

# Frequency Offset Estimation With Improved Convergence Time and Energy Consumption

M. Josie Ammer and Jan Rabaey  
 {mjammer, jan}@eecs.berkeley.edu  
 University of California, Berkeley

**Abstract**—An approach to simultaneously improve the energy consumption and convergence time (given the input SNR and required estimation variance) of feed-forward data-aided frequency estimation is presented. Four well-known frequency estimation algorithms are compared using actual ASIC hardware implementations to verify the results. It is demonstrated how a modification to the algorithms can simultaneously achieve lower energy consumption and improved convergence time. For example, for an input SNR of 12dB and required estimation variance of  $2 \times 10^{-5}$ , convergence time is decreased by a factor of 4 while decreasing the energy consumption by a factor of 4.3. Directions on how to apply these algorithms to spread spectrum systems are provided.

## INTRODUCTION

In a typical wireless communication system, imperfect up- and down-conversion caused by nonidealities in the transmitter and receiver local oscillators (LO) result in a carrier offset at the receiver. This offset causes a continuous rotation of the signal constellation, and must be corrected for reliable demodulation of the received signal. In some communication systems, carrier recovery is performed by a phase-locked loop (PLL). However, if the carrier offset is greater than the pull-in range of the PLL, a coarse feed-forward frequency estimation and correction must be performed before the signal enters the PLL [4]. In some systems, the PLL is not used in favor of an all-feed-forward algorithm.

Convergence time of the synchronization subsystem is critical in wireless systems. If implementing a standardized wireless system, such as 802.11b, synchronization time is limited by the standard's preamble or synch word. Improved convergence time for one synchronization parameter may mean being able to increase system performance and reliability by allocating more of the synch word to the estimation of this or other synchronization parameters. In the case of a non-standardized wireless system, improved convergence time allows the use of a shorter synch word, reducing the packet overhead and thereby reducing the system power consumption.

Since most wireless communication devices are battery-powered, low power operation is a primary concern. Low-power research is concentrated in the RF, MAC, Network and Application layers of the wireless device while the synchronization system is often overlooked. However, synchronization systems for several common wireless

standards, such as Bluetooth and 802.11a take up more than 15% of the physical layer die area. While the power consumption numbers are not separately reported, we can assume that the synchronization system power consumption is significant because of its significant area and high clock rates. Indeed, in our own work, we have found the synchronization to have significant power consumption [4] (approx 15% of the system power).

This work examines four feed-forward data-aided frequency offset estimation algorithms and compares the estimation performance and power consumption of each over estimation length and input SNR. A modification of these algorithms is presented that simultaneously achieves lower power and faster convergence time.

## FREQUENCY ESTIMATION ALGORITHMS

As described in [1] and [2], in the absence of ISI and with moderate frequency offset (less than ~15% of the symbol rate), the sampled output of the matched filter (at one sample per symbol), assuming perfect symbol synchronization, is given by

$$r_n = a_n e^{j(\phi + n\Delta\omega T)} + w_n, \quad (1)$$

where  $a_n$  is the  $n$ th (complex) data symbol,  $\phi$  is the carrier phase,  $\Delta\omega$  is the carrier frequency offset,  $T$  is the symbol duration, and  $w_n$  is a complex Gaussian white noise process with independent, zero-mean real and imaginary parts each with variance  $\sigma^2 = N_0 / (2E_s)$  where  $E_s$  is the symbol energy and  $N_0$ , the one-sided spectral density of the noise. Also, as in [2], we use the convenient notation of normalized frequency offset, defined as  $\Omega = \Delta\omega T$ .

We examine the case of data-aided estimation where the known data symbols are removed before frequency estimation. This reduces to the problem of frequency estimation with an unmodulated carrier [2]. This is the most common use of frequency estimation in systems where a synchronization header is used, such as 802.11b.

