Application Note AN107A May 12, 1997

Using SystemView by ELANIX to Generate Bit Error Rate (BER) Curves

Maurice L. Schiff, Ph.D., Chief Scientist, ELANIX, Inc.

The most important figure of merit in a communications system is the bit error rate (BER). This application note is designed to aid the user in generating such plots.

We will start with the mechanics of generating the BER plot. **Figure 1** is the simplest of communication systems. We are adding noise (AWGN) to a data source token (token 0) of amplitude A at 1 sample/bit, and then making a hard decision (token 5) on the output. If the noise sample is greater than the signal and is of opposite sign, an error has occurred. This model is the vector channel representation of antipodal signaling such as PSK.

The first step is to calibrate the signal to noise ratio, which is generally expressed in energy per bit Eb divided by the noise power density No (W/Hz)/ By definition for a baseband signal,

$$Eb / No = A^2T / No = A^2 / NoR$$

where A is the signal amplitude (normally set to unity), and R = 1/T is the data rate of the source. Now set No in the noise token such that Eb/No = 1 (0 dB) or,

$$No = A^2 / R$$

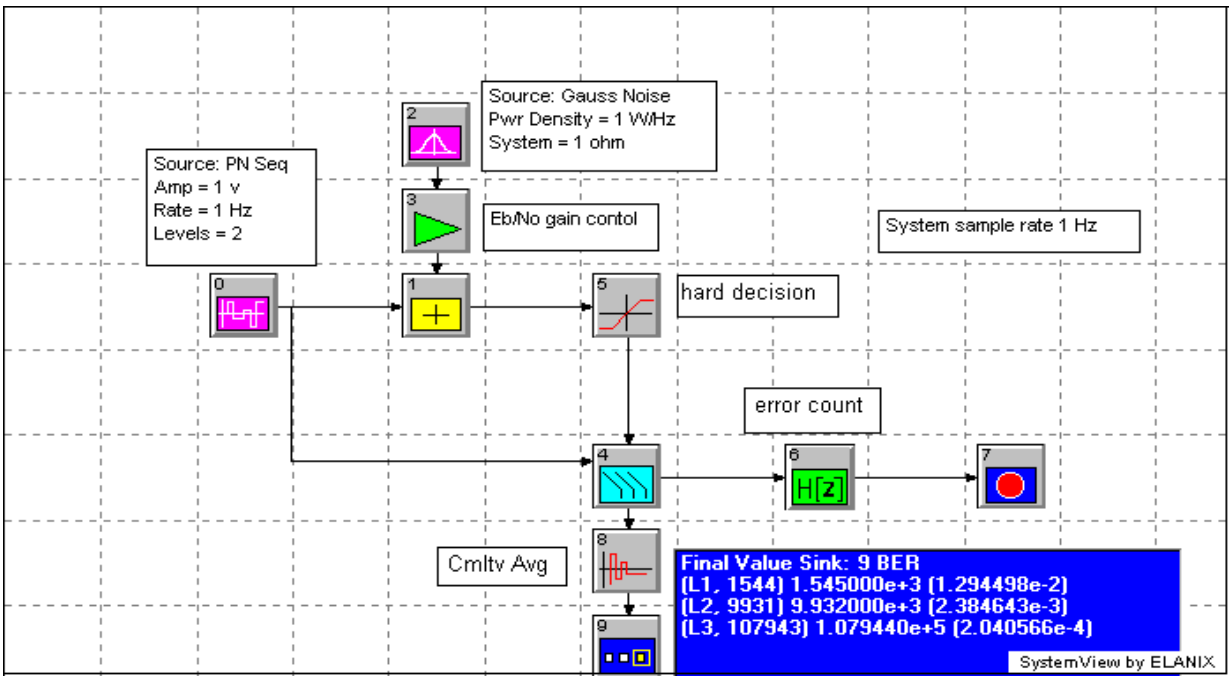

Figure 1. BER set-up for simple communications system

Enter the value of No calculated from this equation into the noise source token, using the W/Hz parameter window.

