RadiSim – A Fast Digital RF Behavioral Simulator including Bit Error Rate Assessment for System Exploration, Validation, and Tuning

Joseph P. Skudlarek
Cypress Semiconductor Corporation
9125 SW Gemini Drive Suite 200
Beaverton, Oregon 97008
Jskud@cypress.com

Abstract

RadiSim is a fast, functional, and accurate RF (radio frequency) system simulator which estimates system performance, including bit error rate (BER), in minutes instead of hours or days. Since RadiSim simulates the entire transmit-through-receive path quickly, we were able to search a large design space while creating our commercial radio chip. We describe the background and motivation; chip system model; fundamental design decisions; a novel way to estimate minimum E_b/N_0 given maximum BER; optimal aggregation of BER estimates; our validation strategy and results; and some essential chip improvements enabled by RadiSim.

1. Background and Motivation

Bluetooth™ is a wireless digital communication standard transmitting binary data at 1 Mb/s employing Gaussian Frequency Shift Keying (GFSK), with a nominal deviation of 160 KHz on 79 carriers spaced 1 MHz apart, starting at 2.402 GHz. [1]

Designing a complete RF system requires numerous tradeoffs, including filter widths and circuit complexity. To explore the design space and validate the architecture for the receive path of our Bluetooth chip, we were using analog and mixed-signal simulation technologies. However, they had two drawbacks; they took too long to run, and they could not determine the BER (the fraction of bits received in error) directly.

To address these issues, we decided to create a fast simulator based on digital signal processing techniques, to embed the bit error rate calculations within it, and to construct the system simulation model from a small set of modular components. The result is RadiSim, a fast system level simulator for digital radio.

The primary design goals for RadiSim were *accuracy*, *speed*, and *flexibility*. Most of the simulator development effort was spent not on implementation, but on validation. We needed to *know* that the simulator predictions were accurate enough to support fundamental design decisions, decisions which included using simpler circuitry and removing unneeded functionality.

Given the size of the design space we needed to explore, we had to perform hundreds of experiments, most involving hundreds of thousands of transmitted bits. On a 500 MHz processor, RadiSim can simulate over 10,000 samples a second, end-to-end; when sampling at 37× the bit rate, the throughput is 280 bits per second, which is hundreds of times faster than conventional analog simulation, and more than 30× faster than a simplified subset of the system (no limiters or filters) simulated in SPICE with behavioral models at baseband. Nonetheless, we consumed over 500 hours of CPU time running RadiSim in the first 6 months alone.

The chip architecture, while stable, was not final. Aspects such as filter widths, frequency offsets, and subsystem implementations were changing. In short, the design was still evolving; and RadiSim had to be flexible enough to handle the revisions.

2. Chip System Model

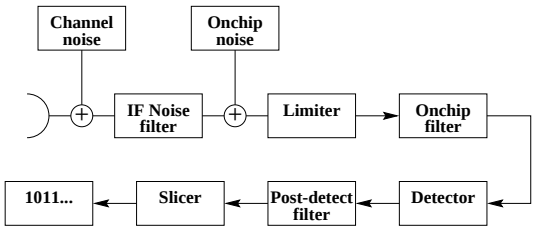

Figure 1: System model of chip receive path.

The chip's transmit path is implemented with mixed signal techniques, while the receive path is pure analog. RadiSim uses DSP techniques to model both paths. Because the goal of RadiSim was to explore the receive path, we use a simple behavioral model of the modulator circuit.

The (random) bits to be transmitted start out as square pulses, either +1 or -1, each of duration T_b. The pulse train is Gaussian filtered to reduce the bandwidth, producing $m(t)$, which is scaled by the constant $2\pi f_{deviation}$ and integrated producing $\phi(t)$. $\phi(t)$ is used to modulate a carrier, producing a real-valued signal which is transmitted through the channel. The channel combines additive white Gaussian noise (AWGN), and possibly an interfering signal, with the desired signal.

The system model of the chip's receive path is shown in Fig. 1. The physical signal is received with a conventional low-IF (low intermediate frequency) image reject mixer [2], using high side injection to produce two related channels, the in-phase I and quadrature Q channels, on a 2 MHz intermediate frequency. RadiSim models the signal coming from the image reject mixer degraded by noise and interference with a complex exponential at baseband, $e^{j\phi(t)}$, to which it adds AWGN and optional interference.

