exact $h(t)$ filter.

A direct-form FIR structure consists of a tapped digital pulse delay line shown in Figure 19. Input pulses $\Delta_x(t)$ travel from the left to right and are tapped at various positions. Graphically, the position of a tap indicates its relative delay in the delay line. The taps at times t_l with $v_l = -\Delta$ will invert any pulses, while taps with $v_l = +\Delta$ preserve the pulse sign. The tapped pulses are all merged together (e.g., on a tri-state bus) and then finally multiplied by the constant Δ to give the correct weight. By scaling all pulses by Δ at the end, the filter output bounds are minimized.

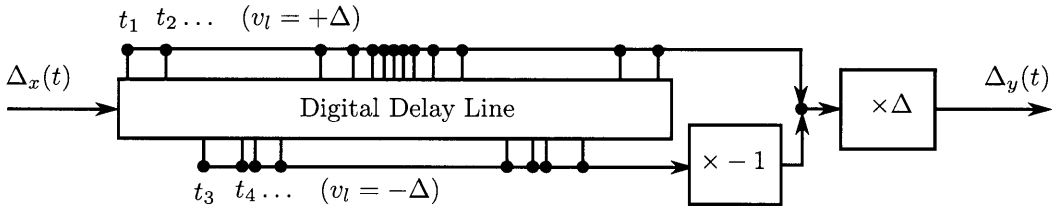


Figure 19: An FIR structure for low-pass filtering in the pulse domain.

This method of filtering can be verified by calculating the gap bounds on the output. For I&F signals, we consider filtering of $\Delta_x(t)$ with times $t_k \in \tau_x$, weights $v_k \in \{-\Delta, \Delta\}$ and gap $g_x(t) \in [-\Delta_x^L, \Delta_x^U)$. The FIR filter has impulse response $\Delta_h(t)$ with $t_l \in \tau_h$, weights $v_l \in \{-\Delta, \Delta\}$ and gap $g_h(t) \in [-\Delta_h^L, \Delta_h^U)$. The result of convolution is a signal composed of Dirac deltas,

$$\begin{aligned}
\Delta_y(t) &= \Delta_x(t) * \Delta_h(t) \\
&= \left(\sum_k v_k \cdot \delta(t - t_k)\right) * \left(\sum_l v_l \cdot \delta(t - t_l)\right) \\
&= \sum_k \sum_l v_k \cdot v_l \cdot \delta(t - t_k - t_l)
\end{aligned}$$

The total number of deltas is the product of the number in the constituents. The deltas are each weighted by $v_k \cdot v_l$ and can be either $+\Delta^2$ or $-\Delta^2$ depending on whether or not v_k and v_l are the same sign. These weights cannot be represented with the same digital pulses used to represent weights of Δ–the squared pulses

[2]Higher-order pulse modulation can be used for better noise-shaping.

are generally smaller than Δ because $\Delta \ll 1$. To correct this, a constant Δ is factored out and multiplied at the end. The result $\Delta_y(t)$ is the I&F encoding of a filtered signal.

The bounds on $\Delta_y(t)$ with respect to exact convolution $y(t) = x(t) * h(t)$ can be found,

$$\begin{aligned} u(t) * \Delta_y(t) &= u(t) * \Delta_x(t) * \Delta_h(t) \\ &= u(t) * (x(t) - g'_x(t)) * (h(t) - g'_h(t)) \\ &= \underbrace{u(t) * x(t) * h(t)}_{Y(t)} - \underbrace{(g_x(t) * h(t) \; + \; g_h(t) * x(t) \; - \; g_x(t) * g'_h(t))}_{g_y(t)} \end{aligned} \tag{9}$$

Each of the three terms in $g_y(t)$ can be bounded to give the overall output gap bound. By Young's theorem, the bounds on the infinity-norm of the first and second terms are,

$$\begin{aligned} \|g_x(t) * h(t)\|_\infty &\leq \max(\Delta_x^L, \Delta_x^U) \cdot \|h(t)\|_1 \\ \|g_h(t) * x(t)\|_\infty &\leq \max(\Delta_h^L, \Delta_h^U) \cdot \|x(t)\|_1 \end{aligned}$$

For the third term note that,

$$\begin{aligned} \|g'_h(t)\|_1 &= \|\Delta_h(t) - h(t)\|_1 \\ &\leq \|\Delta_h(t)\|_1 + \|h(t)\|_1 \end{aligned}$$

so that,

$$\| - g_x(t) * g'_h(t)\|_\infty \leq \max(\Delta_x^L, \Delta_x^U) \cdot (\|\Delta_h(t)\|_1 + \|h(t)\|_1)$$

The bound on $g_y(t)$ is the bound on the sum of the three terms. A pulse-domain FIR filter then has the following bounds,

$y(t) = x(t) * h(t)$	$\Delta_y^L = \max(\Delta_x^L, \Delta_x^U) \cdot (\|\Delta_h(t)\|_1 + 2\|h(t)\|_1) + \max(\Delta_h^L, \Delta_h^U) \cdot \|x(t)\|_1$
	$\Delta_y^U = \Delta_y^L + \Delta$

The extra Δ term added to Δ_y^U comes from the final multiplication by Δ in the filter. Now, in practice the $\|x(t)\|_1$ may be very large, but in fact if $\Delta_h(t)$ is finite in length (FIR), then this L_1-norm can be replaced with the largest absolute-integral of $x(t)$ over a period of time equal to the filter length. Further, the actual range for $g_y(t)$ tends to be much smaller than the hard bounds suggests.

A practical issue concerns the large number of pulses that merge before the multiplication by Δ. Collision rates increase dramatically during the merge, adding potentially significant error. Breaking the addition and by Δ into steps (e.g. with a binary tree averaging signal pairs at each level) is one possibility to avoid as many collisions. Unfortunately, this increases the overall output gap from the roundoff at each level. A different solution involves counting networks. These structures can perform the operation exactly without changing the output gap bounds while distributing the combined task of merging and multiplying by Δ.

3.4.5 Counting Networks

Many pulse-domain operations rely on counting as a core mechanism; pulses from one or more inputs drive a counter and outputs are generated based on the counter state. There is need for fast and flexible counting structures that can reliably count pulses from multiple asynchronous inputs. Individual asynchronous ripple counters are efficient but do not support sharing with multiple inputs and are intolerant to state corruption. As an alternative, counting networks[32] are a distributed counting structure that provide a shared counter

with many asynchronous inputs. They are non-blocking, and can trade network depth for lower input contention and improved fault tolerance[33]. Counting networks find many uses in pulse-domain processing.

Counting networks are comprised of two-input two-output elements called balancers that are connected to one another. A balancer is symbolized in Figure 20 by two dots and a vertical line. It directs input pulses to the top or the bottom output instantaneously in an alternating fashion. For each input pulse, a state is toggled. Regardless of which input line a pulse arrives on, it is directed to the top if the state is 1 and to the bottom if the state is 0. This evenly balances pulses between the two outputs. In the figure, the input pulses are numbered sequentially in absolute order and the resulting output pulses shares the same number. Balancers can be extended to support negative input pulses as well[34]. For input pulses of either sign, the

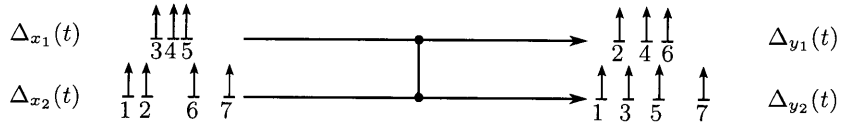


Figure 20: A balancer routes pulses so that they alternate on either line.

state toggles as before, however, negative input pulses are instead directed to the top if the state is 0 and to the bottom if the state is 1. This is shown in Figure **??**.

If the two inputs to a balancer are I&F signals, each of the two outputs correspond to a pulse-domain averaging operation. The emitted pulses are equivalent to merging the inputs and then divided by 2 yielding the operation $y_1 = y_2 = (x_1 + x_2)/2$. The two outputs encode the same value but perform decimation by 2 with different initial conditions; if the outputs are later combined the result is identical to merging the original inputs $\Delta_{x_1}(t)$ and $\Delta_{x_2}(t)$. Balancer outputs can be fed into inputs of other balancers and certain connections form counting networks. A counting network extends the balancing operation to any number N pulse signals and generates outputs in a round-robin cycle regardless of the particular input wire from which a pulse arrives. These networks were first studied by Aspnes, Herlihy, and Shavit [32]. In general,

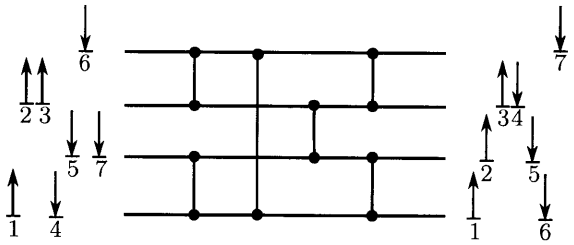


Figure 21: An example counting network with width of 4.

for a counting network of width N, an output pulse is emitted on the wire with index equal to the signed cumulative sum of all input pulses $\mod N$. Design of large efficient counting networks is an area of active research but many methods exist. Any input / output width and ratio is possible by using irregular counting networks[35].

Counting networks can be used directly in pulse processing for constant multiplication by rational numbers. For example, to multiply the sum of two inputs by the rational constant $\frac{3}{8}$, a counting network of input width 2 and output width 8 is constructed. 3 of the outputs are merged to specify the numerator (the underlying balancers for the unused inputs and outputs can be pruned away).

In general, the correct counting network output wires to select can be determined based on the pattern

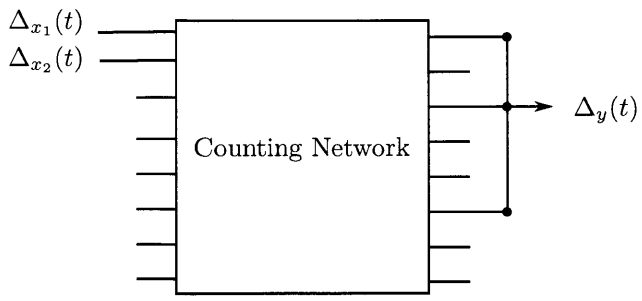


Figure 22: A counting network implementation of $y = \frac{3}{8}(x_1(t) + x_2(t))$.

of decimation prescribed for constant multiplication. For the $\frac{3}{8}$, incrementing a 3-bit ($2^3 = 8$) counter by 3 results in $3i \mod 8 = 3, 6, 1^*, 4, 7, 2^*, 5, 0^*$ (repeated). The starred numbers indicate overflows. Thus, only 3rd, 6th and 8th outputs give emissions. Corresponding wires are selected on the counting network output.

A counting network is especially useful when many inputs are merged and then multiplied by a small constant. The FIR filter is an important example. Many signals are tapped from delays (based on the filter specification), merged and then multiplied by Δ. Normally, this suffers from high collision frequency during the merging step. With a counting network, the merging and final multiplication can be combined, avoiding many more collisions as each constituent balancer only handles a fraction of the pulses.

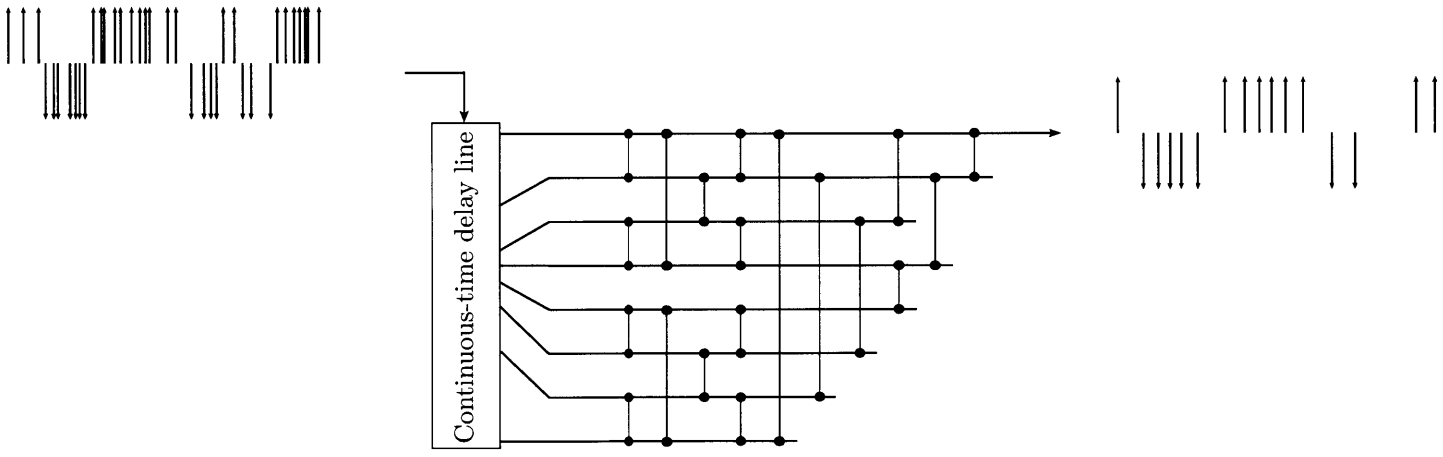


Figure 23: An example of a an 8-tap pulse-domain FIR filter based on a pruned counting network.

3.5 Piecewise-linear Functions

The inclusion of non-linear pulse-domain operations would enable many more pulse-processing applications beyond linear-time invariant systems. Here we determine that the general class of memoryless nonlinear operations that can be implemented in the pulse-domain are piecewise-linear (PWL) functions. These functions can all be implemented using simple asynchronous finite-state machines that we call asynchronous pulse machines (APMs). The finite-state machine for constant multiplication is one example of an APM. In general, an APM consists of a number of states over which transitions occur in response to input pulses from one or more inputs. During the transition, an output pulse may be emitted, depending on the particular input source and the current state. A key property of APMs is that they are incapable of changing state or generating output pulses without an originating input pulse; pulses are only passed through to the output or blocked. APMs are "passive" digital systems–in principle, one could construct APMs without an external energy source beyond the input pulses themselves–a small amount of energy from a pulse can cause the state transition while the remaining energy may pass through to the output.

The design of APMs is based on careful arguments about the possible set of signals encoded by input pulses. While these designs are different than any know published results, some are generalizations of earlier developments in discrete-time bit-stream processing systems (see tutorials [3, 4, 5, 6, 7]).

The behavior of an APM implies that the sequence of state transitions depend only on the relative ordering of input pulses. For example, if all inputs to the APM are doubled in rate, the outputs will necessarily double in rate but the signs and state ordering are otherwise the same. If the pulse signal is encoded with I&F, changing the rate–or time-warping–corresponds to scaling the encoded signal (a rate scaling $t \to \alpha t, \quad \alpha > 0$ corresponds to scaling the encoded signal by α). Therefore, any memoryless function f constructed with a APM must satisfy: $f(\alpha x_1, \ldots, \alpha x_N) = \alpha f(x_1, \ldots, x_N)$ for all $\alpha > 0$. This indicates the function must be positive-homogeneous of order 1–a generalization of linear functions that includes *abs*, *min* and *max*. General piecewise-linear functions can then be constructed with the inclusion of a constant pulse input source.

3.6 Absolute Value

In principle, absolute value consists of determining the sign of the encoded signal and inverting pulse signs whenever it is negative ($x(t) < 0$). For example, with a CT-Δ signal, the sign can be determined using an up/down reconstruction counter counting the input Δ pulses. Because $sign(Q(x(t))) = sign(x(t))$, the sign of the count gives the sign of the signal. To perform the absolute value, pulses are passed unchanged to the output when the counter value is positive, but flipped in sign when the counter is negative. The resulting stream is exactly the CT-Δ modulation of the absolute value of the signal. Unfortunately this technique cannot be transferred directly to I&F as done previously. Absolute value of a signal integral is not the same as the integral of the absolute value of the signal because the operation isn't linear.

In fact with an I&F pulse encoding, absolute value is not nearly as straightforward. A pathological example can be constructed in which an input $x(t)$ alternates rapidly in sign such that its integral varies but never crosses a Δ level to form a pulse in its I&F encoding $\Delta_x(t)$. While the true integral of $abs(x(t))$ grows unbounded (and would cross many levels if encoded with I&F), there are no $\Delta_x(t)$ pulses from which to form an output $\Delta_y(t)$! Further, a signal such as $x(t) = 0$ has the same I&F encoding as the alternating signal. In this case, the correct output is ambiguous. The opposite problem can also occur. For example, consider a different input $x(t)$ formed from two merged pulse streams each encoding the same constant magnitude but with opposite sign and a small relative delay. $\Delta_x(t)$ consists of alternating positive and negative pulses and encodes $x(t) = 0$ with gap bounds $\Delta_x^L = \Delta_x^U = 1$. Now, it is clear that the correct output $\Delta_y(t)$ encoding $y(t) = abs(x(t)) = 0$ is formed by emitting no pulses (blocking all input pulses). On the other hand, an alternative $x(t)$ with the same $\Delta_x(t)$ may be non-zero and the correct behavior may be to pass on every input pulse as a positive output pulse.

These problems arise from the ambiguity of $x(t)$ based only on $\Delta_x(t)$. One way forward is to choose a

unique signal $x^*(t)$ from the set of possibilities for $x(t)$ consistent with $\Delta_x(t)$ and to form a $\Delta_y(t)$ encoding $y(t) = abs(x^*(t))$. The approach taken here is to choose the $x^*(t)$ that is zero whenever zero is a possible value for the signal given the pulses in $\Delta_x(t)$ and the known input bounds. This way, the output pulses still maintain an encoding of the infimum of $abs(x(t))$ and $\Delta_y^L = 0$. As a consequence, we may miss some positive output pulses and cannot guarantee a finite upper bound on the gap ($\Delta_y^U = \infty$). However, these errors can only occur during changes in the sign of $x(t)$. Indeed, it can be shown that only $\Delta_x^L + \Delta_x^U$ missing pulses are possible per sign change of $x(t)$ in an extreme case. If the signal $x(t)$ does not alternate between signs too rapidly, these missing pulses are actually rare. Therefore, we can consider them spurious exception errors as with pulse collisions and exclude them from the stated Δ_y^U output bound.

In order to produce the correct output pulses to encode $y(t) = abs(x^*(t))$, it is only necessary to determine when the encoded input signal $x(t)$ could be zero while still satisfying the observed pulse times and the known input bounds. A signal $x(t)$ may equal zero over any time interval that the integral $X(t)$ may be constant. Given the input pulses and the input bounds, we can keep track of the possible range of changes in $X(t)$ since the last emitted pulse. If this range does not include zero, we are guaranteed that the integral has changed by at least Δ so that the integral of the absolute value has also changed by at least Δ and a positive output pulse should be emitted.

This reduces to simple APM structures shown in Figure 24). The number of states is simply the size of the entire input gap bound range, $\Delta_x^L + \Delta_x^U$, plus one.

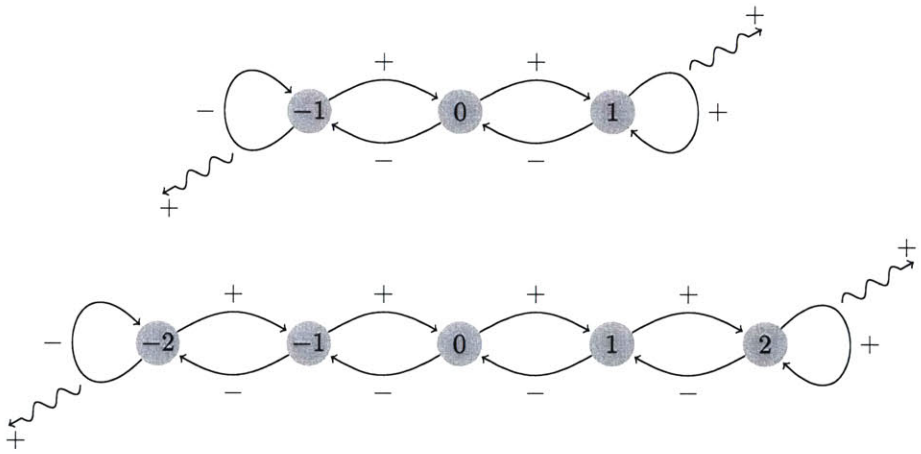


Figure 24: Two different APMs for the pulse-domain absolute value function. Three states are required if $\Delta_x^U = \Delta_x^L = 1$ while five states are required if, $\Delta_x^U = \Delta_x^L = 2$.

The bounds for the pulse-domain absolute value are simply the sum of the input bounds if we exclude the transient issues when $x(t)$ changes sign:

$y(t) = abs(x(t))$	$\Delta_y^L = 0$	$\Delta_y^U = \Delta_x^L + \Delta_x^U$
*Exception point errors apply during sign transitions		

(10)

3.7 Min and Max Operations

The min and max operations are important functions that can be constructed using a simple identity,

$$min(x_A(t), x_B(t)) = \frac{x_A(t) + x_B(t) - |x_A(t) - x_B(t)|}{2}$$

$$max(x_A(t), x_B(t)) = \frac{x_A(t) + x_B(t) + |x_A(t) - x_B(t)|}{2}$$

Using the known absolute value operation, the constituent operations can all be performed in the pulse domain. All of the operations can be combined into a single APM. A derivation is shown in Figure 25. The resulting finite-state machine still only has as many states as the constituent absolute value operation. In the notation, transition A^+ corresponds to a positive input pulse from input $\Delta_A(t)$ while B^- corresponds to a negative input pulse from input $\Delta_B(t)$. Multiple kinds of input pulses may cause the same transition but the output pulse depends on the particular input pulse.