The control of Eb/No can be accomplished by inserting a gain token after the source, or after the noise. The latter is preferred (token 3) if your system has a tracking loop, where the bandwidth is dependent on the amplitude of the incoming signal.

Setting Up the BER Token and Its Outputs

The BER token (token 4) has several parameters as described in the token definition section. Set the number of trials to 1. The token output is 1 if there is an error, and 0 if correct. To obtain a running BER use the cumulative average token (token 8) found in the Function library. The only value of the average that will be used is the final value at the end of the run. Thus, we use a final value sink (token 9). This saves the memory requirements of having to hold onto all of the intermediate values which are unnecessary.

The number of points that must be run in a simulation depends on the BER level desired. For a BER of say 1e-4 at least 1e-5 trials must taken to obtain a statistically significant number of errors. If the BER desired is 1e-2, then at least 1e- 3 trials are required, etc.

One way to generate the BER curve is to set a large number of trials in the time window, large enough to accommodate the number of trials required for the lowest desired BER. The problem with this method is, at the higher BER rates, this large number of trials is an overkill. What is desired is a mechanism where the number of trials is related to the BER at hand. This can be accomplished by running the simulation long enough to accumulate a fixed number of errors. The number of trials will vary in this case. However, at each BER level only the time required to obtain a sufficient accuracy is used. This procedure optimizes the simulation run time.

A simple counter can be implemented by a digital filter $H(z) = 1/(1-z^{-1})$. This is shown as token (token 6) the figure. The output of the counter is sent to a sink token (token 7) with a stop sink definition. One of the options in the stop sink token is to go to the next loop when the threshold is reached. Thus, if we set the threshold to, for example, 20, the simulation at a given loop will continue until 20 errors occur, and then the simulation will advance to the next loop. In the time window, set the run time per loop large enough to accommodate the lowest desired BER. The loop will run until the stop token threshold is crossed or until the run time indicated is satisfied. Again, the simulation will spend only that time required to obtain a significant answer at each loop for each BER.

To control the Eb/No per loop use the global parameter link option in the token pull down menu. Select the gain token (token 3). In the panel F(Gi,Vi), write a simple algebraic expression that controls the noise gain token (token 3) in dB. For example, suppose that we wish to start Eb/No at a value of 4 dB and increment by 2 dB on each loop. Then the equation F(Gi,Vi)= -2*cl-2 will produce the desired results. The minus sign is used since we are decreasing the noise, not raising the signal power in this example. At the end of each loop, the final value BER sink will have the desired data as shown in the parameter window in **Figure 1.**

To obtain the desired BER plot, got to the analysis window. Open the sink calculator, and select the 'style' tab. Select the BER plot option. In the start parameter box enter 4, and for the increment parameter enter 2. Select the sink token that contains the final value BER information and click OK. The BER plot as shown in **Figure 2** will appear. You can annotate this plot as desired.