The physical I and Q channels are filtered with a single 1.2 MHz complex bandpass filter; each channel is hard limited to remove amplitude modulation and reduce noise; the pair is complex bandpass filtered again to remove the harmonics introduced by hard limiting and to reduce onchip noise, then fed to a complex detector which reproduces a noisy version of $m(t)$ which is lowpass filtered and fed to a slicer. The receive path circuitry introduces additional noise, which RadiSim models with input-referred AWGN introduced just before the hard limiters.

3. RadiSim Design Decisions

The major RadiSim design decisions are described next. More general formulations of some of these ideas and efficient alternatives can be found in [3], [4], and [5].

DSP Techniques, FFT-based FIR Filters RadiSim is built around sampled signal values processed primarily with finite impulse response (FIR) filters efficiently implemented via the Fast Fourier Transform (FFT).

Given sufficient sampling rates and filter taps, DSP techniques [6] using FIR filters can approximate bandlimited linear time-invariant continuous-time systems arbitrarily well [7, Section 3.4, 7.0], [8, Section 10.1]. Most RF transmit and receive circuits are bandlimited linear time-invariant; and with care, the non-linear portions can also be modeled with DSP techniques.

Since the chip architecture has no feedback between blocks, and since we had ready access to machines with a minimum of 256 MB of main memory, each block can efficiently process its entire input sequence in isolation.

Complex Representation at Baseband Consider a carrier, $\cos(\omega t)$, which is amplitude, frequency, and/or phase modulated to create a signal $v(t)$. The signal can be expressed as $v(t) = I(t) * \cos(\omega t) - Q(t) * \sin(\omega t)$. Define the complex envelope $g(t) = I(t) + jQ(t)$; if the modulations are not too fast relative to the carrier, then the information content of the signal is contained in $g(t)$ [9, page 56], [10, page 151]. "In digital computer simulations of bandpass signals, the sampling rate used in the simulation can be minimized by working with the complex envelope, $g(t)$, instead of with the bandpass signal, $v(t)$, because $g(t)$ is the baseband equivalent of the bandpass signal." [11]

Direct BER Determination given E_b/N_0 The BER for a given signal-to-noise ratio (SNR) is a key figure of merit for digital communications. SNR is often expressed in the modulation independent manner of E_b (energy per bit) divided by N_0 (noise power spectral density), or E_b/N_0 [12, p. 118]. Many of the Bluetooth specs are expressed in BER. In order to determine BER, one must compare transmitted bits with received bits, something which is not that easy to do in, say, SPICE. Also, it is difficult to introduce AWGN in a SPICE simulation. RadiSim addresses both of these issues by directly computing BER, given E_b/N_0.

Configurable, including Oversampling Rate Numerous aspects of the models and simulation run are configurable. Parameters include: number of data bits to simulate, oversampling rate (number of samples per the 1μs bit time), frequency deviation, IF carrier frequency, filter widths and offsets, noise and interference levels and offsets, models to use, amplitude limiter settings, and slicer settings.

Initially, oversampling at 16× was expected to be sufficient. The most direct way to validate that claim was to vary the oversampling rate, and demonstrate that 16× was sufficient. We discovered that oversampling at 32× (when simulating at baseband) gave better, more stable, results. We also needed to prevent the hard limiter's odd harmonics from being aliased into the passband. Our default oversampling rate at baseband is 37×, and we often oversample at 60× or higher when simulating with a non-zero IF or with adjacent channel interference.

Sweep Design Space, Repeat Runs, Log Results Consider the incoming noise limiting filter in the receiver. If it is too narrow, then much of the intended signal will be filtered out. If it is too wide, then too much noise will be admitted. We sweep the design space to determine (or confirm) the best filter width. In addition to filter widths and offsets, RadiSim can also automatically sweep oversampling rate, slicer location, frequency deviation, IF carrier frequency, and noise levels.

For each run, RadiSim logs the simulator version and the settings of all parameters. We can archive every version of the simulator, and the output of every run. This provides an audit trail from which we can reproduce any experiment, and re-examine the details used to make crucial decisions.