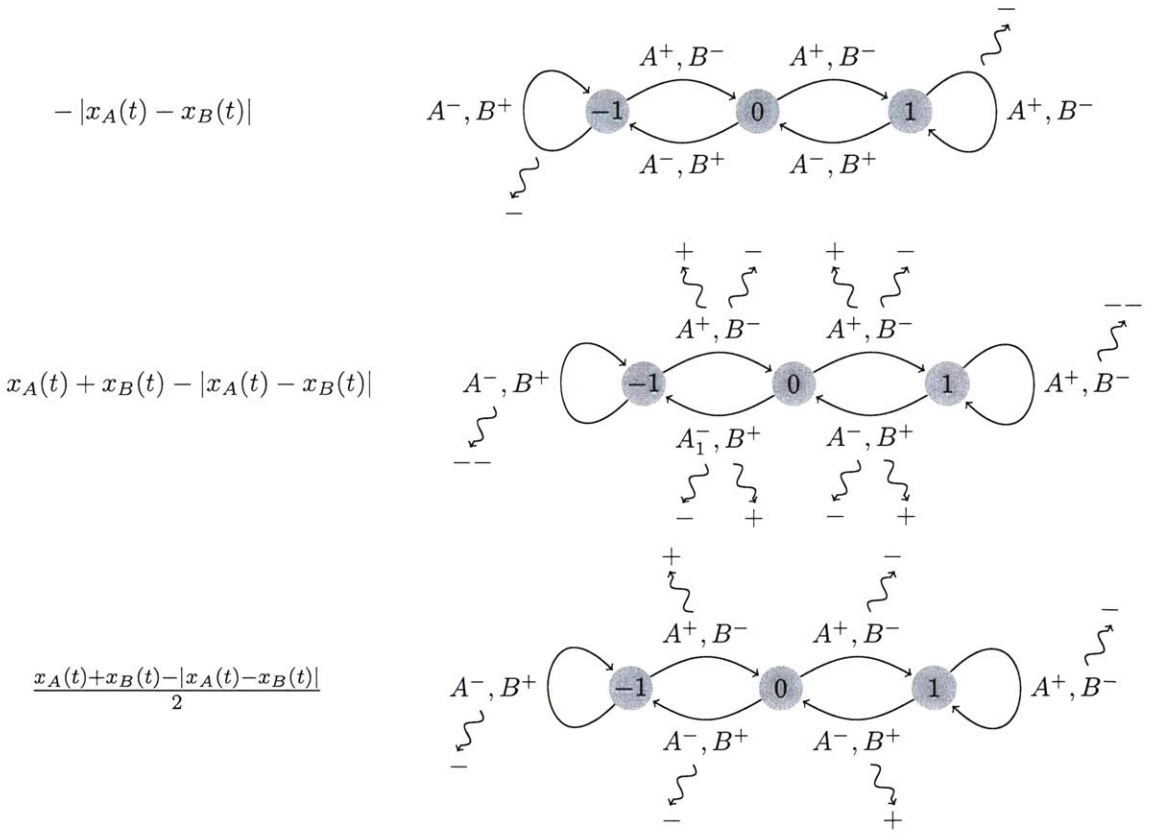


Figure 25: Deriving the APM for the pulse-domain min operation. Here, $\Delta_A^L = \Delta_B^L = 0$ and $\Delta_A^U = \Delta_B^U = 1$.

The APM for the max operation is derived along the same lines. It can equivalently be arrived at by directly implementing $max(x_A(t), x_B(t)) = -min(-x_A(t), -x_B(t))$.

Conveniently, min and max use the same finite-state machines and one pulses only when the other does not. In particular, merging the outputs of min and max will always give an output equal to the original inputs merged, i.e., $min(A, B) + max(A, B) = A + B$. Therefore, both the min and max operation can be

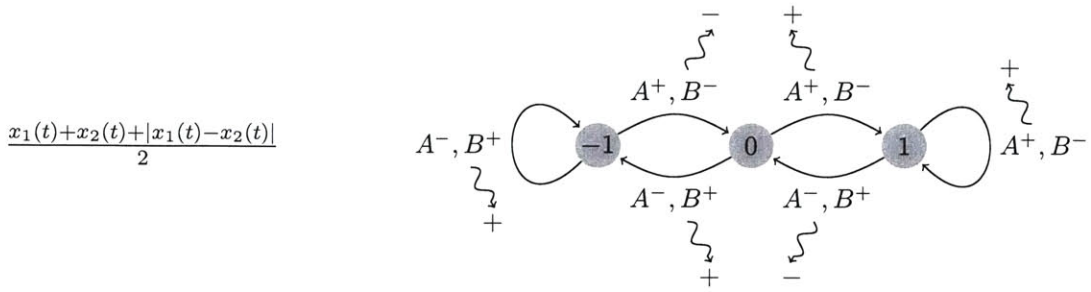


Figure 26: The APM for the pulse-domain *max* function. Here, $\Delta^L_{x_1} = \Delta^L_{x_2} = 0$ and $\Delta^U_{x_1} = \Delta^U_{x_2} = 1$.

computed in parallel with a single state machine that sorts input pulses into a min and max output channel. Any input pulse is passed to an output based on the current state and immediately afterwards the state changes depending on the type of the input pulse.

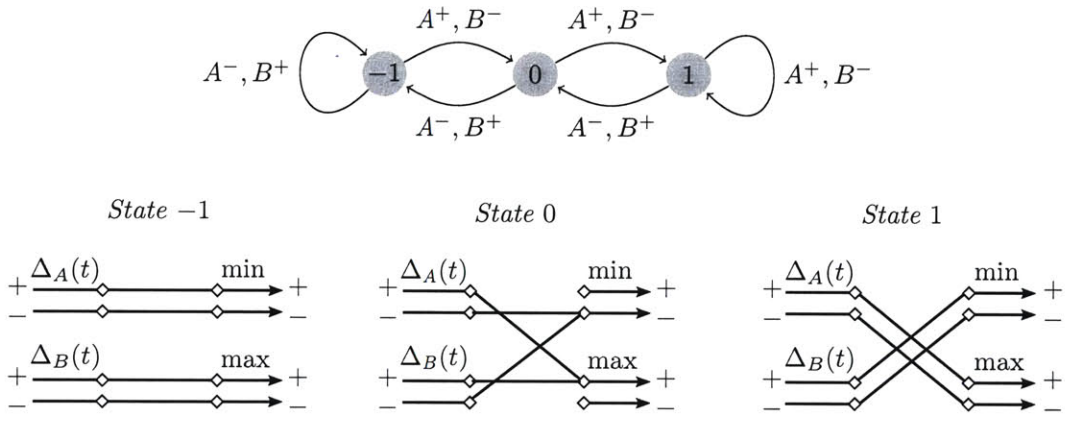


Figure 27: An alternative "pulse sorting" representation of the pulse-domain min and max operations.

In some applications, the input pulses are always positive for both inputs. If this is the case, the APM is simplified by removing the extra negative inputs. This results in the APMs shown in Figure 28.

The min and max operations have many applications including the construction of any piecewise-linear function. For example, a convex piecewise linear functions can be written as,

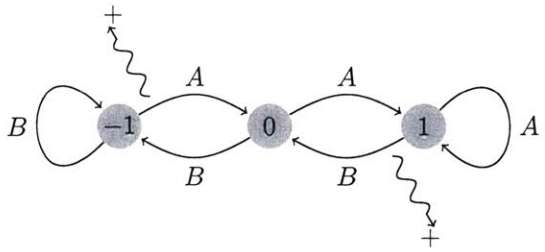
$$f(x) = max(\alpha x + a, \beta x + b, \ldots)$$

Also, a network of min and max operations can build a sorting network for continuous-time order statistic calculations. Useful signal processing tasks include median filters for impulsive noise removal, image edge detection, and comparing the outputs of matched filters. Another powerful application is the min-sum algorithm which is applied to forward error correction in Section 5.5.

3.8 Choosing a Pulse Encoding

While this thesis focuses on the I&F encoding, other continuous-time encodings have relative advantages and disadvantages. A pulse processing system may use multiple signal representations as conversion is often

$\Delta^U_{x_A} = \Delta^U_{x_B} = 1$
$min(x_A(t), x_B(t))$



$\Delta^U_{x_A} = \Delta^U_{x_B} = 1$
$max(x_A(t), x_B(t))$

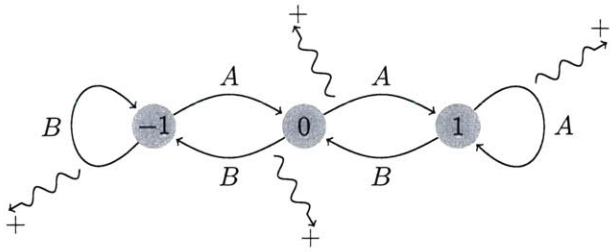


Figure 28: Min/Max operations restricted to only positive input pulses. In all cases, $\Delta^L_{x_A} = \Delta^L_{x_B} = 0$.

simple. A brief comparison among continuous-time formats is shown in the table below. Continuous-time pulse-code modulation (CT-PCM) consists of standard fixed-point computation in continuous-time while CT-Δ simplifies this by only transmitting pulses when PCM quantization levels step up or down.

	CT-16-bit PCM	CT-Δ	CT-$\Sigma\Delta$ (I&F)	CT-Analog
Transistors Required				
Addition	10^2	10^1	0	0
Constant Multiplication	$> 10^3$	10^2	10^2	10^1
Signal Multiplication	$> 10^3$	10^2	Difficult	10^1
Comparitors	10^3	10^2	10^1	10^1
Delays	10^2	10^1	10^1	10^1
Other functions	All possible	All possible	Piecewise linear (PWL)	Log/Exp/PWL
Characteristics				
Quantiz. Noise PSD	Broadband	Broadband	High-pass	N/A
Timing Jitter PSD	Moderate, low-freq.	Small, low-freq.	Small, high frequency	N/A
Digital Error Spectrum	Large, broadband	Infinite, low freq.	Small, broadband	N/A
Error Floor Effects	No	No	No	Yes
Power Consumption	$\propto$ derivative	$\propto$ derivative	$\propto$ amplitude	$\propto$ both
Dynamic Range Limits	$\pm 2^{15}$	Counter size	Pulse rate limited	Linearity limited
System Overhead				
Requires handshaking	Yes	Yes	No	No
Number of wires	16	1-2	1-2	1-2
A/D & D/A Complexity	High	High	Low	None

4 Error Sensitivity

Digital pulses are a relatively robust signal encoding. Once an analog signal is converted to pulses, it can be amplified with a non-linear device and subject to considerable noise without corrupting the signal. The basic detection of a pulse above the noise floor requires a single comparator and if pulses are of equal size, they can be restored to an ideal height and width and retransmitted without influences from previous noise. Pulses are also easy to generate with little power and the efficiencies explain much of the interest in ultra-wideband communication, class-D amplifiers and switched-mode power supplies. Of course, this interest isn't new; pulses were fundamental to the first wired and wireless communication systems including the telegraph and spark-gap transmitters. The robustness and simplicity of pulses remain as important today as they were for Marconi's first transatlantic transmissions.

However, pulses do admit certain kinds of distortion. In general, pulse signals exhibit two basic types of error: timing variation and pulse deletion / insertion. Timing variation or jitter involves pulses being delayed in time by different amounts. Electronic timing noise has many causes including temperature variations or supply voltage fluctuations. In addition, with pulse processing, the process of pulse queuing can be interpreted as systematic timing variation. Pulse loss occurs when a transmitted pulse is undetected by a receiver. This can occur when the pulse shrinks in amplitude or duration or a pulse collision occurs. A pulse insertion occurs when a pulse is misdetected. Processing hardware may contribute to misdetection; transistors occasionally switch improperly from influences by external radiation and cosmic rays especially as dimensions shrink and the critical charges are lowered.

Exactly how timing variation and pulse loss affect a reconstruction is a property of the particular encoding. With I&F and a filter-based reconstruction, there is a natural robustness to both types of error. Analysis of the distortion on an I&F encoding is now examined.

4.1 Characterization of Pulse Timing Jitter

Timing variation can be modeled as addition of an error signal that subtracts a pulse at the correct time and adds or inserts a pulse at the delayed time. For example, if a single pulse has moved from time t_0 to $t_0 + \tau$, the error signal is,

$$e(t) = \Delta\,\delta(t - (t_0 + \tau)) - \Delta\,\delta(t - t_0)$$

The effect of this error on the reconstructed signal depends on the pulse encoding type. For filter-based reconstruction of CT-Δ and CT-$\Sigma\Delta$ (I&F), the distortion can be gauged from the power spectrum of the error. The Fourier transform of $e(t)$ is,

$$\begin{aligned} E(j\omega) &= \mathcal{F}\{e(t)\} \\ &= \Delta\, e^{-j\omega(t_0+\tau)} - \Delta\, e^{-j\omega t_0} \\ &= \Delta\, e^{-j\omega(t_0+\tau/2)} \left(e^{-j\omega\tau/2} - e^{j\omega\tau/2}\right) \\ &= 2\Delta\, e^{-j\omega(t_0+\tau/2)-\pi/2}\, \sin(\frac{\tau}{2}\omega) \end{aligned}$$

Ignoring the phase term, $E(\omega)$ is sinusoidal in frequency. For $|\tau\omega| \ll 1$, a 1st-order Taylor series approximation of sin indicates that $|E(\omega)| \propto |\Delta\tau\omega|$. The error energy within a small bandwidth ω_{BW} is,

$$\begin{aligned} E_{\Sigma\Delta} &= \int_{-\omega_{BW}}^{\omega_{BW}} |E(j\omega)|^2 d\omega \\ &\propto \Delta^2\tau^2\omega_{BW}^3 \end{aligned}$$

Therefore, with I&F, small τ timing perturbations of pulses produce high-frequency distortion which mostly eliminated by the reconstruction low-pass filter.

How does CT-Δ compare? Without pre-integration during the modulation, CT-Δ requires the pulses to be integrated before the reconstruction filter is applied. The error signal is also integrated which results in a new distortion spectrum. The Fourier transform is,

$$\begin{aligned}\mathcal{F}\{u(t) * e(t)\} &= \frac{E(j\omega)}{j\omega} \\ &= -2\Delta\, e^{-j\omega(t_0+\tau/2)}\, \frac{\sin(\frac{\tau}{2}\omega)}{\omega}\end{aligned}$$

The amplitude is now a sinc function in frequency. For $|\tau\omega| \ll 1$, the magnitude is nearly a flat constant in frequency $\propto |\Delta\tau|$. The resulting error energy over a small bandwidth f_{BW} is then,

$$\begin{aligned}E_\Delta &= \int_{-\omega_{BW}}^{\omega_{BW}} \left|\frac{E(j\omega)}{j\omega}\right|^2 d\omega \\ &\propto \Delta^2\tau^2\omega_{BW}\end{aligned}$$

From these calculations, it is apparent that the reconstruction error power-spectral-density differs in shape between the two pulse encodings. Pulse timing variation in I&F gives error with more energy at higher frequencies while CT-Δ incurs mostly a broadband or low-frequency distortion. If the reconstruction filter has small bandwidth, the distortion caused by I&F can be much less than CT-Δ.

4.2 Characterization of Pulse Loss

To understand pulse loss, it is sufficient to consider the effect of a single lost pulse. Pulse insertion produces the same error only negated in sign and multiple pulse loss is the sum of individual pulse losses. Loss at time t_0 involves adding an error signal of simply,

$$e(t) = -\Delta\, \delta(t - t_0)$$

The Fourier transform is, $\mathcal{F}\{e(t)\} = -\Delta$. The error energy is therefore linearly proportional to the bandwidth of interest, $E_{\Sigma\Delta} \propto \Delta \cdot \omega_{BW}$. The error energy from pulse loss is generally higher than pulse timing variation. However, compared with other signal encodings such as fixed-point binary numbers, the loss is small (similar to a change in the least significant bit).

CT-Δ modulation is more dramatically affected by pulse loss. With the reconstruction integration, the error signal is now,

$$e(t) = -\Delta\, u(t - t_0)$$

Unfortunately, this error extends for all time after t_0; the reconstruction counter is effectively off from the error time on and the total energy of the error is infinite. Pulse loss with CT-Δ encodings can cause serious distortion especially if multiples errors accumulate over time. For this reason, the design architecture for CT-Δ processing must protect from pulse loss. An alternate solution is to use a leaky integration for reconstruction instead of the counter or integrator. In this case, very low-frequency information is lost but error does not accumulate.

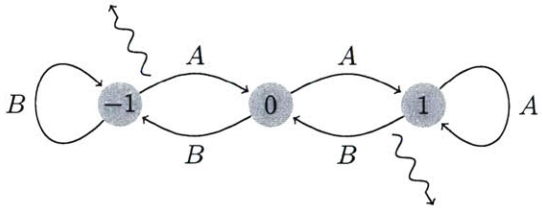
In summary, two types of pulse errors can affect a pulse encoding. Pulse jitter introduces high frequency error in I&F (sine spectrum) and low/broadband noise in CT-Δ (sinc spectrum). Spurious pulse loss/gain creates moderate broadband error in I&F and large low frequency error in CT-Δ.

4.3 Noisy Pulse Processing Machines

In order to estimate how pulse timing variations affect the results after processing with a PPM, we analyze the case where pulses are randomly jittered to the point of behaving like a stochastic Poisson Process. Normally PPMs are designed based on the gap bounds of the inputs. When the inputs violate these bounds, the PPM does not necessarily perform the correction function.

The limiting case of an unbounded pulse signal is a Poisson process (e.g., from merging many independent pulse signals or randomizing pulse times). We can analyze the effect of the Poisson input process by modeling the PPM as a Markov chain. With Poisson inputs, state transitions occur randomly with transition probabilities proportional to input pulse rates. The time evolution of the Markov chain is a continuous-time Markov process. We can therefore calculate the stationary distribution for the chain and determine the steady-state output pulse rate as a function of the input rates.

As an example, consider the $\min(A, B)$ operation (with strictly positive input and output pulses). We examine a 3-state min operation designed for gap bounds of $\Delta^L = 0$ and $\Delta^U = 1$ for both inputs. The PPM is as follows,



If the two inputs $\Delta_A(t)$ and $\Delta_B(t)$ are independent Poisson processes rather than I&F signals, the output $\Delta_C(t)$ is random but a reconstruction can still precede by low-pass filtering to recover the average pulse rate. The question is whether the output rate is close to the min of the input rates despite the Poisson statistics. We assume the Poisson inputs have locally constant rates equal to λ_A and λ_B respectively and that the chain is well-mixed. We would like to compute the output pulse rate λ_C for the PPM given these input pulse statistics.

The analysis follows from standard Markov process theory. In steady-state, the frequency of transitioning into any given state must equal the frequency of transitioning out of that state. Also the rate of a given transition rate is equal to the steady-state probability of being in the initial source state multiplied by the rate of the Poisson input causing the transition. This, along with the restriction that the sum of the steady-state probabilities must be 1, constitute the balance equations which can be solved for a steady-state distribution. If π_i is taken as the steady-state probability for being in the i'th state, the balance equations for the 3-state I&F min operation are,

$$\begin{aligned}
\lambda_A \pi_{-1} &= \lambda_B \pi_0 \\
\lambda_A \pi_0 &= \lambda_B \pi_1 \\
\pi_{-1} + \pi_0 + \pi_1 &= 1
\end{aligned}$$

This system of equations is then solved to give steady-state probabilities for the various states,

$$\pi_{-1} = \frac{\lambda_B^2}{\lambda_A^2 + \lambda_A\lambda_B + \lambda_B^2} \qquad \pi_0 = \frac{\lambda_A\lambda_B}{\lambda_A^2 + \lambda_A\lambda_B + \lambda_B^2} \qquad \pi_1 = \frac{\lambda_A^2}{\lambda_A^2 + \lambda_A\lambda_B + \lambda_B^2}$$

The output pulse rate is then computed as the total rate of transitions from state -1 or 1 into state 0,

$$\begin{aligned}
\lambda_C &= \pi_{-1}\lambda_A + \pi_1\lambda_B \\
&= \frac{\lambda_B^2\lambda_A + \lambda_A^2\lambda_B}{\lambda_A^2 + \lambda_A\lambda_B + \lambda_B^2}
\end{aligned}$$

This function is plotted for a few fixed values of λ_B in the purple curves of the left graph of Figure 29. Clearly, the output rate λ_C is no longer the precise min (pink lines) as is obtained with true I&F inputs with bounds $\Delta^L = 0$ and $\Delta^U = 1$. It is observed that the output rate is always less than $\min(\lambda_A, \lambda_B)$ and differs most when λ_A is equals λ_B.

For min PPMs with more states (designed for wider input gap bounds), the output rates come closer to $\lambda_C \approx \min(\lambda_A, \lambda_B)$. This is shown in the next two graphs. While the PPMs with more states give more accurate output rates, the tradeoff is that the mixing times of the chains become longer as the number of states increase and the steady-state distribution takes longer to become valid. If the inputs have time-varying rates, an increased number of states lengthens the mixing time of the chain and leads to inaccurate results. Even when complete mixing does occur, the reconstruction of an output signal with random pulses is less accurate than with a true I&F output.

