Chapter 2
The Electra Radio

Edgar Satorius, Tom Jedrey, David Bell, Ann Devereaux,
Todd Ely, Edwin Grigorian, Igor Kuperman,
and Alan Lee

This chapter provides an overview of the Electra radio [1]. This is the first programmable software radio that has been developed for space missions. The radio currently accommodates digital binary phase shift keying (BPSK) modulation with both suppressed- and residual-carrier capabilities. The radio is designed to operate over a wide range of data rates from 1 kilobit per second (kbps) to 4 megabits per second (Mbps) and must accommodate frequency uncertainties up to 20 kHz with navigational Doppler tracking capabilities. As such, it is highly programmable and incorporates efficient front-end digital decimation architectures to minimize power consumption requirements. The Electra radio uses field programmable gate array (FPGA) technology to provide the real-time and programmable capabilities. Emphasis in this chapter is focused on the programmable features of the software algorithms implemented in the Electra transceiver as well as the hardware functional specifications.

The objective of the Electra radio, which is based on the original Micro Communications and Avionics Systems (MCAS) prototype [2], is to develop programmable telecommunications systems to meet the unique needs of the National Aeronautics and Space Administration (NASA) for low-power space and planet-surface communications. NASA is moving into an era of much smaller space exploration platforms that require low mass and power. This new era will usher in increasing numbers of miniature rovers, probes, landers, aerobots, gliders, and multiplatform instruments, all of which have short-range communications needs (in this context short-range is defined as non-deep-space links). Presently these short-range (or in situ) communications needs are being met by a combination

of modified commercial solutions (e.g., Sojourner) and mission-specific designs [e.g., Mars '96, Mars '98, Space Technology-2 (ST-2)]. The problem with commercial-based solutions is that they are high-power, high-mass, and single-application-oriented; achieve low levels of integration; and are designed for a benign operating environment. The problem with the mission-specific designs is that the resultant short-range communication systems do not provide the performance and capabilities to make their use for other missions desirable.

Electra is primarily targeted at potential Jet Propulsion Laboratory (JPL) users in the space exploration arena, such as the Mars Exploration Office, which will use this for various microspacecraft short-range communication links, such as orbit–lander, orbiter–rover, orbiter–microprobe, orbiter–balloon, orbiter–sample return canister), ST-4 (orbiter to/from lander link), ST-3 (inter-spacecraft links), and multiple proposed Discovery missions (e.g., balloons, gliders, probes). Electra also has applicability to any space mission that has a short-range communications requirement, such as International Space Station intravehicular and extravehicular wireless communications, X-33 wireless sensor and short-range ground links, Moon missions (e.g., Moonrise), etc. To this end, Electra has been designed to be compatible with the Proximity-1 Space Link Protocol [3].

The ultimate goal of Electra is to achieve a higher level of system integration, thus allowing significant mass, power, and size reductions, at lower cost, for a broad class of platforms. The realization of this goal has resulted in maximizing the transceiver functions performed in the digital domain. These digital functions are implemented with space-qualified FPGA technology. The receiver functions that must be in the analog domain consist primarily of the radio frequency (RF) up- and down-conversions. A space-qualified RF design has been developed through proper selection of parts.

The functionality of the digital portion of the Electra transceiver is exhibited in the block diagram presented in Fig. 2-1. Emphasis in this chapter will be focused primarily on the programmable digital portions of the Electra transceiver (receiver and digital modulator). The programmable digital receiver front-end processing is described in Section 2-1, and the Electra data demodulator is discussed in Section 2-2. Finally, the programmable digital modulator is described in Section 2-3.

2.1 Electra Receiver Front-End Processing

In this section, we describe the Electra receiver front end. With reference to Fig. 2-1, this comprises the automatic gain control (AGC), the analog-to-digital converter (ADC), and the digital downconverter/decimator. These are described separately in this section.

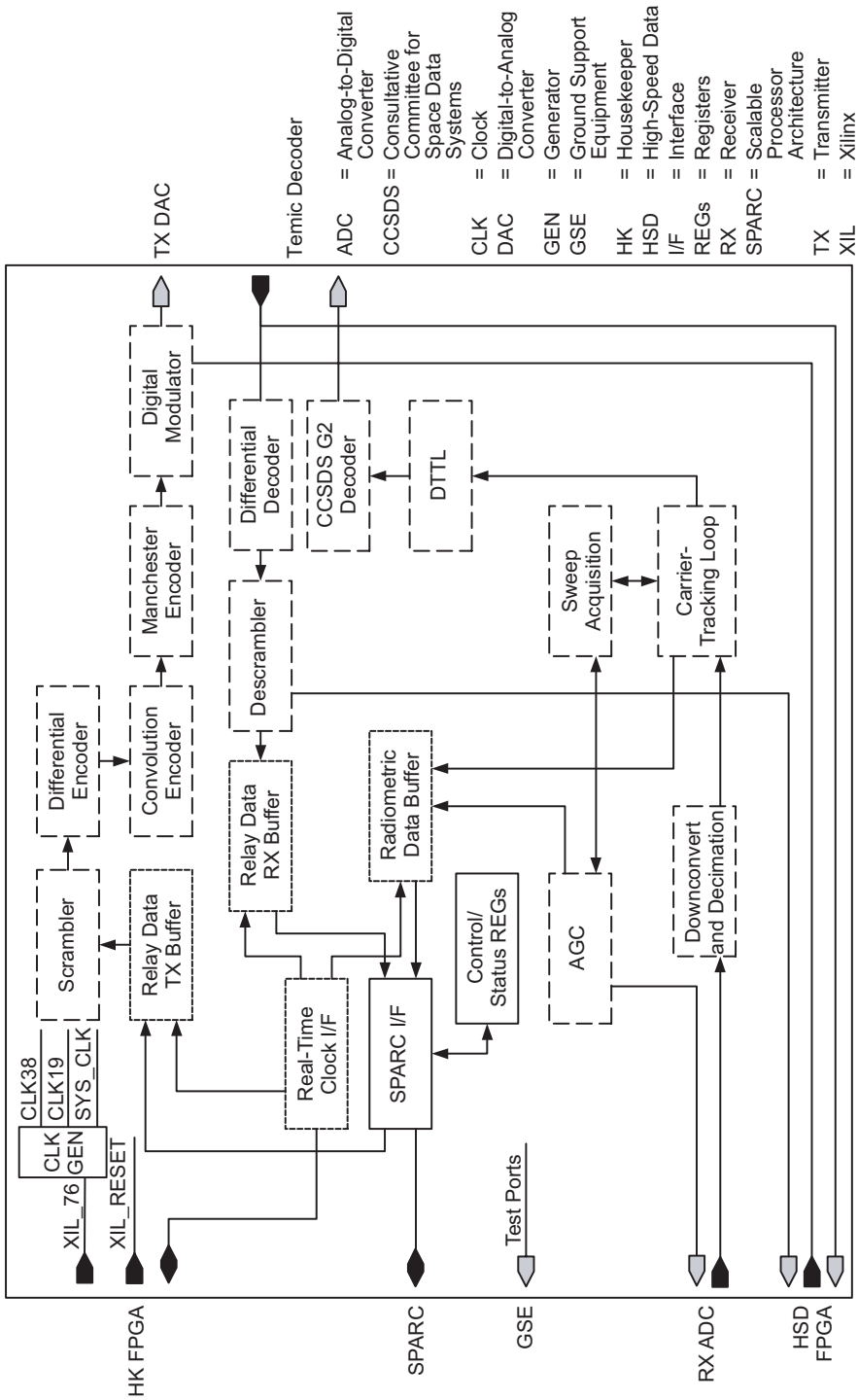

Fig. 2-1. Electra transceiver block diagram.

2.1.1 AGC

The AGC controls the voltage level input to the ADC based on a control voltage signal generated digitally in the FPGA. In particular, the AGC is based on a single feedback control loop design with the AGC control voltage extending back from the carrier-tracking loop (CTL) arm-filter outputs, as indicated in Fig. 2-2. The digital AGC error signal, E_{AGC}, is generated from the CTL arm-filter outputs, I and Q, via

$$E_{\mathrm{AGC}} = K_{\mathrm{gain}} \cdot (1 - \sqrt{I^2 + Q^2}) \tag{2-1}$$

where K_{gain} controls the time constant of the AGC as well as the variance of the resulting amplitude gain estimate. Note that K_{gain} is the only programmable AGC constant. Typically, $K_{\mathrm{gain}} = 2^{-15}$ provides a reasonable compromise between a fast AGC response time (<10 ms) and a low noise gain estimate.

