Whole Symbol Moments SNR Estimator (WSME) Analysis and Implementation

Igor Kuperman* and Edgar Satorius*

ABSTRACT. — Adaptive data rate (ADR) functionality was added to the Mars Reconnaissance Orbiter (MRO) Electra software-defined radio (SDR) via software and firmware uploads to its modem in October 2011. An integral part of ADR is the symbol signal-to-noise ratio (SNR) estimation algorithm. The Whole Symbol Moments SNR Estimator (WSME) algorithm has been developed for ADR control and was included in the October 2011 upload. It is the subject of this article. The WSME architecture is described and performance results are presented via analysis, simulation, and data from an inflight MRO test on March 5, 2012.

I. Introduction

The Mars Reconnaissance Orbiter (MRO) was launched in 2005 with the Electra software-defined radio (SDR) as the telecommunications payload for proximity communications with rovers and landers on Mars. Adaptive data rate (ADR) functionality was added to the MRO Electra SDR via software and firmware uploads to its modem in October 2011. An integral part of ADR is the symbol signal-to-noise ratio (SNR) estimation algorithm. The Whole Symbol Moments SNR Estimator (WSME) algorithm has been developed for ADR control and was included in the October 2011 upload. It is the subject of this article.

The WSME is required to operate at all symbol rates between 1 and 4096 ksps. This is particularly challenging at the highest symbol rates, especially when there are not an integer number of samples per symbol (as in the case of MRO). For example, a nominal 19.18-MHz sampling rate and a 4096-ksps symbol rate correspond to approximately 4.68 samples per symbol, in which case each symbol output from the symbol data-transition tracking loop (DTTL) will comprise either four samples (32 percent of the time) or five samples (68 percent of the time). Originally it was envisioned that a split-symbol signal-to-noise ratio (SNR) estimator[1] [1,2] could be used for ADR. However, given that each half symbol at 4096 ksps comprises only two or three samples, the resulting SNR estimates exhibit an extremely large variability that render this technique impractical for MRO. Consequently, a modification to the split-symbol SNR estimator has been developed that allows the utilization of whole

* Flight Communications and Systems Section.

[1] M. Wang, "Adaptive Data Rate Design Implementation in Verilog," JPL Interoffice Memorandum (internal document), Jet Propulsion Laboratory, Pasadena, California, May 31, 2005.

The research described in this publication was carried out by the Jet Propulsion Laboratory, California Institute of Technology, under a contract with the National Aeronautics and Space Administration.

symbols. As discussed in Section II, this is accomplished by removing the data polarity of the binary phase-shift keyed (BPSK) data symbols, thereby resulting in more samples used in generating the sample SNR estimates. This significantly stabilizes the resulting symbol SNR estimates. Implementation of the WSME is described in Section III.

II. WSME Analysis

A block diagram of the WSME is presented in Figure 1 (including the fixed-point word-lengths). This represents a modification of the original split-symbol estimator [1,2] in that the SNR estimate is obtained from whole (soft) symbols out of the DTTL (as indicated in Figure 1). The basic concept is to remove the data polarity of the BPSK data symbols, thereby allowing whole symbols to be used in estimating the SNR. This concept is related to a carrier synchronization approach developed in [3] wherein data stripping was used to improve carrier tracking losses. It also represents a high SNR limit to the maximum-likelihood SNR estimator.

To better understand the performance of this estimator, we denote the soft symbol stream at the input to the WSME by

$$Y(k) = \sqrt{P_s} \cdot a_k + n(k)$$

where $a_k = \pm 1$ are the BPSK data symbols, P_s represents the signal power, and $n(k)$ represents the additive noise samples. Following [3], we can express the product $Y(k) \cdot sign(Y(k))$ as

$$Y(k) \cdot sign\big(Y(k)\big) = \sqrt{P_s} \cdot a_k \cdot \hat{a}_k + N(k)$$

where $\hat{a}_k$ denotes the estimate of the $k-th$ data symbol (generated by $sign(Y(k))$) and $N(k) \equiv n(k) \cdot sign(Y(k))$. The outputs $Y_{aj}(k)$ and $Y_{bj}(k)$ (Figure 1) are given by

$$\begin{aligned} Y_{aj}(k) &= Y(k) \cdot sign\big(Y(k)\big) \\ Y_{bj}(k) &= Y(k-1) \cdot sign\big(Y(k-1)\big) \end{aligned}$$

With reference to Figure 1, we have that

$$\begin{aligned} m_p = \langle Y_{aj} \cdot Y_{bj} \rangle_M &= \langle P_s \cdot a_k \cdot \hat{a}_k \cdot a_{k-1} \cdot \hat{a}_{k-1} \rangle_M + \big\langle N(k) \cdot \sqrt{P_s} \cdot a_{k-1} \cdot \hat{a}_{k-1} \big\rangle_M \\ &\quad + \big\langle N(k-1) \cdot \sqrt{P_s} \cdot a_k \cdot \hat{a}_k \big\rangle_M + \langle N(k) \cdot N(k-1) \rangle_M \\ &\approx P_s \cdot \langle a_k \cdot \hat{a}_k \cdot a_{k-1} \cdot \hat{a}_{k-1} \rangle_M \end{aligned}$$

where $\langle \cdot \rangle_M$ denotes an average over M symbols. We also have that

$$\begin{aligned} m_{ss} = \big\langle (Y_{aj} + Y_{bj})^2 \big\rangle_M &= \langle P_s \cdot (a_k \cdot \hat{a}_k)^2 \rangle_M + \langle P_s \cdot (a_{k-1} \cdot \hat{a}_{k-1})^2 \rangle_M + 2\langle P_s \cdot a_k \cdot \hat{a}_k \cdot a_{k-1} \cdot \hat{a}_{k-1} \rangle_M \\ &\quad + N\langle (k)^2 \rangle_M + \langle N(k-1)^2 \rangle_M + 2\langle N(k) \cdot N(k-1) \rangle_M \\ &\quad + 2\big\langle N(k) \cdot \sqrt{P_s} \cdot a_k \cdot \hat{a}_k \big\rangle_M + 2\big\langle N(k-1) \cdot \sqrt{P_s} \cdot a_{k-1} \cdot \hat{a}_{k-1} \big\rangle_M \\ &\quad 2\big\langle N(k) \cdot \sqrt{P_s} \cdot a_{k-1} \cdot \hat{a}_{k-1} \big\rangle_M + 2\big\langle N(k-1) \cdot \sqrt{P_s} \cdot a_k \cdot \hat{a}_k \big\rangle_M \\ &\approx 2P_s + 2P_s \langle a_k \cdot \hat{a}_k \cdot a_{k-1} \cdot \hat{a}_{k-1} \rangle_M + \langle N(k)^2 \rangle_M + \langle N(k-1)^2 \rangle_M \end{aligned}$$