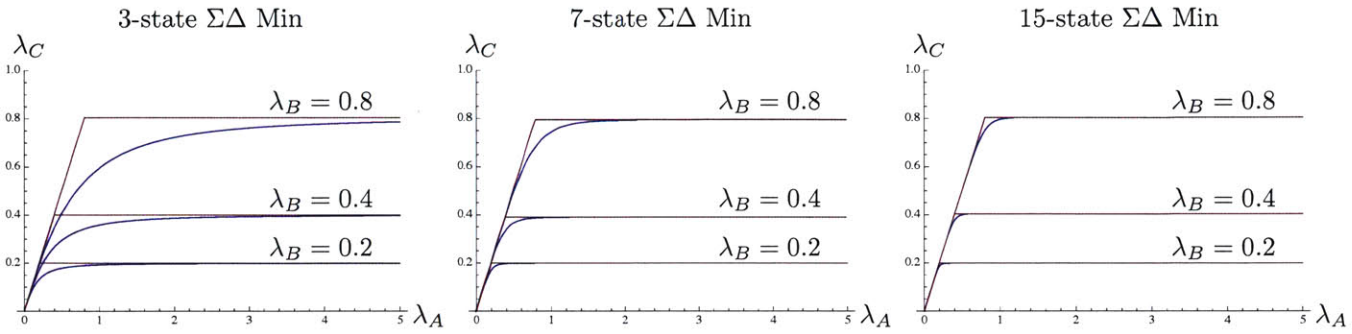


Figure 29: The steady-state output rate λ_C of a pulse-domain min operation as a function of inputs rates λ_A and λ_B for I&F (pink) and Poisson (purple) input encodings.

The output pulse rate for a max state machine with Poisson inputs can also be computed by the same procedure. The output rate for a 3-state *max* function is,

$$\begin{aligned}\lambda_C &= \pi_{-1}\lambda_B + \pi_0(\lambda_A + \lambda_B) + \pi_1\lambda_A \\ &= \frac{\lambda_B^3 + \lambda_B^2\lambda_A + \lambda_A^2\lambda_B + \lambda_A^3}{\lambda_A^2 + \lambda_A\lambda_B + \lambda_B^2}\end{aligned}$$

This function is always greater than $\max(\lambda_A, \lambda_B)$, but again converges for PPMs with enough states. This analysis supports the accuracy of the I&F min and max PPMs even when the input bounds fail to match those used for the design. This is important because in certain applications these bounds are not exactly known (e.g., pulse-domain systems with feedback loops). These results generally extend to the other pulse-domain operations. However, any passive pulse machine can be analyzed as a continuous-time Markov chain for determining its exact behavior with stochastic inputs.

5 Applications

Various applications of pulse processing were examined as part of the thesis. A custom software simulator was combined with an FPGA platform to demonstrate pulse processing in a filter bank, a differential analyzer for solving ODEs, a pulse-domain linear & quadratic program solver, a min-sum solver and LDPC decoder, and a test application for receiving RF pulses from a wireless pulse-based EMG sensor. These applications show the versatility of the pulse processing paradigm and demonstrate real performance benefits that compete with state-of-the-art fixed-point or analog systems.

5.1 Sigma Delta Event Simulator

Pulses processing occurs in continuous-time so that standard discrete-time numerical methods are not an exact simulation. Instead, a special purpose discrete-event simulator was developed as part of this thesis that exactly simulates continuous-time digital systems on a general purpose computer. The Sigma Delta Event Simulator (SDES) was written in C for Linux and OS X PCs with an emphasis on efficiency and portability for both desktop and embedded applications. SDES enables study of large-scale pulse processing systems by managing time-stamped tokens passed between operation modules on a graph. A library of modules is included with operations such as pulse-domain multiplication, piecewise-linear functions, filtering, etc... Each module has ports which can be interconnected via wires for exchanging positive and negative pulses.

The SDES simulation is governed by a global priority queue which maintains a sorted list of times that pulses arrive at modules. Simulation commences by examining the next pulse event scheduled to occur and calling the destination module's handler function. This function updates the module's internal state and optionally generate new output pulses by adding new events back to the queue. The new events are either inserted at the top of the queue or at some time in the future–e.g., in the case of a delay module. After the handler returns, the next pulse event is read off of the global priority queue. In this way, events are handled efficiently and with floating-point time resolution. On a modern desktop, processing rates of about 100 million pulses per second are attained. Currently the priority queue is implemented with a splay tree in software but a hardware queue would enable greater performance. Initial results suggest that a general purpose pulse-processing DSP might be feasible solely by performing discrete-event simulation.

In SDES, various input/output modules are provided: for reading and writing files, performing separate discrete-time DSP, connecting to MATLAB and interacting with the sound card for real-time audio processing. A field-programmable gate array platform was also developed to implement parallel pulse processing logic. The desktop and FPGA interact semi-asynchronously by sending pulse event codes over a custom USB interface. SDES is released under an open source license and is available at `http://web.mit.edu/mrtn/www/`

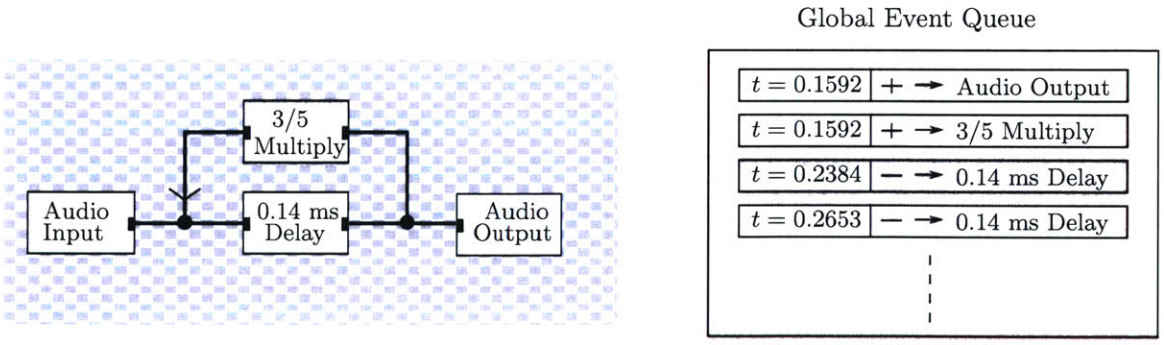
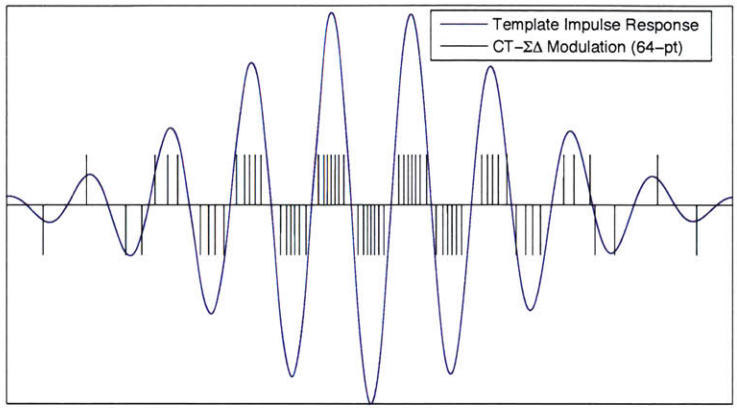


Figure 30: The ΣΔ event simulator (SDES) was developed for researching CT-ΣΔ and Δ pulse processing.

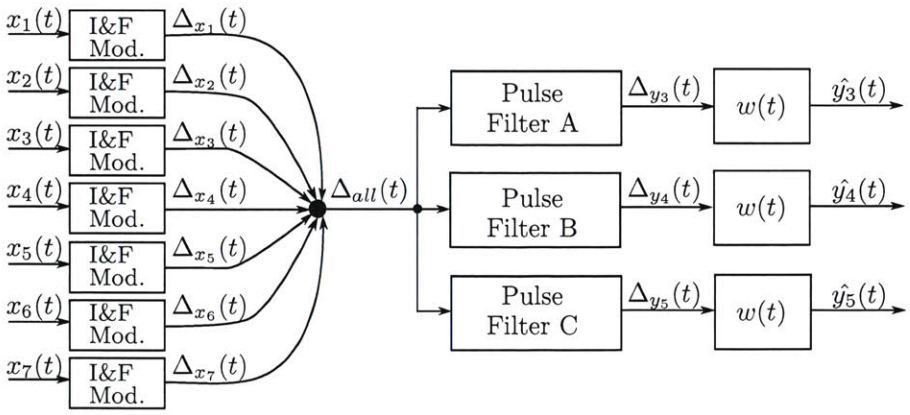
5.2 Filter Bank

A filter bank is simulated in SDES to demonstrate pulse-domain filter performance. Three pulse-domain FIR filters are constructed. Each consists of a continuous-time delay line with 64 taps. The tap time positions are designed to give a bandpass response. The design begins with a continuous impulse response template built from a simple Gaussian-windowed sinusoid. This waveform is then modulated into a sequence of signed pulses using a I&F modulator. An example is shown below. The time between these pulses defines the tap positions in a pulse-domain band-pass filter. Three different template sinusoid frequencies are used to generate three different bandpass filter bands.



To form an input test signal, seven different input sinusoids, $\{x_1(t), x_2(t), \ldots, x_7(t)\}$ are generated. Each are converted to the pulse-domain by I&F modulator. All seven pulse signals are merged into a single stream of indistinguishable pulses, $x_{all}(t)$ which serves as the test input to the filters. Three different sinusoids are within the passband of a unique pulse-domain filter A, B and C. The rest are out of band and filtered out by the filters.

After filtering, simple reconstruction from the output pulse streams is performed by computing the moving-average with a rectangular window function, $w(t) = \frac{u(t)-u(t-W)}{W}$. The reconstruction should approximately recover a unique sinusoid from each band-pass filter.



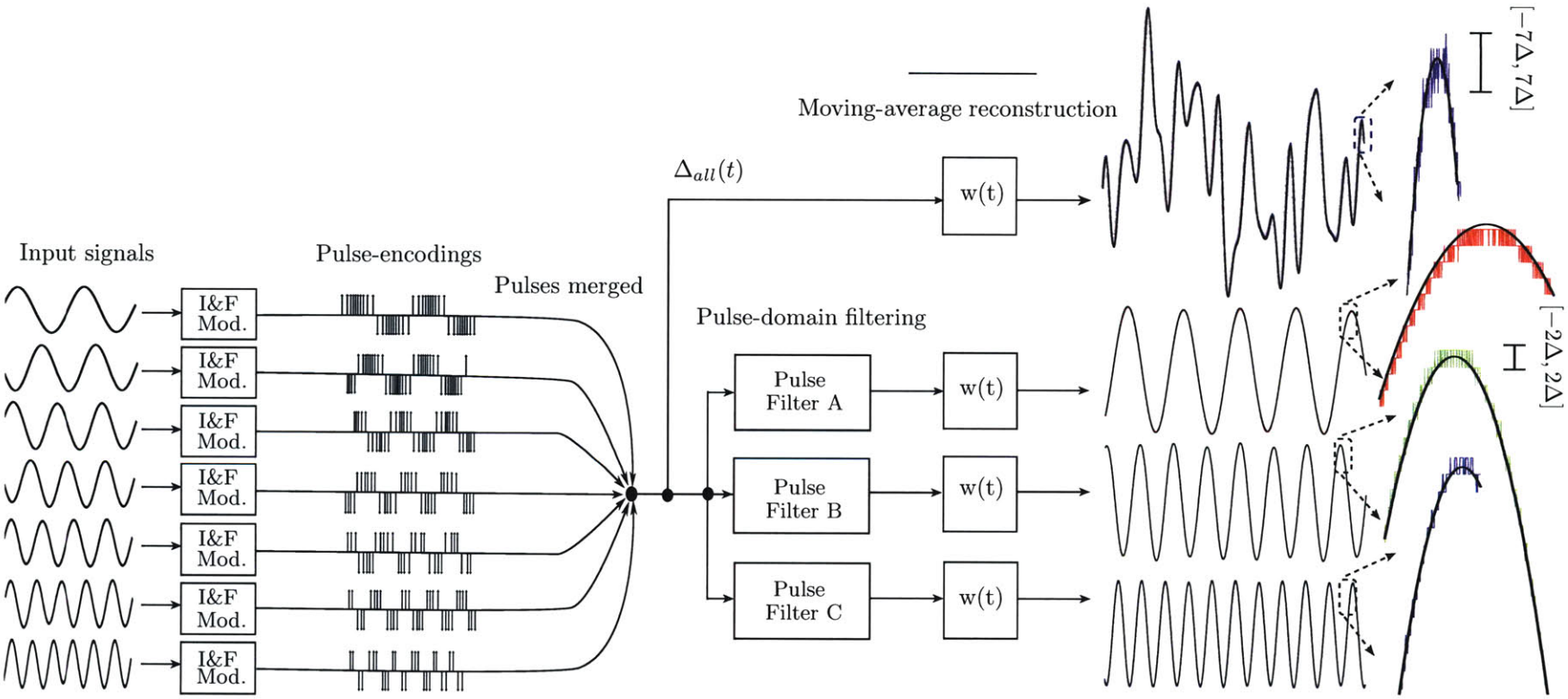


Figure 31: A pulse-domain filterbank example. Seven sinusoids are passed into I&F modulators and the results are merged into a single pulse stream. Three components are separated again by pulse-domain bandpass filters.

For comparison, a floating-point discrete-time filter bank is also constructed using the template impulse responses. These outputs are filtered by the same $w(t)$ window functions to produce the ideal results for comparison. The pulse-domain filter bank output moving-averages are shown above (colored) and superimposed on the corresponding floating point result (black). Visual comparison between the reconstructed pulse-domain filter outputs and the ideal filter outputs show a close match. It is confirmed that the outputs are bounded to within $[-2\Delta, 2\Delta]$ of the ideal outputs. The reconstruction of $\Delta_{all}(t)$ is also shown superimposed on the moving-average of the original sum of seven sinusoids. Notice that the reconstruction error bounds are wider before filtering $[-7\Delta, 7\Delta]$ than after. Using a different reconstruction filter for $w(t)$ gives improved results.

5.3 The Min-sum Algorithm

Min-sum is a powerful message passing algorithm with applications in speech recognition, DNA sequencing, medical diagnosis, digital error correction and computer vision[54]. The algorithm was first established by Michael Tanner[55] and later recognized by Niclas Wiberg[56] as a direct generalization of the soft-output Viterbi algorithm–a longstanding workhorse in communication systems. The Viterbi algorithm[57] itself is an efficient method for maximum-likelihood sequence detection of latent variables in discrete-time models observed with memoryless noise. Min-sum expands on this model to the efficient computation of maximum a-posteriori estimates of general high-dimensional distributions that factor into a product of local functions. Indeed, it is one of the best methods known for many complex inference tasks. Min-sum[3] has a straightforward implementation in the pulse-domain by using the operations for *min* and *sum*. Reduced hardware complexity and fundamental improvements in the algorithm's performance are demonstrated.

Min-sum seeks the arguments to a global minimization problem. For some problems, the algorithm is guaranteed to find the global minimum while for others, it may find an approximate solution. In general, we consider an arbitrary function of N variables where we seek the values for $x_1, x_2, \ldots, x_N$ that minimize F,

$$(x_1^*, x_2^*, \ldots, x_N^*) = \underset{x_1, x_2, \ldots, x_N}{\operatorname{argmin}} F(x_1, x_2, \ldots, x_N) \tag{11}$$

We will consider situations where the variables are defined over discrete domains. This problem is NP-complete in general–it requires testing all possible value combinations for the variables and the computational complexity is proportional to the product of the domain size in each variable. However, if the function can be decomposed into a sum of lower-dimensional functions, the computation may be simplified. For example, suppose the function decomposes as,

$$\begin{aligned} &F(x_1, x_2, x_3, x_4, x_5) = \\ &f_A(x_1, x_2) + f_B(x_2, x_3) + f_C(x_2, x_4, x_5) + f_1(x_1) + f_2(x_2) + f_3(x_3) + f_4(x_4) + f_5(x_5) \end{aligned}$$

Here functions f_A, f_B, f_C depend on various subsets of the variables while functions $f_1, \ldots, f_5$ each depend on their associated single variable. In this particular case, a minimization can be simplified by reordering and distributing the *mins* over the *sums* as far to the right as possible and evaluating inner terms first,

$$\min_{x_1,x_2,x_3,x_4,x_5} (f_A(x_1, x_2) + f_B(x_2, x_3) + f_C(x_2, x_4, x_5) + f_1(x_1) + f_2(x_2) + f_3(x_3) + f_4(x_4) + f_5(x_5)) =$$

$$\min_{x_1} \Bigg(\underbrace{\min_{x_2} \Bigg(f_A(x_1, x_2) + \underbrace{\min_{x_3} (f_B(x_2, x_3) + f_3(x_3))}_{\mu_{f_B \to x_2}(x_2)} + \underbrace{\min_{x_4, x_5} (f_C(x_2, x_4, x_5) + f_4(x_4) + f_5(x_5))}_{\mu_{f_C \to x_2}(x_2)} + f_2(x_2) \Bigg)}_{\mu_{f_A \to x_1}(x_1)} + f_1(x_1) \Bigg)$$

Intermediate functions in the calculation are highlighted. For example, the quantities denoted $\mu_{f_B \to x_2}(x_2)$ and $\mu_{f_C \to x_2}(x_2)$ are each functions of x_2 alone. If the variable x_2 is defined over a discrete domain of size M_2, each function can be stored as a single vector in $\mathbb{R}^{M_2}$ that is then passed to the next minimization.

The argument to the final minimization over x_1 gives the argument for the global minimization, x_1^*. The distribution of the *mins* can then be rearranged so that minimization of a different variable appears last. The rest of the arguments can be obtained in this fashion. To further reduce computation, intermediate $\mu_{f \to x_i}(x_i)$ calculations from before are reused. With these techniques, overall computation is significantly less than an explicit minimization of F over the entire domain.

[3] Also, Pearl's sum-product belief propagation algorithm[58] is closely related to min-sum by shared semiring properties[59]. However, the need for multiplication in sum-product complicates a pulse domain implementation.

The min-sum algorithm is a simple way of coordinating the reuse of these intermediate calculations as with dynamic programming. The process can be represented by local computations with message passing on a graphical representation of the function called a *factor graph* (see [60]). In a factor graph , each low-dimensional function is represented by a square (called a *factor node*) and each variable is represented by a circle (called a *variable node*). Edges are placed between variable nodes and factor nodes according to the functions in which the variables appear. The result is a bipartite graph that completely represents the function F. For the given example the factor graph is,

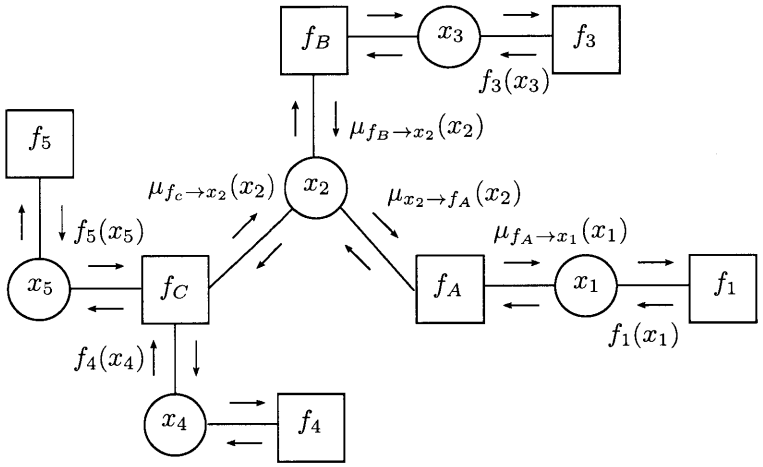


Figure 32: A factor graph representation an example function F with messages passed between nodes.

Min-sum is a message passing algorithm defined over the factor graph that consists of passing messages between nodes in both directions over every edge. Messages are passed from factors to variables (denoted $\mu_{f \to x_j}(x_j)$) and from variables to factors (denoted $\mu_{x_j \to f}(x_j)$). Each message is a function of the single variable it involves and can be represented by a vector in $\mathbb{R}^{M_j}$ when the variable x_j is defined over a discrete domain of size M_j. Messages are calculated locally at each node based on their inputs.

The method for computing the message from a given node depends on the node type and the previous input messages to that node. An output message from a variable node is computed by summing all incoming messages from neighbors of x (denoted $N(x)$) but excluding the output message's recipient (denoted $\backslash\{f\}$).

$$\mu_{x_j \to f}(x_j) = \sum_{\hat{f} \in N(x_j) \backslash \{f\}} \mu_{\hat{f} \to x_j}(x_j) \tag{12}$$

(In the special case where the variable node has only two neighbors, it simply passes messages through.)

A factor node forms its output messages differently. It begins by summing the previous input message functions (again, excluding the recipient, $\backslash\{x_j\}$). But now each message is over a different domain–if there are K variables attached to a factor, the resulting sum is a K-dimensional table of values instead of a single vector. This table is then added element-wise to the factor's local function (also a K-dimensional table of values) and a minimization[4] is performed over all variables except for the recipient variable x_j. An output message from factor f to variable x_j is then a function of x_j only,

$$\mu_{f \to x_j}(x_j) = \min_{N(f) \backslash \{x_j\}} \left(f(\{N(f)\}) + \sum_{\hat{x} \in N(f) \backslash \{x_j\}} \mu_{\hat{x} \to f}(\hat{x}) \right) \tag{13}$$

(When a factor is connected to a single variable, it sends its own function: $\mu_{f \to x_i}(x_i) = f_i(x_i)$.)

[4]It is assumed that there is a unique minimum.