There are two well-known algorithms for frequency estimation operating with timing information derived from the maximum likelihood equations. The difference between the two depends on whether or not the angle of  $r_n$  is taken before deriving the ML algorithm. If the angle is taken before, the result is the Kay estimator[2],

This work is funded by DARPA, GSRC, and the BWRC member companies.

$$\hat{\Omega} = \sum_{n=1}^{L-1} b_n \arg\{r_n r_{n-1}^*\}, \quad (2)$$

if the angle is taken after, the result is the Meyr estimator [1],

$$\hat{\Omega} = \arg\left\{ \sum_{n=1}^{L-1} b_n (r_n r_{n-1}^*) \right\}, \quad (3)$$

where

$$b_n = \frac{6n(L-n)}{L(L^2-1)}. \quad (4)$$

Neither algorithm requires phase unwrapping, and both are limited to frequency offsets that obey

$$|\Omega| < \pi \quad (5)$$

It should be noted, that while different weighting functions,  $b_n$ , can be used, the one given in (4) is optimal. A simplification, suggested in [1], that is often used in practice, substitutes an integrate-and-dump filter ( $b_n=1/L$ ) that computes an unweighted average, for the filter function in (4) that computes a weighted average. We apply this simplification to both the Meyr and Kay estimators to expand the number of estimators considered here to be four. The variance of the weighted  $\hat{\Omega}_{Mw}$  and unweighted  $\hat{\Omega}_{Mu}$  versions of the Meyr estimator are given in [1] as,

$$Var[\hat{\Omega}_{Mw}] = \frac{12}{L(L^2-1)} \frac{1}{2E_s/N_0} + \frac{12}{5} \frac{1}{L} \frac{L^2+1}{L^2-1} \frac{1}{(2E_s/N_0)^2} \quad (6)$$

and

$$Var[\hat{\Omega}_{Mu}] = \frac{1}{L^2} \frac{2}{2E_s/N_0} + \frac{1}{L} \frac{2}{(2E_s/N_0)^2} \quad (7)$$

The simulated performance of the four estimators (Meyr weighted and unweighted and Kay weighted and unweighted) is shown in Figure 1.

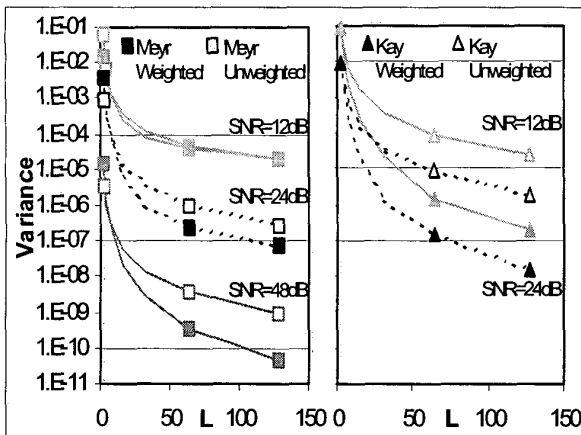


Figure 1: Meyr and Kay Weighted and Unweighted Performance.

The simulations match the performance predicted by Meyr very closely. As expected, at high SNR, the performance of the two weighted estimators approach the Modified Cramer-Rao bound given in [2] as

$$MCRB(\Omega) = \frac{6}{L(L^2-1)(E_s/N_0)}. \quad (8)$$

While these algorithms have been derived for the flat fading channel, in practice, they also work for the frequency selective fading channel.

The improved convergence time is achieved by exploiting a little-known modification to these algorithms described in [1]. In the estimator equations, the product,  $(r_n r_{n-1}^*)$  is replaced with  $(r_n r_{n-D}^*)$ . While this is not a new result, it is often overlooked, for instance in [3]. The variance of the estimator is improved roughly as  $D^2$ . For instance, the performance of the unweighted Meyr estimator is given in [1] by

$$Var[\hat{\Omega}_{Mu}] = \frac{1}{D^2} \left( \frac{D}{L^2} \frac{2}{2E_s/N_0} + \frac{1}{L} \frac{2}{(2E_s/N_0)^2} \right). \quad (9)$$

The algorithms are now limited to frequency offsets that obey

$$|\Omega D| < \pi. \quad (10)$$

In practice, many systems can tolerate  $D>1$ . If following the rule of thumb that frequency offset should be less than 15% of the symbol rate, then  $D \leq 3$  is possible. In 802.11b, with a 25ppm carrier offset from a 2.4GHz reference, the maximum frequency offset is +/- 120KHz, allowing  $D=4$  to be used. Figure 2 shows that even  $D=2$  yields a huge improvement (decrease) in  $L$  for a given variance.