The AGC error signal, Eq. (2-1), is chosen such that the AGC forces the complex magnitude of the CTL arm-filter outputs, $\sqrt{I^2 + Q^2}$, to be unity on average. This in turn helps to regulate the CTL loop bandwidth over a reasonably wide range of input signal levels. The error signal, Eq. (2-1), is integrated in the AGC loop filter, i.e.,

$$V_{\mathrm{out}}[k+1] = V_{\mathrm{out}}[k] + E_{\mathrm{AGC}}[k] \tag{2-2}$$

and the magnitude of the result, $|V_{\mathrm{out}}|$, is used to generate the AGC gain, K_{AGC}, via the nonlinear transfer curve, $f(\cdot)$, i.e.,

$$K_{\mathrm{AGC}}\ (\mathrm{dB}) = f\big(|V_{\mathrm{out}}|\big) \tag{2-3}$$

This gain then is used to scale the AGC input.

A critical issue with this approach is the impact of the AGC on the operation of the ADC as well as the internal digital arithmetic implemented in the FPGA. Ideally the input ADC voltage is scaled to achieve an optimal trade-off between ADC quantization noise and clipping distortion. In contrast, the AGC loop attempts to maintain the complex magnitude of the CTL arm-filter outputs to be unity on average. Thus, there is no guarantee that this criterion of unity root-mean-square (rms) CTL arm-filter outputs will enable the ADC to operate at its optimal input scaling (loading) point or even prevent the ADC from saturating. To alleviate this situation, fixed gains are distributed throughout the digital data paths. These gains are programmable, dependent upon the data rate, and used for purposes of minimizing the effects of digital quantization noise and saturation.

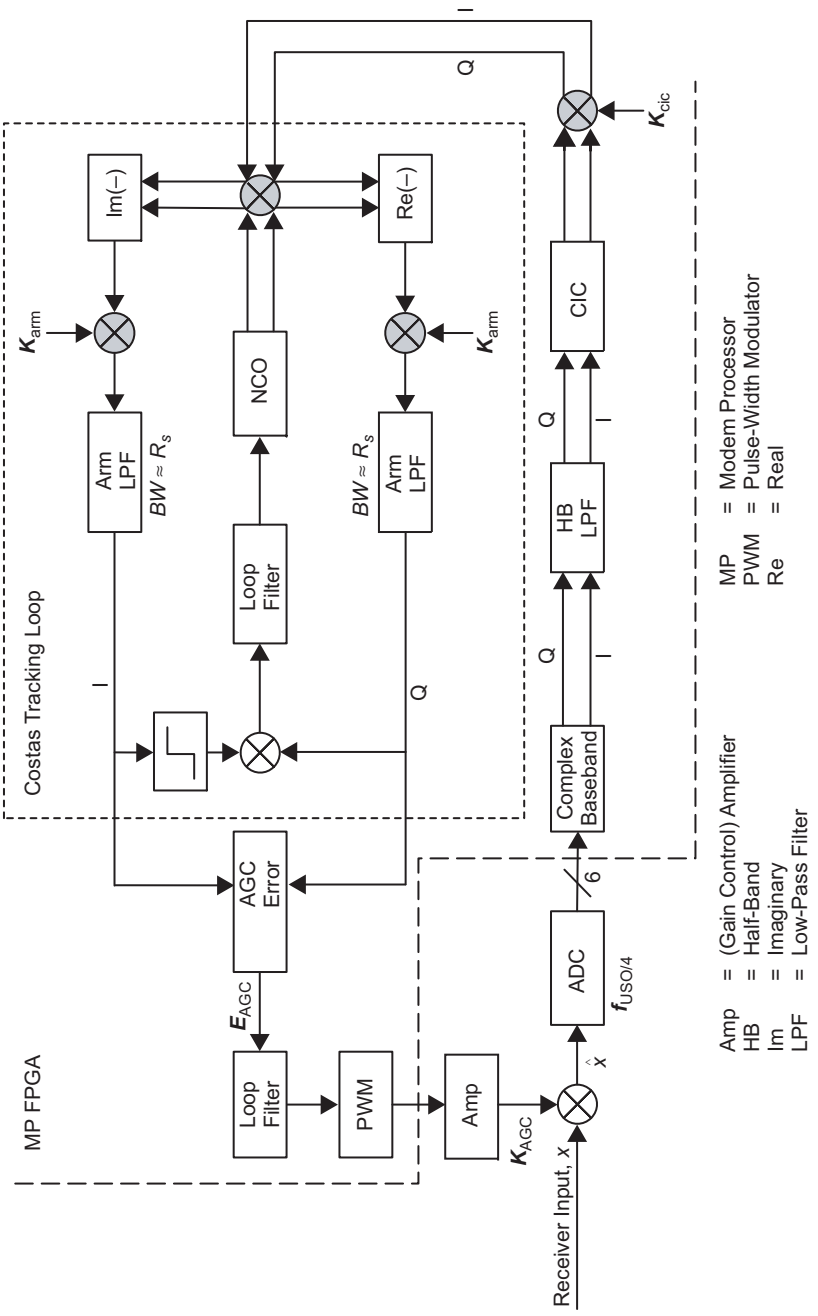

Fig. 2-2. AGC control loop.

2.1.2 ADC

The Electra receiver employs first-order, bandpass sampling wherein the intermediate frequency (IF) band is mapped directly down to digital baseband. Denoting the IF and ADC sampling frequencies by f_{IF} and F_s, respectively, then as long as the frequency band $f_{\mathrm{IF}} - F_s/4 \le f \le f_{\mathrm{IF}} + F_s/4$ coincides with one of the image bands, $kF_s/2 \le f \le (k+1)F_s/2$ for some integer k, the input at f_{IF} will be mapped into the baseband interval $0 \le f \le F_s/2$ as a result of bandpass sampling. This leads to the following condition on f_{IF} and F_s:

$$f_{\mathrm{IF}} = (2n+1) \cdot \frac{F_s}{4} \tag{2-4}$$

where n is a positive integer. Choosing f_{IF} and F_s to satisfy Eq. (2-4) guarantees that the IF frequency will be mapped down to the center of the Nyquist band, i.e., down to $F_s/4$. Furthermore, to maintain the lowest possible ADC sample rate and avoid aliasing, it is desired that F_s just exceed twice the IF filter bandwidth ($2B_{\mathrm{IF}}$).

In designing the bandpass sampling system, the ADC sampling rate was chosen to accommodate an integral, power-of-two number of samples per symbol at all symbol rates: 4.096 megasymbols per second (Msps), 2.048 Msps, $\cdots$, 1 kilosymbol per second (ksps). To achieve a minimum of 4 samples per symbol at the highest symbol rate of 4.096 Msps, an ADC sampling rate in excess of 16.384 MHz is required. In the Electra receiver, F_s is a sub-multiple of the master clock frequency that is provided by an ultra-stable oscillator (USO). A nominal USO frequency, F_{USO}, is 76.72 MHz, and thus the lowest sub-multiple of 76.72 MHz that meets the 4.096-MHz requirement is $F_s = 19.18$ MHz, as indicated in Fig. 2-1. This in turn results in a digital IF frequency of $F_s/4 = 4.795$ MHz. Given F_s, admissible IF frequencies are obtained from Eq. (2-4). An IF near the standard 70 MHz is desired. The closest admissible IF frequency satisfying Eq. (2-4) occurs when $2n + 1 = 15$, corresponding to $f_{\mathrm{IF}} = 15 \cdot 19.18/4$ MHz $= 71.775$ MHz.

Note that different ADC sampling rates may be accommodated simply by choosing different sub-multiples K of the USO frequency, i.e., $F_s = F_{\mathrm{USO}}/K$. This would allow the Electra receiver the flexibility of operating the ADC at slower rates for lower data rate missions, which in turn saves power. However, this flexibility is quite limited since changing the data rate typically requires a change in the IF frequency by virtue of Eq. (2-4), and the analog IF frequency is fixed in the Electra design. For example, if $K = 8$, then $F_s = F_{\mathrm{USO}}/8 = 9.57$ MHz, and the nominal IF frequency, $f_{\mathrm{IF}} = 71.775$ MHz is no longer an odd multiple of F_s. This of course would result in severe digital aliasing.