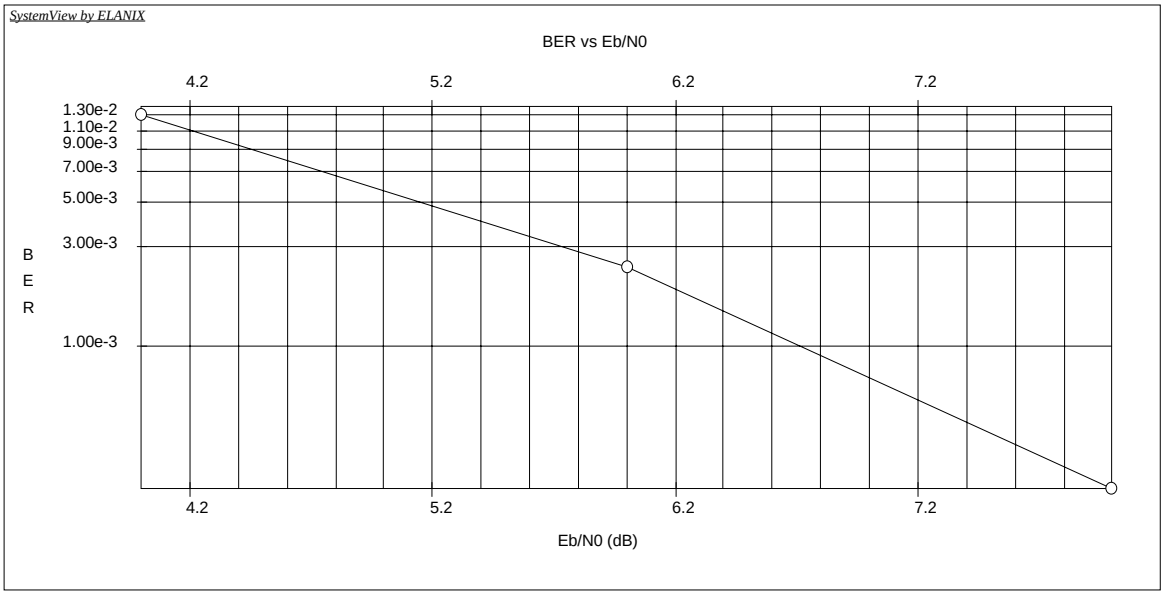

Figure 2. Typical BER curve

Another requirement for calculating BER curves centers on the timing of the signal as it passes through the system. **Figure 3** shows a more complicated system than **Figure 1**. In this case we are encoding the data via a (7,4) BCH block code (token 2). The BER token (token 8) requires two inputs which have the exact same time base. The time base rate is one sample per data bit. However, due to group delays through the simulation, the recovered clock while having the correct rate may be shifted in phase from the clock associated with the truth data.

In **Figure 3** the source data (token 0) is sampled at a rate of 1 sample per bit or 1 Hz in this case. The sampler token then produces a time base at this rate with the first sample stating a t=0. The BCH coder token gathers 4 bit blocks of the input and produces 7 bit blocks which occupy the same time span (4 sec). Thus, each encoded bit is 4/7=0.5714 sec. long. At this point the output clock of the encoder is at a 1.75 Hz rate with the first sample at t=0. After the encoder the hold token (token 4) reestablishes the system sample rate.

The channel adds AWGN noise (token 11), and the receiver must then recover the data.

The optimum detector for any signal with AWGN is the matched filter. In this case the matched filter is a simple integrate and dump (I&D) operation (token 5) set to the encoded bit time 4/7 sec.. The data is recovered by sampling (token 6) this filter once per bit at a data rate of 1.75 Hz. It is important to note that the first valid data bit out of the sampler is at t = 4/7 sec not at t = 0. This is the group delay through the I&D token. To properly decode the blocks of data, the start time parameter of the BCH decoder (token 9) is set for 4/7 sec. The decoders first output is then 4 4/7 sec., the next is at 8 4/7 sec, etc. The data bits (4 per block) are 1 sec in duration *but* the first bit is at 4/7. Thus, there is a 4/7 sec time difference between the two 1 Hz clocks which are carrying data to the BER token. The combination of the hold token (token 21) and the sampler token (token 22) reestablishes this clock phase of the clock from the BCH decoder to 0, 1, 2, sec, etc.

The remaining task is to determine the overall group delay through the system. This is accomplished by using the Correlation/ Convolution option in the sink calculator. The sample delay token (token 20) is initially set to 0. Run the system for a short period of time.

Now in the Analysis Window go into the sink calculator. Choose the Correlation/ Convolution tab, and select the cross correlation option. In the upper right text box choose one of the two sinks entering the BER token (token 19), in the lower box choose the other (token 18). The resulting plot will have a sharp peak. **Figure 4** shows this type of plot for this case.