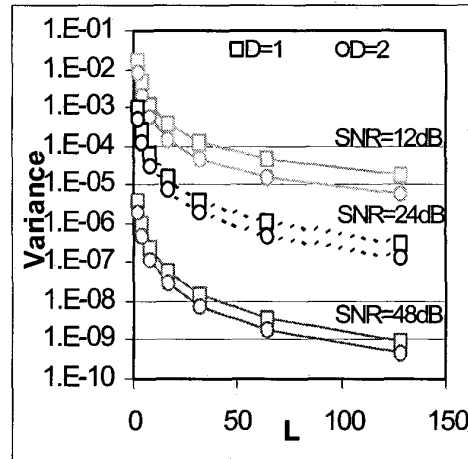


Figure 2: Meyr Weighted D=1,2 Performance.

The block diagrams for the Kay and Meyr weighted estimators are shown in Figure 3 and Figure 4. To implement the unweighted estimators, one or two scalar multipliers are removed from the Kay or Meyr estimators respectively.

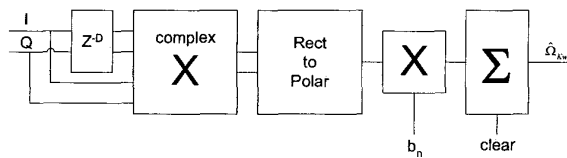


Figure 3: Block Diagram of the Weighted Kay Estimator.

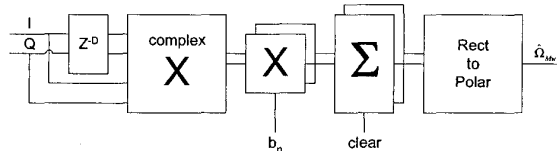


Figure 4: Block Diagram of the Weighted Meyr Estimator.

The goal is to choose the lowest power frequency estimation algorithm to achieve a given variance. The Mery and Kay algorithms seem to have similar hardware complexity upon first glance because they consist of the same operations but in a different order. However, the ordering of operations in hardware can have a large impact on the power consumption. The simplification suggested in [1] of  $b_n=1/L$ , reduces the hardware, but incurs a performance penalty. It is unclear at what point, if any, this hardware simplification will actually decrease energy consumption. Increasing  $D$  requires marginally more hardware but gives a significant improvement in performance. It is expected that this will be a good tradeoff because the hardware cost is so small. However, it is unknown by only looking at the estimator equation and the variance performance which algorithm to choose for a low power system.

#### POWER ESTIMATION METHODOLOGY

Each frequency estimation algorithm was coded as a parameterized module in a high-level hardware description language in Synopsys Module Compiler. Each module was synthesized in Module Compiler for a range of parameters, such as input SNR and estimation length. Each synthesized VHDL netlist from Module Compiler is incrementally compiled in Synopsys Design Compiler to insert a clock tree and to add buffer delays to fix hold time violations. This step is very important to get accurate power consumption because the clock tree and delay buffers can account for 30-50% of the block power depending on the ratio of registers to combinational logic.

The block is then simulated at the gate level in ModelSim using realistic input vectors to verify functionality and to determine the switching activity on each node. Simulation using realistic vectors is important because it accurately characterizes the correlation in the data stream that often exists in communication systems which results in reduced power consumption versus using statistical switching activity.