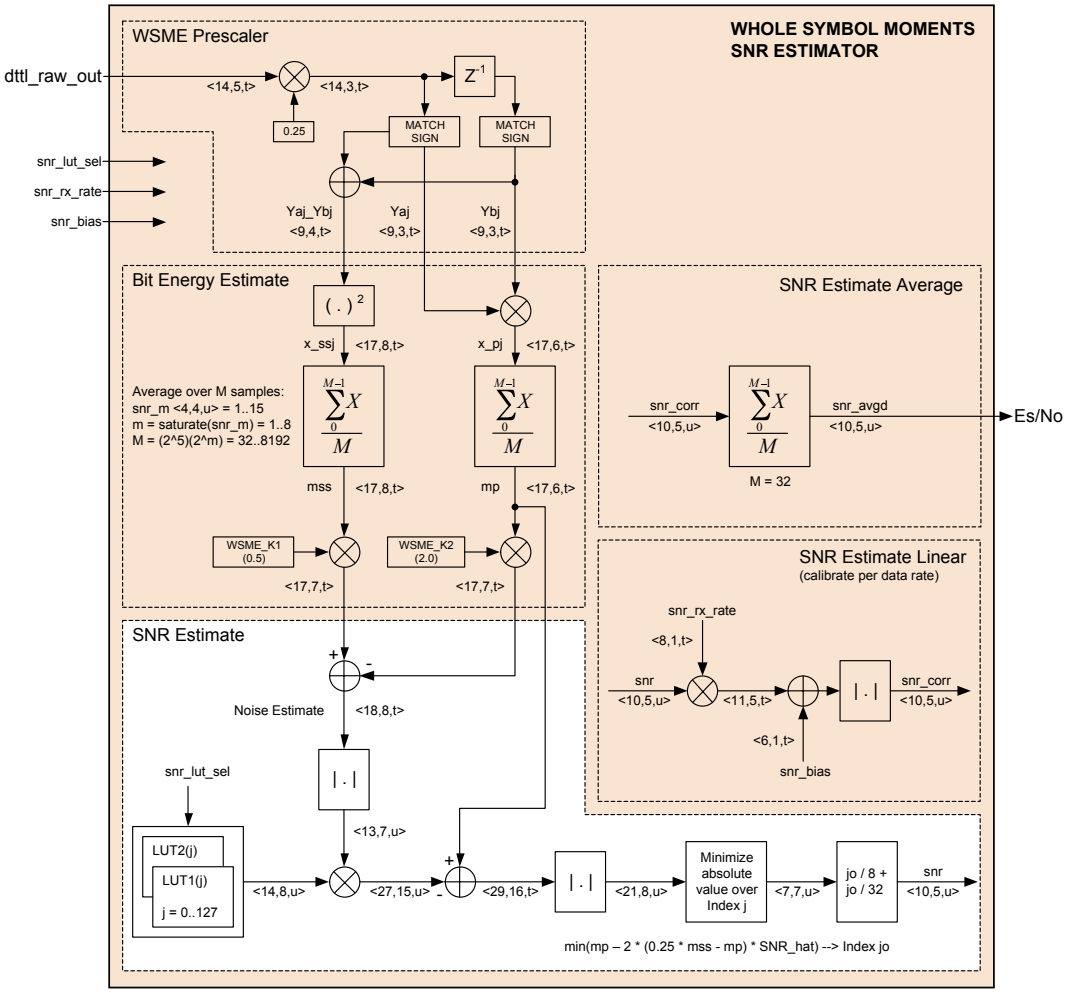


Figure 1. WSME architecture.

Thus, from Figure 1 we have that

$$\begin{aligned} \textit{Noise Estimate} = 0.5 \cdot m_{ss} - 2 \cdot m_p \approx P_s - P_s \cdot \langle a_k \cdot \hat{a}_k \cdot a_{k-1} \cdot \hat{a}_{k-1} \rangle_M \\ + 0.5 \cdot \langle N(k)^2 \rangle_M + 0.5 \cdot \langle N(k-1)^2 \rangle_M \end{aligned}$$

For large M, we can approximate the time averages, $\langle \cdot \rangle_M$, with statistical expectations, $E(\cdot)$, in which case we have

$$\begin{aligned} m_p &\approx P_s \cdot E\left(a_k \cdot \hat{a}_k \cdot a_{k-1} \cdot \hat{a}_{k-1}\right) = P_s \cdot E\left(a_k \cdot \hat{a}_k\right) \cdot E\left(a_{k-1} \cdot \hat{a}_{k-1}\right) \\ &= P_s \cdot (1-2p)^2 \end{aligned}$$

$$\textit{Noise Estimate} = 0.5 \cdot m_{ss} - 2 \cdot m_p \approx P_s - P_s \cdot (1-2p)^2 + E\left(N^2\right)$$

where p denotes the probability that $a_k \neq \hat{a}_k$ and we have assumed that

$$E\left(N(k)^2\right) = E\left(N(k-1)^2\right) \equiv E\left(N^2\right)$$

We also assume independence of data symbols such that

$$E\left(a_k \cdot \hat{a}_k \cdot a_{k-1} \cdot \hat{a}_{k-1}\right) = E\left(a_k \cdot \hat{a}_k\right) \cdot E\left(a_{k-1} \cdot \hat{a}_{k-1}\right)$$

Note that for BPSK, $p = 0.5 \cdot erfc\left(\sqrt{SNR_0}\right)$, where SNR_0 denotes the true symbol SNR. Thus, the output SNR estimate is given by

$$SNR_{est} = \frac{m_p}{Noise\ Estimate} \approx \frac{P_s \cdot (1-2p)^2}{P_s - P_s \cdot (1-2p)^2 + E(n^2)}$$
$$= \frac{SNR_0 \cdot \left\{1 - erfc\left(\sqrt{SNR_0}\right)\right\}^2}{4 \cdot SNR_0 \cdot erfc\left(\sqrt{SNR_0}\right) \cdot \left\{1 - erfc\left(\sqrt{SNR_0}\right)\right\} + 1}$$

A plot of SNR_{est} versus SNR_0 is presented in Figure 2.

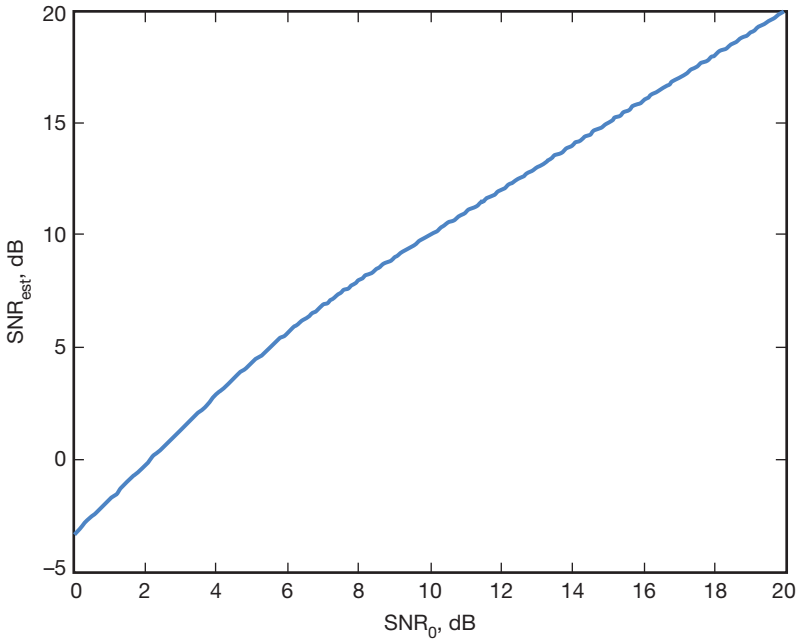


Figure 2. WSME output SNR versus true symbol SNR.