2.1.3 Digital Downconversion and Decimation

Programmable digital downconversion and decimation directly follow the ADC. The digital complex baseband downconversion scheme depicted in Fig. 2-3 is used and comprises Eq. (2-1), digital complex mixing from $F_s/4 = 4.795$ MHz down to baseband, followed by Eq. (2-2), and digital decimation via a first-order, cascaded integrator-comb (CIC) filter [4,5]. Note that the digital mixing functions do not require multiplication, and furthermore the CIC filters are multiplierless; thus, the entire structure can be implemented efficiently in the FPGA.[1]

The decimation factor M is programmable and is dependent upon the input data rate. To accommodate symbol-timing recovery, M typically is chosen so that there are at least 16 samples per symbol after decimation, except at the highest data rates. So at 1.024 Msps, 2.048 Msps, or 4.096 Msps, M will nominally be set to 1 (no decimation), in which case the remainder of the digital receiver (CTL, symbol-timing recovery, etc.) will run at the input sampling rate, F_s. As the data rate is lowered below $R_s = 1.024$ Msps down to 8 ksps, M is increased proportionately such that

$$\frac{F_s}{R_s \cdot M} = \frac{F_s}{1.024 \text{ MHz}} \tag{2-5}$$

Below $R_s = 8$ ksps, M remains fixed at 128 to accommodate Doppler offsets. Note that as M increases up to 128, more of the input noise to the ADC is filtered out by the CIC filters, thereby reducing the total CIC output power. This is compensated by the AGC as well as programmable, fixed gains following the CIC filters.

2.2 Electra Demodulation

In this section, the various programmable elements of the demodulation process are presented, including the frequency-acquisition and carrier-recovery loop (Section 2.2.1), the Doppler frequency extraction for navigation (Section 2.2.2), and the symbol-timing recovery (Section 2.2.3). In addition, a description of the Electra symbol signal-to-noise (SNR) estimator and convolutional node synchronization (sync) algorithm also are provided (Section 2.2.4). The remaining demodulation functions depicted in Fig. 2-1 (de-scrambling and decoding) are primarily non-programmable, standard functions and thus are not discussed here. In fact, the decoder is implemented outside of the FPGA.

---

[1] Additional, fixed digital (multiplierless) half-band filters [5] are also included in the digital downconverter to remove images—especially when digital decimation is not used ($M = 1$).

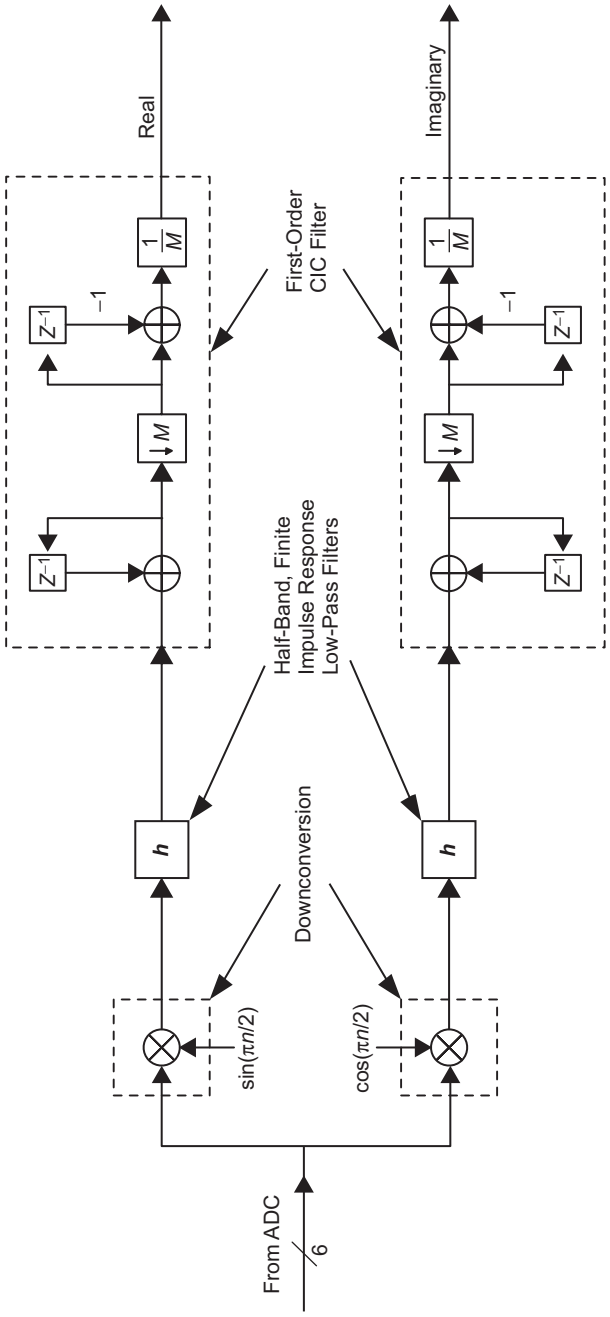

Fig. 2-3. Digital complex basebanding and decimation.

2.2.1 Frequency-Acquisition and Carrier-Tracking Loop

The Electra frequency acquisition and CTL are fully programmable and are designed to acquire and track the frequency and phase of the received signal. The signal can be suppressed-carrier BPSK, residual-carrier BPSK with a modulation index of $\pi/3$, or unmodulated.[2] The CTL operates at all of the required symbol rates from 1 ksps to 4 Msps, signal-to-noise ratios, and CIC-filter-decimated sampling rates. It tracks the carrier reliably when the received signal strength varies over many orders of magnitude. The tracking-loop bandwidth is programmable from 10 Hz to 10 kHz to meet the tracking and acquisition requirements for various communications scenarios. In addition, the frequency acquisition acquires and tracks received signals with maximum frequency offsets of ± 20 kHz, which are typical of ultra-high frequency (UHF), but is programmable to accommodate larger offsets, e.g., at S-band (around 2 GHz) or X-band (around 8 GHz). The CTL also supports navigation by supplying the instantaneous phase of the received signal.

Figure 2-4 shows the block diagram of the Electra CTL, including frequency acquisition. The loop follows the ADC, digital downconverter, and CIC decimation filter as indicated in Fig. 2-1. The complex baseband loop input is multiplied by the complex output of the numerically controlled oscillator (NCO). The product of the complex multiplication is split into the real and the imaginary data paths.

Both the real and imaginary signals are filtered by a pair of identical low-pass arm filters, $G(f)$, with a programmable cut-off frequency. After the arm filters, one or both of the arm filter outputs is used to form the input to the loop filter, $F(f)$, depending on whether the tracking loop is operated in the suppressed-carrier- (Costas) or the residual-carrier-tracking mode (labeled PLL mode in Fig. 2-4). There are three switches (SWs) in the CTL. SW1 and SW2 are selected depending on whether the loop is operated in the Costas loop or the residual-carrier-tracking mode. SW3 is used in the Costas-loop mode. Its position is chosen depending on whether the tracking loop is in the acquisition mode or the tracking mode.

Aside from SW1–3, the CTL loop bandwidth, B_L, as well as the arm filters are programmable. The arm filters are discrete implementations of a first-order low-pass Butterworth filter with programmable cut-off frequency. The arm filters are used to reduce noise in the carrier-tracking loop, but the cut-off frequency should not be so low that the signal power is reduced appreciably by the arm filters. It is found that the cut-off frequency that minimizes the tracking-loop error for the arm filters is approximately equal to the received symbol rate, R_s, for non-return to zero (NRZ)-coded data. Therefore, for the Electra receiver, the

[2] Currently, the Electra FPGA is being re-programmed to also accommodate suppressed-carrier quadrature phase-shift keying (QPSK).