It is sufficient to describe the message passing as an iterative algorithm. Messages are initialized to uniform constants and then for every iteration, each node receives unique messages from each of its neighbors and sends a unique message to each of its neighbors. The iterations can be written as successive substitutions of state into a multidimensional piecewise-affine function F consisting of all the message update equations. The state $\boldsymbol{\mu}^{(t)}$ contains the messages in both directions for every edge in the graph at time t.

$$\boldsymbol{\mu}^{(t+1)} = F(\boldsymbol{\mu}^{(t)}) \tag{14}$$

If the factor graph has a tree topology (i.e., singly-connected), the messages will converge to a unique fixed-point $\boldsymbol{\mu}^*$ after the number of iteration reaches the diameter of the graph. The fixed-point solves the minimization problem and gives the arguments that achieve the global minimum. The global minimum can be recovered by computing the sum of the messages being sent to any given variable and choosing the minimum value. This minimization is equivalent to the outer min in the complete minimization over F and gives the the argument for that variable in the global minimization,

$$x_j^* = \operatorname*{argmin}_{x_j} \sum_{\hat{x} \in N(x_j)} \mu^*_{\hat{x} \to x_j}(x_j) \tag{15}$$

All x_j^* can be found by performing this calculation for each variable. A proof of convergence to the exact solution for the tree topology is given in [56].

For many important problems such as computer vision and error correction, the function has a factor graph that is *not* singly-connected and contains loops. In these cases, the min-sum iterations can be applied regardless but convergence to a fixed-point is not guaranteed. However, it is empirically observed that when min-sum does converge, it is often a global minimum. Indeed, according to Weiss [61] fixed-points are a 'neighborhood minimum' (stronger than a local minimum). For many error correction applications, if one only considers convergent cases, min-sum decoding even outperforms the competing sum-product method of decoding. Unfortunately, min-sum is less likely to converge so that overall performance is inferior.

Consequently, a number of adjustments have been proposed to improve convergence of min-sum on loopy graphs. For example, rather than updating all messages in parallel as in (14), they can be computed according to a schedule or by randomly updating messages for different nodes. Another powerful method is successive relaxation or 'damping'. Successive relaxation is a general method for improving the iterative solution of nonlinear equations so that the evolution follows a continuous-time system more closely. Here, the min-sum update is modified to use a convex combination of the previous two iterations,

$$\boldsymbol{\mu}^{(t+1)} = F(\beta \boldsymbol{\mu}^{(t)} + (1-\beta)\boldsymbol{\mu}^{(t-1)})$$

Any fixed-points of this equation are also fixed-points of (14). Successive relaxation is known to significantly improve convergence of min-sum if β is small (i.e., underrelaxation) by reducing overshoot. However, as a consequence of the smaller steps, convergence is slowed and the number of iterations greatly increases.

Considering that successive relaxation improves min-sum convergence by more closely approximating a continuous-time system, it is natural to consider the direct implementation of min-sum in continuous-time. A number of such implementations have been demonstrated using analog circuits (e.g., [62, 63, 64]). In practice however, these analog decoders do not scale well beyond very small systems; manufacturing variances and other analog issues compound to limit the size of the graph.

Simulation of continuous-time min-sum for large LDPC decoders was considered by Saied Hemati in [65]. Results show that the hypothetical decoder can gain 1dB and maintains a much lower error floor compared with discrete-time floating-point LDPC decoders. Unfortunately, these gains are not realized in practice due to shortcomings in analog VLSI.

As an alternative, we propose a continuous-time digital pulse processing implementation. Min-sum has a number of properties that make a pulse-domain implementation appear promising. The algorithm is known to be robust to occasional random transient errors in the messages (see Varshney [66]) allowing for occasional

pulse collisions and aggressive hardware simplifications when computing with the representation. I&F is also very accurate at the low frequencies observed in the time evolution of min-sum messages. Any error in the representation is effectively bounded. In addition, min-sum is inherently parallel and has a natural implementation directly in hardware. With fixed-point arithmetic, a full-parallel implementation requires unreasonable logic, routing and power consumption. Pulse-domain operations are much lower complexity and the single-wire data path reduces routing enough to make a fully parallel design viable.

5.3.1 A Pulse-domain Min-Sum Algorithm

To construct a pulse-domain min-sum solver, pulse-domain min and sum circuits are directly connected to form a fully parallel system. Individual circuits are used for each variable node and factor node. The variable node circuits compute equations (12) and the factor node circuits compute equations (13).

Min-sum is often robust to moderate pulse loss so that unique bidirectional methods can be employed. Until now, pulse processing has been described in terms of operations taking input pulses and producing output pulses; each wire has an implicit direction with one circuit a source and the other a destination. This is a common design principle in digital systems, but it is an unnecessary restriction. In reality, circuits can communicate in both directions over a single wire.

Frequently, bidirectional communication is achieved by a tri-state bus, with the direction of communication coordinated by a clock or asynchronous handshaking. With I&F, the coordination can be removed. The implementation is now straightforward: each circuit sharing the bus normally listens and receives pulses when the bus is pulled high briefly by other circuits. When transmitting its own pulse, it disables its receiver to prevent counting its own pulse and then pulses the bus high for detection by others. During the transmission, everyone is blind to other pulses and random loss is potentially incurred.

5.3.2 Variable Nodes

Bidirectional communication can be leveraged further to simplify the variable node computation. Merging pulses on a single wire called the *variable bus* gives all participating circuits the full pulse-domain sum of input messages. A circuit's own signal can be subtracted off by not including its own pulses. This effectively computes the variable node output message (12) for all directions. Figure 33 depicts the process. We denote the continuous-time I&F encoding of the kth element in a message from f to x_j by $\Delta^{(k)}_{f\to x_j}(t)$.

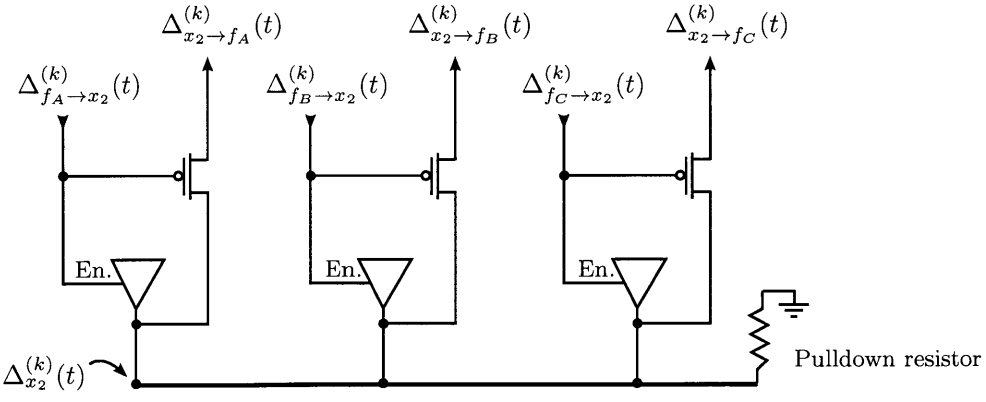


Figure 33: A pulse-domain computation for element k in variable node x_2 using a shared tri-state bus. An actual asynchronous tri-state implementation requires care to prevent glitches in the pulse block.

This distributed computation replaces what normally amounts to 5 additions with a single variable bus

wire and 3 tri-state buffers. The circuit is repeated with a variable bus wire for each $k \in \{1, \ldots, M_2\}$. This is significantly reduced compared with the $8 \times M_2$ or more wires required with a fixed-point implementation.

The 'variable node portion' of the implementation will be associated only with the variable bus wires. The tri-state circuitry will be incorporated in the factor node portion. With no local point of computation, input wires need not converge to a single variable node computation module (only connect to the variable bus). This can significantly reduce routing complexity.

5.3.3 Factor Nodes

The factor node computation can take advantage of bidirectional signaling as well. To describe the circuit, a special *min-broadcast* element is introduced. Each of the three interfaces on the element are bidirectional. It computes the pulse-domain min operation in the forward direction, but broadcasts pulses in the reverse direction:

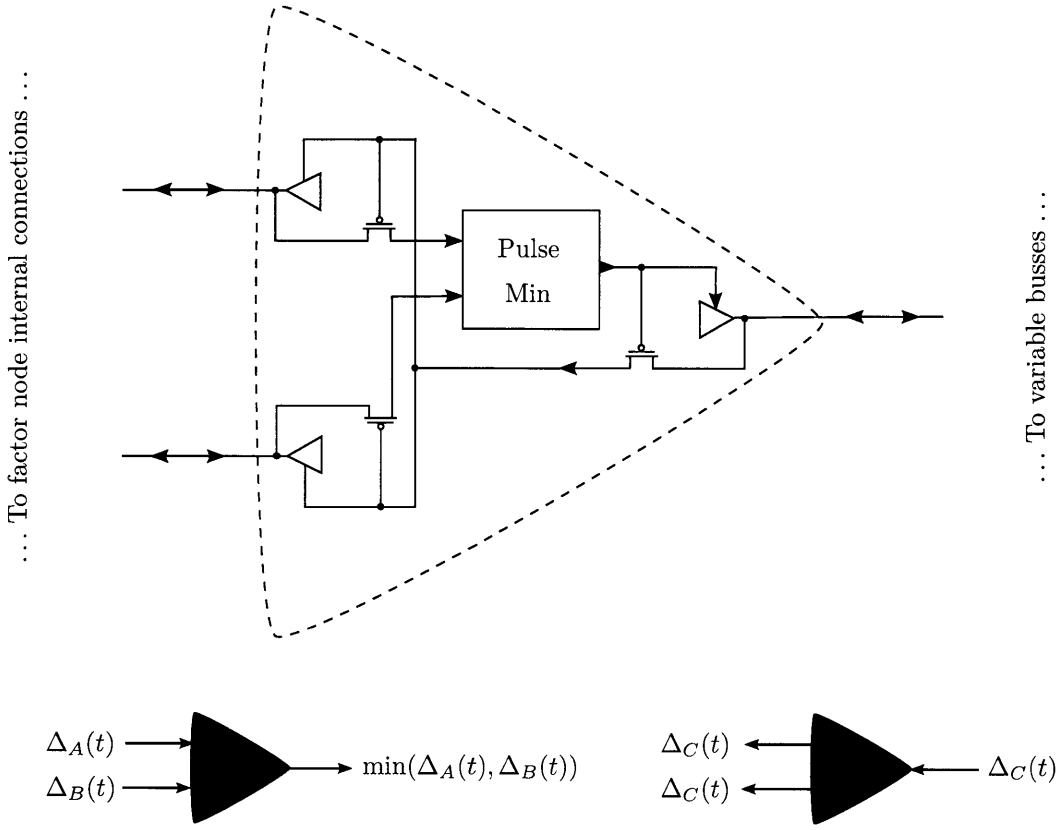


Figure 34: The min-broadcast element performs the pulse-domain min operation in the forward direction and broadcasts copies of pulses in the reverse direction.

In general, $(N : 1)$ degree min-broadcast elements can be constructed that still compute *min* in one direction and broadcast in the other. The min-sum computations for factor nodes uses min-broadcasts with degree depending on the sparsity of the factor function.

These min-broadcast elements are the connected together to give the min-sum calculation for factor nodes defined in equation (13). A single wire is defined for each finite entry in the factor's function. A pulse modulator sends pulses to this wire using the constant value associated with the entry in the function. The

wire is also shared with min-broadcast components that are associated with each element in each variable's domain. A connection is made when a corresponding element participates in the wire's entry. For example, the wire for $f_C(x_2 = q,\ x_4 = r,\ x_5 = s)$ is connected to the min-broadcast components that connect to variable bus wires $\Delta_{x_2}^{(q)}(t)$, $\Delta_{x_4}^{(r)}(t)$ and $\Delta_{x_5}^{(s)}(t)$.

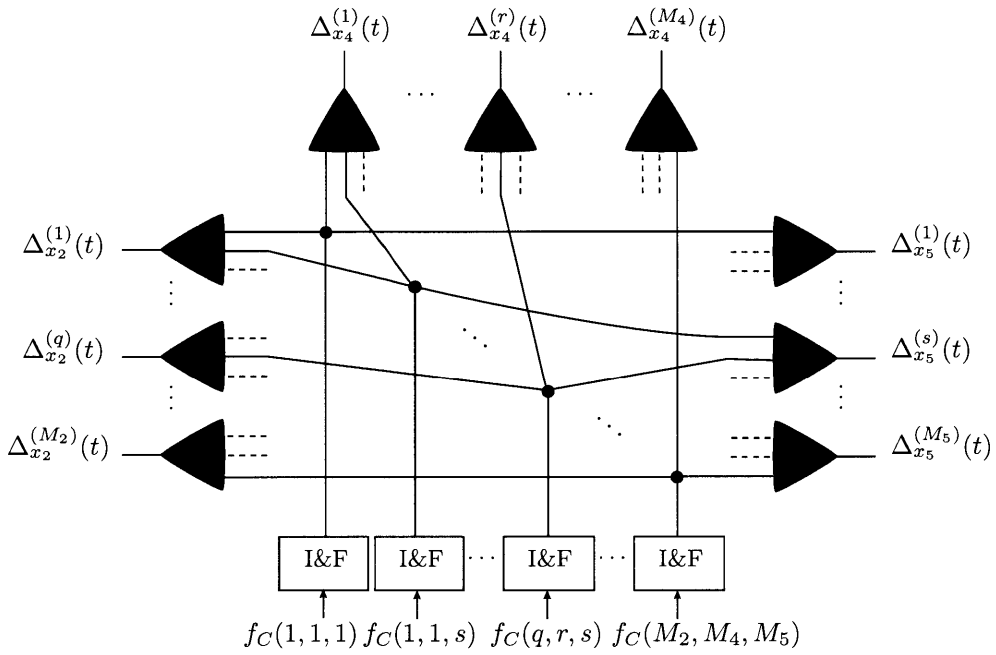


Figure 35: A pulse-domain computation for the factor node $f_C(x_2, x_4, x_5)$. Each internal factor bus corresponds to a finite entry in the factor function.

If an entry in the function is infinite, it does not require a wire or I&F modulator. This value will never be the minimum of any calculation and the wire can be omitted entirely. In some cases where min-sum is performed on factor graphs with loops, the values encoded on the variable busses may still become unbounded even with finite functions. Offset normalization is the common solution. This can also be performed in the pulse domain by subtracting,

$$\min_k \Delta_{x_i}^{(k)}(t)$$

from each of the x_i variable bus wires, $\Delta_{x_i}^{(k)}(t)$.

By connecting factor node computation modules together via variable busses, a continuous-time pulse-domain min-sum solver is constructed. In steady state, the minimum values encoded on the variable busses determine min-sum's (possibly approximate) solution to equation (11). In practice, this pulse-domain system converges much more readily than a discrete-time floating point implementation. As an example, an application of min-sum to forward error correction is demonstrated with a fully-parallel FPGA implementation containing thousands of factors and variable busses.

5.4 Forward Error Correction

Min-sum is a powerful approach to finding maximum a posteriori (MAP) assignments in probabilistic models. This is particularly useful for decoding error-correcting codes. For example, in a coding model, a codeword $X = (x_1, \ldots, x_N)$ is selected from a code $\mathcal{C}$ and transmitted over a channel to form a received sequence $Y = (y_1, \ldots, y_N)$. Given Y, the MAP assignment for the codeword X is,

$$\begin{aligned} X^* &= \underset{X}{\operatorname{argmax}} \left(P(X|Y)\right) \\ &= \underset{X}{\operatorname{argmax}} \left(\frac{P(Y|X)P(X)}{P(Y)}\right) \\ &= \underset{X}{\operatorname{argmax}} \left(P(Y|X)P(X)\right) \end{aligned} \tag{16}$$

$P(Y|X)$ is determined by the type of channel. We know that for a memoryless channel the distribution factors,

$$P(Y|X) = \prod_{i=1}^{N} P(y_i|x_i)$$

$P(X)$ is the a priori distribution for the transmitted codewords. If this distribution is uniform over the codewords in $\mathcal{C}$, it is proportional to the following indicator function,

$$P(X) \propto \mathbb{1}\{(x_1, \ldots, x_N) \in \mathcal{C}\}$$

Here we consider low-density parity check (LDPC) codes. LDPC is a linear block code over $GF(2)$ that was invented by Robert Gallager in 1960 and enable communication very close to the Shannon limit over AWGN channels[67]. Valid codes are those that live in the null-space of a given sparse matrix H. For an example H, the a priori distribution is,

$$P(X) \propto \mathbb{1}\left\{ \underbrace{\begin{pmatrix} 1 & 0 & 1 & 1 & 0 & 0 & \ldots & 0 & 0 \\ 1 & 1 & 0 & 0 & 1 & 0 & \ldots & 0 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 1 & 0 & 1 & 1 & 0 & \ldots & 0 & 0 \end{pmatrix}}_{H} (x_1, \ldots, x_N)^T = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{pmatrix} \right\}$$

This indicator function factors into a product of indicators for the parity check in each row,

$$P(X) \propto \mathbb{1}\{x_1 \oplus x_3 \oplus x_4 = 0\} \cdot \mathbb{1}\{x_1 \oplus x_2 \oplus x_5 \oplus x_N = 0\} \ldots \mathbb{1}\{x_2 \oplus x_4 \oplus x_5 = 0\}$$

Because log is monotonic and $\max(a, b) = -\min(-a, -b)$, equation (16) is equivalent to,

$$X^* = \underset{X}{\operatorname{argmin}} \left(-\log P(Y|X) - \log P(X)\right)$$

Now by defining the following functions, the LDPC problem is brought into the form defined earlier for the general min-sum algorithm,

$$f_i(x_i) = -\log P(y_i|x_i)$$

$$f_A(x_1, x_3, x_4) = -\log(\mathbb{1}\{x_1 \oplus x_3 \oplus x_4 = 0\}) = \begin{cases} 0, & x_1 \oplus x_3 \oplus x_4 = 0 \\ \infty, & else \end{cases}$$

$$f_B(x_1, x_3, x_4) = -\log(\mathbb{1}\{x_1 \oplus x_2 \oplus x_5 \oplus x_N = 0\}) = \begin{cases} 0, & x_1 \oplus x_2 \oplus x_5 \oplus x_N = 0 \\ \infty, & else \end{cases}$$

etc...

Finding the MAP assignment for the codeword becomes,

$$X^* = \underset{X}{\operatorname{argmin}} \left(\sum_{i=1}^{N} f_i(x_i) \; + \; f_A(x_1, x_3, x_4) + f_B(x_1, x_2, x_5, x_N) + \ldots + f_M(x_2, x_4, x_5) \right) \tag{17}$$

All functions (and min-sum messages) contain negative log of probabilities. These are non-negative values because $[0,1] \xrightarrow{-\log} [0,\infty]$. All signals can therefore be represented entirely with positive pulses.

The factor output message calculations are particularly straightforward for the parity-check function because functions $f_A, f_B, \ldots, f_M$ contain only the values ∞ or 0. Any infinite values can be omitted from the minimizations while the zeros do not require I&F modulators. For example, the $f_A(x_1, x_3, x_4)$ is a parity check factor of degree 3. All 6 output messages are calculated as follows,

$$\begin{aligned}
\mu_{f_A \to x_1}(0) &= \min(\mu_{x_3 \to f_A}(0) + \mu_{x_4 \to f_A}(0),\ \mu_{x_3 \to f_A}(1) + \mu_{x_4 \to f_A}(1)) \\
\mu_{f_A \to x_1}(1) &= \min(\mu_{x_3 \to f_A}(1) + \mu_{x_4 \to f_A}(0),\ \mu_{x_3 \to f_A}(0) + \mu_{x_4 \to f_A}(1)) \\
\\
\mu_{f_A \to x_3}(0) &= \min(\mu_{x_1 \to f_A}(0) + \mu_{x_4 \to f_A}(0),\ \mu_{x_1 \to f_A}(1) + \mu_{x_4 \to f_A}(1)) \\
\mu_{f_A \to x_3}(1) &= \min(\mu_{x_1 \to f_A}(1) + \mu_{x_4 \to f_A}(0),\ \mu_{x_1 \to f_A}(0) + \mu_{x_4 \to f_A}(1)) \\
\\
\mu_{f_A \to x_4}(0) &= \min(\mu_{x_3 \to f_A}(0) + \mu_{x_1 \to f_A}(0),\ \mu_{x_3 \to f_A}(1) + \mu_{x_1 \to f_A}(1)) \\
\mu_{f_A \to x_4}(1) &= \min(\mu_{x_3 \to f_A}(1) + \mu_{x_1 \to f_A}(0),\ \mu_{x_3 \to f_A}(0) + \mu_{x_1 \to f_A}(1))
\end{aligned} \tag{18}$$

These computations are all accomplished by connecting together 6 min-broadcast elements as shown below. Each of the four internal wires join together one of the four valid configurations of a 3-input parity check: $(0,0,0)$, $(0,1,1)$, $(1,0,1)$ and $(1,1,0)$.