As is seen, the effect of multiplying by $sign(\cdot)$ is to underestimate the true symbol SNR at low symbol SNRs (below 4 dB). However, even at $SNR_0 = 0$ dB, $p = 0.5 \cdot erfc\left(\sqrt{SNR_0}\right) \sim 0.08$, which results in an underestimate of the true symbol SNR by only about 3 dB. Furthermore, this underestimation can easily be accounted for via the table lookup procedure used to generate the final output SNR estimate (see Figure 1) and it significantly stabilizes the resulting symbol SNR estimates, as noted above. Another source of error not included in the above analysis is the impact of bandpass filtering, which distorts the received symbol pulses and produces intersymbol interference, especially at high data rates [1]. However, the table lookup approach will also correct for this distortion.

Another important consideration is the variance of the SNR estimate provided by the WSME. As pointed out to the authors by Ken Andrews,[2] this is a special case of the analysis originally presented by Dolinar and Vilnrotter[3] and later published in [4] where it was shown that the variance of the WSME estimator is comparable to that of the split-symbol estimator for positive dB SNR (SNR >1) but degrades significantly for negative dB SNR. However, it is this range of positive dB SNR that is of particular interest for ADR, and thus WSME is a good candidate for the ADR application. Furthermore, SNR compression at 4096 ksps and high SNR severely limit the split-symbol SNR estimator, making the WSME the best choice for the ADR application.

In generating the symbol WSME SNR estimates, smoothed measurements of m_p/*Noise Estimate*, (defined above) are collected at different symbol SNRs. A table of ratios, i.e., m_p/*Noise Estimate* is then created by curve-fitting the measurements. The resulting table comprises 128 ratios corresponding to symbol SNRs ranging from 0 dB to (20 – 5/32) dB in 5/32 dB steps. The table lookup procedure is provided in Section III. As currently implemented, two tables are used: one for the symbol rates 2048 and 4096 ksps (high-rate table) and the second one for the rates 1024 ksps and below (low-rate table). The nominal low-rate and high-rate table values are provided in Section III, Tables 3 and 4.

The table values are determined by a combination of fixed-point Matlab simulation and laboratory testing. The latter was limited by the availability of various intermediate data (m_p, *Noise Estimate*, etc.). Consequently, a scheme was devised to make table corrections based only on output SNR estimates from the laboratory test setup. Note that in Tables 3 and 4 (Section III), the high-rate ratios approach a much smaller value (~24) than the low-rate ratios (~132). Initially, Matlab fixed-point simulations indicated more comparable ratios; however, laboratory measurements revealed significant compression for larger SNRs — especially at the highest data rate. This was corrected using the measured output SNR values.

The correction process comprises the following steps: (1) using the initial Matlab-generated table, lab SNR measurements are collected versus the true input symbol SNR (as determined from a calibrated source); (2) given the measured SNR values, the corresponding ratio m_p/*Noise Estimate* can be determined from the initial Matlab-generated table; (3) establish a new (corrected) table by associating the true input symbol SNR with the ratio determined from step (2), and (4) interpolate the corrected table to a finer grid (128 entries and 5/32 dB grid spacing), which is the final (corrected) table. The corresponding table basically eliminates the compression observed with the initial Matlab-generated table.

The basic procedure is illustrated in Table 1, corresponding to a segment of laboratory SNR data collected at 1024 ksps. In this segment, the original Matlab-generated table (based on simulations at 1024 ksps) produced laboratory WSME SNR estimates that transition from overestimating the true input SNR (e.g., 9.18 dB versus a true SNR of 9 dB) to underestimating the true input SNR (e.g., 12.85 dB versus a true SNR of 13 dB). To correct this, we modify the Matlab-based ratios m_p/*Noise Estimate*, as indicated in Table 1.

---

[2] K. Andrews, personal communication, Section 332B, Jet Propulsion Laboratory, Pasadena, California, April 2012.

[3] S. Dolinar and V. Vilnrotter, "Seminar 331 Viewgraphs on SNR Estimation," JPL Interoffice Memorandum 331-86.2-119 (internal document), Jet Propulsion Laboratory, Pasadena, California, February 13, 1986.

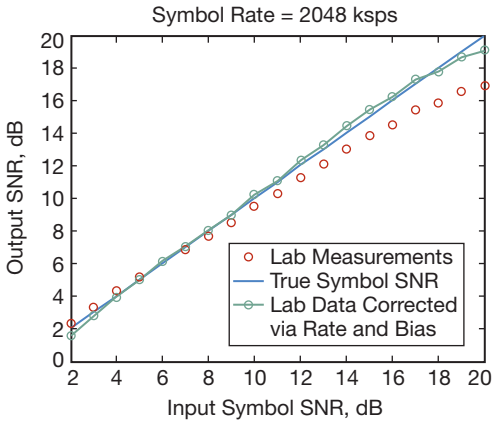
Table 1. Segment of Matlab-generated table of ratios and the corresponding measured SNR values from lab data at 1024 ksps.

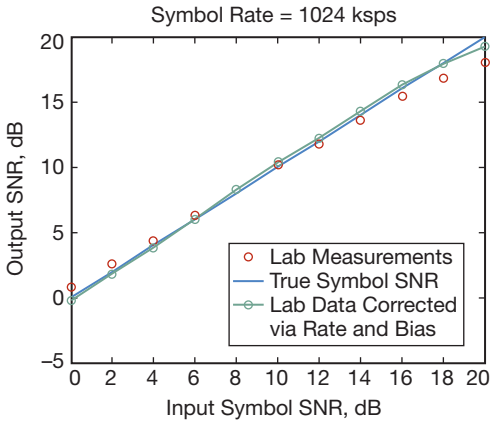
True Input Symbol SNR	Matlab-Generated m_p/*Noise Estimate*	Lab-Measured SNR Using the Matlab Ratios	New Ratios
9	14.1809	9.18	15.048
9.2	15.048		
9.4	15.6207		
9.6	16.1474		
9.8	17.3106		
10	17.8045	10.13	18.607
10.2	18.607		
10.4	19.4431		
10.6	20.3138		
10.8	21.2233		
11	22.1742	11.05	22.1742
11.2	23.1639		
11.4	24.1937		
11.6	25.2718		
11.8	26.3942		
12	27.5613	11.96	27.5613
12.2	28.7771		
12.4	30.047		
12.6	31.3664		
12.8	32.7371		
13	34.1673	12.85	32.7371

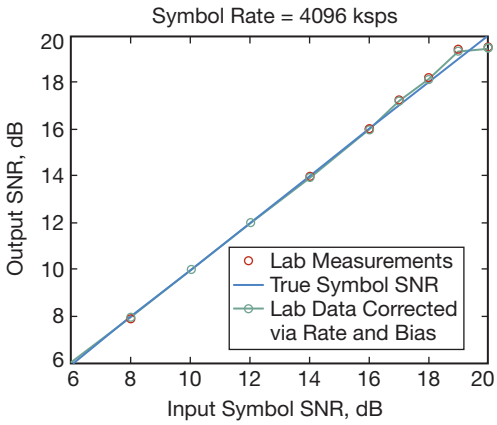
This correction process is carried out at 4096 ksps in the case of the high-rate table and at 512 ksps for the low-rate table. Additional corrections for the other rates are made via the rate and bias parameters indicated in Figure 1: *snr_rx_rate*, *snr_rx_bias*. Representative plots of the symbol SNR estimates from lab measurements are presented in Figure 3. As is seen, after applying rate and bias parameters to the measured data, the compression is practically eliminated and the agreement is quite good. Matlab simulation results indicating anticipated hardware performance at 8 ksps are presented in Figure 4. The SNR table used in these simulations was generated from Matlab simulations at 512 ksps and matched the table used in creating the 1024 ksps estimates in Figure 3. Again, it is seen that the performance of the WSME is quite good.