Abstract and Derived Models We needed to explore the design space to help guide the circuit design, so we had to use abstract models initially. We used a collection of ideal and less than ideal models. For example, we implemented ideal two and three pole Bessel filters. But to model expected behavior, we implemented a gain-limited differentiator with two pole roll off, with one pole at 10 MHz, and the other pole 4 MHz below f_{max}.

Another reason to provide ideal models is to aid validation. We compared RadiSim's results with different theories, some of which use ideal circuit elements; cf. Section 6.

To better estimate the implemented circuit's behavior, RadiSim also incorporates models derived from SPICE circuit simulations; these models incorporate artifacts of implementation, including amplitude ripple and tilt, and non-constant group delay.

"The filters and differentiator are all analog circuits. In order to represent these circuits accurately with RadiSim, a SPICE AC (small signal) simulation is performed to generate a point-by-point complex transfer function in the frequency domain. The resulting equally spaced real and imaginary pairs cover the relevant frequency response up to +/− 10 MHz. Since the filters are complex, they must be simulated in two steps to acquire both the positive and negative frequency responses. This is done by first sweeping with a −90 degree phase relationship between the I and Q channels, followed by the same frequency sweep with a +90 degree phase offset. The resulting two sets of frequency response data are combined in the proper order, interpolated based on the sample rate of the RadiSim simulation [and passed through an inverse discrete Fourier transform] to create an FIR filter." [Mark Gehring, personal communication, September 2001].

Visualization and Monitoring In order to diagnose problems and understand how the system is behaving, RadiSim provides a number of optional outputs. One of the most important is a standard *eye diagram* [13] of the input into the slicer, as shown in Fig. 2.

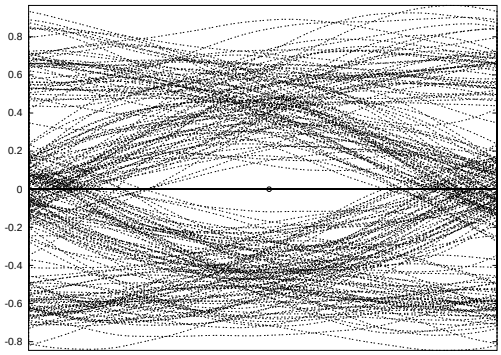

Figure 2: Simulated Eye Diagram at BER=1e-3.

Additionally, the Gaussian filtered binary data signal (the input to the modulator), the output of the modulator, the input to the demodulator, and the output of the demodulator can all be displayed, either as separate I and Q channels over time, or as an eye diagram. In all cases, the original data is displayed with the intermediate data to provide a reference. The average power and the power spectral density for the signal at most intermediate stages can also be reported. Lastly, results from multiple simulation runs are manually extracted and used to create plots.

Most of the RadiSim building blocks have unit tests and optional debugging modes. For example, filter construction can optionally display the filter's frequency response, its group delay, and its impulse response.

Development and Run Environments GNU/Linux[1] [14], [15] is a free, full-featured, readily available development environment which works well with few restrictions. Octave [16] is a free, robust, feature-rich MATLAB[2] clone, which runs under many operating systems, including Linux, Solaris, and Windows.

RadiSim (including test scaffolding and proofs of concept) is composed of 20,000 lines of code in dozens of M-files, and runs under Octave. Our primary development environment is GNU/Linux running on a laptop. Most of our experiments are run on our laptop, or on dual processor servers running Linux; we also run experiments under Solaris and Windows.

4. Estimating E_b/N_0, given BER

Simulations usually predict bit error rates based on E_b/N_0. Fig. 3 shows the predicted bit error rate, as well as the sample standard deviation associated with the predictions, based on simulating 9 trials (of 20K bits) at each value of E_b/N_0. Note that the graph uses log-log axes, so the error bars are distorted.

But sometimes we need to invert this relationship. Recall that Bluetooth performance requirements specify the required BER and the minimum signal-to-noise ratio needed to achieve it. In order to determine the margin in the system, one needs to determine the minimum E_b/N_0 which delivers that BER.