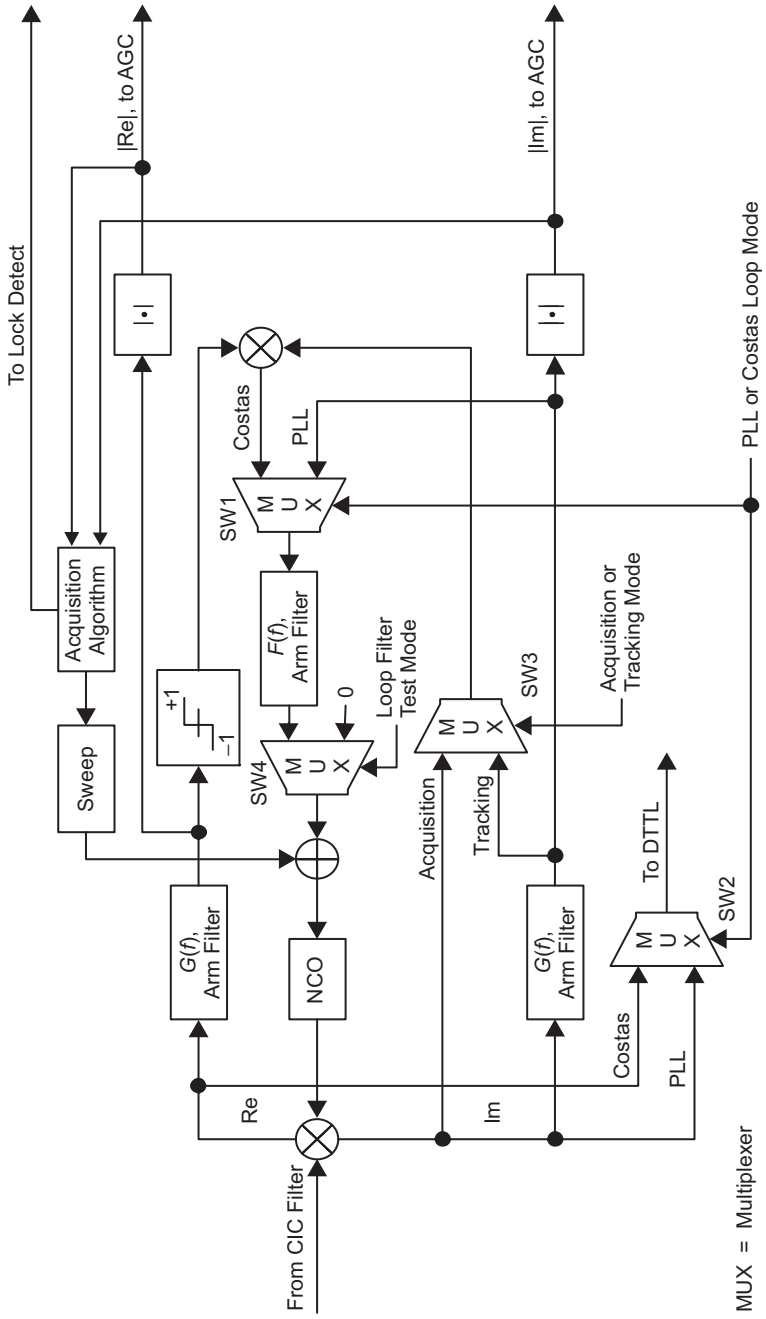

Fig. 2-4. Block diagram of the Electra carrier-tracking loop.

cut-off frequency needs to be programmable between $f_s/128$ and $f_s/4$, where $f_s = F_s/M$ is the decimated sampling rate.

The tracking-loop bandwidth, B_L, should be chosen small enough to provide sufficient loop SNR,

$$\rho \equiv \frac{P \times S_L}{N_0 B_L} \tag{2-6}$$

(P is the signal power input to the loop, S_L is the loop squaring loss, and $N_0/2$ is the two-sided noise power spectral density), and yet large enough to reduce acquisition time. Maintaining a loop SNR ρ in the 15- to 20-dB range requires that B_L vary between 10 Hz and 10 kHz, depending on the data rate and the symbol energy-to-noise spectral level, $E_s/N_0 = P/(N_0 \cdot R_s)$. For example, $B_L = 10$ kHz is acceptable at $R_s = 4.096$ Msps and $E_s/N_0 = 0$ dB (corresponding to a 23-dB loop SNR when $S_L = 3$ dB), whereas $B_L = 10$ Hz is more appropriate at $R_s = 1$ kHz.

The CTL bandwidth thus is chosen as the largest value that satisfies the 15- to 20-dB loop SNR requirement (assuming $E_s/N_0 = 0$ dB). These values for B_L are pre-computed as a function of the data rate and are programmed into the FPGA. In general, two sets of B_L settings are programmed: one for tracking and one for acquisition. The frequency-acquisition algorithm is fully programmable and uses simple saw-tooth sweeping. In particular, the frequency sweep starts at F_{init} (a programmable input parameter) and is incremented in frequency steps, f_{step}, i.e., $F_{acq_{n+1}} = F_{acq_n} + f_{\text{step}}$. The f_{step} is also a programmable input parameter that is typically set to $0.375 \cdot B_L$. At each frequency step, either the real arm output from the CTL, I_n (residual-carrier mode), or the difference between the magnitude of the real and imaginary arm outputs from the CTL, $|I_n| - |Q_n|$ (suppressed-carrier mode), is accumulated over N_{dwell} samples (at the decimated sampling rate) and compared to a threshold Z_{thresh}.

Both N_{dwell}, the number of samples per dwell, and Z_{thresh} are programmable input parameters that are pre-computed based on data rate and modulation type (residual or suppressed carrier). Once Z_{thresh} is exceeded, the sweep is terminated and data demodulation begins. If the threshold is not exceeded, the sweep continues to a maximum of $N \times f_{\text{step}}$ (programmable input parameter dependent upon the sweep range and CTL bandwidth B_L), at which point the sweep returns to the starting frequency, F_{init}, and is repeated until the threshold test succeeds. By re-programming the various input parameters (f_{step}, N, N_{dwell}, and Z_{thresh}), a wide range of Doppler search uncertainties and data rates can be accommodated.

2.2.2 Navigation: Doppler Phase Measurement

Missions like the Mars Relay Orbiter will be required to provide Doppler estimates derived from the received signal. This section describes how Electra Doppler frequency estimates are obtained. The method described herein is applicable to either the suppressed- or residual-carrier CTL mode of operation. The technique basically derives the Doppler frequency estimate from the difference between two instantaneous phase outputs from the phase register of the NCO. The resulting frequency estimate is equivalent to counting the elapsed phase cycles over a fixed time interval, T.

The Electra Doppler frequency estimate, f_{est}, is given by

$$f_{est} \equiv \frac{\theta(t+T) - \theta(t)}{2\pi T} \tag{2-7}$$

where T is the time between two instantaneous NCO phase measurements, $\theta(t)$ and $\theta(t+T)$. Assuming that T is sufficiently large such that $\theta(t+T)$ and $\theta(t)$ are independent (i.e., $T \gg 1/B_L$), then the variance of f_{est}, σ_f^2, is approximated by

$$\sigma_f^2 = \frac{2\sigma_\theta^2}{(2\pi T)^2} \tag{2-8}$$

where σ_θ^2 denotes the variance of the NCO phase measurements. Simulation results have verified the accuracy of Eq. (2-8) for both the suppressed- and residual-carrier CTL modes [2]. As seen from Eq. (2-8), the standard deviation of the Doppler frequency estimation error is inversely proportional to T. Thus, T should be as large as possible while still yielding a meaningful frequency estimate. Typically, T is on the order of 10s to 60s for a 10-minute pass [6]. It should be noted that a model suitable for operational use requires a more detailed development than is presented here [6].

2.2.3 Symbol-Timing Recovery

The Electra symbol-tracking loop is based on a digital data-transition tracking loop (DTTL) with a window size of half-symbol period [7]. The block diagram shown in Fig. 2-5 includes both the DTTL and the DTTL lock-detection logic and provides the word lengths used in the DTTL implementation, e.g., the notation "⟨6,1,t⟩" indicates that the data path is 6 bits wide with 1 integer bit and twos complement representation. The notation "⟨6,1,u⟩" also indicates that the data path is 6 bits wide with 1 integer bit but that the arithmetic representation is unsigned. The primary function of the DTTL is to synchronize the receiver