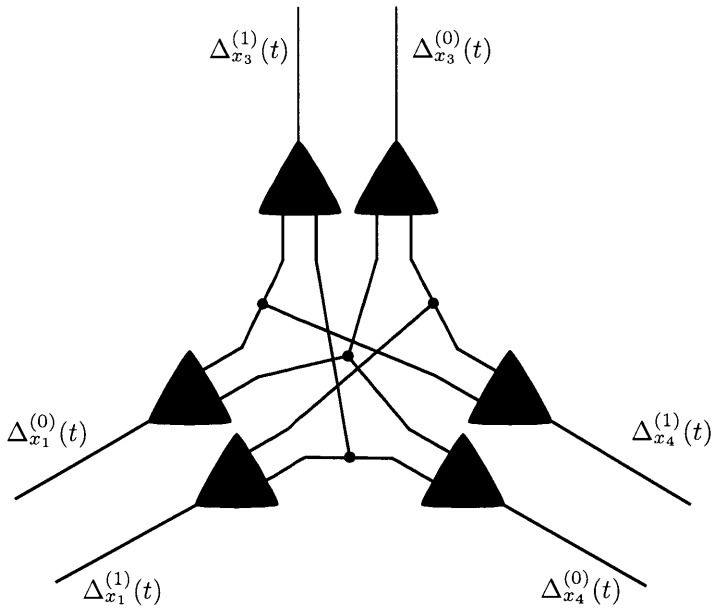


Figure 36: Pulse-domain 3-input parity-check factor node for $f_A(x_1, x_3, x_4)$.

Due to a well-known symmetry, any parity-check function with greater than 3 inputs can be factored into multiple 3-input parity-check functions. For example four, five and six-input factor can be constructed as in Figure 37.

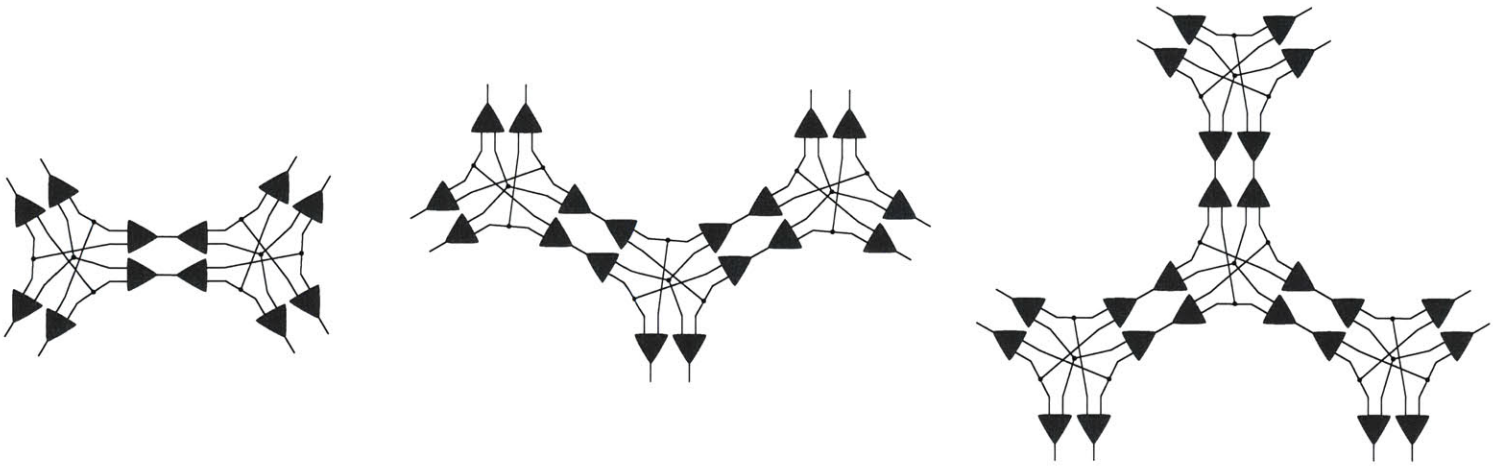


Figure 37: Pulse-domain 4-input, 5-input and 6-input parity-check factor nodes.

These party-check factors are then connected by wires (variable busses) when they share a variable.

A final component is necessary to complete the LDPC decoder. As with all min-sum implementations on factor graphs with loops, offset normalization is required to prevent signals from growing unbounded. Best performance is observed when the offset-normalization is applied to each signal sent from a variable bus to a factor,

$$\mu'_{x_i \to f}(1) = \mu_{x_i \to f}(1) - \min\left(\mu_{x_i \to f}(0), \mu_{x_i \to f}(1)\right) \tag{19}$$

$$\mu'_{x_i \to f}(0) = \mu_{x_i \to f}(0) - \min\left(\mu_{x_i \to f}(0), \mu_{x_i \to f}(1)\right) \tag{20}$$

Both calculations can be performed entirely with a single simple passive pulse machine shown below. Input pulses from the encodings for $\mu_{x_i \to f}(1)$ cause transitions labeled (1) while inputs from $\mu_{x_i \to f}(0)$ cause transitions labeled (0). Output pulses are given for the pulse encoding of $\mu'_{x_i \to f}(1)$ and $\mu'_{x_i \to f}(0)$. The number of states should equal the number of other factors that share the variable bus for the bounds to be appropriate.

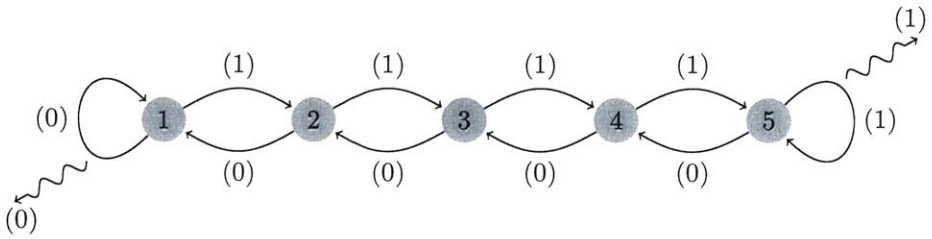


Figure 38: A PPM for offset normalization of messages to parity-check factors.

The inputs messages come from the factors $f_i(x_i) = -\log P(y_i|x_i)$. In a communications setting, these (positive) values are computed from the received (noisy) modulation symbols. If the scheme is BPSK or QPSK over a memoryless channel with additive white Gaussian noise, the received amplitudes r_i can be used directly for both parts of these messages. Indeed,

$$\begin{aligned} f_i(0) - f_i(1) &= -\log P(y_i|x_i = 0) + \log P(y_i|x_i = 1) \\ &= -\log\left(\frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{-(r_i+1)^2}{2\sigma^2}}\right) + \log\left(\frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{-(r_i-1)^2}{2\sigma^2}}\right) \\ &= \frac{2}{\sigma^2} r_i \end{aligned}$$

If this is a positive quantity, offset-normalized values of the form (19,20) give $f_i'(0) = \frac{2}{\sigma^2} r_i$ and $f_i'(1) = 0$. Otherwise, if $\frac{2}{\sigma^2} r_i$ is negative, $f_i'(1) = -\frac{2}{\sigma^2} r_i$ and $f_i'(0) = 0$. Further, the constant of proportionality $\frac{2}{\sigma^2}$ does not affect the results of the min-sum optimization (17) so that $\pm r_i$ can be used directly. The pulse-domain input messages are then straightforward. The analog signal r_i is passed to a I&F modulator and *positive* output pulses give $\Delta^{(0)}_{f_i \to x_i}(t)$ while *negative* output pulses give $\Delta^{(1)}_{f_i \to x_i}(t)$.

The complete LDPC decoder is shown below. Input pulses are generated on the left by the modulators and distributed on the variable busses $\Delta_{x_i}(t) = \{\Delta^{(1)}_{x_i}(t), \Delta^{(0)}_{x_i}(t)\}$. Pulses on each bus are received by all of the connected parity-check factor nodes. Input pulses to the factor nodes are normalize according to equations (19,20) and then processed using the min-broadcast networks shown in Figure 37. The factor nodes may pass pulses out to other variable busses depending on the state of their internal min-broadcast elements. This process may involve occasional collisions (depending on the pulse widths) but the distortion is generally small enough to leave the decoding unaffected. Finally, modules pictured on the right listen on the variable busses and continuously determine the sign of $\Delta^{(0)}_{x_i}(t) - \Delta^{(1)}_{x_i}(t)$. The sign determines the output bits D_i that are passed to the codeword verification circuit. When the verification circuit detects a valid codeword, the decoding is stopped.

The connections between check nodes and variable busses defines the structure of the LDPC code. These single-wire connections can be fixed or determined by a programmable crossbar switch.

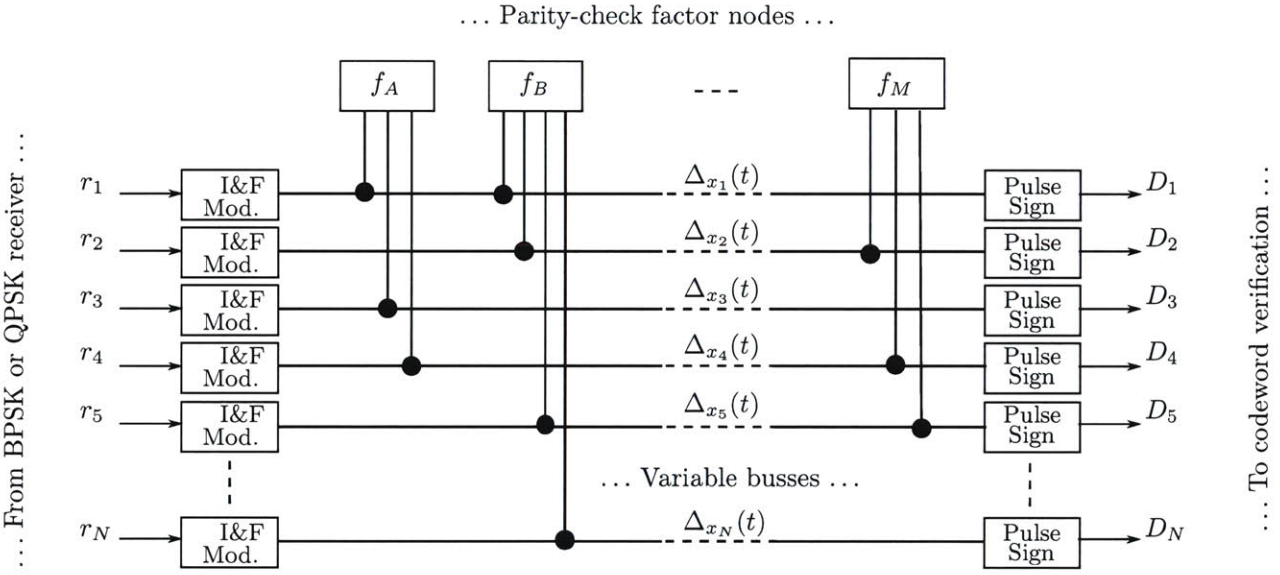


Figure 39: Overview of the fully-parallel pulse-domain LDPC decoder

5.4.1 Pulse-domain LDPC Decoder Prototypes

In order to demonstrate the performance of the pulse-domain LDPC decoder, a hardware implementation was constructed on two FPGAs, a Virtex II (XUPV2P) and a more modern Virtex V5 (XUPV5-LX110T). The older Virtex II contains internal tri-state buffers for implementing the variable busses directly using the bidirectional design. The newer Virtex V5 has greater logic resources but lacks internal tri-states and the variable busses are instead emulated with logic gates. Both approaches were tested to explore the design tradeoffs. In addition, decoders were simulated in SDES in order to analyze the effects of artificially induced timing jitter and hardware failure. Simulations using SDES were performed using supercomputing resources from the Extreme Science and Engineering Discovery Environment (XSEDE) provided by the National

Science Foundation. In all, seven decoders were built for multiple irregular WiMAX 802.16e codes, the digital video broadcasting satellite second generation (DVB-S2) code, and other codes.

5.4.2 Virtex II Implementation (internal tristates)

The smallest decoder prototype was built on a Virtex II development board. This older FPGA is fairly small, so that only a small (528,264) WiMAX 802.16e code was able to meet the resource constraints. Further, the 528 I&F modulators that generate input pulses did not fit on the FPGA with the rest of the decoder. For this reason, the modulators were simulated in real-time in SDES on a quad-core PC and output pulse events were sent to the FPGA over a custom USB interface.

Only the decoder was implemented on the FPGA. Random message generation, encoding and QPSK transmission through a memoryless AWGN channel is all performed by software on the desktop with routines adapted from code by Radford Neal[67]. The signal amplitudes returned from a simulated QPSK receiver are fed to SDES where a simulated array of I&F modulators generated pulses. For each pulse, an 11-bit code indicating the source modulator and the pulse sign is sent over the USB interface to the FPGA. On the FPGA, the rest of the decoder is implemented in a fully parallel design. A bank of comparators constantly check the input code values and generate pulses on their variable bus using internal tri-state buffers. Each variable bus is a pair of internal tri-state busses with one for positive or $\Delta_{x_i}^{(0)}(t)$ pulses and the other for negative or $\Delta_{x_i}^{(1)}(t)$ pulses. 264 I&F parity-check factor nodes of degree 6 and 7 are connected to these busses. The various offset-normalization and min-broadcast state machines asynchronously process pulses. Finally, an array of sign detectors determine the sign of the signal encoded by the pulses on each bus to form a tentative decoded codeword. These bits are passed to a tree of NXOR and AND gates that detect if they satisfy all of the parity-check constraints of the LDPC code. When a valid codeword is detected, decoding is immediately halted and the results are sent back to the computer. The computer logs whether the codeword was indeed what was transmitted and keeps bit-error rate (BER) and frame (full codeword) error rate (FER) statistics.

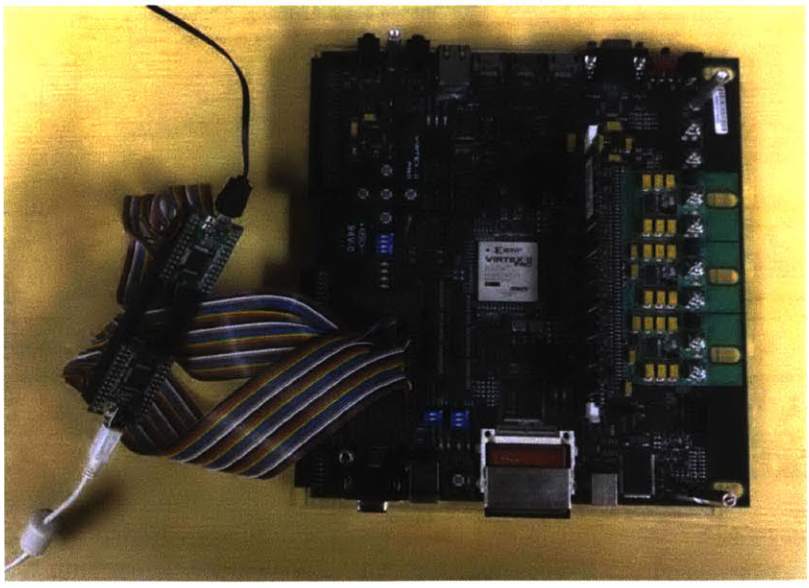


Figure 40: Virtex II LDPC decoder implementation with custom dual-USB pulse addressing interface.

5.4.3 Virtex V Implementation (emulated tristates)

On the newer FPGA, a larger (1056, 528) WiMAX 802.16e was implemented. Again, the codewords are generated on the PC but instead of sending pulse codes, 15 bit receiver LLRs are sent directly over a PCIe interface. The I&F modulators and the rest of the decoder are implemented on the FPGA. Each modulator consists of a single overclocked 16-bit accumulator stepping by its given LLR value and with the output pulses formed from the the regularly-timed carry bit. The decoder implementation otherwise mirrors the Virtex II design but with variable busses implemented with multiplexers. The PCIe interface supports decoding at approximately 1 Gbit/sec and significantly outperforms the Virtex II implementation. Still, the communication over the PCIe interface is the bottleneck limiting the decoder throughput. In principle, the design is capable of decoding at well over 100 Gbit/sec.

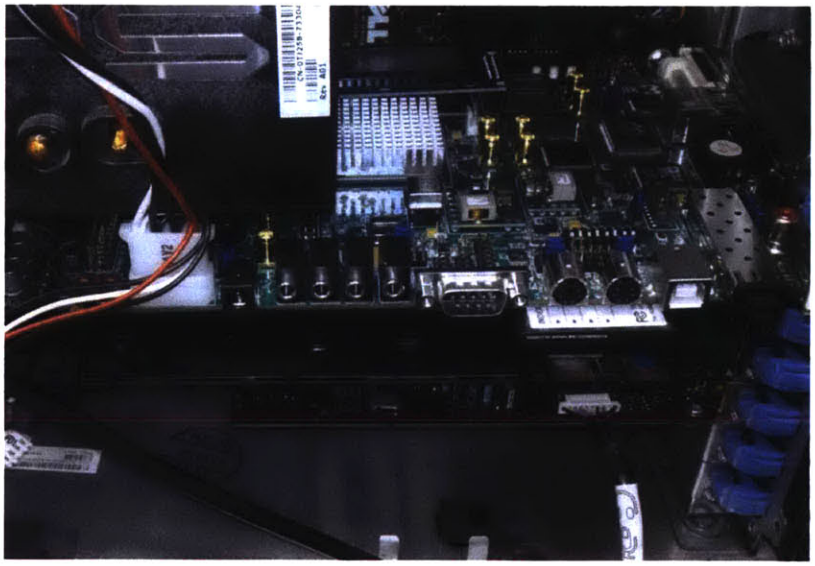


Figure 41: Virtex V LDPC decoder implementation with high speed PCIe interface.

5.4.4 Simulations

Software prototypes were also developed in SDES and enable testing of extreme conditions. For example, the introduction of additional timing-jitter, static delays or pulse loss was tested to determine design sensitivity. It was found that static delays have very little effect on the decoder performance, while severe timing jitter (on the order of the pulse rates) does increase convergence time by up to a factor of two and adversely affects decoding BER. The simulation also allows the decoding dynamics to be examined in detail. In FIgure 42 we show typical pulse activity on a few of the variable busses. Here pulses are shown during the decoding of a random (528,264) WiMAX code at a 2.5dB noise level. About four pulses per bus are output by I&F modulators on each bus during the time needed to decode the correct codeword. The rest of the pulses shown correspond to additional pulses produced by parity-check nodes. The selected subset of signals illustrates many of the variables where the received LLR's having the wrong sign (because AWGN noise has flipped the bit entirely) so that the pulses are initially of the wrong sign but are later corrected by the decoder. The signals are color-coded based on their (correct) decoded end sign.

Notice that some pulse timing collisions are observed. These are manifest as shortened pulses that may not be detected by the receiving circuitry. However, the timing collisions are found to have only small effect

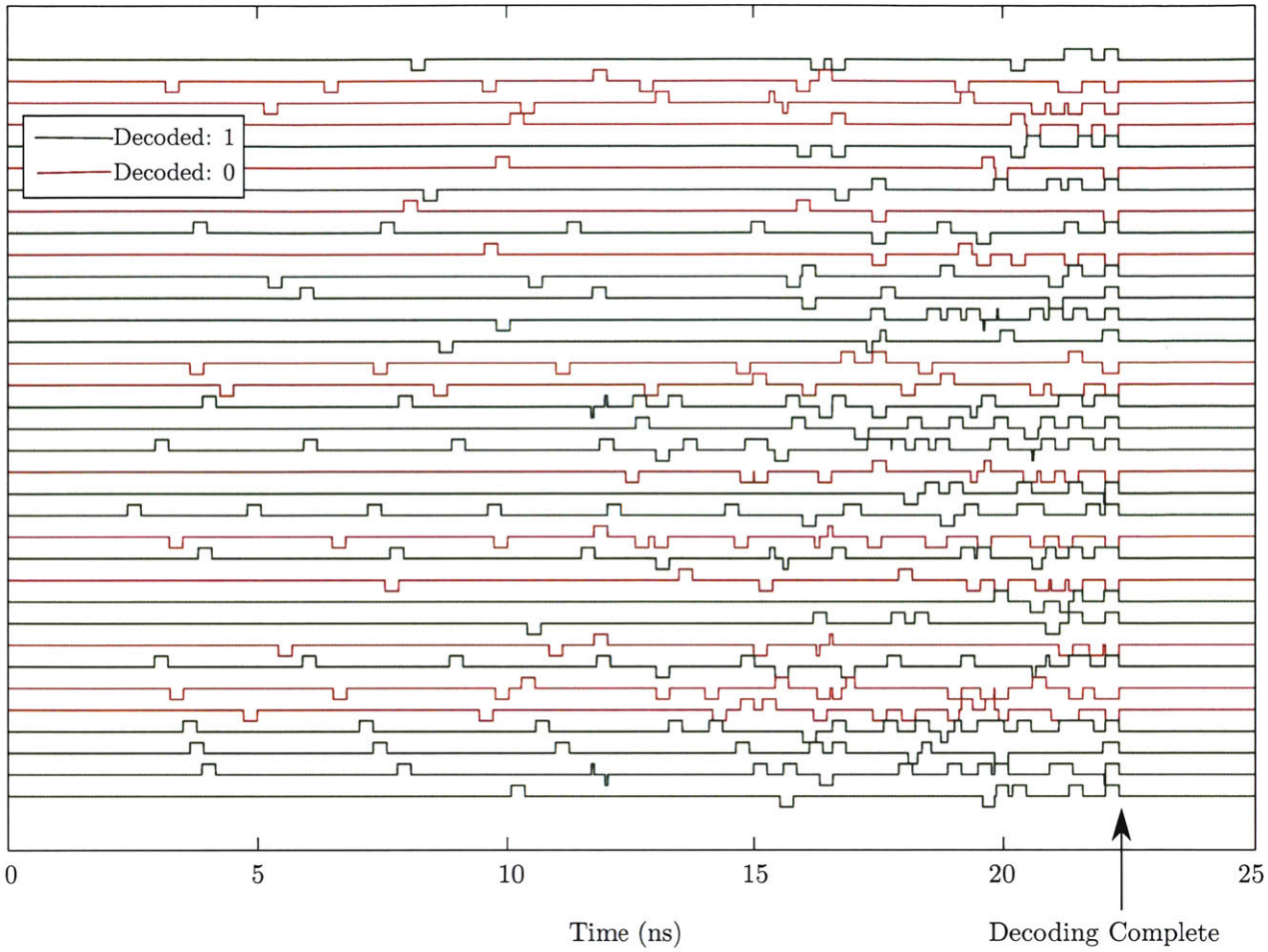


Figure 42: A subset of the 528 variable bus signals while decoding a WiMAX (528,264) code at 2.5dB AWGN.