III. WSME Implementation

The WSME is implemented based on the functional diagram shown in Figure 1. The WSME and the previous Split-Symbol Moments SNR Estimator (SSME) designs both utilize similar architectures. This in turn minimizes implementation changes and enables easier module reuse. A couple of clear advantages over the legacy design should be noted:







- 2048, 4096 results were obtained with modified Matlab table generated at 4096 ksps symbol rate.
- 1024 results were obtained directly from a Matlab table generated at 512 ksps symbol rate.
- Except at the extreme symbol SNR limits, max error between the true symbol SNR and the measured data (corrected by rate and bias) is still less than 0.25 dB.

Figure 3. Measured WSME estimates at 1024, 2048, and 4096 ksps.

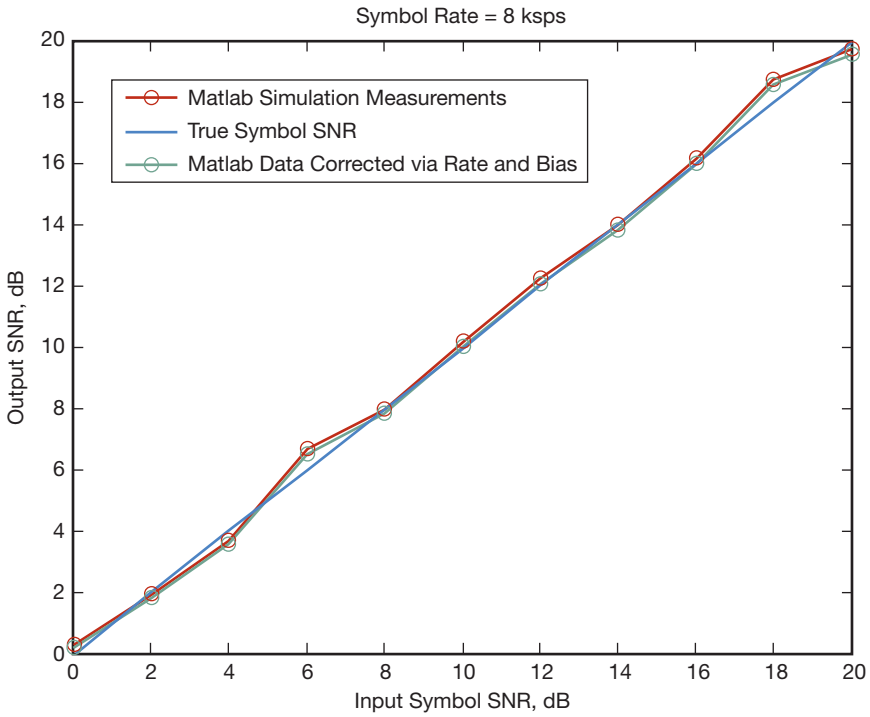


Figure 4. Simulated WSME at 8 ksps.

- Because whole DTTL symbols are used instead of half-symbols, twice as many samples are available for processing.
- Implementation is immune to designs with inconsistent or odd number of samples per symbol.

This implementation of the SNR Estimator no longer operates on half-symbol outputs from the DTTL, thus only a single data input is required. Raw DTTL symbol output (*i_dttl_raw_out_i*) is used in conjunction with several control signals to produce valid Es/No estimates. The entire WSME is operated using a single clock source *i_sys_clk* with *i_sym_clk* being used only as a symbol valid strobe. Both *i_dttl_raw_out_i* and *i_sym_clk* are synchronized to the positive edge of the *i_sys_clk* before entering the SNR Estimator top-level module.

In addition to WSME Prescaler logic, the SNR Estimator is further divided into two submodules: Bit Energy Estimator and SNR Estimator Core. Figure 5 shows the schematic of the SNR Estimator top-level wrapper that illustrates signal interconnects between the WSME Prescaler and two other submodules. I and Q channel power estimations obtained from the sweep acquisition module are added to the SNR estimator top-level wrapper to maintain backward compatibility with the data buffer module that resides in the modem processor (MP) field-programmable gate array (FPGA) top level.

To enable the use of a previous SSME architecture design, a simple WSME Prescaler is added. This prescaler formats the DTTL output data before it enters the Bit Energy Estimator module. First, adjacent DTTL symbols are split into two separate channels *yaj* and *ybj*. Then both channel amplitude signs are matched to emulate DTTL half-symbols. Finally, *yaj* and *ybj* signals are added together to form "super symbols" — *yaj_ybj*. Individual symbols are then quantized to <9,3,t>, while the "super symbols" are quantized to <9,4,t> to avoid saturation.

The main function of the Bit Energy Estimator module is to calculate the signal power and the signal plus noise power. Figure 6 shows the corresponding implementation schematic of the Bit Energy Estimator module.

Three signals are fed from the WSME Prescaler: *i_dttl_yaj*, *i_dttl_ybj*, and *i_dttl_raw_out_i*. Signal plus noise power component (*mss*) is derived by multiplying *i_dttl_yaj* and *i_dttl_ybj* inputs together and then by averaging their product with a sum-and-dump filter. The *i_dttl_raw_out_i* input is squared and then averaged with a sum-and-dump filter to produce the signal power component (*mp*). The sample size *wire_m* of both sum-and-dump filters is programmable through a user-defined variable *snr_m*, represented as <4,4,u> and ranging from 0 to 15. To limit width of sum-and-dump filter accumulators and minimize resource utilization, another variable, *m,* is defined using the following relationship:

$$m = \begin{cases} snr\_m, & if\ snr\_m \leq 8 \\ 8, & otherwise \end{cases}$$

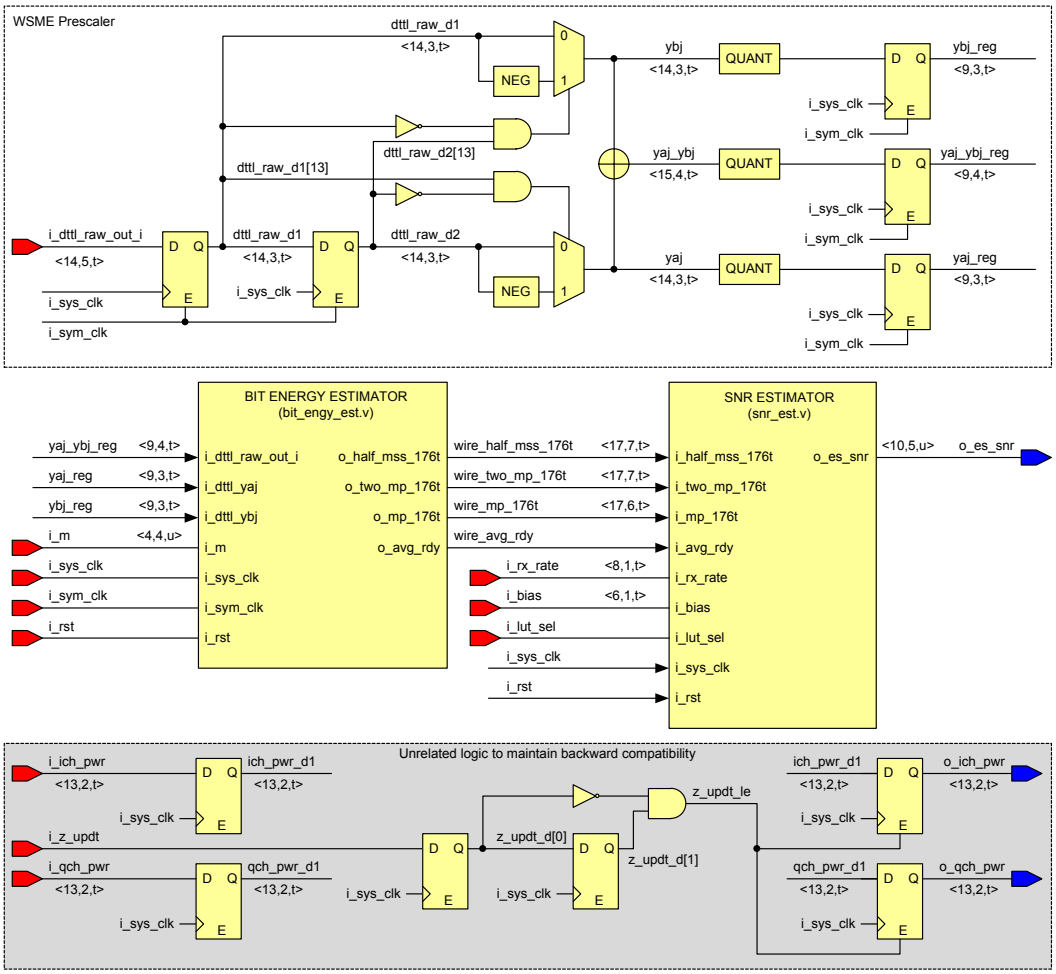


Figure 5. WSME top-level wrapper schematic.

Then the actual value of the sample size is defined using the formula

$$wire\_m = (2^5)(2^m)$$

The relationship of the sum-and-dump filter sample size (*wire_m*) to *snr_m* is shown in Table 2.

Both 30-bit sum-and-dump filter accumulators run at the symbol rate, but are clocked with *i_sys_clk* with *sym_ena* used as a gate to preserve a single-clock, single-edge synchronous design approach throughout the WSME implementation. A 13-bit counter is used to time outputs of the accumulators. Analogous to the accumulators, the counter also increments at the symbol rate on the positive edge of *i_sys_clk* while gated by the *sym_ena* signal. The outputs of both accumulators are then normalized by the sum-and-dump filter sample size. This is achieved by dividing each output by *wire_m*, which is equivalent to the right-shift by (*m* + 5). Normalization is a two-step process. First, the output of each accumulator is divided by 2^m using a shift-right function. Then, because the remaining division is always constant (2^5), it is performed by changing the signal representations from <30,19,t> to <30,14,t> and from <30,21,t> to <30,16,t>, without affecting actual numerical values.

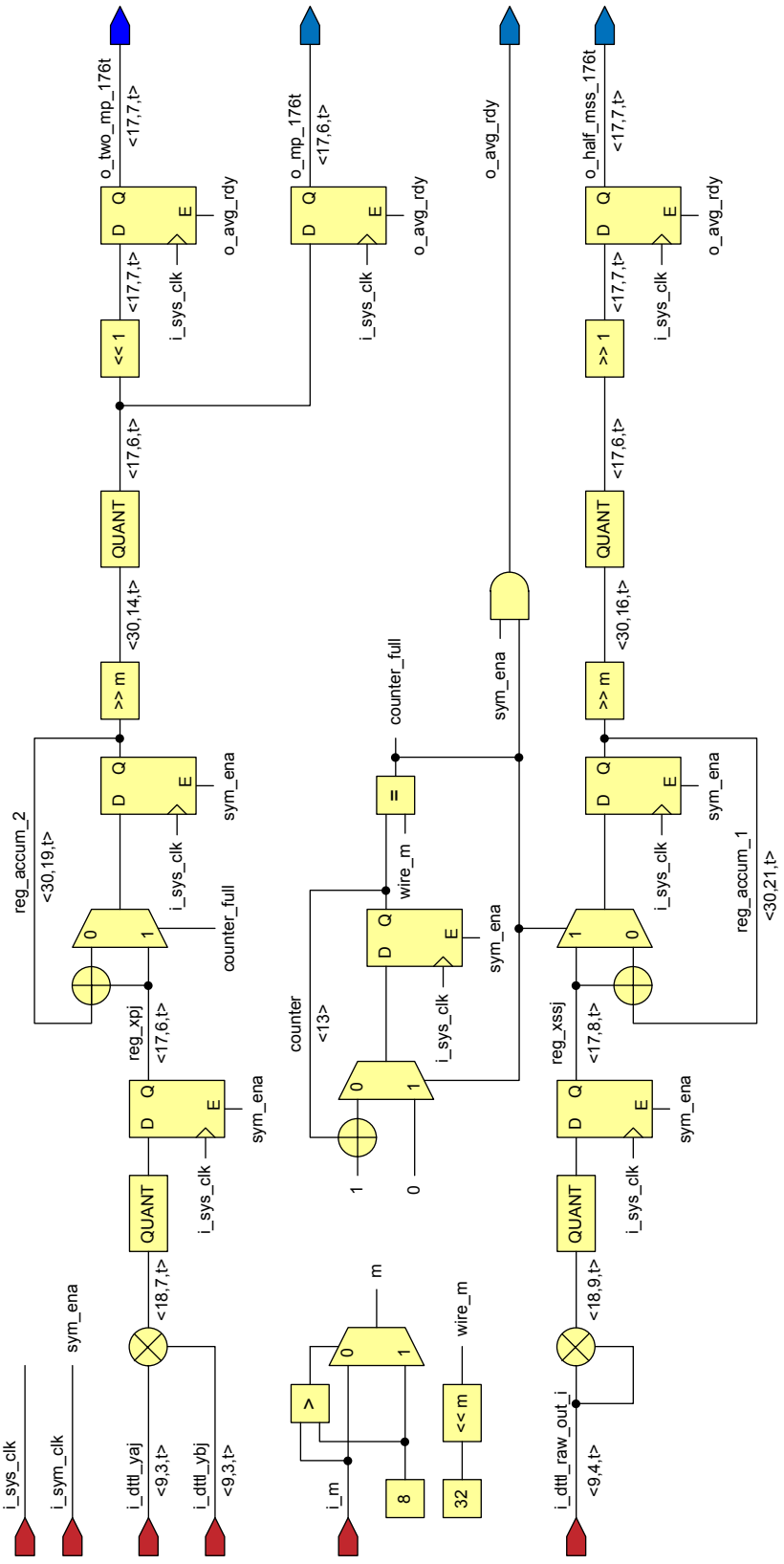


Figure 6. Bit Energy Estimator schematic.

Table 2. Sum-and-dump filter sample size values.

snr_m	*m*	*wire_m*
0	0	32
1	1	64
2	2	128
3	3	256
4	4	512
5	5	1,024
6	6	2,048
7	7	4,096
8..15	8	8,192

The Bit Energy Estimator module produces three signal outputs that are fed into the SNR Estimator core module. The output of the *mss* accumulator is divided by two (shifted right by one), quantized to <17,7,t> and output as *o_half_mss_176t*. Output *o_two_mp_176t* is taken from the *mp* accumulator by multiplying it by two (shifting left by one) and then quantizing to <17,7,t>. Finally, *o_mp_176t* is the *mp* accumulator value quantized to <17,6,t>.