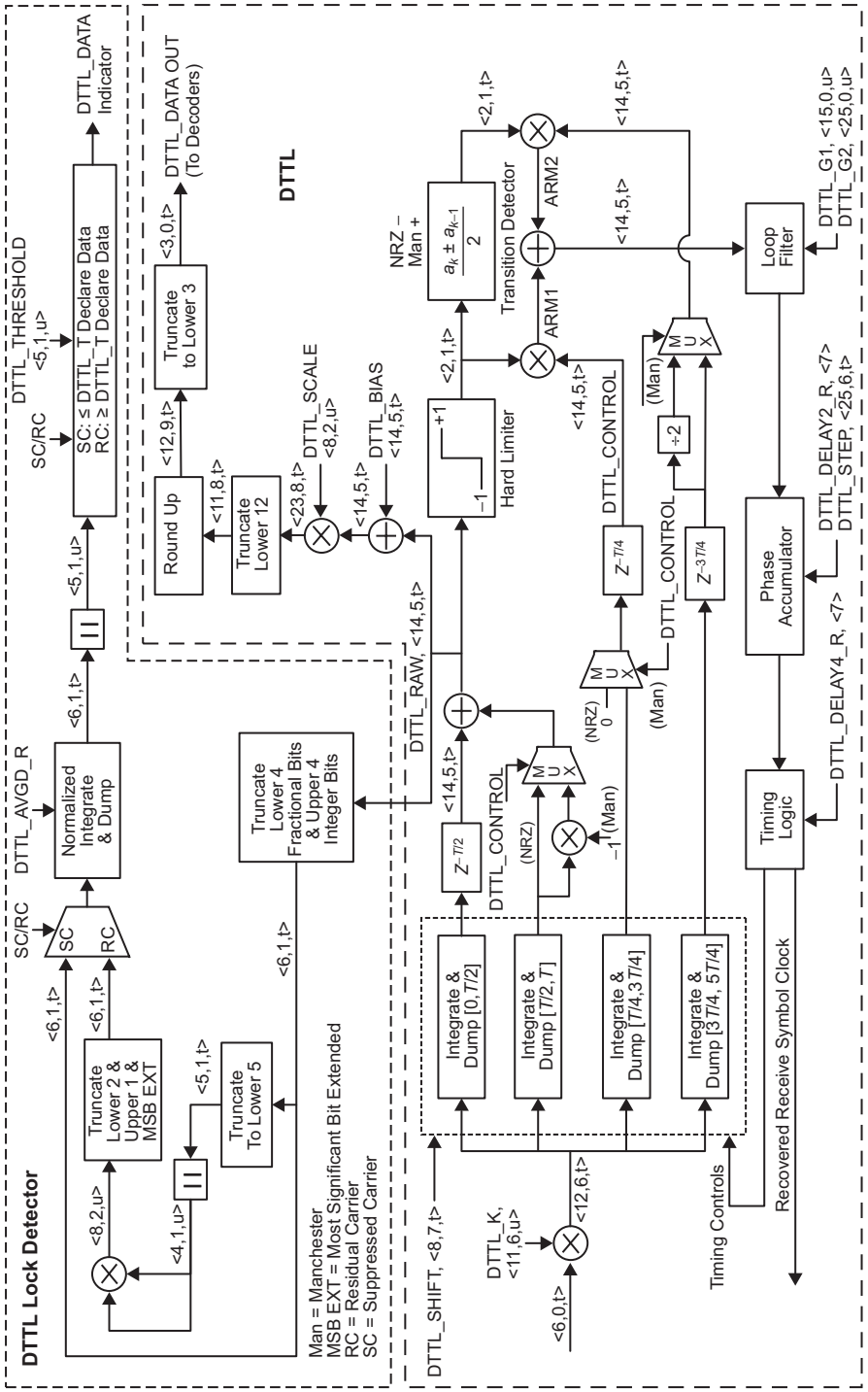

Fig. 2-5. Block diagram of the Electra symbol synchronization architecture.

logic to the received bit stream with the aid of the symbol transitions. The loop is designed to track the symbol boundaries for both NRZ and Manchester-coded data. As a result, a window size of 1/2 a symbol is chosen for the design. The DTTL lock-detection logic is used to determine if the received bit stream has only the carrier or carrier with active symbols. It is used by the downstream Viterbi decoders and to determine when the output of the DTTL has valid symbols.

The DTTL begins with a scaling stage. The only purpose for this stage is to normalize the amplitude of the input to the DTTL to 1. The scaling factor is entered via a programmable input parameter called "DTTL_K." The values for this parameter are obtained via simulation. It is a function of data rate and signal-to-noise ratio. Varying this scaling factor can change the bandwidth of the DTTL. The constant is pre-computed as a function of data rate and is programmed into the FPGA.

The next stage of the DTTL contains a set of integrators. Each integrator is designed to integrate the symbol level over a different segment of the symbol period. Since the window size of this DTTL is 1/2 a symbol, four integrators must be used. Two of them are used for detecting symbol transitions at symbol boundaries, and two are used for the mid-symbol boundaries. The timing of these integrators is controlled by the DTTL timing logic. "DTTL_SHIFT" is a programmable input parameter used to scale the integrator output by $1/2^{\text{DTTL\_SHIFT}}$. Again, DTTL_SHIFT is pre-computed as a function of data rate and is programmed into the FPGA.

The DTTL employs a second-order loop that requires the programmable loop bandwidth input parameter, B_{LD}. Like the CTL bandwidth parameter, the DTTL bandwidth should be chosen small enough to provide sufficient DTTL SNR yet large enough to track data rate changes—which is especially important given that the master (e.g., USO) clock is not necessarily an integral multiple of the data rate. Based on measurements, B_{LD} is computed as a function of data rate and is programmed into the FPGA. There are various other programmable DTTL parameters used to control the integrators and the DTTL phase accumulator (Fig. 2-5). All of these are pre-computed as a function of data rate and programmed into the FPGA.

Finally we describe the DTTL lock detector. A typical Proximity-1 session [3] starts with carrier-only transmission. Therefore, carrier-lock detection alone cannot determine the presence of valid data. This is the purpose for introducing the DTTL-lock detector. It is used to calculate the "power" of the received bit stream. The difference between the current power level and that for the carrier-only case hence can be used to determine the presence of valid data. The top half of Fig. 2-5 depicts the design of the DTTL lock detector. The DTTL lock detector is essentially a power-threshold detector. It converts the signal

level to signal power and compares that against a user input value to determine the lock detection.

Due to the difference in format between the data from the residual-carrier mode (Manchester coded) and those from the suppressed-carrier mode (NRZ coded), the raw data from the DTTL are reformatted before its power can be calculated. The next step is to calculate the respective signal power. The output of this stage is sent to an integrate-and-dump (or sum-and-dump) logic that produces an averaged power measurement on the incoming bit stream. The size of the sample space used for calculating the average is determined by the programmable input parameter "DTTL_AVGD_R." There are only 8 possible values for "DTTL_AVGD_R" ranging from 64 samples to 8192 samples. These are precomputed as a function of data rate and are programmed into the FPGA.

The absolute value of the output from the integrate-and-dump is checked against a programmable threshold called "dttl_threshold." Taking the absolute value of the integrate-and-dump output causes the drop of the sign bit and reduces the data size by 1 bit. The output of the integrate-and-dump is always positive, and the sign bit is superfluous. Depending on whether the received signal is in the residual-carrier mode or the suppressed-carrier mode, the threshold comparison is performed differently. In the case of the suppressed-carrier mode (NRZ), the averaged signal power must be less than or equal to the threshold to be indicative of the valid data. For the residual-carrier case (Manchester), the averaged signal power must be greater than or equal to the threshold to indicate the presence of valid data. The final DTTL data detection flag (dttl_lock) is gated by carrier lock to avoid false indication. The Viterbi decoder is constantly monitoring this flag to determine when the data are valid for processing.

2.2.4 Viterbi Node Sync and Symbol SNR Estimation

The Viterbi decoder is utilized to provide error correction of the received symbols that were convolutionally encoded at the transmitter. Proper Viterbi decoding requires that the correct pair of demodulated data symbols be assigned to the Viterbi code segment. This is termed Viterbi node sync. The method used by Electra to achieve node sync is based on re-encoding [8] as depicted in Fig. 2-6. The Viterbi decoding and re-encoding functions are performed outside of the FPGA on the commercial Temic chip. As is seen from Fig. 2-6, the key behind node sync is the proper choice of the threshold, T. Based on extensive testing, we find that T is a function of the symbol SNR, E_s/N_0. A typical empirically determined threshold curve is defined as follows:

$$T = \begin{cases} -50 \cdot E_s/N_0 + 730, & \text{if } E_s/N_0 \leq 7.5 \\ 355, & \text{if } E_s/N_0 > 7.5 \end{cases} \tag{2-9}$$

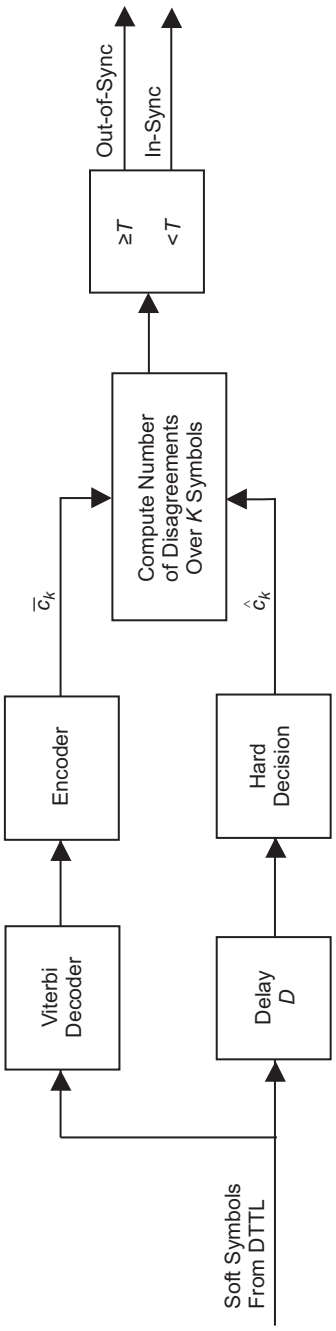

Fig. 2-6. Viterbi code node sync based on re-encoding.