5.4.5 Results

Bit-error-rate (BER) curves for two different LDPC codes are examined here. For the curve generation, early termination is used with each trial (i.e., if the parity check functions are all satisfied, the decoder ends immediately and the codeword is verified against the correct codeword). Otherwise, a timeout condition determines a decoding failure. For traditional discrete-time decoders, the timeout is often stated in terms of a maximum number of iterations. Here, I&F input pulses are used in lieu of iterations; if the the average number of pulses generated by the I&F modulators exceeds the threshold, the decoding is declared a failure. It is observed that the number of pulses required is very comparable to the number of iterations in a discrete-time floating-point decoder.

The BER curve for a pulse-domain min-sum decoder using a (1056,528) WiMAX 802.16e code are shown in Figure 43. To guarantee a reliable curve, each data-point required at least 100 independent codeword errors. Two other curves are shown for comparison. A discrete-time sum-product decoder with ideal floating-point arithmetic serves as the traditional benchmark. While most decoder implementations require fixed-point quantization, they generally maintain BERs near this curve. However, nearly all existing decoders are

implemented in a semi-parallel fashion where multiple clock cycles are needed for each full iteration of the algorithm. Fully-parallel designs such as this pulse-domain system are rare because the usual fixed-point implementations have logic area and routing requirements that are impractical for all but the smallest codes. However, a stochastic LDPC decoder by Tehrani, Mannor and Gross [68] is an efficient alternative and can serve as a state-of-the-art comparison in the class of fully-parallel designs. This design uses stochastic computation for variable and parity-check computations of the sum-product algorithm. The resulting circuits require significantly less logic than fixed or floating-point arithmetic and indeed requires only slightly more gates than this pulse-domain hardware implementation. The BER curve for the (1056,528) code from this paper is compared in Figure 43.

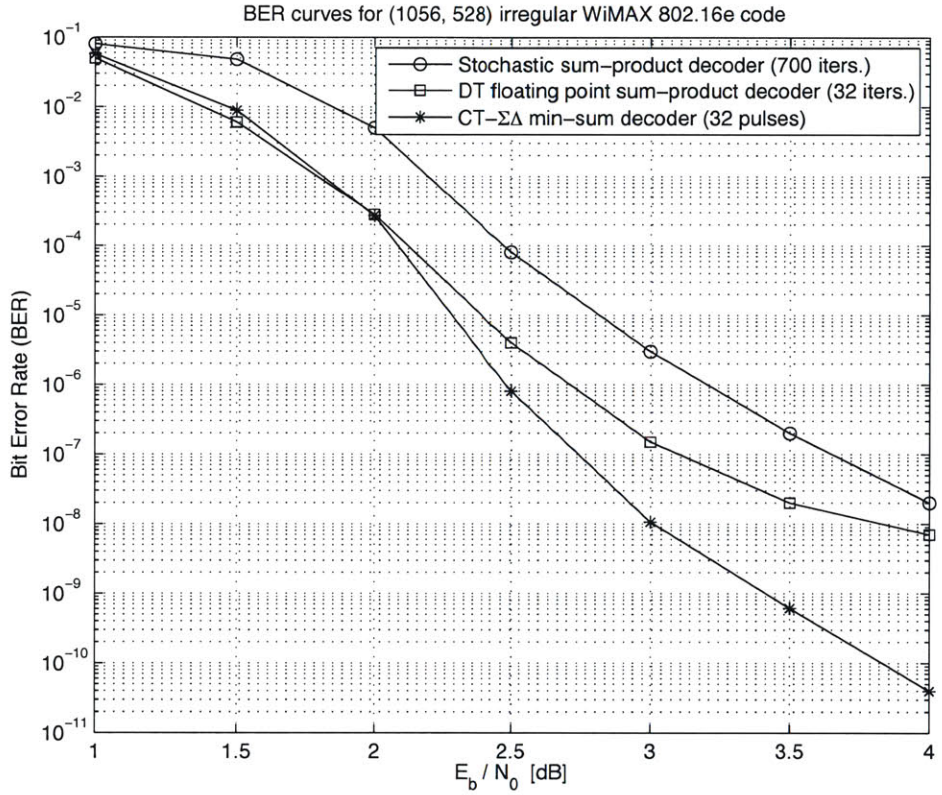


Figure 43: Bit-error rates for the pulse-domain domain min-sum decoder using a $\frac{1}{2}$-rate WiMAX 802.16e LDPC code.

It is apparent that the pulse-domain decoder exhibits at least a 1 dB gain over the stochastic decoder. The added structure and accuracy of deterministic pulse-domain encodings appears to have a distinct advantage over the slowly-converging stochastic computation. The stochastic decoder requires a maximum of 700 iterations compared with the 32 iterations with floating-point decoders or the 32 pulses with the pulse-domain decoder. This seems to limit the practical throughput and latency that would be otherwise desirable in a fully-parallel implementation. The pulse-domain decoder frame-error-rate as a function of the maximum I&F input pulse threshold is shown in Figure 44. The pulse-domain implementation only requires an average of 3 input I&F pulses per variable before the codeword is decoded and exhibits very good throughput. However what is more striking is the pulse-domain decoder produces substantially better BERs than even

the ideal floating-point sum-product decoder. In fact, the error floor remains largely undetected up to the limit of SNRs that can be accurately ascertained.

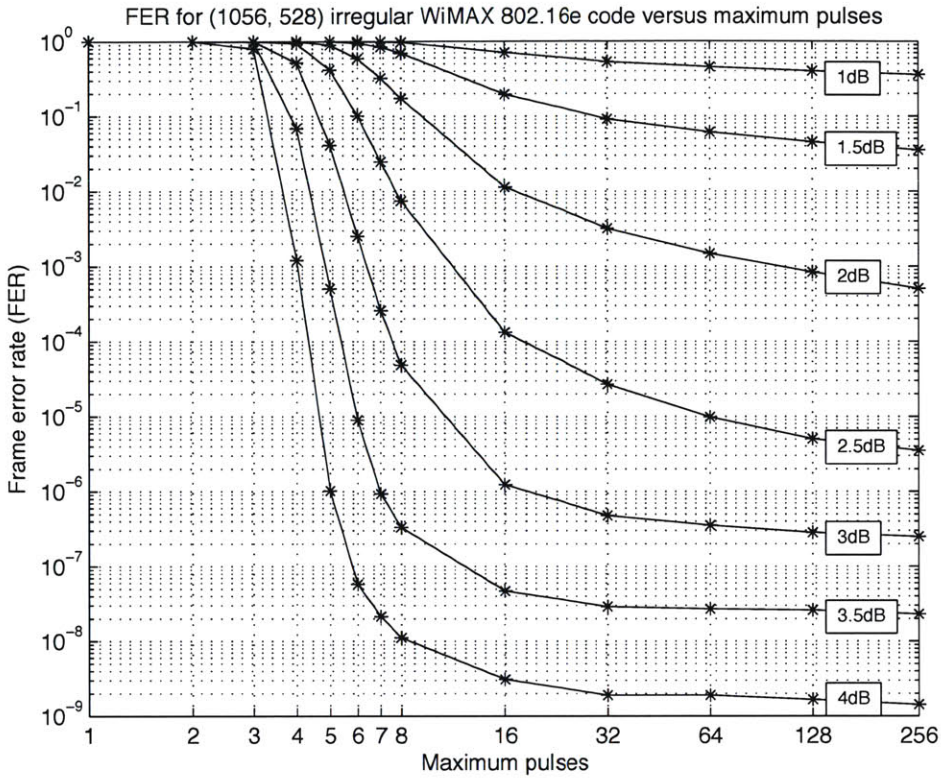


Figure 44: The frame error rate (FER) as a function of maximum number of I&F input pulses before declaring failure.

This gain is attributed to the continuous-time aspect of the processing; in comparison to a discrete-time floating-point decoder, a continuous-time implementation will not experience overshoot artifacts during its evolution. In fact, continuous-time gains were predicted by simulation of ideal analog LDPC decoders by Hemati and Banihashemi [65] using successive relaxation with small β. Continuous-time analog LDPC simulations show a consistent 0.7-1dB improvement over the discrete-time floating point solvers. Yet continuous-time analog decoders have not found practical construction for anything but very small codes (e.g., (7,4) Hamming codes) due to the challenges of large analog designs. It appears the pulse-domain design side-steps the difficulty of an analog design while maintaining the continuous-time gains and small hardware size.

To make a closer comparison with the expected results of an ideal analog decoder. The same regular (504,252) regular code in [65] was decoded with a pulse-domain decoder. The results show a very close match. In fact, the pulse-domain decoder actually slightly outperforms the paper's analog simulations. Perhaps this is due to the infeasibility of truly simulating the continuous-time analog decoder precisely. The successive relaxation methods used in the paper may not have been enough to give an exact continuous-time model without excessive iterations. It is expected that the analog results will converge to or exceed these pulse-domain results when sufficiently small β is used.

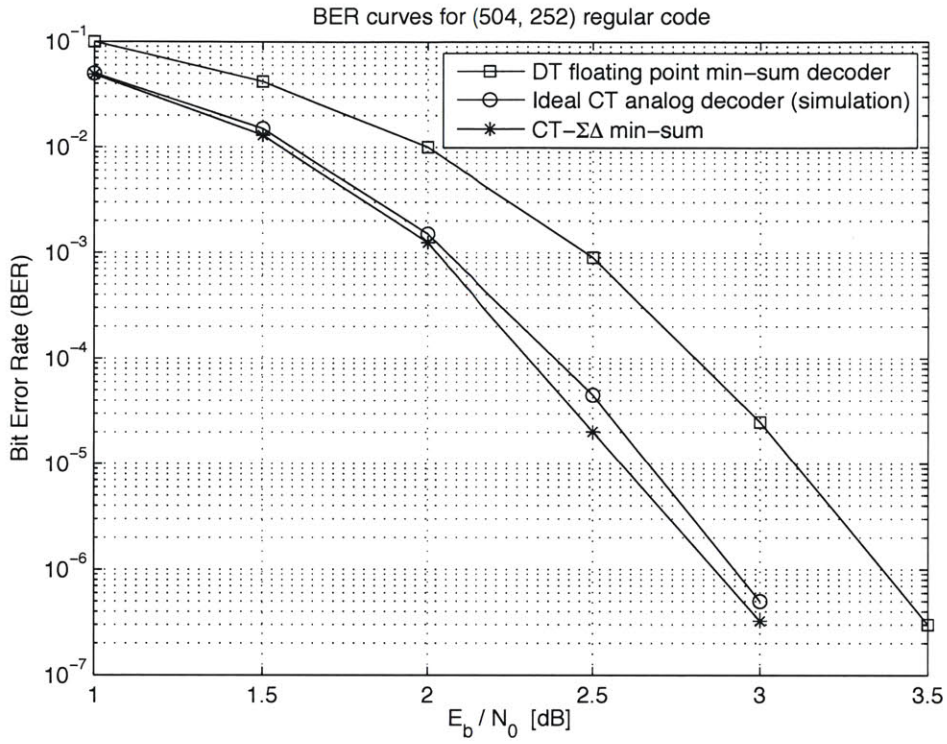


Figure 45: Pulse-domain versus ideal analog min-sum decoder using a (504,252) code examined by [65].

5.4.6 Hardware Resources

Resource usage for FPGAs scales roughly linearly with the codeword size. For example, the (1056,528) code requires about twice the resources of the (528,264) code. We note that the pulse domain decoder requires slightly less FPGA resources than the stochastic decoder in [68]. However, the prototype here uses a very low-end device with fewer slices and so a substantial fraction of the device is utilized.

Resource	Utilization
4-Input LUTs	31,445 / 67,584 (46%)
FFs	20,905 / 67,584 (30%)
TBUFs	4,400 / 16,896 (15%)
Occupied Slices	20,447 / 33,792 (60%)

Table 1: Virtex-2 XC2V6000-4BF957 Device Utilization for (528,264) WiMAX 802.16e irregular code.

Resource	Utilization
6-Input LUTs	58,118 / 69,120 (84%)
FFs	40,595 / 69,120 (59%)
TBUFs	(Not available, converted to logic)
Occupied Slices	17,280 / 17,280 (100%)

Table 2: Virtex-5 XC5VLX110T-3FF1136 Device Utilization for (1056,528) WiMAX 802.16e irregular code.

The construction of a continuous-time decoder may offer a solution to the difficult sampling challenges in ultra-wideband receivers. Recent work in [69] shows that codeword synchronization can be performed after LDPC decoding by using the output of the codeword verification circuit. This suggests that by using a continuous-time pulse-domain or analog filter-bank front-end, a completely clockless receiver system can be constructed and the difficulties of sampling bypassed.

5.4.7 Alternative Form

An alternative min-sum LDPC decoder design was also implemented using LLR-domain computation within the parity-check factors. It exhibits slightly degraded performance in exchange for a decrease in design area. The message calculations (18) are modified with a change of variable $L_{f_A \to x_1} = \mu_{f_A \to x_1}(0) - \mu_{f_A \to x_1}(1)$. For example, the first calculation is now,

$$\begin{aligned} L_{f_A \to x_1} &= \min\left(\mu_{x_3 \to f_A}(0) + \mu_{x_4 \to f_A}(0),\ \mu_{x_3 \to f_A}(1) + \mu_{x_4 \to f_A}(1)\right) \\ &\quad - \min\left(\mu_{x_3 \to f_A}(1) + \mu_{x_4 \to f_A}(0),\ \mu_{x_3 \to f_A}(0) + \mu_{x_4 \to f_A}(1)\right) \\ &= \frac{|L_{x_3 \to f_A} + L_{x_4 \to f_A}| - |L_{x_3 \to f_A} - L_{x_4 \to f_A}|}{2} \\ &= \operatorname{sign}(L_{x_3 \to f_A}) \cdot \operatorname{sign}(L_{x_4 \to f_A}) \cdot \min(|L_{x_3 \to f_A}|, |L_{x_4 \to f_A}|) \end{aligned}$$

Offset-normalization is no longer required (although the absolute values used have an identical circuit). Overall the complexity of one min is removed in exchange for two sign calculations (based on the state of the absolute value PPMs) and multiplication with exclusive-ORs.

5.5 Differential Equations

Finding solutions to continuous-time differential equations is a fundamental problem in many fields. A common numerical computing solution proceeds by transforming the differential equation into a discrete-time finite-difference equation. For example, consider the first-order differential equation:

$$\frac{dx(t)}{dt} = \alpha\, x(t), \quad x(0) = 1$$

The exact analytic solution is known to be $x(t) = e^{\alpha t}$. In principle, this might be computed in hardware with a differential analyzer type machine by feeding a continuous-time integrator with it's output scaled by α. By setting the integrator state to 1 at $t = 0$, the state will grow as $e^{\alpha t}$. Unfortunately, an analog integrator is difficult to build exactly and any real implementation would suffer from compounding inaccuracies. As an alternative, a discrete-time digital computer can evaluate a finite-difference equation that approximates the continuous problem. One of the simplest approaches is the Euler method. This transforms the problem into a recursive computation finding values of $x(t)$ at integer multiples of a step size h,

$$x(t+h) = x(t) + h\alpha x(t)$$

In this case, the n'th point of the calculation is precisely $x(nh) = (1+h\alpha)^n$ which approximates $x(t) = e^{\alpha t}$ when h is small. Still as t grows, the Euler method begins to deviate exponentially from the exact solution.

Here, we instead consider quantizing amplitude rather than time by using the pulse encoding of $\frac{dx(t)}{dt}$ instead. The problem is transformed into a continuous-time digital differential equation where the solution involves calculating the exact (pulse) times when the solution function changes quantization levels. For this example,

$$\begin{aligned}\Delta_{\frac{dx}{dt}}(t) &= \text{I\&F}\{\ \alpha x(t)\ \} \\ &= \frac{d}{dt} Q\left(\int \alpha\, x(t) dt\right)\end{aligned}$$

A solver still consists of a continuous-time integrator fed by the output scaled by α, but now this is encoded with a I&F modulator and the integrator is an up/down pulse counter as shown in Figure 46.

While this system is not necessarily easier to build as a continuous-time circuit, it can be accurately simulated. The I&F modulator output pulse times can be calculated by a discrete-event simulator such as SDES. The calculation is often very simple. In fact, for this problem the n'th time interval between pulses has a closed form–it is always equal to a single quantization interval Δ divided by the input to the modulator over the time interval $(n\alpha\Delta)$,

$$\begin{aligned}t_n - t_{n-1} &= \frac{\Delta}{n\alpha\Delta} \\ &= \frac{1}{n\alpha}\end{aligned}$$

Interestingly, the pulse timing is independent of the value of Δ. With $t_0 = 0$, this defines a pulse signal where the time of the n'th pulse, t_n, is a harmonic series. The harmonic series was also studied by Euler, who gave an equivalent formula,

$$\begin{aligned}t_n &= \frac{1}{\alpha}\sum_{k=1}^{n}\frac{1}{k} \\ &= \frac{1}{\alpha}\left(\ln(n) + \frac{1}{2n} + \gamma\right)\end{aligned}$$

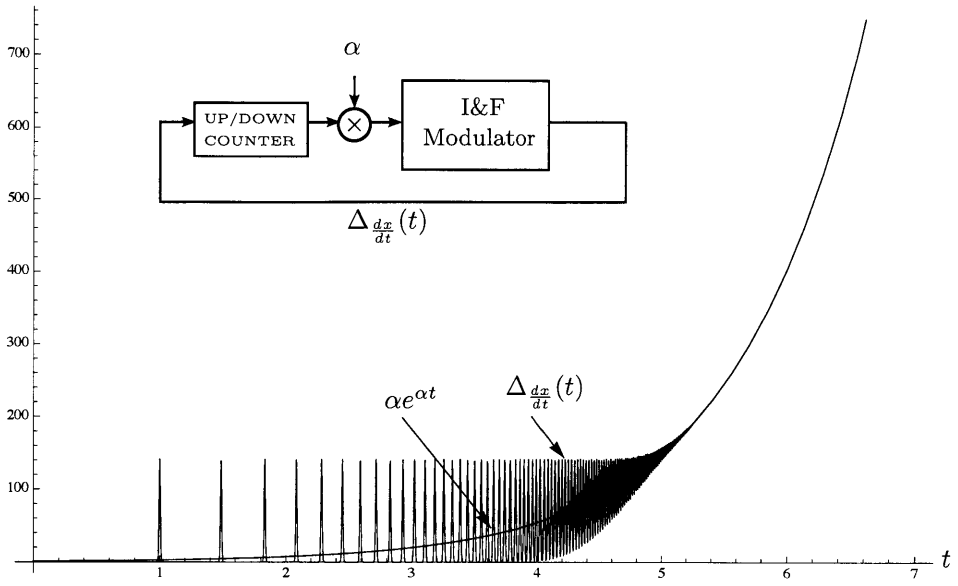


Figure 46: A simple first-order I&F differential analyzer. In the plot, $\Delta_{\frac{dx}{dt}}(t)$ pulses are smoothed by a narrow Gaussian function. This reconstruction converges exactly to $\alpha e^{\alpha t}$ when the rate is high.

where $\gamma \approx 0.57721$ is known as Euler's constant. Based on this, we see that if Dirac deltas of measure $\Delta = e^{\frac{\gamma}{\alpha}}$ is used, a reconstruction (by smoothing with a narrow Gaussian) will be close to the exact solution. In fact, the reconstruction converges to the correct function $\alpha e^{\alpha t}$ as t increases. This is an improvement over the discrete-time Euler method that diverges with increasing t. Of course, the rate at which new pulses occur increases over time so that the computation in a discrete-event simulator grows unbounded. However, for differential equations where the solution has a bounded amplitude, this problem is avoided. These results suggest that a "pulse-domain differential analyzer" is feasible to simulate and has benefits over traditional discrete-time numerical methods.

5.6 Neural Networks

An important class of dynamical systems are artificial neural networks. Neural networks (neuromorphic systems) are based on simple models of animal nervous systems and include non-linear functions in their formulation. Neural networks can also be implemented in the pulse-domain as long as only piece-wise linear operations are required. This includes Hopfield and McCulloch-Pitts-type networks with simple-limiters ($max(0, x)$), *signum* functions or saturation-limiters (but not sigmoid functions).

Neural networks can be used to solve certain optimization and learning problems. For example, neural network solvers exist for linear and quadratic programs and other optimizations[71]. An efficient neural network for solving linear and quadratic programs was proposed by Ghasabi-Oskoei and Mahdavi-Amiri in [72] and is implemented here in the pulse-domain. The system converges to the solution to the quadratic program,

$$\vec{x}^* = \underset{\vec{x}}{\text{argmin}} \left(\frac{\vec{x}^T A \, \vec{x}}{2} + \vec{c}^T \vec{x} \right)$$
$$\text{subject to:} \quad D\vec{x} = \vec{b}, \; \vec{x} \geq 0$$

A potential framework for analyzing these problems is called time-scale calculus[70].