The SNR Estimator core module includes logic to estimate Es/No values based on noise estimates. The noise power is derived from the outputs of the Bit Energy Estimator block by subtracting the scaled signal power component (*i_two_mp_176t*) from the scaled signal plus noise power component (*i_half_mss_176t*). Figure 7 depicts the corresponding implementation schematic of the SNR Estimator Core module.

The signal-to-noise ratio is calculated as follows:

$$Es/No = P_{SIGNAL}/P_{NOISE}$$

To avoid the high logical cost of a divider, a lookup table (LUT)–based approach is used to estimate the Es/No value. From the above equation, we have

$$(Es/No)(P_{NOISE}) - P_{SIGNAL} = 0$$

By stepping through every entry in the Es/No LUT, we find the index that produces the minimum value of *sig_diff_abs_q* to satisfy the following relationship:

$$\min\left|(Es/No)(P_{NOISE}) - P_{SIGNAL}\right|$$

The resulting LUT index is used to find the Es/No estimate, represented as <10,5,u> and calculated using the following formula:

$$Es/No(dB) = \frac{index}{8} + \frac{index}{32}$$

Two independent LUTs are used in this design to determine the SNR estimate. The primary LUT is used for receive symbol rates of 1024 ksps and below. An alternative LUT is automatically selected for 2048 ksps and 4096 ksps receive rate configurations. Both LUTs are hard-coded and stored in the FPGA's distributed read-only memory. Tables 3 and 4 show the current values of the SNR Estimator's primary and alternative LUTs.

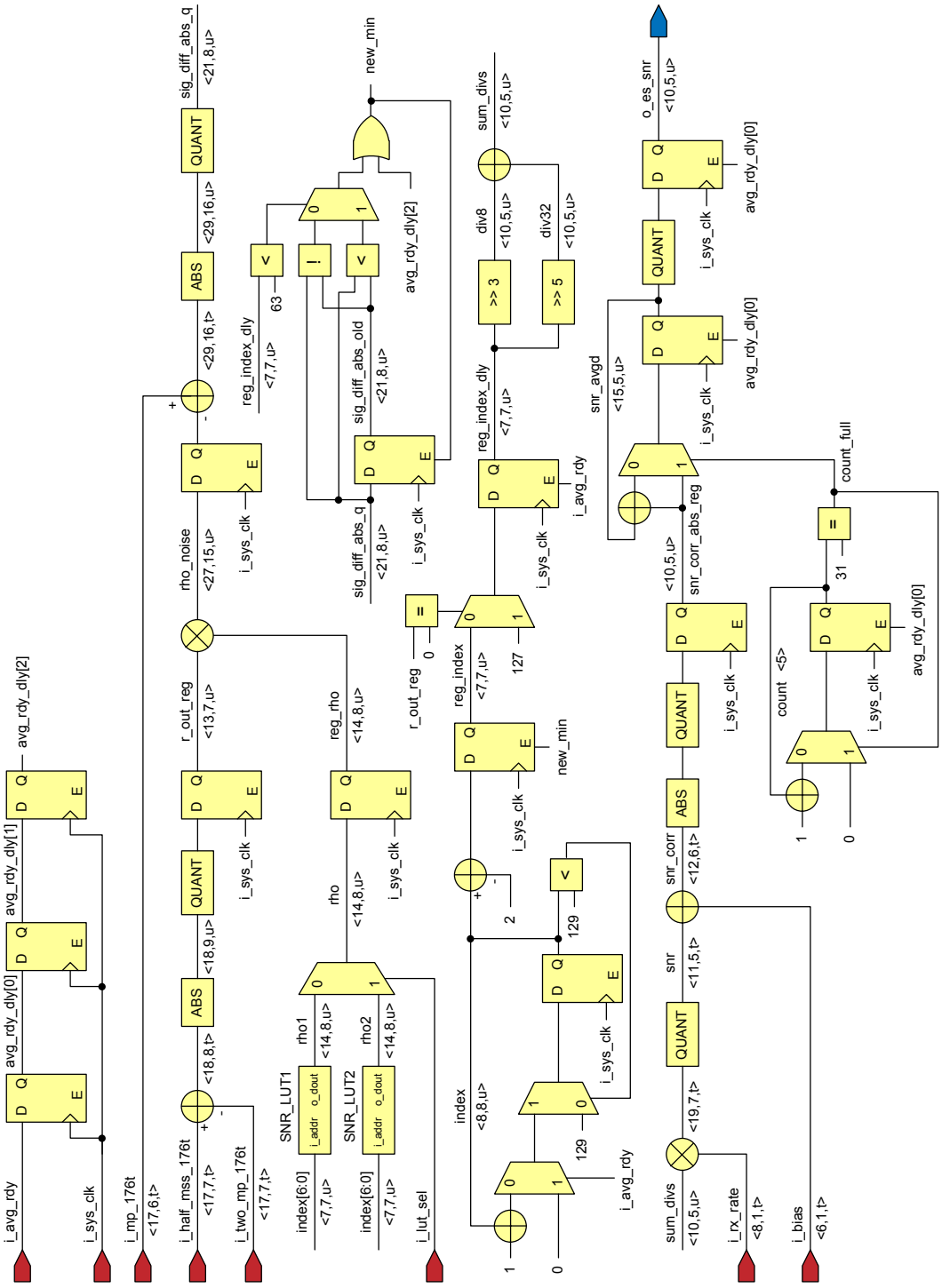


Figure 7. SNR Estimator core schematic.

Table 3. Primary Es/No 20 db LUT (Rx rates from 1 to 1024 ksps).