Thus, once E_s/N_0 is estimated (as described below), it is converted to an error threshold (count) T via Eq. (2-9).

A flow diagram of the node sync algorithm that has been incorporated into Electra is presented in Fig. 2-7. There are three basic components of the algorithm: (1) the acquisition portion, wherein initial symbol-sequence phasing is determined, (2) the tracking portion, wherein lock status is continuously updated, and (3) threshold computation, which uses a combination of symbol SNR estimates and empirically derived estimates based on measurements of the decoder errors. These are described herein.

Viterbi sync acquisition starts once the DTTL is in lock. A large error-count threshold of $T_\Phi = 255$ is used to prevent the hardware (Temic) chip from prematurely halting its operation. The first steps in the acquisition process are computation of errors for the two node phases. This is done sequentially. The results of these error counts are denoted by the "Integrate $\Phi_0(S_0, N_\Phi)$" and "Integrate $\Phi_1(S_i, N_\Phi)$" blocks in Fig. 2-7. N_Φ is the number of bits used in the integration process (a multiple of 1000 bits), and S_i denotes error count for each code phase, Φ_i. If the difference in the error counts is sufficiently large, $|S_0 - S_1| \geq \alpha_0$ (α_0 is a pre-determined threshold between 0 and 255) *and* the smallest of the two error counts, $S_{\Phi\,\min}$, is less than a second pre-determined threshold α_1 (between 0 and 2048), then the node phase is set to the phase associated with the minimum error count and acquisition is completed. If not, then the decoder resets and the above process is repeated.

Once acquisition is declared, the node sync algorithm enters the tracking phase. The first steps in the tracking portion are to (1) initialize an un-lock counter U_L to zero and (2) initialize a running error-count average, X_A. During tracking, the number of bits used in computing the error counts is set to a multiple, $K = 1, 2, 4, 8, \cdots$, of N_Φ used for acquisition. Consequently, X_A is initialized to $K \cdot S_{\Phi\,\min}$. This is used in setting the tracking-error-count threshold T_S, as will be described in the next paragraph. Once this initialization occurs, then the error count is measured (S_S in Fig. 2-7) and compared against T_S. If $S_S < T_S$, then U_L is maintained at 0 and the error count average, X_A, is updated via $X_A = (X_A + S_S)/2$. Note that this represents a simple, exponential running average of the form

$$X_A(n) = \alpha \cdot X_A(n-1) + (1-\alpha) \cdot S_S(n) \tag{2-10}$$

where $\alpha = 1/2$ and n is a sequential time index. This average converges quite quickly (in just a few steps) and provides a reasonably smoothed estimate of the error count. If $S_S \geq T_S$ in Fig. 2-7, then U_L is incremented by 1 and compared against a threshold, L_T, between 0 and 15. As long as $U_L \leq L_T$, then the tracking loop proceeds. If not, then the decoder resets and acquisition is re-initiated.

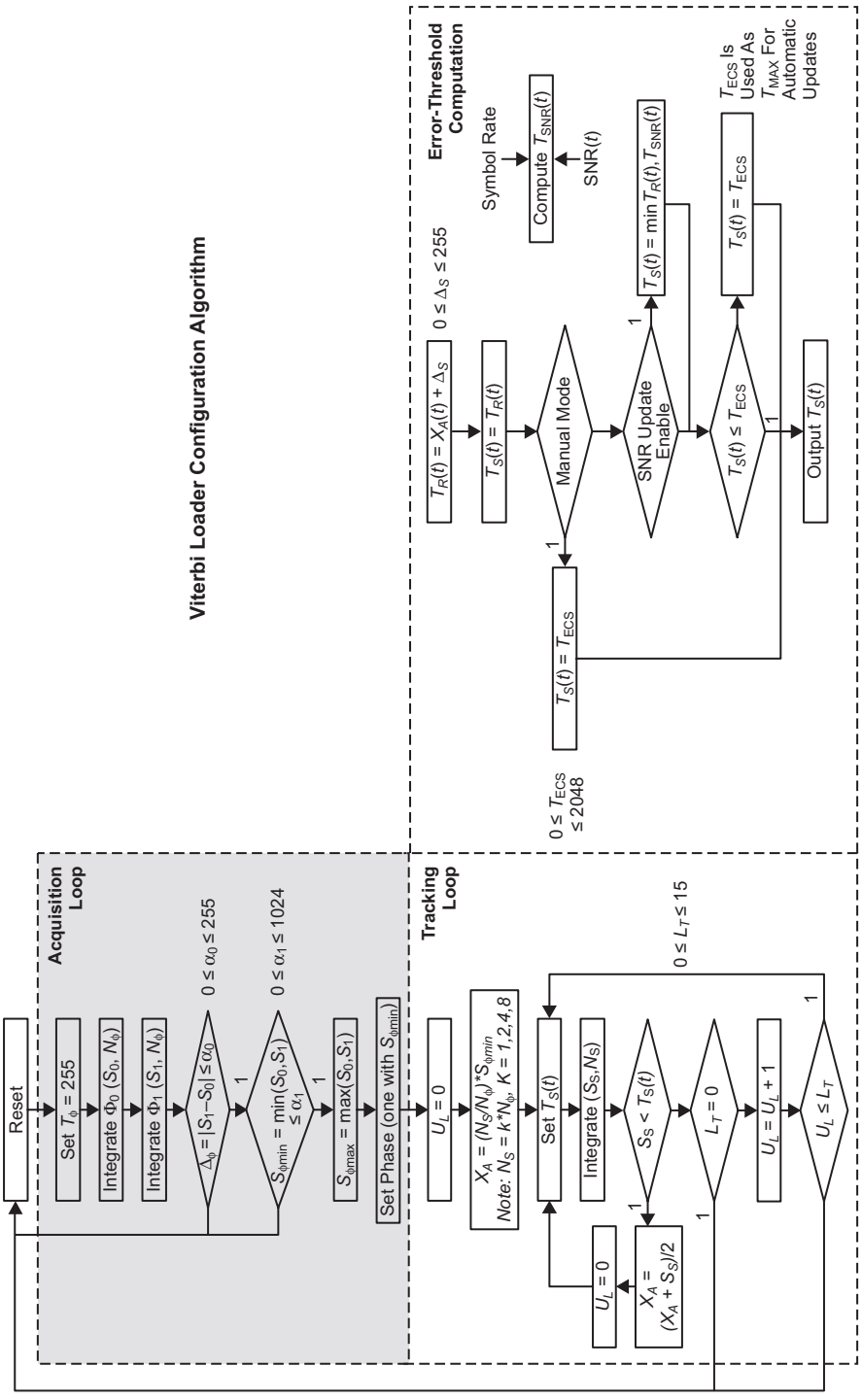

Fig. 2-7. Flow diagram of the Viterbi code node synchronizer.

The tracking-error-count threshold T_S is computed based on both the averaged error count X_A and the symbol SNR estimate. Initially, the threshold is set equal to $T_R \equiv X_A + \Delta_S$ (Δ_S is a pre-determined constant between 0 and 255). Then, assuming the symbol SNR estimator is enabled, a threshold $T_{\rm SNR}$ is computed from the above empirical relationship, Eq. (2-9). The tracking-error-count threshold T_S then is set to the minimum of T_R and $T_{\rm SNR}$ (if the symbol SNR estimator is not enabled, $T_S = T_R$). In addition, a manual override threshold, $0 \le T_{ECS} \le 2048$, is available if desired.

The various parameters defined above (e.g., T_{ECS}, α_0, α_1, N_Φ, etc.) are programmable and typically are determined from calibration measurements made on a given Electra unit. The symbol SNR, E_s/N_0, on the other hand, must be estimated from the received data. Techniques for estimating symbol SNR are described in Chapter 6. A different technique has been incorporated into the current Electra radio based more on FPGA constraints than on performance considerations. However, the Electra FPGA will be re-programmed to accommodate a better symbol SNR estimator based on the techniques described in Chapter 6.