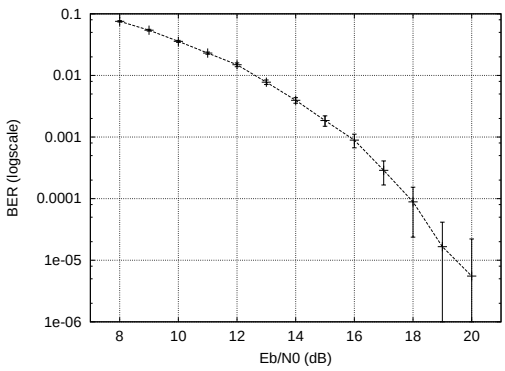

Figure 3: BER vs. E_b/N_0 in dB for our Bluetooth receiver, based on simulating 9 trials of 20K bits at each E_b/N_0 value; error bars are one sample standard deviation around the mean.

Simulating the bit error rate is similar to determining the fairness of a coin based on repeated flips of the coin. A fair coin will not always produce exactly half as many heads as tails; there is inherent variance in the outcome of a set of trials—it's a noisy process.

One way to determine the E_b/N_0 value for a given BER value is to sweep out the BER versus E_b/N_0 curve as was done in Fig. 3, and calculate where the curve crosses the desired BER line. However, sweeping out the BER versus E_b/N_0 curve can consume a lot of simulation cycles. For example, RadiSim took about 2 hours 15 minutes on a 500 MHz processor to create Fig. 3, from which we can estimate E_b/N_0 corresponding to BER=1e-3 by linear extrapolation with about 0.6% standard error.

Rather than sweeping out the entire curve, RadiSim contains a novel iterative solver with improved speed and comparable accuracy that handles the variance inherent in Monte Carlo simulation. It took about 6 minutes to determine the E_b/N_0 value which produced the target BER of 1e-3, with about 1% standard error. That is a simulation speedup of over 20X, with a minor decrease in accuracy. The speed improvement is a factor of 10 when the solver is compared with a simple sweep (using 80K bit samples at each E_b/N_0 value) which also produces an E_b/N_0 estimate with 1% standard error at BER=1e-3.

The RadiSim solver iterates through a series of trials, performing simulation at each trial; 5 trials are sufficient in most cases. The multiple simulations reduce the number of bits required in each simulation, and the directed simulations reduce the number of simulations required.

The curve of log(BER) versus E_b/N_0 in dB is fairly linear if E_b/N_0 does not vary too widely; see Fig. 4. The RadiSim solver uses every trial outcome to update a linear least squares fit of log(BER) versus E_b/N_0. To establish a solid pair of points for the initial linear fit, it uses a user-supplied guess for E_b/N_0 in dB, call it ξ_1, and computes $\xi_0 = \xi_1 - 2$; the RadiSim solver then performs trials at ξ_0 and ξ_1. The result of the simulation at ξ_0 has a larger BER with less variance, and helps to anchor the fitting process. After trial

[1]Linux is a registered trademark of Linus Torvalds.

[2]MATLAB is a registered trademark of The Mathworks, Inc.

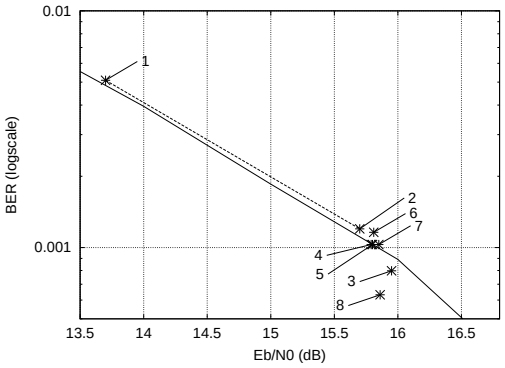

Figure 4: BER versus E_b/N_0 curve, overlayed with initial linear fit; directed trials are indicated by isolated points, labeled by order of prediction.