Index	Ratio	Ratio, BIN	SNR, dB	Index	Ratio	Ratio, BIN	SNR, dB
0	2.453125	00000010011101	0.00000	64	22.328125	00010110010101	10.00000
1	2.500000	00000010100000	0.15625	65	22.625000	00010110101000	10.15625
2	2.562500	00000010100100	0.31250	66	23.093750	00010111000110	10.31250
3	2.625000	00000010101000	0.46875	67	23.906250	00010111111010	10.46875
4	2.703125	00000010101101	0.62500	68	24.734375	00011000101111	10.62500
5	2.781250	00000010110010	0.78125	69	25.578125	00011001100101	10.78125
6	2.859375	00000010110111	0.93750	70	26.468750	00011010011110	10.93750
7	2.828125	00000010110101	1.09375	71	27.375000	00011011011000	11.09375
8	2.968750	00000010111110	1.25000	72	28.906250	00011100111010	11.25000
9	3.140625	00000011001001	1.40625	73	30.296875	00011110010011	11.40625
10	3.296875	00000011010011	1.56250	74	31.328125	00011111010101	11.56250
11	3.375000	00000011011000	1.71875	75	33.062500	00100001000100	11.71875
12	3.578125	00000011100101	1.87500	76	34.625000	00100010101000	11.87500
13	3.718750	00000011101110	2.03125	77	35.796875	00100011110011	12.03125
14	3.765625	00000011110001	2.18750	78	37.000000	00100101000000	12.18750
15	3.828125	00000011110101	2.34375	79	38.250000	00100110010000	12.34375
16	3.953125	00000011111101	2.50000	80	39.531250	00100111100010	12.50000
17	4.171875	00000100001011	2.65625	81	40.843750	00101000110110	12.65625
18	4.265625	00000100010001	2.81250	82	42.218750	00101010001110	12.81250
19	4.343750	00000100010110	2.96875	83	43.609375	00101011100111	12.96875
20	4.593750	00000100100110	3.12500	84	45.062500	00101101000100	13.12500
21	4.781250	00000100110010	3.28125	85	47.468750	00101111011110	13.28125
22	4.796875	00000100110011	3.43750	86	49.656250	00110001101010	13.43750
23	5.078125	00000101000101	3.59375	87	51.265625	00110011010001	13.59375
24	5.234375	00000101001111	3.75000	88	52.937500	00110100111100	13.75000
25	5.406250	00000101011010	3.90625	89	55.718750	00110111101110	13.90625
26	5.500000	00000101100000	4.06250	90	56.031250	00111000000010	14.06250
27	5.625000	00000101101000	4.21875	91	56.343750	00111000010110	14.21875
28	5.906250	00000101111010	4.37500	92	56.656250	00111000101010	14.37500
29	6.062500	00000110000100	4.53125	93	56.984375	00111000111111	14.53125
30	6.296875	00000110010011	4.68750	94	57.296875	00111001010011	14.68750
31	6.531250	00000110100010	4.84375	95	57.609375	00111001100111	14.84375
32	6.828125	00000110110101	5.00000	96	57.921875	00111001111011	15.00000
33	7.046875	00000111000011	5.15625	97	58.234375	00111010001111	15.15625
34	7.234375	00000111001111	5.31250	98	60.093750	00111100000110	15.31250
35	7.593750	00000111100110	5.46875	99	60.859375	00111100110111	15.46875
36	7.796875	00000111110011	5.62500	100	62.015625	00111110000001	15.62500
37	8.093750	00001000000110	5.78125	101	63.984375	00111111111111	15.78125
38	8.375000	00001000011000	5.93750	102	64.781250	01000000110010	15.93750
39	8.828125	00001000110101	6.09375	103	68.578125	01000100100101	16.09375
40	9.078125	00001001000101	6.25000	104	73.796875	01001001110011	16.25000
41	9.328125	00001001010101	6.40625	105	78.453125	01001110011101	16.40625
42	9.750000	00001001110000	6.56250	106	84.906250	01010100111010	16.56250
43	9.890625	00001001111001	6.71875	107	91.171875	01011011001011	16.71875
44	10.234375	00001010001111	6.87500	108	96.734375	01100000101111	16.87500
45	10.765625	00001010110001	7.03125	109	102.578125	01100110100101	17.03125
46	11.203125	00001011001101	7.18750	110	104.968750	01101000111110	17.18750
47	11.781250	00001011110010	7.34375	111	106.812500	01101010110100	17.34375
48	11.906250	00001011111010	7.50000	112	109.953125	01101101111101	17.50000
49	12.234375	00001100001111	7.65625	113	113.171875	01110001001011	17.65625
50	12.828125	00001100110101	7.81250	114	116.468750	01110100011110	17.81250
51	13.187500	00001101001100	7.96875	115	119.843750	01110111110110	17.96875
52	13.984375	00001101111111	8.12500	116	120.109375	01111000000111	18.12500
53	14.859375	00001110110111	8.28125	117	120.375000	01111000011000	18.28125
54	15.406250	00001111011010	8.43750	118	120.640625	01111000101001	18.43750
55	15.734375	00001111101111	8.59375	119	120.890625	01111000111001	18.59375
56	16.265625	00010000010001	8.75000	120	121.156250	01111001001010	18.75000
57	17.187500	00010001001100	8.90625	121	121.421875	01111001011011	18.90625
58	17.734375	00010001101111	9.06250	122	121.687500	01111001101100	19.06250
59	18.171875	00010010001011	9.21875	123	121.953125	01111001111101	19.21875
60	18.812500	00010010110100	9.37500	124	122.218750	01111010001110	19.37500
61	19.468750	00010011011110	9.53125	125	122.468750	01111010011110	19.53125
62	20.156250	00010100001010	9.68750	126	122.734375	01111010101111	19.68750
63	21.296875	00010101010011	9.84375	127	123.000000	01111011000000	19.84375

Table 4. Alternative Es/No 20 db LUT (Rx rates 2048 and 4096 ksps).

Index	Ratio	Ratio, BIN	SNR, dB	Index	Ratio	Ratio, BIN	SNR, dB
0	2.546875	00000010100011	0.00000	64	11.265625	00001011010001	10.00000
1	2.562500	00000010100100	0.15625	65	11.500000	00001011100000	10.15625
2	2.578125	00000010100101	0.31250	66	11.750000	00001011110000	10.31250
3	2.593750	00000010100110	0.46875	67	11.984375	00001011111111	10.46875
4	2.609375	00000010100111	0.62500	68	12.218750	00001100001110	10.62500
5	2.640625	00000010101001	0.78125	69	12.468750	00001100011110	10.78125
6	2.656250	00000010101010	0.93750	70	12.703125	00001100101101	10.93750
7	2.687500	00000010101100	1.09375	71	12.937500	00001100111100	11.09375
8	2.734375	00000010101111	1.25000	72	13.171875	00001101001011	11.25000
9	2.765625	00000010110001	1.40625	73	13.406250	00001101011010	11.40625
10	2.812500	00000010110100	1.56250	74	13.640625	00001101101001	11.56250
11	2.859375	00000010110111	1.71875	75	13.875000	00001101111000	11.71875
12	2.906250	00000010111010	1.87500	76	14.109375	00001110000111	11.87500
13	2.953125	00000010111101	2.03125	77	14.328125	00001110010101	12.03125
14	3.015625	00000011000001	2.18750	78	14.562500	00001110100100	12.18750
15	3.062500	00000011000100	2.34375	79	14.781250	00001110110010	12.34375
16	3.125000	00000011001000	2.50000	80	15.015625	00001111000001	12.50000
17	3.203125	00000011001101	2.65625	81	15.234375	00001111001111	12.65625
18	3.265625	00000011010001	2.81250	82	15.453125	00001111011101	12.81250
19	3.343750	00000011010110	2.96875	83	15.671875	00001111101011	12.96875
20	3.421875	00000011011011	3.12500	84	15.890625	00001111111001	13.12500
21	3.500000	00000011100000	3.28125	85	16.093750	00010000000110	13.28125
22	3.578125	00000011100101	3.43750	86	16.312500	00010000010100	13.43750
23	3.656250	00000011101010	3.59375	87	16.531250	00010000100010	13.59375
24	3.750000	00000011110000	3.75000	88	16.734375	00010000101111	13.75000
25	3.843750	00000011110110	3.90625	89	16.937500	00010000111100	13.90625
26	3.937500	00000011111100	4.06250	90	17.156250	00010001001010	14.06250
27	4.046875	00000100000011	4.21875	91	17.359375	00010001010111	14.21875
28	4.140625	00000100001001	4.37500	92	17.562500	00010001100100	14.37500
29	4.250000	00000100010000	4.53125	93	17.750000	00010001110000	14.53125
30	4.359375	00000100010111	4.68750	94	17.953125	00010001111101	14.68750
31	4.468750	00000100011110	4.84375	95	18.156250	00010010001010	14.84375
32	4.578125	00000100100101	5.00000	96	18.343750	00010010010110	15.00000
33	4.703125	00000100101101	5.15625	97	18.546875	00010010100011	15.15625
34	4.828125	00000100110101	5.31250	98	18.734375	00010010101111	15.31250
35	4.953125	00000100111101	5.46875	99	18.937500	00010010111100	15.46875
36	5.109375	00000101000111	5.62500	100	19.125000	00010011001000	15.62500
37	5.250000	00000101010000	5.78125	101	19.312500	00010011010100	15.78125
38	5.421875	00000101011011	5.93750	102	19.500000	00010011100000	15.93750
39	5.593750	00000101100110	6.09375	103	19.687500	00010011101100	16.09375
40	5.765625	00000101110001	6.25000	104	19.859375	00010011110111	16.25000
41	5.953125	00000101111101	6.40625	105	20.046875	00010100000011	16.40625
42	6.156250	00000110001010	6.56250	106	20.234375	00010100001111	16.56250
43	6.359375	00000110010111	6.71875	107	20.406250	00010100011010	16.71875
44	6.562500	00000110100100	6.87500	108	20.593750	00010100100110	16.87500
45	6.765625	00000110110001	7.03125	109	20.765625	00010100110001	17.03125
46	6.984375	00000110111111	7.18750	110	20.937500	00010100111100	17.18750
47	7.203125	00000111001101	7.34375	111	21.109375	00010101000111	17.34375
48	7.421875	00000111011011	7.50000	112	21.281250	00010101010010	17.50000
49	7.656250	00000111101010	7.65625	113	21.453125	00010101011101	17.65625
50	7.875000	00000111111000	7.81250	114	21.625000	00010101101000	17.81250
51	8.109375	00001000000111	7.96875	115	21.796875	00010101110011	17.96875
52	8.343750	00001000010110	8.12500	116	21.953125	00010101111101	18.12500
53	8.578125	00001000100101	8.28125	117	22.125000	00010110001000	18.28125
54	8.828125	00001000110101	8.43750	118	22.281250	00010110010010	18.43750
55	9.062500	00001001000100	8.59375	119	22.437500	00010110011100	18.59375
56	9.312500	00001001010100	8.75000	120	22.593750	00010110100110	18.75000
57	9.546875	00001001100011	8.90625	121	22.750000	00010110110000	18.90625
58	9.796875	00001001110011	9.06250	122	22.906250	00010110111010	19.06250
59	10.031250	00001010000010	9.21875	123	23.046875	00010111000011	19.21875
60	10.281250	00001010010010	9.37500	124	23.203125	00010111001101	19.37500
61	10.531250	00001010100010	9.53125	125	23.343750	00010111010110	19.53125
62	10.765625	00001010110001	9.68750	126	23.484375	00010111011111	19.68750
63	11.015625	00001011000001	9.84375	127	23.625000	00010111101000	19.84375