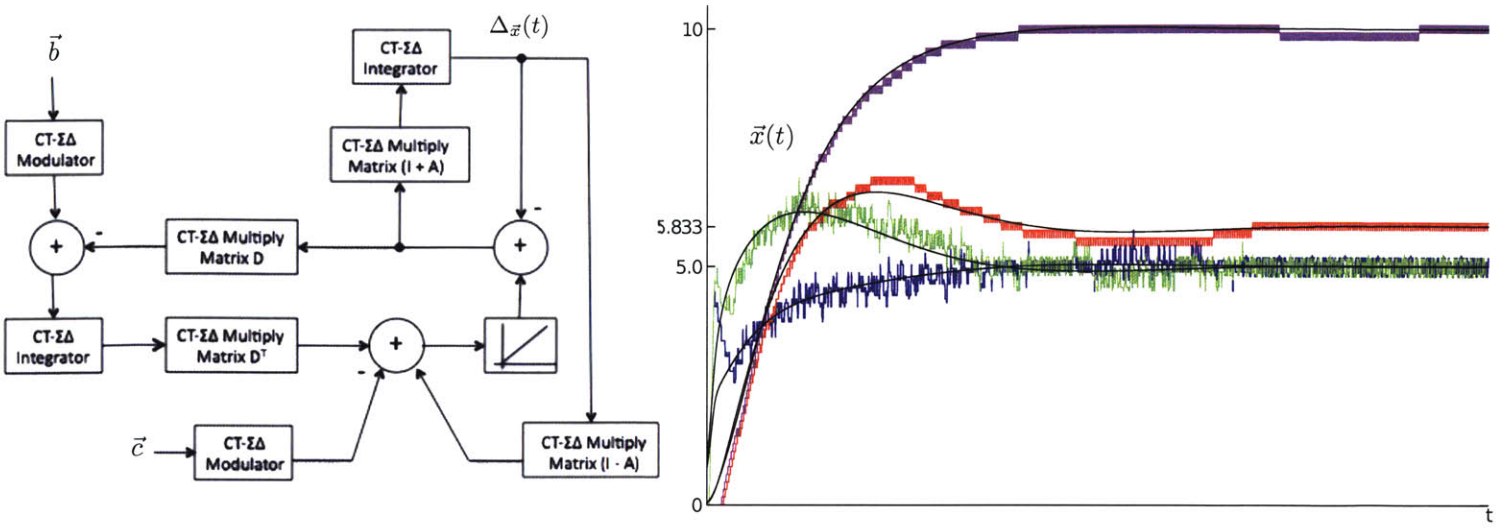


Figure 47: A pulse-domain linear and quadratic program solver based on a neural network by Ghasabi-Oskoei and Mahdavi-Amiri.

The solver mirrors the structure of the original design but implements each operation in the pulse-domain. It incorporates the entries of matrix D in a matrix multiplier composed of pulse-domain additions and constant multiplications. A required piecewise-function, $max(0, x_i)$ is performed in the pulse-domain using a passive pulse machine (Figure 48). The vectors $\vec{b}$ and $\vec{c}$ are constant inputs that are converted to pulses. In steady-state, the encoded solution is found in the pulse rates of $\Delta_{\vec{x}}(t)$.

As a demonstration, we show the output while solving the example problem given in [72],

$$
\begin{aligned}
(x_1^*, \ldots, x_6^*) = \min_{(x_1,\ldots,x_6)} & \left(x_1^2 + x_2^2 + x_1 x_2 - 30x_1 - 30x_2\right) \\
\text{s.t.} \quad & \frac{5}{12}x_1 - x_2 + x_3 = \frac{35}{12} \\
& \frac{5}{2}x_1 + x_2 + x_4 = \frac{35}{2} \\
& x_5 - x_1 = 5 \\
& x_2 + x_6 = 5 \\
& (x_1, \ldots, x_6) \geq 0
\end{aligned}
$$

The pulse-domain system converges to the correct solution $\vec{x}^* = (5, 5, 5.8333333, 0, 10, 0)$. A precise pulse reconstruction to find these values involves averaging the output pulses over time. As a comparison, the same problem solved by a a floating-point version of the system in MATLAB is superimposed in black.

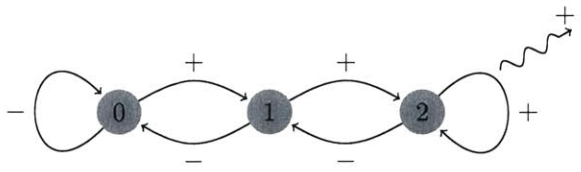


Figure 48: PPM for the pulse-domain implementation of $max(0, x_i)$.

6 Circuits for I&F Modulation and Processing

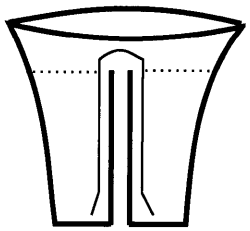


Figure 49: The Pythagorean cup.

One of the oldest known man-made systems exhibiting integrate-and-fire dynamics is the Pythagorean cup of ancient Greece. This cup was invented by Pythagoras around 500 BC to discourage gluttony and excess. It fills with wine (integrates) only to a critical level before a siphon process is activated and it empties itself entirely onto the drinker's lap (fires). While the cup remained a novelty item, it later inspired the hydraulic components in the robotic systems of Heron of Alexandria. Today, hydraulic robots have given way to electrical systems but integrate-and-fire dynamics still frequently appear.

6.1 Frequency Modulation (FM)

An integrate-and-fire encoding is formed by the output of a voltage controlled oscillator. The zero crossings of frequency modulation correspond to pulse times of integrate-and-fire. If an FM signal is given by,

$$y(t) = \sin\left(\frac{\pi}{\Delta}\int_0^t x(\tau)d\tau\right)$$

then times at which $y(t) = 0$ are the times that the integral of $x(\tau)$ intersects a Δ level crossing (taking the sin of an integer multiple of π). This link appears to have been first made in [73].

6.2 Neon-discharge Tubes

These circuits involve a capacitor charged by a current source. The capacitor is placed in parallel with a threshold device, such as a neon lamp, that exhibits high conductance above a threshold voltage (e.g., gas ionization) and continues to conduct in an avalanche manner until the capacitor is fully discharged. This forms a series of discharge pulses in time with the rate based on the constant current source refilling the capacitor. Early circuits of this type were configured for constant pulse rates and found application as the time bases in early oscilloscopes and for stroboscopic effects. However, by using a time-varying current source as an input signal, intergrate-and-fire modulation is produced.

Unfortunately, in many cases the threshold voltage is very large (e.g. neon tubes, thyratron, diacs, other negative resistance devices) but some low threshold devices do exist. In the course of this thesis, a unijunction transistor-based design was developed that can be configured for a threshold of 2 to 3 volts.

6.3 Unijunction Transistor (UJT) Modulator

A unijunction transistor can be paired with a single capacitor to form a semiconductor relaxation oscillator with a low threshold voltage. A UJT is a rarely used 3-terminal device that peaked in popularity in the 1960's among electronic hobbyists. It consists of a lightly doped N-type bar and a heavily doped P-type region, forming a single P-N junction. The N-type bar acts as a resistor and divides the 5 volt source to a form a potential of about 2 volts near where the P-type doping is placed. The P-type region is called the emitter terminal while the top and bottom of the N-type bar are called base one and two respectively. Input current is integrated on the capacitor, raising the voltage at the emitter. This shrinks an insulating depletion region that forms at the P-N interface. After shrinking enough, conduction occurs and persists until the capacitor is drained. Conduction continues because the resistance of the lower half of the N channel decreases when electrons are flowing, changing the voltage divider so that the P-N threshold is lowered.

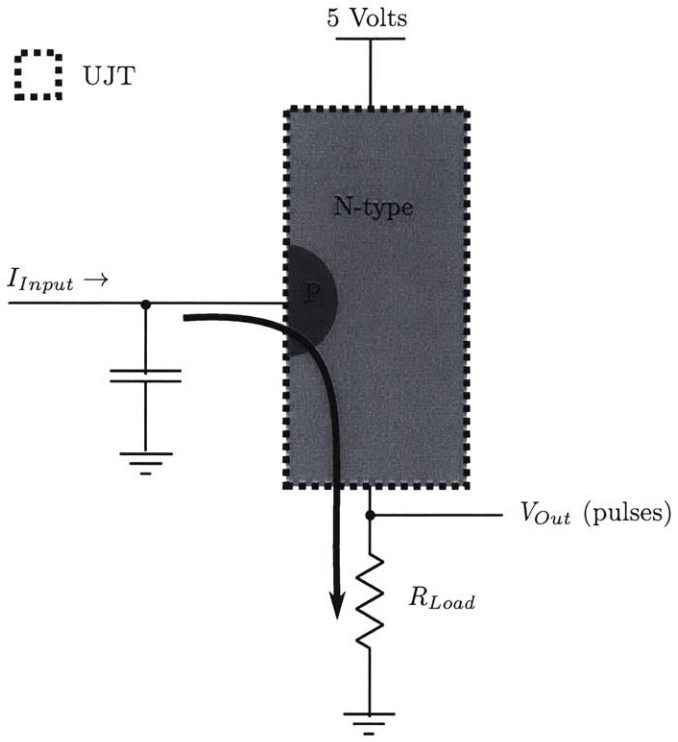


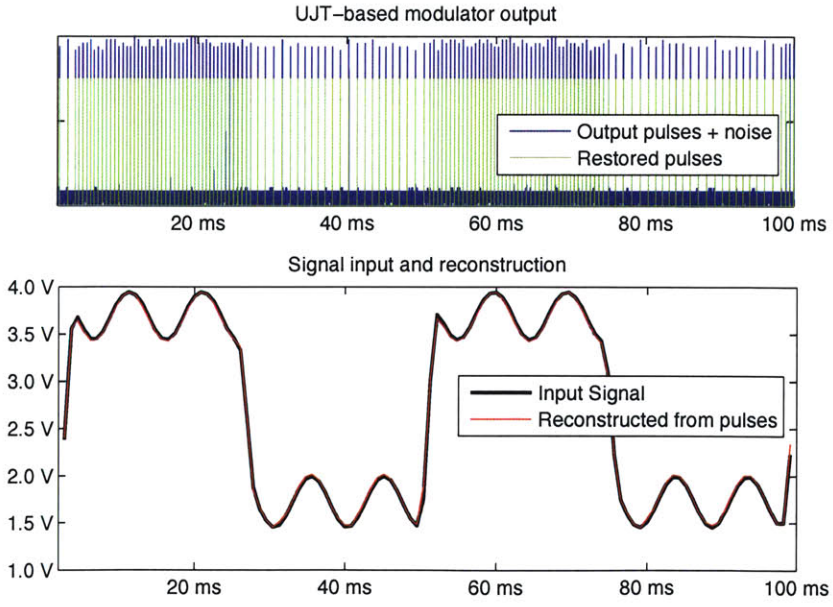
Figure 50: Unijunction transistor (UJT) based integrate-and-fire modulator. Current flows from the P-type emitter to the base when an insulating P-N depletion layer shrinks enough.

An example of this circuit was constructed and found to pulse at rates up to 1 million pulses per second while consuming a few tens of μA from a 5 volt supply (nearly all of the power comes from the input signal itself). A transconductance amplifier is generally required to convert voltage-mode signals to current-mode signals, but here it was found that a simple resistor from a voltage source to the capacitor is sufficient.

In the figure below, the output of a UJT integrate-and-fire modulator with a $0.01\mu F$ capacitor is shown. A voltage-mode input signal was connected with a $100K\Omega$ series resistor to the input capacitor. The output pulses were passed to an LED and the signal detected by a receiving photodiode was measured. These measurements were contaminated with noise, nonlinear distortion, bias and gain, but ideal pulse height and width were restored with a monostable multivibrator (one-shot pulse generator). Restored pulses were passed through a $sinc^3$ filter to give a good real-time reconstruction of the original input waveform.

In the example below, the pulse rate is very low (about 1 thousand pulses per second), but with a smaller capacitor (100pF), the UJT was found to generate pulses at 250 thousand pulses per second or more. The circuit can also be extended to perform biphasic integrate-and-fire with an additional complementary UJT (CUJT) that swaps the the N and P-type semiconductors and enables formation of negative pulses.

This circuit is being tested at Professor Todd Coleman's Neural Interaction Lab at UCSD as part of a simple electromyography (EMG) signal encoding and transmission scheme for use with epidermal electronic tattoos[74]. Very low power processing using pulse-domain techniques might enable medical devices and vital sign monitors with extremely low power and thermal budgets. Pulses can be transmitted optically through LEDs or as an electromagnetic pulse and can be multiplexed by using different pulse colors and shapes or with different base-line pulse rates.



6.4 Neuromorphic Circuits

Efficient integrate-and-fire circuits are also found abundantly in the neuromorphic computing field for modeling the biological action potential. The original VLSI design dates back to Carver Mead's six-transistor/two-capacitor Axon Hillock circuit[1] in 1989. Soon improvements and lower power versions were developed (e.g., [75]). A biphasic integrate-and-fire circuit has a number of variations as well including [16]. SPICE models and experimentation with discrete components suggest that these circuits consume significantly more dynamic power than a UJT implementation. However, variations can generate pulses at rates up to 1 billion per second to accurately encode signals with bandwidths on the order of MHz.

6.5 $\Sigma\Delta$ Modulators

$\Sigma\Delta$ is a low cost, high precision/dynamic range A/D and D/A converter architecture (see [80]). These converters employ discrete-time oversampled $\Sigma\Delta$ modulation where for each sample an analog signal is coarsely quantized, often only measuring the sign, but the error stored in an analog form for compensation in later samples. In the 1st-order design, negative quantization error feedback causes first-differencing of the error signal moving it to higher frequencies. A lowpass filter applied to the sign bitstream can remove a significant portion of the error. Normally a down-sampled fixed-point output from a digital lowpass filter then serves as the converted signal for a digital signal processor to manipulate. However, the output of a $\Sigma\Delta$ modulator can also be processed directly using synchronous logic or used to create pulses that are processed asynchronously. In fact, integrate-and-fire is closely related to the discrete-time $\Sigma\Delta$ modulator. Indeed, a clocked 1st-order modulator circuit can be used directly as an I&F modulator for pulse processing. The sole difference between the exact I&F and a discrete-time 1st-order $\Sigma\Delta$ modulator is a small pulse timing jitter due to alignment of pulses to the oversampling clock period. A basic block diagram is shown in Figure 51.

Circuit design for discrete-time $\Sigma\Delta$ modulation is well developed (see [80]). Many implementations use switched-capacitors to perform the integration (discrete-time). Alternatively, so-called continuous-time modulators use an active RC filter to perform the integration, however, the sampling is still usually discrete-time). These circuits are relatively cheap compared with other conversion architectures.

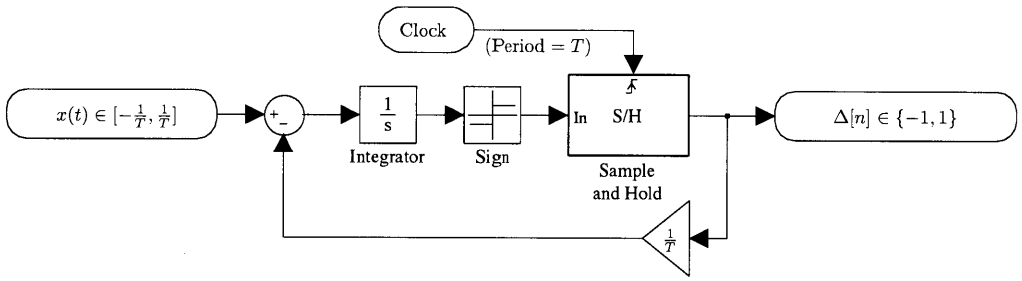


Figure 51: A 1st-order discrete-time $\Sigma\Delta$ modulator.

On the rising edge of the clock, a $\Sigma\Delta$ modulator samples the integrator sign and holds this value for one clock period. The held value ± 1 is fed back and subtracted from the integrator after scaling by $\frac{1}{T}$. Integration occurs over one sample period (T sec.) and the net contribution to the integrator is exactly ± 1. $\Sigma\Delta$ modulators usually limit the input to the range $[-\frac{1}{T}, \frac{1}{T}]$ in order to ensure that the integrator state

Clocked $\Sigma\Delta$ was first described in a patent by Cutler in 1960[76], analyzed further by Inose and Yasuda in 1962[77] and later adapted for modern oversampled A/D converters by Goodman[78]. A related asynchronous structure was invented by Kikkert and Miller 13 years later[79] but is different than I&F.

maintains a bounded value. If the input is bounded, it is easy to prove that the integrator state $g(t)$ does not exceed the interval $[-2, 2]$ and the system is stable. This can be shown by an inductive argument,

$$x(t) \in [-\frac{1}{T}, \frac{1}{T}], \quad g(nT) \in [-2, 2]$$

$$\Longrightarrow g((n+1)T) = \left(\underbrace{\int_{nT}^{(n+1)T} x(t')dt'}_{\in[-1,1]} + \underbrace{\begin{cases} g(nT)+1, & g(nT) \leq 0 \\ g(nT)-1, & g(nT) > 0 \end{cases}}_{\in[-1,1]} \right) \in [-2, 2]$$

The diagram for the 1st-order $\Sigma\Delta$ modulator looks very similar to the I&F system except for the zero-order hold. The $\Sigma\Delta$ modulator feeds back a signal that integrates to -1 or 1 in one sample period T while I&F feeds back an instantaneous pulse that integrates to 0 or 1. The difference in these values is easily corrected with an offset and scaling, while the fact that the pulses have different widths is unimportant because the next sampling always occurs after the pulse has been completely deposited in the integrator. If the input is bounded, the pulse rate will not exceed the clock rate and the quantizer sampling is equivalent to sampling an SR latch that is external to the feedback loop.

A model of a 1st-order $\Sigma\Delta$ modulator built from an I&F modulator is shown in Figure 52. The input to I&F is offset by $+\frac{1}{T}$ to span the interval $[0, \frac{2}{T}]$. The threshold of I&F is set to $\Delta = 1$ yielding an output pulse rate between 0 and $\frac{1}{T}$ pulses/second depending on the input. These pulses are then captured by a latch that asynchronously sets to 1 whenever an input pulse arrives. The latch is sampled and then reset to 0 on every rising clock edge. The sampled value is held until the next clock period with a zero-order-hold. The resulting $\Delta[n]$ binary signal is identical to the bitstream produced by a $\Sigma\Delta$ modulator with the same analog input.

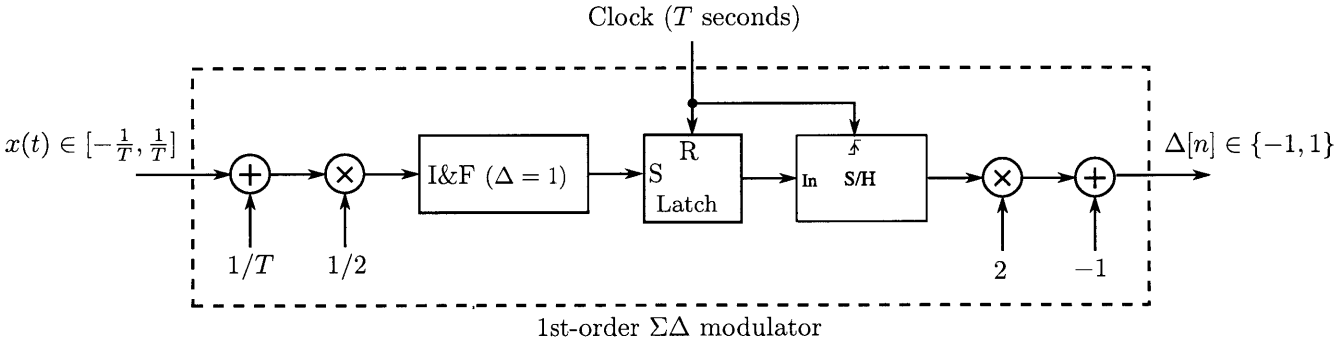


Figure 52: An equivalent representation of a 1st-order $\Sigma\Delta$ modulator built from an I&F modulator.

The connection between $\Sigma\Delta$ and I&F can be exploited for building modulators. In some cases, it may be easier to construct a 1st-order $\Sigma\Delta$ modulator than asynchronous I&F. When the oversampling clock is at GHz rates, the pulse timing jitter is negligible. To construct an I&F modulator from a discrete-time $\Sigma\Delta$ modulator, the input must be scaled and offset as shown in Figure 53. When the output of the modulator is 1, a pulse is generated at the rising edge of the clock. When the output is -1, no pulse is generated. If it is acceptable to have a pulse width of T seconds, the 1 output of the $\Sigma\Delta$ modulator can be used directly as the digital pulse send to pulse processing logic. The only difference between this implementation and that of an exact I&F circuit is the pulse timing jitter and the limitation that the input must be restricted to the interval $[0, \frac{1}{T}]$. To support negative pulses for signed I&F, a ternary 1st-order $\Sigma\Delta$ modulator can be employed in a similar configuration.