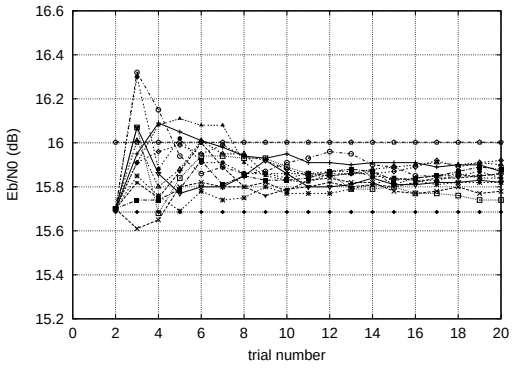

Figure 5: Estimated E_b/N_0 in dB at BER=1e-3 versus trial number; first trial (not shown) at E_b/N_0=13.70 dB. Bounds are 1% above and below final trial's mean value of 15.84 dB.

n the solver sets ξ_n to the updated forecast of the E_b/N_0 solution, allowing ξ_n to change by at most 4 dB for each successive trial. Note how this procedure concentrates the trials in the region which is predicted to yield the desired bit error rate.

The application of this procedure is illustrated in Fig. 4. The experimentally determined BER versus E_b/N_0 curve based on 180K simulated bits for each value of E_b/N_0 is shown; just above it is the initial linear fit, based on simulations run with E_b/N_0 set equal to 13.7 and 15.7. This line was extrapolated to predict the intersection with the BER = 1e-3 line, a forecast of $E_b/N_0 = 15.95$. However, the simulation at that point produced a BER estimate of 7.98e-04, below target. ξ_0, ξ_1, and ξ_2 were then used to fit another straight line, which predicted the intercept at $E_b/N_0 = 15.79$, and the process continued. The first 8 trials are illustrated.

Note that the simulated bit error rate is not deterministic: the 8th trial is well placed on the E_b/N_0 axis, yet the result is very noisy. Since the simulated bit error rate is not deterministic, the RadiSim solver is more than traditional root finding. There is inherent noise in each observation; therefore unlike simple bisection search or a Newton-Raphson solver, the solution technique must take these (random) variations into consideration.

Fig. 5 illustrates the convergence, robustness and accuracy of the solution technique, using 13.70 dB as the initial guess. Similar results are obtained using initial starting values of 10.50 dB and 14.50 dB [17].

5. Average is Maximum Likelihood Estimator

To validate the simulator and tune the simulator parameters, we need to have an accurate representation of the shape of the error curve. One way to generate such a representation is to perform numerous simulations at each different value of E_b/N_0, and combine the results.

One can combine multiple trials for a fixed E_b/N_0 value and a fixed number of transmitted bits to estimate the standard deviation of the underlying process. Simulating a fixed number of transmitted bits also makes the simulation run time deterministic.

What's the best way to combine multiple trials of constant size when estimating the bit error rate with simulation? One obvious method is to combine all the trials, and compute the grand average bit error rate; but how good is that? In this section, we show that the grand average is the maximum likelihood estimator (MLE) for an underlying Poisson process; therefore, in the absence of any other information, this is the best one can do.

Maximum Likelihood Estimation refers to determining the values of parameter(s) most likely to have produced the observations; by that criterion, they are the *best* estimates of the underlying process. See, for example, [18, Chapter 15].

Consider T trials, where trial i simulates n bits and produces t_i errors. If the underlying Poisson process has parameter λ, the likelihood of observing $t_1, t_2, ..., t_T$ errors is

$$\prod_{i=1}^{T} \frac{\mathrm{e}^{-\lambda}\lambda^{t_i}}{t_i!}$$

which we can rewrite as

$$\mathrm{e}^{-\lambda T} \prod_{i=1}^{T} \frac{\lambda^{t_i}}{t_i!}$$

Call the grand average c; that is, $c = \frac{1}{T}\sum_{i=1}^{T} t_i$; then the equation can be rewritten

$$\mathrm{e}^{-\lambda T} \frac{\lambda^{cT}}{\prod_{i=1}^{T} t_i!}$$

To determine the MLE, we maximize over all λ; therefore, the denominator, a constant, drops out, and we're left with

$$\max_{\lambda>0} \left(\mathrm{e}^{-\lambda}\lambda^{c}\right)^{T}$$

T is a fixed positive integer, so to maximize the expression, we can maximize $\mathrm{e}^{-\lambda}\lambda^c$; to solve for λ, set the derivative equal to 0

$$\mathrm{e}^{-\lambda}(-1)(\lambda^c) + \mathrm{e}^{-\lambda}(c\lambda^{c-1}) = 0$$

since $\mathrm{e}^{-\lambda}$ is non zero

$$(c\lambda^{c-1} - \lambda^c) = 0$$

and dividing both sides by λ^{c-1} gives

$$c = \lambda$$

which was to be proved.