A block diagram of the symbol SNR estimator currently used in Electra is presented in Fig. 2-8. The one-bit control signal, pll_mode, indicates whether suppressed carrier (pll_mode = 0) or residual carrier (pll_mode = 1) is being received. In the former case, the carrier-tracking loop is set to the Costas mode and the input data to the SNR indicator comprise time-averaged samples of the magnitude I- and Q-arm filter outputs (see Figs. 2-2 and 2-4). When pll_mode = 1, the carrier-tracking loop is set to the phase-locked loop (PLL) mode and the input data to the SNR indicator comprise time-averaged samples of the I- and residual-I-arm filter outputs. In practice, symbol SNR estimates are obtained via a table lookup procedure. Specifically, smoothed measurements of I_amp_k/Q_amp_k are collected at different symbol SNRs (see Fig. 2-8). A table of ratios then is created by curve fitting the measurements (two such tables are created, corresponding to suppressed- or residual-carrier mode). The resulting table comprises 64 ratios corresponding to symbol SNRs ranging from 0 dB to $(10 - 1/64)$ dB in 1/64-dB steps. Given the measurements, I_amp_k, Q_amp_k, the lookup procedure comprises the following steps: (1) Find $\min_j |\rho * Q\_amp_k * \text{Table}\ (j) - I\_amp_k|$, where "Table (j)" denotes the 64-element table of ratios and ρ is a constant that is nominally set to 1 but can be fine-tuned so that the single table can accommodate the different data rates, and (2) given the table index j_0 corresponding to the smallest magnitude residual in the first step, compute the symbol SNR estimate (dB) from $\hat{E}_s/N_0 = j_0/8 + j_0/32 + \text{bias}$. The bias is nominally zero but is also fine-tuned for different data rates.

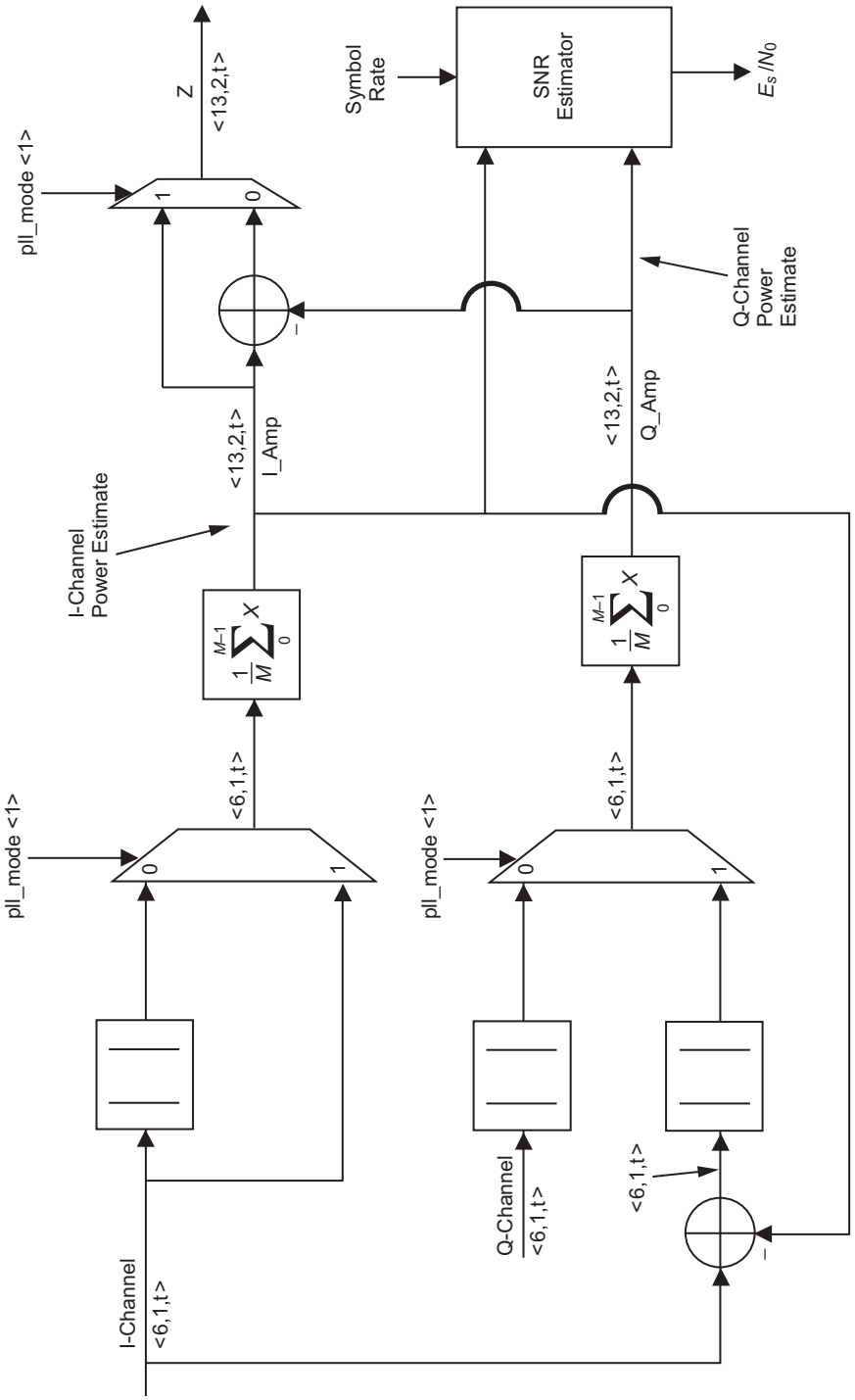

Fig. 2-8. Symbol SNR architecture.

As currently implemented in the residual-carrier mode, ρ is always equal to 1 for all data rates. The table entries "Table (j)" and bias offsets are obtained during system calibration and are uploaded to the Electra FPGA.

2.3 Electra Digital Modulator

The Electra modulator is fully programmable and is implemented on the FPGA. A block diagram of the modulator is depicted in Fig. 2-9. The coding (differential, convolutional encoding) and scrambling functions also are implemented on the FPGA. However, these typically are not programmed and thus are not described herein. Instead, we describe the programmable digital modulator section extending from the coders to the digital-to-analog converter (DAC) outputs. It is noted that in order to support carrier-only operations as required for the Proximity-1 protocol, a user-controlled variable called "enc_enable" is provided to force the final output of the encoder chain to a constant when needed. The output of the encoder is forced to 0 when setting "enc_enable" to 0. It is also noted that the modulator can accommodate dual-channel data inputs (a_n, b_n as depicted in Fig. 2-9). For BPSK modulation, only one input is required (and thus only one coder). However, the modulator also can accommodate QPSK modulation and in fact currently is being reprogrammed to include the suppressed-carrier, QPSK mode. The general purpose software that implements a protocol on an actual spacecraft is described in [9].

Manchester encoding typically is enabled for residual-carrier transmissions. It follows the coder chain as indicated in Fig. 2-9 and is considered part of the digital modulator. When enabled, the encoder converts NRZ waveforms to biphase-level-represented waveforms. In either case (Manchester or NRZ), the output is initially a binary (1,0) stream. After Manchester or NRZ encoding, the encoded binary data are converted to bipolar data. Typically, a logical zero is mapped into a phase of zero radians, and a logical one is mapped into a phase of π radians. This requires that for suppressed carrier, BPSK, the in-phase channel depicted in Fig. 2-9 (with $\cos\beta = 0$) is always zero. However, for hardware implementation, it is difficult to output an algebraic zero to the analog modulator (following the DACs). Thus, for the suppressed-carrier BPSK mode, the data are put on both I and Q channels ($a_n = b_n$), which essentially rotates the BPSK modulation by 45 deg. For the residual-carrier mode, the in-phase channel in Fig. 2-9 (prior to multiplication by $\cos\beta$) is always set equal to 1.