The SNR Estimator Core module also includes linear correction logic. Available user programmable scale (*i_rx_rate*, represented as <8,1,t>) and bias (*i_bias*, represented as <6,1,t>) parameters are utilized to further calibrate and tune the final SNR estimate value for different data rates. True linear correction is achieved using the following equation:

$$Es/No\_corr(dB) = (i\_rx\_rate)(Es/No) + i\_bias$$

Linear correction can easily be bypassed by setting *i_rx_rate* = 1 and *i_bias* = 0. Final SNR estimate is calculated using a 32-sample average of *Es/No_corr* to reduce output jitter.

IV. Conclusions

A summary has been presented of the WSME algorithm that is currently implemented in MRO for ADR applications. Sample data from inflight loop-back tests show good agreement with predicts. An example was presented from an inflight MRO test on March 5, 2012.[4] The results are presented in Figure 8, showing the WSME output (the Es_No curve) along with an SNR estimate based on Viterbi reencoding (the Vit_Es/No curve) as developed by Tom Jedrey.[5]

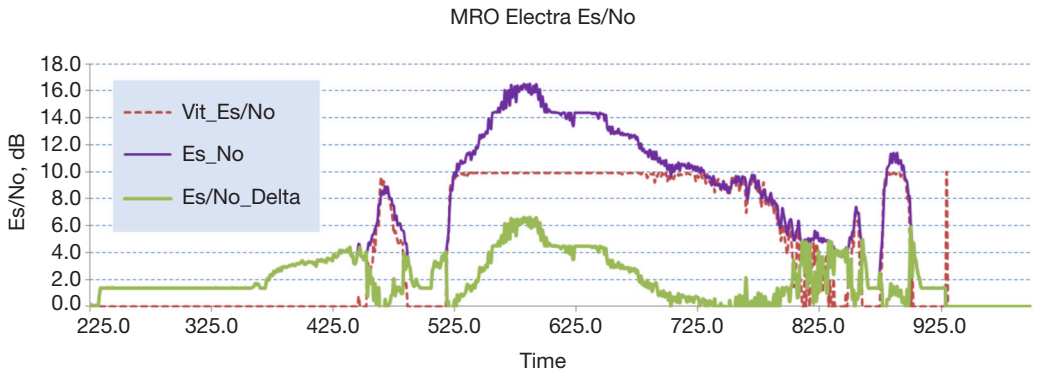


Figure 8. SNR data from an inflight MRO test on March 5, 2012.[6]

The WSME output provides a reliable SNR estimate over the majority of the pass. It is noted that the Viterbi SNR estimator saturates in regions of high SNR where there are zero observed bit errors between the Viterbi reencoded data and the output symbol hard decisions as evidenced by the large residual errors between the two estimates (the Es/No_Delta curve), e.g., between times 525 and 725 in Figure 8. Nevertheless, the Viterbi SNR estimator is desirable for ADR, which is highly dependent on estimates of the bit error rate.

Current improvements to the WSME SNR estimator are in progress and may be uploaded to MRO after the Mars Science Laboratory landing in August 2012.

---

[4] T. Jedrey, "MRO Electra/MER-B Pass Analysis DOY 065 2012," presentation to the Mars Project Office (internal document), Jet Propulsion Laboratory, Pasadena, California, May 16, 2012.

[5] T. Jedrey, personal communication, Section 337, Jet Propulsion Laboratory, Pasadena, California, April 2012.

[6] T. Jedrey, May 16, 2012, op cit.

References

[1] B. Shah and S. Hinedi, "The Split-Symbol Moments SNR Estimator in Narrow-Band Channels," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 26, no. 5, pp. 737–747, September 1990.

[2] M. Simon and S. Dolinar, "Turning Fiction Into Non-fiction for Signal-to-Noise Ratio Estimation — The Time-Multiplexed and Adaptive Split-Symbol Moments Estimator," *The Telecommunications and Mission Operations Progress Report,* vol. 42-162, Jet Propulsion Laboratory, Pasadena, California, pp. 1–9, August 15, 2005. http://tmo.jpl.nasa.gov/progress_report/42-162/162H.pdf

[3] M. Simon and V. Vilnrotter, "Iterative Information-Reduced Carrier Synchronization Using Decision Feedback for Low SNR Applications," *The Telecommunications and Data Acquisition Progress Report*, vol. 42-130, Jet Propulsion Laboratory, Pasadena, California, pp. 1–21, August 15, 1997. http://tmo.jpl.nasa.gov/progress_report/42-130/130A.pdf

[4] M. Simon and S. Dolinar, "Signal-to-Noise Ratio Estimation," Chapter 6, *Autonomous Software-Defined Radio Receivers for Deep Space Applications*, Jon Hamkins and Marvin K. Simon, editors, DESCANSO Monograph Series, Pasadena, California: NASA Jet Propulsion Laboratory, 2006. http://descanso.jpl.nasa.gov/Monograph/series9_chapter.cfm