6. Simulator Validation

By far, the most difficult task in developing RadiSim was to validate the results. As part of the ongoing development, almost every claim about the system model or the simulation techniques was challenged and verified. This ongoing validation was key to developing a solid model and simulator; in particular, many mistaken assumptions were discovered early, avoiding inaccurate simulations, erroneous conclusions, and bad decisions. In addition, we unit tested the building blocks, interactively checked intermediate results, and relied on the following techniques.

Classic Coherent and Noncoherent Theory The theory for optimal coherent and noncoherent detection of FSK signals is well established [12] [19] [9] [10]. Initially, RadiSim only modeled and simulated at baseband. To allow a comparison with these theories, we added non-zero-IF capabilities and both coherent and noncoherent matched filters; we used the existing flexibility to bypass unused portions of the system, such as Gaussian filtering of the incoming digital signal. As illustrated in Fig. 6, our simulations were consistent with theory, which indicated that the basic infrastructure, including noise generation and BER evaluation, was working.

One subtle issue is that noncoherent detection theory does not apply when using moderate carriers—the theory is predicated on transmitting narrow-band bandpass signals, for which the carrier frequency is much greater than the data rate, so that the complex correlation coefficient defined as

$$\rho = \frac{1}{2\mathcal{E}} \int_0^T \tilde{s}_1(t)\tilde{s}_2^*(t)\,\mathrm{d}t$$

[9, p. 202] is a valid approximation, where $\tilde{s}(t)$ is the baseband equivalent of $s(t)$. In our experiment, we transmitted binary data at 1 MHz using 160 KHz deviation on a 20 MHz carrier, and oversampled at 60×.

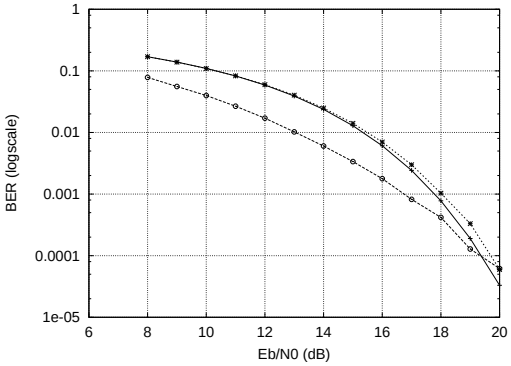

Figure 6: Lower line is simulated BER vs. E_b/N_0 in dB for chip design; upper lines are noncoherent matched filter simulation results and noncoherent theory prediction.

Consistent Results at Baseband and IF The most direct way to demonstrate fidelity of baseband simulation is to compare it with the simulated non-zero-IF results. Table 1 displays the strong consistency of the computed minimum E_b/N_0 required to achieve BER=1e-3 for an earlier version of the system using an ideal FM discriminator.

Table 1: Simulation Results at Varying Intermediate Frequencies of E_b/N_0 Needed to Achieve BER=1e-3.

run	IF (MHz)	E_b/N_0
run.2001.1009.0448.log	0	15.48
run.2001.1009.0452.log	2	15.50
run.2001.1009.0451.log	3	15.60
run.2001.1009.0450.log	4	15.47
run.2001.1009.0449.log	5	15.89

Comparison with SPICE Generating random inputs, including noise, for a SPICE transient simulation is problematic. We compared the simulations of SPICE and RadiSim for a deterministic system: transmitting a chirp data signal $(+1, -1, -1, \ldots)$ in the presence of a relatively prime interfering sinusoid.

Narrow-Band Digital FM Error Rate Theory Our system includes a hard limiter, and so does the system analyzed in [20]. We extended RadiSim to model that system; the results of runs with different IFs are summarized in Table 2, where BT is the $B_{if} * T_b$ product, and h is the modulation index. The theory values were read from the figures in [20], and are only approximate.