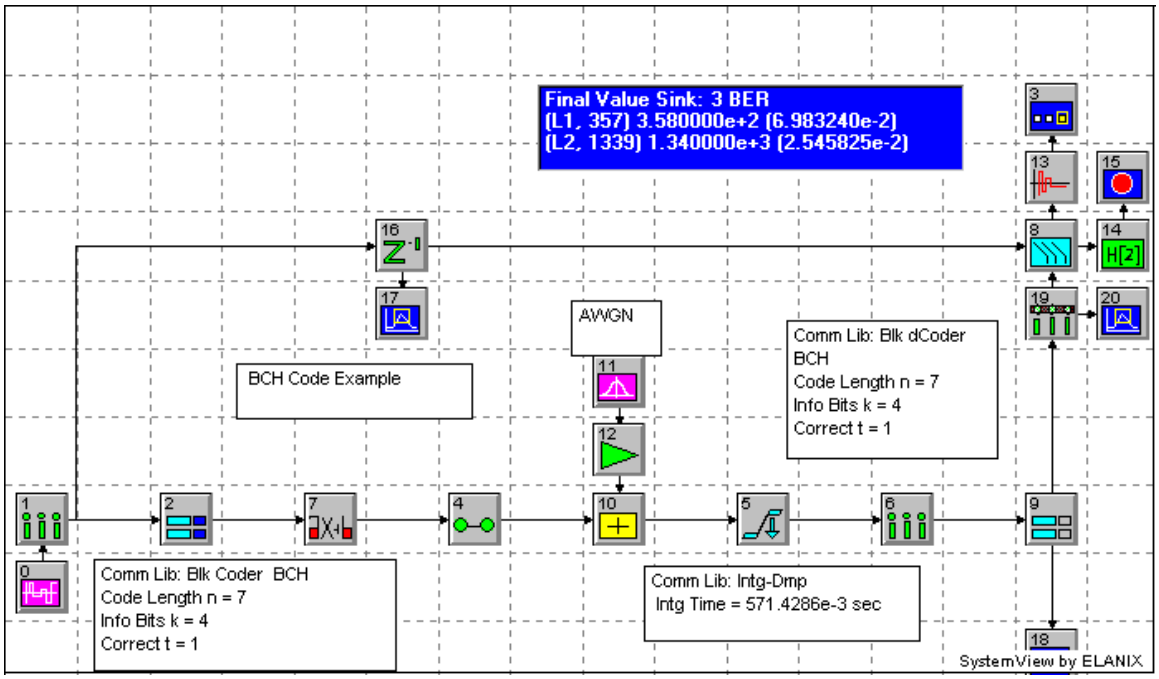

Figure 3. Simulation system with group delay

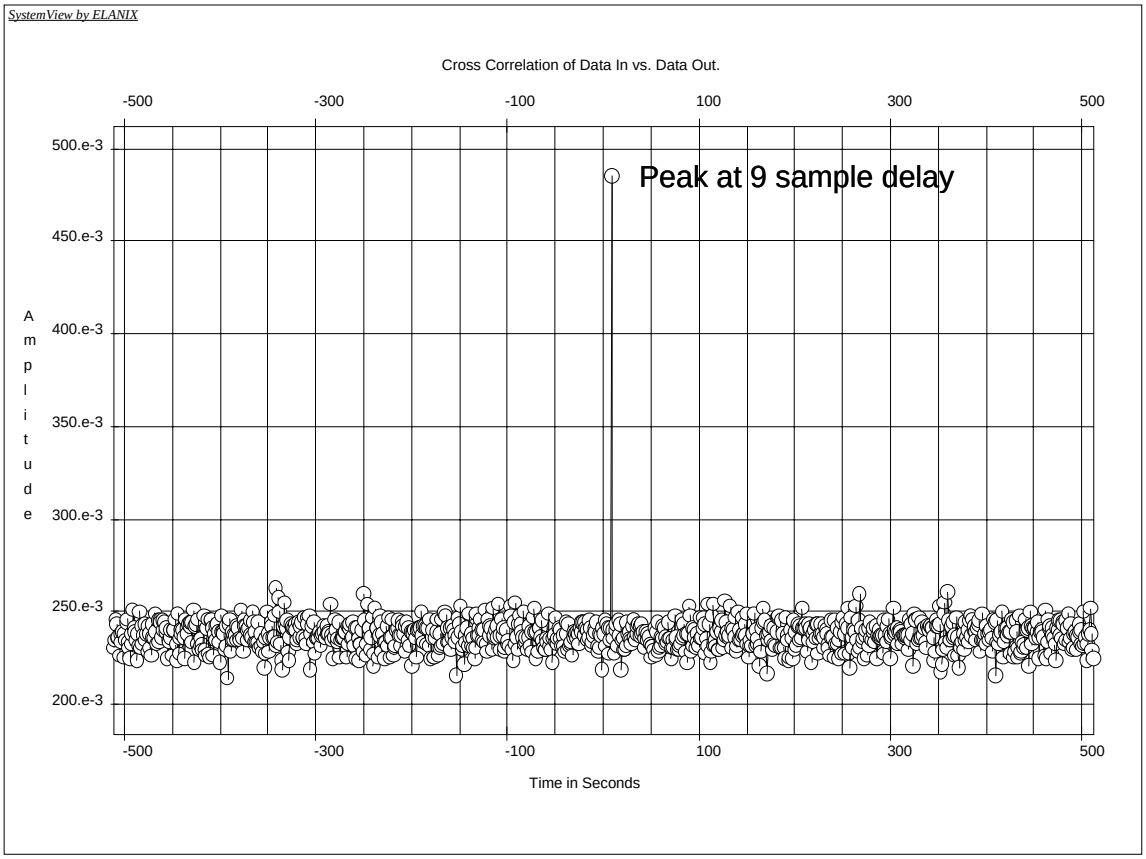

Figure 4. Cross correlation of BER input data stream

Place your mouse on the peak correlation point and read in the box at the upper right the coordinates of the point and in particular the location of the point with respect to zero shift : S8 in this case. Enter this integer number into the sample delay (token 18) and rerun the simulation. There should be no errors.

Your system is now properly aligned. Reestablish the noise and setup the simulation as described above.

Other Timing Issues

A common operation in communications systems is to recover a bit or symbol after a matched filter or other filtering options. The SystemView sample token is designed for this purpose. However, the sampler starts sampling at t = 0, T, 2T, etc. It may happen in your system that the group delay through the various token elements is not an integer multiple of T.

In this case, the sampler would not recover the correct information. To compensate for this offset, add a delay token before the sampler and add the necessary value to move the total group delay to a multiple of T. This is equivalent to establishing bit or symbol synchronization. Now the sampler token retrieves the correct information.

Group Delay Through SystemView Tokens

- **FIR filter:** For an N tap FIR filter operating at a rate fs, the group delay T is T=(N-1)fs/2. In order to keep T an integer multiple of fs, N should be an odd number.
- **IIR filter:** The is no simple formula available. In the SystemView linear system token window click on the 'group delay' button in the lower left hand corner of the form. Expand the plot around the frequency range where the filter response is maximum. Read the group delay calibrated in units of the sample time. Note that the group delay need not be an integer number of 1/fs.