Synopsys Power Compiler is used to estimate the power consumption of the block using the back annotated switching activity and statistical wire load models. Our own experiments on several frequency estimation blocks with

varying input and output bit-widths have shown this gate-level estimation method to be accurate to within 15% of the power consumption estimated by extracting parasitics from a post-place-and-route block and simulating using a switch-level simulator like PowerMill or NanoSim. Our own experience and reports from our foundry have shown that switch-level simulations give 10-15% agreement with power consumption of actual chips. Gate-level power estimation is used because it is over 50 times faster than switch-level simulation (not including the time it takes to place-and-route the block as required to extract accurate parasitics). The total time to characterize all 21 different chosen instantiations (3 different SNR's, 7 different  $L$ 's) of each algorithm is under 3 hours using the gate-level method.

Energy, rather than power, is used as the cost metric for each block. This is because the frequency estimation takes a different number of cycles depending on the input SNR, required estimation variance, and which algorithm is selected. Aggressive low power designs will gate the clock and power rails to the frequency estimation block when not in use. Therefore, the way to fairly compare different blocks is the energy consumption, which is the power consumed when the block is on times the amount of time the block needs to be on to achieve the desired variance.

The energy consumption reported here is for a 0.13um CMOS process. While the actual energy consumption will change for different processes, the comparison of one algorithm vs. another is valid for most contemporary processes. Obviously, the ratio of leakage power to switching power and power consumed in the wires will vary between process and this will alter the crossover points of the curves, however the general results will remain true.

#### ALGORITHM COMPARISON AND RESULTS

For each implementation, it is assumed that the number of bits at the input to the estimator is scaled depending on the input SNR. This is a reasonable assumption because most systems employ good AGC and would not pay the cost penalty of implementing an ADC that converted more bits than necessary nor a frequency estimator that achieved better precision than was needed. The bit widths are scaled up in subsequent blocks to accommodate the growing precision. The accumulators are pre-scaled to accommodate the summation of  $L$  samples, and the precision of the weighting taps,  $b_n$ , is increased with  $L$ . The  $b_n$  coefficients are hard-wired before synthesis for the lowest power operation. The rectangular-to-polar conversion is performed by a CORDIC [6] and the number of CORDIC stages is increased depending on the required precision. These adjustments ensure that the hardware is not significantly limiting the expected variance.

The resulting energy consumption of each estimator is shown versus variance for a range of input SNR and  $L$ . Since lower variance and lower energy consumption are desired, data points to the bottom and left are better. While this is the right presentation of the data for optimizing the energy of the frequency estimator in isolation,  $L$  must be considered if a system-wide reduction in power consumption is to be

achieved because the RF and analog front-end are on for different amounts of time. For instance, in the case where the front-end power dominates that of the frequency estimation, choosing an algorithm with smaller L may optimize system energy even if it has higher frequency estimation energy. Since the absolute energy consumption and the L for each data point is given in the graphs, the designer can make the appropriate trade-off. Obviously, cases where both the power consumption and convergence time (L) are decreased for the same variance are hands-down winners.

Figure 5 compares the energy consumption vs. variance of the weighted and unweighted versions of the Meyr algorithm. At low SNR and at high required variance, it is more energy efficient to use the non-weighted version. Here, there is a small difference in variance between the two algorithms, so the hardware simplification of unweighted combining pays off. However, at high SNR or low variance, it is more energy efficient to use the weighting function. Here the energy

savings from the unweighted averaging are outweighed by the longer correlation times required to overcome the degradation in variance. For instance, at 24db SNR, and a required variance of  $3 \times 10^{-7}$ , the unweighted estimator converges in 128 samples, whereas the weighted estimator takes only 64 samples and as a result, consumes marginally less energy.

Figure 6 compares the weighted and unweighted versions of the Kay algorithm. For the Kay estimator, it is almost always better to use the weighted version of the algorithm. This is to be expected because the variance of the unweighted version of the Kay algorithm severely under performs the weighted version. In this case, the hardware simplification of an unweighted average is not worth the degradation in variance. For instance, at 24db SNR, and a required variance of  $2 \times 10^{-6}$ , the unweighted estimator converges in 128 samples, whereas the weighted estimator takes only 32 samples and as a result, consumes 1/3 as much energy.