Concurrence appears to degrade with a non-zero-IF; our assessment is that FM click noise, which in the continuous case is independent of carrier, but in the discrete case depends on carrier, is a primary contributing factor. We did not analyze the anomalous results at BT=3, IF=2 MHz and IF=4 MHz.

Table 2: Narrow-Band Digital FM Error Rate Theory versus Simulation of E_b/N_0 predicted for BER=1e-3.

BT	h	theory	0 MHz	2 MHz	4 MHz
1	0.5	10.5	10.29	10.36	10.41
1	0.7	09.2	09.36	09.97	10.14
1	1.0	11.0	10.89	11.37	11.59
2	0.5	11.8	11.81	12.40	12.36
2	0.7	11.1	11.33	12.36	12.36
2	1.0	11.6	11.82	12.63	12.61
3	0.5	13.2	13.23	14.33	14.12
3	0.7	12.7	12.87	14.32	14.08
3	1.0	13.0	13.14	14.63	14.22

Dead Reckoning of Noise Levels Many theories use E_b/N_0, while many specifications use SNR. Assuming the incoming signal is fed through a bandpass filter of width B_{if} which passes the entire signal, $N_0 * B_{if}$ is the noise power, and E_b/T_b is the signal power. Therefore, at the output of the filter,

$$\frac{E_b/T_b}{N_0 * B_{if}} = \frac{Signal_{power}}{Noise_{power}}$$

This simple sanity check was instrumental in uncovering a basic flaw in our modeling. We erroneously decided that the signal would be modeled with a complex exponential, forcing the average signal power to be 1, while requiring that N_0 be set to match the environment. The problem was that we discarded half the signal power by ignoring the portion of the signal translated above 4 GHz by the image reject mixer. Therefore, adjusting N_0 within RadiSim was parameterized, allowing it to be increased by 3 dB so that E_b/N_0 would be correct.

7. Chip Design Revisited

RadiSim provided quick and conclusive analysis that enabled us to simplify and correct the implementation after the chip design was well underway.

Remove Filter Centering Circuitry According to the Bluetooth specification, the transmitter can initially be up to 75 KHz from center at the start and drift up to an additional 40 KHz during transmission, resulting in a net offset of 115 KHz from the nominal carrier. To help ensure good reception in this worst case, a centering mechanism was designed for the initial noise limiting bandpass filter. However, the addition of this centering circuitry increased the design and layout time, consumed die area, and reduced the filter's linearity. In about an hour with RadiSim, we were able to sweep the noise filter offset and determine that the system performance did not start to degrade appreciably until the transmitter was off center by 150 KHz (Fig. 7). Therefore, we decided to remove the centering circuitry. In addition to time, area, and linearity benefits, staying centered at nominal also avoided increasing adjacent channel interference.

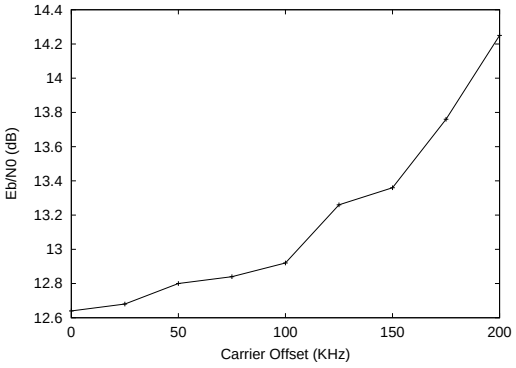

Figure 7: Estimated E_b/N_0 yielding BER=1e-3 versus carrier offset, for nominal deviation of 160 KHz.

Replace AGC Circuit with Hard Limiters Many FM detection techniques, including the one we are using, depend on having constant amplitude signals. We designed an automatic gain control circuit to help meet this need. However, Bluetooth has a relatively short preamble, making the AGC settling time requirements stringent; we needed to have the circuit settle in less than 1 μs, but current designs were taking about 1.5 μs. One classic solution which requires no feedback or settling time is to pass a sinusoid through a limiting amplifier to produce a square wave of constant amplitude, and lowpass filter the result to recover a sinusoid of constant amplitude. Could we *independently* hard limit each of the I and Q channels and still meet spec? With RadiSim, it was easy to add an amplitude limiting amplifier to each channel and demonstrate an acceptable degradation of 0.62 dB in overall system performance for a BER of 1e-3. Therefore, we abandoned the AGC circuit in favor of hard limiting and bandpass filtering.