- **Integrate and Dump matched filter token in Comm library:** The group delay T is the integration time entered in the token parameter window.
- **Average Token:** The group delay T is the time value entered in the parameter window.
- **Block Error Encoder:** For an (N, k, t) block error encoder the group delay is one code block in time. T can be expressed in terms of the input sample rate to the token which is the data or symbol rate R of the information source. **T = k/R**. The encoder token produces N bits out for k bits in at a new rate R' =RN/k. Thus **T = N/R'** is also a valid expression.
- **Block Error Decoder:** Same as Block error encoder.
- **Convolutional Code Encoder:** None.
- **Convolutional Code Decoder:** The convolutional decoder parameters include the number of information bits k, and the path length (PL) used by the Viterbi algorithm to make a decision. If R is the encoded bit rate into the decoder, then the delay T is given by **T = k*(PL+2)/R.**
- **Bit to Symbol**: The token gathers k bits of data having a rate R, and produces an output between 0 and 2^k-1. The group delay is **T = k/R**.
- **Symbol to Bit:** None.
- **Gray Encoder**: The encoder converts one set of k bits of a signal of data rate R (at 1 sample/bit), to a second Gray encoded set. The group delay is **T = k/R.**
- **Gray Decoder**: Same as Gray encoder.
- **Pulse Width Modulator**: None.
- **Pulse Width Demodulator:** The group delay T is the reciprocal of the rate entered in the token parameter widow.
- **Pulse Position Modulator:** None.
- **Pulse Position Demodulator:** The group delay T is the reciprocal of the rate entered in the token parameter widow.