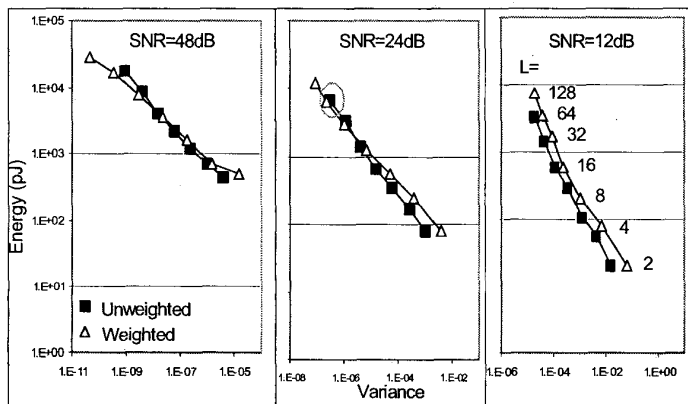


Figure 5: Meyr Weighted vs. Unweighted Comparison.

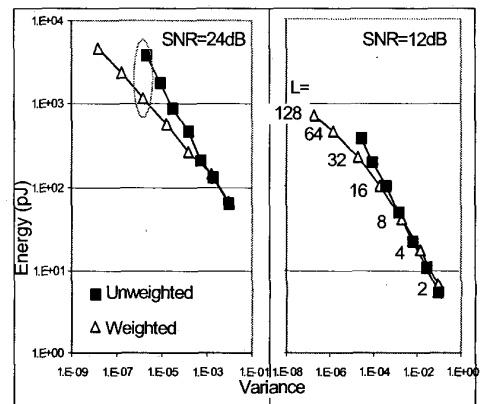


Figure 6: Kay Weighted vs. Unweighted Comparison.

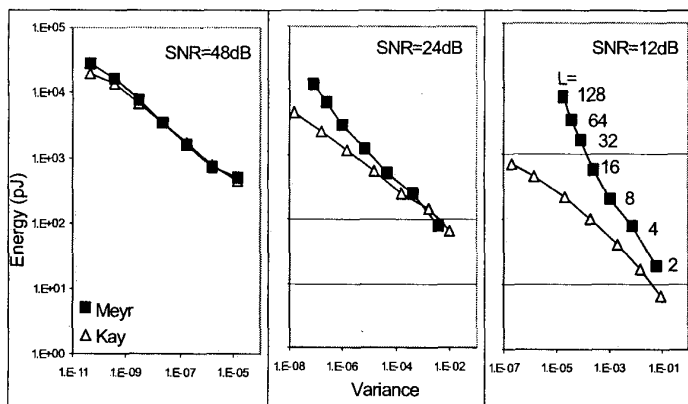


Figure 7: Meyr vs. Kay Weighted Comparison.

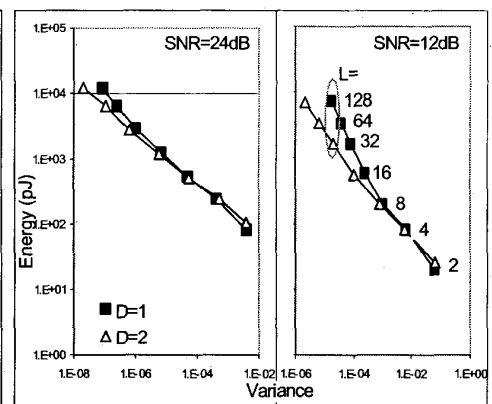


Figure 8: Meyr Weighted D=1 vs. D=2 Comparison.

Figure 7 compares the weighted versions of the Meyr and Kay algorithms. The weighted Kay algorithm is almost always better than or equal to the weighted Meyr algorithm. At low SNR, the marked advantage of the Kay algorithm is due to the combination of achieving better variance and requiring considerably less hardware to implement than the Meyr algorithm. At high SNR where the algorithms have similar variance performance and similar hardware requirements, the minor differences mostly result from the correlation of the data as it flows through the hardware. At high variance there is little difference between the two, while at low variance the Kay algorithm wins out.