Replace 3-Pole Bessel with 3-Pole Elliptic Filter We use a complex differentiator in our detector. Differentiators (with their $\bigvee$ shaped frequency response) inherently amplify any adjacent channel noise and interference which reaches them. Late in the design cycle, we found that we were about 3.35 dB over the adjacent channel interference spec using a 3-pole Bessel low pass filter to feed the slicer. By replacing the 3-pole Bessel with a 3-pole elliptic filter, we were able to recover more than 4 dB of performance, and meet spec. The elliptic filter fit in the silicon area formerly occupied by the Bessel filter.

8. Acknowledgements

Mark Gehring's initial conception and proposal for simulating the complex signal representation at baseband using DSP techniques and his ongoing support and creative input were vital contributions to the design of RadiSim. Lawrence Ragan, Dick Walvis, and Professor Y. C. Jenq each had beneficial discussions with the author on wireless RF in general, and RadiSim in particular. Feedback from Alan Mantooth and anonymous reviewers helped to substantially improve this presentation.

9. References

[1] Bluetooth specification version 1.1, February 22, 2001. http://www.bluetooth.com.

[2] Jan Crols and Michel S. J. Steyaert. A single-chip 900 MHz CMOS receiver front-end with a high performance low-IF topology. *IEEE Journal of Solid State Circuits*, 30(12):1483–1492, December 1995.

[3] Gerd Vandersteen, Piet Wambacq, Yves Rolain, Petr Dobrovolný, Stépahne Donnay, Marc Engels, and Ivo Bolsens. A methodology for efficient high-level dataflow simulation of mixed-signal front-ends of digital telecom transceivers. In *Design Automation Conference*, pages 440–445, 2000.

[4] Gerd Vandersteen, Piet Wambacq, Yves Rolain, Johan Schoukens, Stépahne Donnay, Marc Engels, and Ivo Bolsens. Efficient bit-error-rate estimation of multicarrier transceivers. In *Design, Automation and Test in Europe*, pages 164–168, 2001.

[5] Jess Chen. Extracting and using J-models to estimate ACPR in direct conversion transmitters. http://www.cadence.com/datasheets/dat_pdf/j_ext.pdf.

[6] Richard G. Lyons. *Understanding Digital Signal Processing*. Addison-Wesley, 1997.

[7] Alan V. Oppenheim and Ronald W. Schafer. *Discrete-Time Signal Processing*. Prentice-Hall, Inc., 1989.

[8] Leland B. Jackson. *Digital Filters and Signal Processing*. Kluwer Academic Publishers, 1986.

[9] Sergio Benedetto and Ezio Biglieri. *Principles of Digital Transmission with Wireless Applications*. Kluwer Academic / Plenum Publishers, 1999.

[10] John G. Proakis. *Digital Communications, 4th ed.* McGraw-Hill, 2001.

[11] Leon W. Couch, II. Complex envelope representations for modulated signals. In Jerry D. Gibson, editor, *The Mobile Communications Handbook, 2nd ed.*, chapter 1. CRC Press LLC, 1999.

[12] Bernard Sklar. *Digital Communications, 2nd ed.* Prentice-Hall, Inc., 2001.

[13] Ferrel G. Stremler. *Introduction to Communication Systems, 3rd ed.* Addison-Wesley, 1990.

[14] GNU home page. http://www.gnu.org.

[15] Linux home page. http://www.kernel.org.

[16] Octave home page. http://www.octave.org.

[17] Joseph P. Skudlarek. Estimating bit error rates and noise margins via simulation. http://members.dsl-only.net/~jskud.

[18] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes in C: the Art of Scientific Computing, 2nd ed.* Cambridge University Press, 1992.

[19] Bhagwandas Pannalal Lathi. *Modern Digital and Analog Communications Systems, 3rd ed.* Oxford University Press, Inc., 1998.

[20] R. F. Pawula. On the theory of error rates for narrow-band digital FM. *IEEE Transactions on Communications*, COM-29(11), November 1981.