This model appears to be simpler than other connections between $\Sigma\Delta$ and I&F published previously[81, 82]

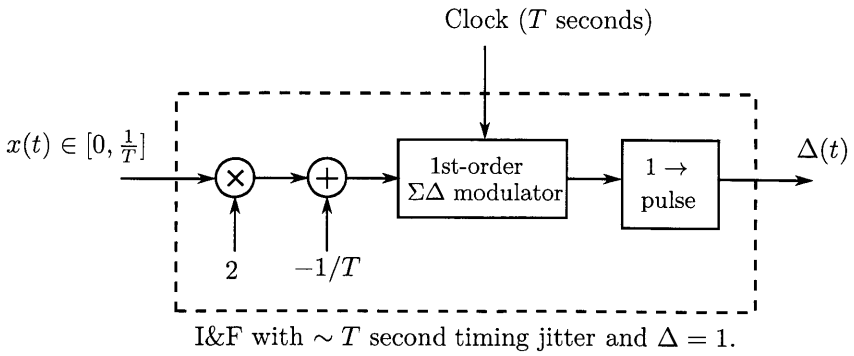


Figure 53: A 1st-order discrete-time ΣΔ modulator is equivalent to I&F with small timing jitter and gain.

6.6 Quantum Modulators

As technology scaling continues, quantum effects can be exploited for pulse modulation purposes. Indeed, pulse processing is the natural process for single-electron computation. As described in Chapter 2, the Coulomb blockade effect[83, 13] produces a type of single electron I&F when electrons tunnel across a barrier. Resonant tunneling diodes are also being considered in clocked ΣΔ modulators designs. These structures integrate well with the standard CMOS process, perform at room temperature and may operate with oversampling frequencies in excess of 100GHz[84].

6.7 Circuits for Pulse Processing

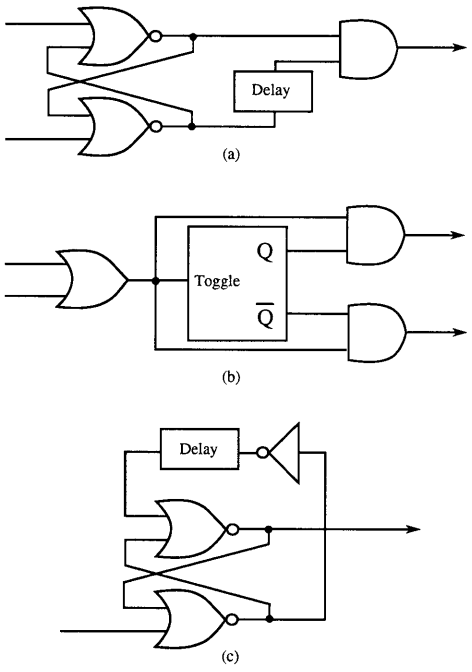


Figure 54: Various circuits for pulses processing: (a) a 2-state min operation (b) a balancer element (c) a pulse regenerator.

The passive pulse machines for pulse processing can be implemented using simple asynchronous circuits. Asynchronous state machines can be constructed using Muller C-elements or toggle gates and feedback (see [85]). Given some timing assumptions, many of the PPMs defined in this thesis reduce to circuits with 10s of transistors. Some simple examples are shown in Figure 54. The fixed delay elements specified in these circuits can be constructed with an even number of cascaded inverters and determine the output pulse width.

Circuit (a) is an example of a 2-state min operation. It consists of an SR latch followed by a pulse generator that emits pulses only during transitions from reset to set. Circuit (b) constructs one of the balancers for use in sorting networks. It uses a falling-edge triggered toggle gate where the toggle occurs right after the incoming pulse ends. The pulse is forwarded to an output based on the state before it is toggled. This type of construction, extends to many of the other passive pulse machines. Circuit (c) shows a basic method for pulse regeneration to restore pulses traveling over long transmission lines. The circuit consists of an SR latch that is set by an incoming pulse and then promptly reset after the delay period.

A toggle gate can be implemented with a pair of Muller C-elements or a self-inverting edge-triggered D-flip flop.

References

[1] C. Mead, *Analog VLSI and neural systems*, Addison-Wesley, 1989.

[2] A.A. Lazar and L.T. Tóth, "Perfect recovery and sensitivity analysis of time encoded bandlimited signals", *Circuits and Systems I: Regular Papers, IEEE Transactions on*, vol. 51, no. 10, pp. 2060–2073, 2004.

[3] V. da Fonte Dias, "Signal processing in the sigma-delta domain", *Microelectronics journal*, vol. 26, no. 6, pp. 543–562, 1995.

[4] H. Fujisaka, N. Masuda, M. Sakamoto, and M. Morisue, "Arithmetic circuits for single-bit digital signal processing", *Proc. 6th IEEE Int. Conf. Electronics, Circuits and Systems ICECS '99*, vol. 3, pp. 1389–1392, 1999.

[5] D. Arfib, F. Keiler, and U. Zölzer, *DAFx-Digital Audio Effects*, J. Wiley & Sons, Chichester, 2002.

[6] D.G. Zrilic, *Circuits and systems based on delta modulation: linear, nonlinear, and mixed mode processing*, Springer, 2005.

[7] A. Pneumatikakis and T. Deliyannis, "Direct processing of sigma-delta signals", in *Electronics, Circuits, and Systems, 1996, Proceedings of the Third IEEE International Conference on*. IEEE, 1996, vol. 1, pp. 13–16.

[8] Y. Tsividis, "Event-driven data acquisition and digital signal processinga tutorial", *Circuits and Systems II: Express Briefs, IEEE Transactions on*, vol. 57, no. 8, pp. 577–581, 2010.

[9] D.R. Cox and H.D. Miller, *The theory of stochastic processes*, vol. 134, Chapman and Hall, 1965.

[10] N. Brunel and M.C.W. Van Rossum, "Lapicques 1907 paper: from frogs to integrate-and-fire", *Biological cybernetics*, vol. 97, no. 5, pp. 337–339, 2007.

[11] R. Jolivet, A. Rauch, HR Lüscher, and W. Gerstner, "Integrate-and-fire models with adaptation are good enough: predicting spike times under random current injection", *Advances in neural information processing systems*, vol. 18, pp. 595–602, 2006.

[12] R. Garrel, R. Scherer, R. Nicollas, A. Giovanni, and M. Ouaknine, "Using the relaxation oscillations principle for simple phonation modeling", *Journal of Voice*, vol. 22, no. 4, pp. 385–398, 2008.

[13] M.H. Devoret and H. Grabert, "Single charge tunneling: Coulomb blockade phenomena in nanostructures", *Proceedings of a NATO Advanced Study Institute on Single Charge Tunneling*, 1992.

[14] K.K. Likharev, "Single-electron devices and their applications", *Proceedings of the IEEE*, vol. 87, no. 4, pp. 606–632, 1999.

[15] Frank Diedrich and Herbert Walther, "Nonclassical radiation of a single stored ion", *Phys. Rev. Lett.*, vol. 58, pp. 203–206, Jan 1987.

[16] M. Rastogi, A. Singh Alvarado, J.G. Harris, and J.C. Principe, "Integrate and fire circuit as an ADC replacement", *Circuits and Systems (ISCAS), 2011 IEEE International Symposium on*, pp. 2421–2424, 2011.

[17] R.M. Gray, "Quantization noise spectra", *Information Theory, IEEE Transactions on*, vol. 36, no. 6, pp. 1220–1244, 1990.

[18] S.B. Furber, *Principles of asynchronous circuit design: a systems perspective*, Springer Netherlands, 2001.

[19] B. Schell and Y. Tsividis, "A continuous-time ADC/DSP/DAC system with no clock and with activity-dependent power dissipation", *Solid-State Circuits, IEEE Journal of*, vol. 43, no. 11, pp. 2472–2481, 2008.

[20] D. Bruckmann, "Design and realization of continuous-time wave digital filters", *Circuits and Systems, 2008. ISCAS 2008. IEEE International Symposium on*, pp. 2901–2904, 2008.

[21] Y.W. Li, K.L. Shepard, and Y.P. Tsividis, "A continuous-time programmable digital FIR filter", *Solid-State Circuits, IEEE Journal of*, vol. 41, no. 11, pp. 2512–2520, 2006.

[22] H. Inose, T. Aoki, and K. Watanabe, "Asynchronous delta-modulation system", *Electronics Letters*, vol. 2, no. 3, pp. 95–96, 1966.

[23] F. De Jager, "Delta modulation: A method of pcm transmission using the one unit code", *Philips Res. Rep*, vol. 7, no. 6, pp. 442–466, 1952.

[24] N. Kouvaras, "Operations on delta-modulated signals and their application in the realization of digital filters", *The Radio and Electronic Engineer*, vol. 48, no. 9, pp. 431–438, 1978.

[25] N. Kouvaras, "Novel multi-input signal-processing networks with reduced quantization noise", *International journal of electronics*, vol. 56, no. 3, pp. 371–378, 1984.

[26] N. Kouvaras, "Modular network for direct complete addition of delta-modulated signals with minimum quantization error", *International Journal of Electronics*, vol. 59, no. 5, pp. 587–595, 1985.

[27] P. O'Leary and F. Maloberti, "Bit stream adder for oversampling coded data", *Electronics Letters*, vol. 26, no. 20, pp. 1708–1709, 1990, First to use the carry adder form for DS adder.

[28] G. Lockhart, "Implementation of delta modulators for digital inputs", *Acoustics, Speech and Signal Processing, IEEE Transactions on*, vol. 22, no. 6, pp. 453–456, 1974.

[29] G. Kim, M.K. Kim, B.S. Chang, and W. Kim, "A low-voltage, low-power CMOS delay element", *Solid-State Circuits, IEEE Journal of*, vol. 31, no. 7, pp. 966–971, 1996.

[30] B. Schell and Y. Tsividis, "A low power tunable delay element suitable for asynchronous delays of burst information", *Solid-State Circuits, IEEE Journal of*, vol. 43, no. 5, pp. 1227–1234, 2008.

[31] D. Brückmann and K. Konrad, "Optimization of continuous time filters by delay line adjustment", in *Circuits and Systems (MWSCAS), 2010 53rd IEEE International Midwest Symposium on*. IEEE, 2010, pp. 1238–1241.

[32] J. Aspnes, M. Herlihy, and N. Shavit, "Counting networks", *Journal of the ACM (JACM)*, vol. 41, no. 5, pp. 1020–1048, 1994.

[33] Y. Ma, *Fault-tolerant sorting networks*, PhD thesis, MIT, 1994.

[34] W. Aiello, C. Busch, M. Herlihy, M. Mavronicolas, N. Shavit, and D. Touitou, "Supporting increment and decrement operations in balancing networks", in *Proceedings of the 16th annual conference on Theoretical aspects of computer science*. Springer-Verlag, 1999, pp. 393–403.

[35] C. Busch and M. Mavronicolas, "An efficient counting network", *Theoretical Computer Science*, vol. 411, no. 34, pp. 3001–3030, 2010.

[36] F. Maloberti, "Non conventional signal processing by the use of sigma delta technique: a tutorial introduction", in *Circuits and Systems, 1992. ISCAS'92. Proceedings., 1992 IEEE International Symposium on*. IEEE, 1992, vol. 6, pp. 2645–2648.

[37] A. K. Lu, G. W. Roberts, and D. A. Johns, "A high-quality analog oscillator using oversampling D/A conversion techniques", *Circuits and Systems II: Analog and Digital Signal Processing, IEEE Transactions on*, vol. 41, no. 7, pp. 437–444, 1994.

[38] P.W. Wong and R.M. Gray, "FIR filters with sigma-delta modulation encoding", *Acoustics, Speech and Signal Processing, IEEE Transactions on*, vol. 38, no. 6, pp. 979–990, 1990.

[39] P.W. Wong, "Fully sigma-delta modulation encoded FIR filters", *Signal Processing, IEEE Transactions on*, vol. 40, no. 6, pp. 1605–1610, 1992.

[40] D.A. Johns and D.M. Lewis, "Design and analysis of delta-sigma based IIR filters", *Circuits and Systems II: Analog and Digital Signal Processing, IEEE Transactions on*, vol. 40, no. 4, pp. 233–240, 1993.

[41] D.A. Johns, D.M. Lewis, and D. Cherepacha, "Highly selective analog filters using $\Delta\Sigma$ based IIR filtering", in *Circuits and Systems, 1993., ISCAS'93, 1993 IEEE International Symposium on*. IEEE, 1993, pp. 1302–1305.

[42] Y. Suzuki, H. Fujisaka, T. Kamio, and K. Haeiwa, "Digital filters built of locally connected nanoelectronic devices", in *Proc. 7th IEEE Conf. Nanotechnology IEEE-NANO 2007*, 2007, pp. 873–878.

[43] T. Yasuno, Y. Suzuki, H. Fujisaka, T. Kamio, Chang-Jun Ahn, and K. Haeiwa, "A single-eletron bandpass digital wave filter", in *Proc. European Conf. Circuit Theory and Design ECCTD 2009*, 2009, pp. 882–885.

[44] T. Ritoniemi, T. Karema, and H. Tenhunen, "A sigma-delta modulation based analog adaptive filter", in *Circuits and Systems, 1992. ISCAS'92. Proceedings., 1992 IEEE International Symposium on*. IEEE, 1992, vol. 6, pp. 2657–2660.

[45] Q. Huang and G.S. Moschytz, "Analog multiplierless LMS adaptive FIR filter structures", *Circuits and Systems II: Analog and Digital Signal Processing, IEEE Transactions on*, vol. 40, no. 12, pp. 790–794, 1993.

[46] Y. Hidaka, H. Fujisaka, M. Sakamoto, and M. Morisue, "Piecewise linear operations on sigma-delta modulated signals", *Proc. 9th Int Electronics, Circuits and Systems Conf*, vol. 3, pp. 983–986, 2002.

[47] K. Hayashi, T. Katao, H. Fujisaka, T. Kamio, and K. Haeiwa, "Piecewise linear circuits operating on first-order multi-level and second-order binary sigma-delta modulated signals", *Proc. 18th European Conf. Circuit Theory and Design ECCTD 2007*, pp. 683–686, 2007.

[48] H. Fujisaka, R. Kurata, M. Sakamoto, and M. Morisue, "Bit-stream signal processing and its application to communication systems", *IEE Proceedings -Circuits, Devices and Systems*, vol. 149, no. 3, pp. 159–166, 2002.

[49] A.Z. Sadik and P.J. O'Shea, "Towards a ternary sigma-delta modulated processor: Adder and integrator", in *Computer and Information Technology Workshops, 2008. CIT Workshops 2008. IEEE 8th International Conference on*. IEEE, 2008, pp. 515–520.

[50] M. Freedman and D.G. Zrilic, "Nonlinear arithmetic operations on the delta sigma pulse stream", *Signal processing*, vol. 21, no. 1, pp. 25–35, 1990.

[51] T.A.D. Riley and M.A. Copeland, "A simplified continuous phase modulator technique", *Circuits and Systems II: Analog and Digital Signal Processing, IEEE Transactions on*, vol. 41, no. 5, pp. 321–328, 1994.

[52] T.R.H. Sizer, *The digital differential analyser*, Chapman & Hall, 1968.

[53] K.J. Astrom, "Event based control", *Analysis and Design of Nonlinear Control Systems: In Honor of Alberto Isidori, Springer Verlag*, pp. 127–147, 2007.

[54] P.F. Felzenszwalb and D.P. Huttenlocher, "Efficient belief propagation for early vision", *International journal of computer vision*, vol. 70, no. 1, pp. 41–54, 2006.

[55] R. Tanner, "A recursive approach to low complexity codes", *Information Theory, IEEE Transactions on*, vol. 27, no. 5, pp. 533–547, 1981.

[56] N. Wiberg and Universitetet i Linköping. Department of Electrical Engineering, *Codes and decoding on general graphs*, Citeseer, 1996.

[57] G.D. Forney Jr, "The viterbi algorithm", *Proceedings of the IEEE*, vol. 61, no. 3, pp. 268–278, 1973.

[58] J. Pearl, *Probabilistic reasoning in intelligent systems: networks of plausible inference*, Morgan Kaufmann, 1988.

[59] S.M. Aji and R.J. McEliece, "The generalized distributive law", *Information Theory, IEEE Transactions on*, vol. 46, no. 2, pp. 325–343, 2000.

[60] H.A. Loeliger, "An introduction to factor graphs", *Signal Processing Magazine, IEEE*, vol. 21, no. 1, pp. 28–41, 2004.

[61] Y. Weiss and W.T. Freeman, "On the optimality of solutions of the max-product belief-propagation algorithm in arbitrary graphs", *Information Theory, IEEE Transactions on*, vol. 47, no. 2, pp. 736–744, 2001.

[62] B. Vigoda, *Analog logic: Continuous-Time analog circuits for statistical signal processing*, PhD thesis, Massachusetts Institute of Technology, 2003.

[63] S. Hemati, A.H. Banihashemi, and C. Plett, "A high-speed analog min-sum iterative decoder", in *Information Theory, 2005. ISIT 2005. Proceedings. International Symposium on*. IEEE, 2005, pp. 1768–1772.

[64] H.A. Loeliger, F. Lustenberger, M. Helfenstein, and F. Tarkoy, "Probability propagation and decoding in analog vlsi", *Information Theory, IEEE Transactions on*, vol. 47, no. 2, pp. 837–843, 2001.

[65] S. Hemati and A.H. Banihashemi, "Dynamics and performance analysis of analog iterative decoding for low-density parity-check (LDPC) codes", *Communications, IEEE Transactions on*, vol. 54, no. 1, pp. 61–70, 2006.

[66] L.R. Varshney, *Unreliable and resource-constrained decoding*, PhD thesis, Massachusetts Institute of Technology, 2010.

[67] D.J.C. MacKay and R.M. Neal, "Near Shannon limit performance of low density parity check codes", *Electronics letters*, vol. 32, no. 18, pp. 1645, 1996.

[68] S. Sharifi Tehrani, S. Mannor, and W.J. Gross, "Fully parallel stochastic LDPC decoders", *Signal Processing, IEEE Transactions on*, vol. 56, no. 11, pp. 5692–5703, 2008.

[69] M.V.R.S. Devarakonda and H.-A. Loeliger, "Continuous-time matched filtering and decoding without synchronization", *Information Theory Workshop*, pp. 139 – 143, 2009.

[70] S. Hilger, *Ein Maßkettenkalkül mit Anwendung auf Zentrumsmannigfaltigkeiten [Calculus on Measure Chains with Application to Central Manifolds]*, PhD thesis, University of Würzburg, Würzburg, 1988.

[71] X.S. Zhang, *Neural networks in optimization*, vol. 46, Springer Netherlands, 2000.

[72] H. Ghasabi-Oskoei and N. Mahdavi-Amiri, "An efficient simplified neural network for solving linear and quadratic programming problems", *Applied mathematics and computation*, vol. 175, no. 1, pp. 452–464, 2006.

[73] JE Flood and MJ Hawksford, "Exact model for delta-modulation processes(mathematical model for pulse waveform identical with single integrator delta modulator, using analog techniques of angle modulation and sampling)", in *Institution of electrical engineers, proceedings*, 1971, vol. 118, pp. 1155–1161.

[74] D.H. Kim, N. Lu, R. Ma, Y.S. Kim, M. McCormick, et al., "Epidermal electronics", *Science*, vol. 333, no. 6044, pp. 838, 2011.

[75] J. Lazzaro, "Low-power silicon spiking neurons and axons", *Circuits and Systems, 1992. ISCAS'92. Proceedings., 1992 IEEE International Symposium on*, vol. 5, pp. 2220–2223, 1992.

[76] C.C. Cutler, "Transmission systems employing quantization", *US Patent 2,927,962*, March 8, 1960 (filed 1954).

[77] H. Inose, Y. Yasuda, and J. Murakami, "A Telemetering System by Code Modulation–Δ-Σ Modulation", *Space Electronics and Telemetry, IRE Transactions on*, pp. 204–209, 1962.

[78] DJ Goodman, "The application of delta modulation to analog-to-PCM encoding", *Bell System Technical Journal*, vol. 48, pp. 321–343, 1969.

[79] CJ Kikkert and DJ Miller, "Asynchronous delta sigma modulation", *Proceedings of the IREE Australia*, vol. 36, no. 4, 1975.

[80] R. Schreier, G.C. Temes, Institute of Electrical, and Electronics Engineers, *Understanding delta-sigma data converters*, IEEE press NJ, 2005.

[81] M. Hovin, D. Wisland, Y. Berg, J.T. Marienborg, and T.S. Lande, "Delta-sigma modulation in single neurons", in *Circuits and Systems, 2002 IEEE International Symposium on*. IEEE, 2002, vol. 5, pp. V–617.

[82] A. Singh Alvarado, M. Rastogi, J.G. Harris, and J.C. Principe, "The integrate-and-fire sampler: a special type of asynchronous Σ-Δ modulator", *Circuits and Systems (ISCAS), 2011 IEEE International Symposium on*, pp. 2031–2034, 2011.

[83] TA Fulton and GJ Dolan, "Observation of single-electron charging effects in small tunnel junctions", *Physical review letters*, vol. 59, no. 1, pp. 109–112, 1987.

[84] K. Maezawa, M. Sakou, W. Matsubara, T. Mizutani, and H. Matsuzaki, "Experimental demonstration of ideal noise shaping in resonant tunneling delta-sigma modulator for high resolution, wide band analog-to-digital converters", *Japanese journal of applied physics*, vol. 45, pp. 3410, 2006.

[85] Richard F. Tinder, *Asynchronous Sequential Machine Design and Analysis: A Comprehensive Development of the Design and Analysis of Clock-Independent State Machines and Systems*, Synthesis Lectures on Digital Circuits and Systems. Morgan & Claypool Publishers, 2009.