Figure 8 compares the weighted version of the Meyr algorithm for  $D=1,2$ . Increasing  $D$  is almost always the right choice, especially for low variance. The power penalty is very small (only one extra register) and the convergence time can be markedly better. For example, for an input SNR of 12dB and required estimation variance of  $2 \cdot 10^{-5}$ , the convergence time is decreased by a factor of 4 while simultaneously decreasing the energy consumption by a factor of 4.3.

#### APPLICATION TO DSSS SYSTEMS

For DSSS, it is sometimes suggested in the literature to apply these frequency estimation algorithms to chips rather than symbols to maximize  $D$ ; this is not usually advantageous. Whereas the normalized frequency offset,  $\Omega = \Delta\omega T$ , is used in this paper, when comparing the variance between chips and symbols, the non-normalized variance,  $\text{Var}[\Delta\omega] = \text{Var}[\Omega]/T^2$ , must be used. When operating on chips rather than symbols,  $D$  is increased by a factor of  $N$  (where  $N$  is the spreading code length) but both  $T$  and SNR are decreased by a factor of  $N$ . Assuming the same header length and minimal power loss in the code correlator due to frequency offset, the performance when operating on chips is significantly worse than when operating on symbols.

The only time one would operate on chips is with a large frequency offset. If, the constraint in (5) is not satisfied for symbol operation, one could operate on chips and still be able to use the algorithms described above without having to resort to more complex FFT-based algorithms. The penalty for performing frequency estimation on chips is reduced when there is severe SNR degradation in the code correlator due to a large frequency offset. For 802.11b-like symbols (11-bit barker sequence spreading, root-raised cosine transmit and receive filters w/ 50% excess bandwidth), the power loss for correlation prior to frequency-offset correction is approximately 3db with a 600Khz offset. Depending on the

required variance, it may be advantageous to do a *coarse* frequency estimation by operating on chips, then perform a coarse frequency correction, correlate to symbols, and perform the *fine* frequency estimation by operating on symbols.

In all cases, because of the SNR degradation due to correlation in the presence of frequency offset, even if frequency-offset *estimation* is performed on symbols, the frequency-offset *correction* should be applied to chips.

#### CONCLUSIONS

Four feed-forward frequency estimators were characterized for energy consumption and variance for a given input SNR and correlation length. It was found that the weighted Kay estimator is a safe bet for all regions of operation, especially for high SNR and low required variance. The unweighted Meyr estimator may be used for low SNR and high required variance. Exploiting  $D$  is the most powerful way to simultaneously decrease convergence time and energy consumption especially for low required variance. It is surprising to find that certain hardware simplifications, such as using  $D=1$  and unweighted averaging does not usually result in lower energy consumption. The degradation in variance due to these simplifications requires longer convergence times and more energy consumption.

#### ACKNOWLEDGMENT

The authors would like to thank Mike Sheets for his valuable help with the power estimation methodology.

#### REFERENCES

- [1] H. Meyr, M. Moeneclaey, and S. A. Fechtel, *Digital Communication Receivers: Synchronization, Channel Estimation and Signal Processing*, Wiley Press, 1998.
- [2] G. Tavares, L. Tavares, and M. Piedade, "Improved Cramer-Rao Lower Bounds for Phase and Frequency Estimation With M-PSK Signals", *IEEE Transactions on Communications*, Vol. 49, No. 12, December 2001.
- [3] K. Barman and V Reddy, "Maximum Likelihood Clock and Carrier Recovery in a Direct Sequence Spread Spectrum Communication System", *Proceedings of the International Conference on Personal Wireless Communication*, 2002.
- [4] M. J. Ammer, M. Sheets, T. Karalar, M. Kuulusa, and J. Rabaey, "A Low-Energy Chip-Set for Wireless Intercom", *Proceedings of the Design Automation Conference (DAC)*, 2003.
- [5] O. Besson and P. Stoica, "On Frequency Offset Estimation for Flat-Fading Channels", *IEEE Communication Letters*, Vol. 5, Issue 10, October 2001.
- [6] K. Turkowski, "Fixed-Point Trigonometry with CORDIC Iterations". Apple Computer White Paper, January 17, 1990.