Prior to describing the different operational modes of the Electra digital modulator, we first point out that there are several programmable parameters that determine the power output from the modulator as well as gain and bias shifts on the two output channels from the digital modulator that are input to the DACs and analog modulator (Fig. 2-9). Specifically, for suppressed-carrier BPSK,

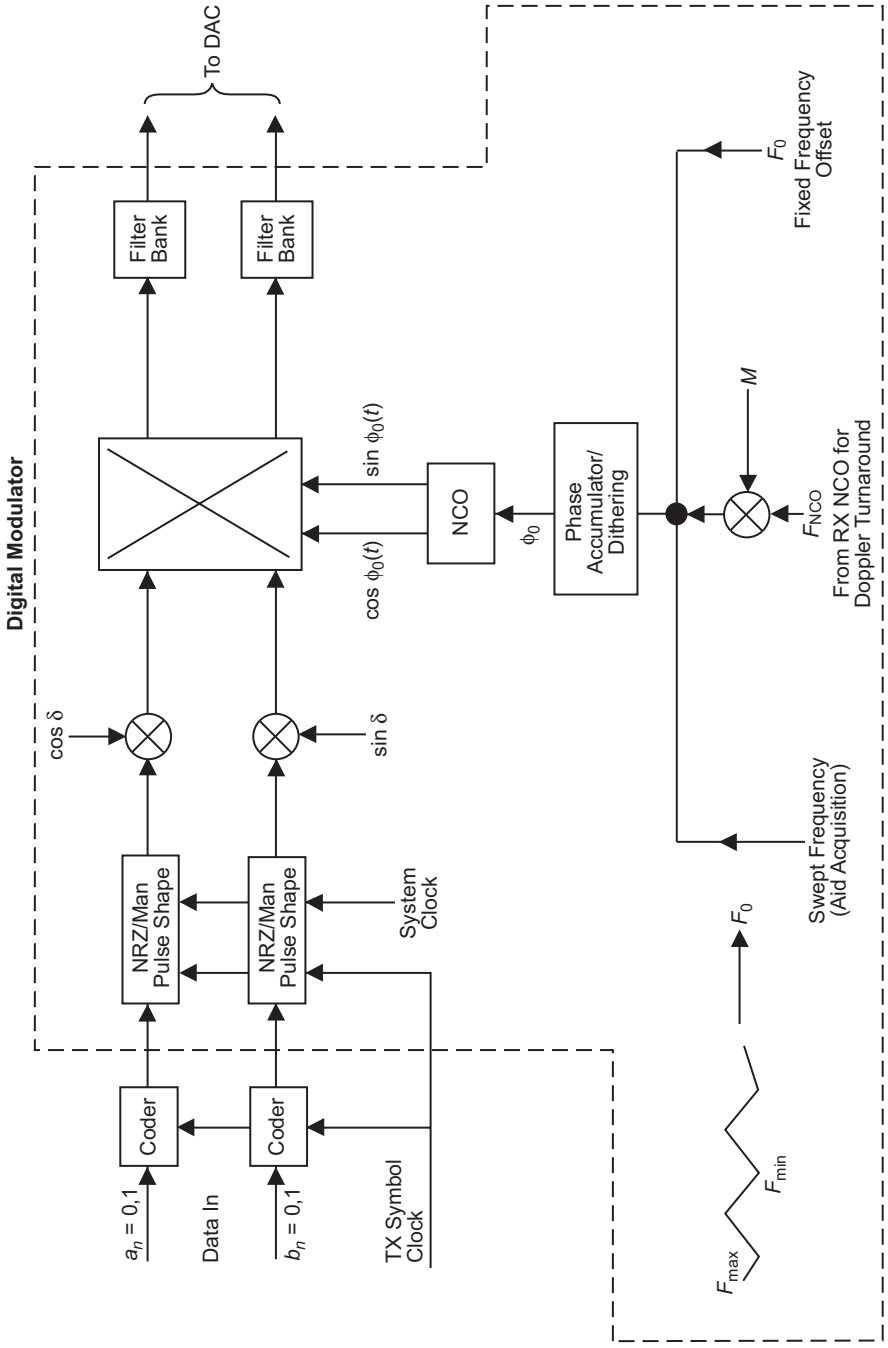

Fig. 2-9. Electra modulator block diagram.

the bipolar input levels to the complex multiplier depicted in Fig. 2-9 are identical and are computed from

$$\pm 0.5 \cdot \sqrt{T_{pow}} \tag{2-11}$$

where T_{pow} is a programmable input parameter controlling the modulator output power. For residual-carrier BPSK, the input levels to the complex multiplier are computed from

$$\text{In-phase} = 0.5 \cdot \sqrt{T_{pow}} \cdot \cos\beta \tag{2-12}$$

$$\text{Quadrature} = \pm 0.5 \cdot \sqrt{T_{pow}} \cdot \sin\beta \tag{2-13}$$

where β is the programmable modulation index parameter (nominally set to $\pi/3$ radians). Note that the in-phase component is unipolar and is the source of the residual carrier. The quadrature channel represents the data component of the residual-carrier waveform. In addition to β and T_{pow}, programmable gain and bias shifts are incorporated on the two output channels from the digital modulator that are input to the DACs. These are critical during system calibration in minimizing mixer images and carrier bleed-through at the output of the analog modulator.

Six operational modes have been implemented in the Electra digital modulator:

(1) Data only

(2) Doppler turnaround

(3) Offset frequency with no Doppler turnaround

(4) Offset frequency with Doppler turnaround

(5) Frequency sweeping with no Doppler turnaround

(6) Frequency sweeping with Doppler turnaround

These different operational modes are controlled via a user-controlled variable modulator_mode (3 bits wide). For mode 0, the modulator NCO is effectively turned off by forcing the NCO outputs to be constant. Mode 1 is the basic Doppler turnaround mode where the input frequency F_{NCO} to the demodulator CTL NCO phase accumulator is multiplied by a turnaround ratio (Fig. 2-9). The turnaround ratio is a programmable input parameter. For Mode 2, the modulator NCO generates in-phase and quadrature sinewave components based on a

fixed carrier offset that is a programmable input parameter. Doppler turnaround is disabled for Mode 2, whereas for Mode 3 it is enabled along with the fixed carrier offset. As indicated in Fig. 2-9, the input to the NCO phase accumulator for Mode 3 is the sum of the turnaround and fixed offset frequencies. Modulator modes 4 and 5 are similar to modes 2 and 3 except that, instead of a fixed-frequency waveform, the modulator NCO generates a swept-frequency waveform that can be used to aid frequency acquisition at the receiver. The transmit sweep parameters (sweep rate, frequency limits, etc.) are all programmable input parameters. These different modes allow a large range of operational scenarios for the Electra transceiver.

References

[1] C. D. Edwards, T. C. Jedrey, E. Schwartzbaum, A. S. Devereaux, R. DePaula, M. Dapore, and T. W. Fischer, "The Electra Proximity Link Payload for Mars Relay Telecommunications and Navigation, IAC-03-Q.3.A06," *International Astronautical Congress 2003*, September–October 2003.

[2] M. Agan, A. Gray, E. Grigorian, D. Hansen, E. Satorius, and C. Wang, "Micro Communications and Avionics Systems First Prototype (MCAS1): A Low Power, Low Mass In Situ Transceiver," *The Telecommunications and Mission Operations Progress Report 42-138, April–June 1999*, Jet Propulsion Laboratory, Pasadena, California, pp. 1–35, August 15, 1999. http://ipnpr.jpl.nasa.gov/progress_report/42-138/138I.pdf

[3] Consultative Committee for Space Data Systems, *CCSDS 211.0-B-1: Proximity-1 Space Link Protocol*, Blue Book, issue 1, October 2002.

[4] E. Hogenauer, "An Economical Class of Digital Filters for Decimation and Interpolation," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 29, no. 2, pp. 155–162, 1981.

[5] A. Kwentus, Z. Jiang, and A. N. Willson, Jr., "Application of Filter Sharpening to Cascaded Integrator-Comb Decimation Filters," *IEEE Transactions on Signal Processing*, vol. 45, no. 2, pp. 457–467, 1997.

[6] T. Ely, "Two Way Electra Phase and Integrated Doppler," *AIAA GNC Conference*, 2005, to appear.

[7] W. C. Lindsey and M. K. Simon, *Telecommunication Systems Engineering*, Englewood Cliffs, New Jersey: Prentice-Hall, 1973; reprinted by Dover Press, New York, 1991.

[8] U. Mengali, R. Pellizzoni, and A. Spalvieri, "Soft-Decision-Based Node Synchronization for Viterbi Decoders," *IEEE Transactions on Communications*, vol. 43, pp. 2532–2539, September 1995.

[9] K. Peters, *Electra Baseband Processor Module (BPM) Payload Controller Software Release Description Document*, JPL D-22719 (internal document), Jet Propulsion Laboratory, Pasadena, California, September 29, 2004.