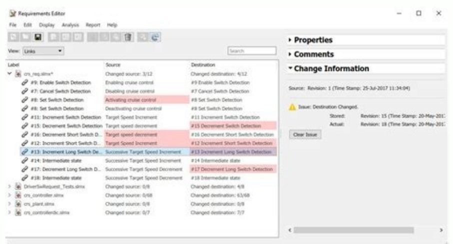
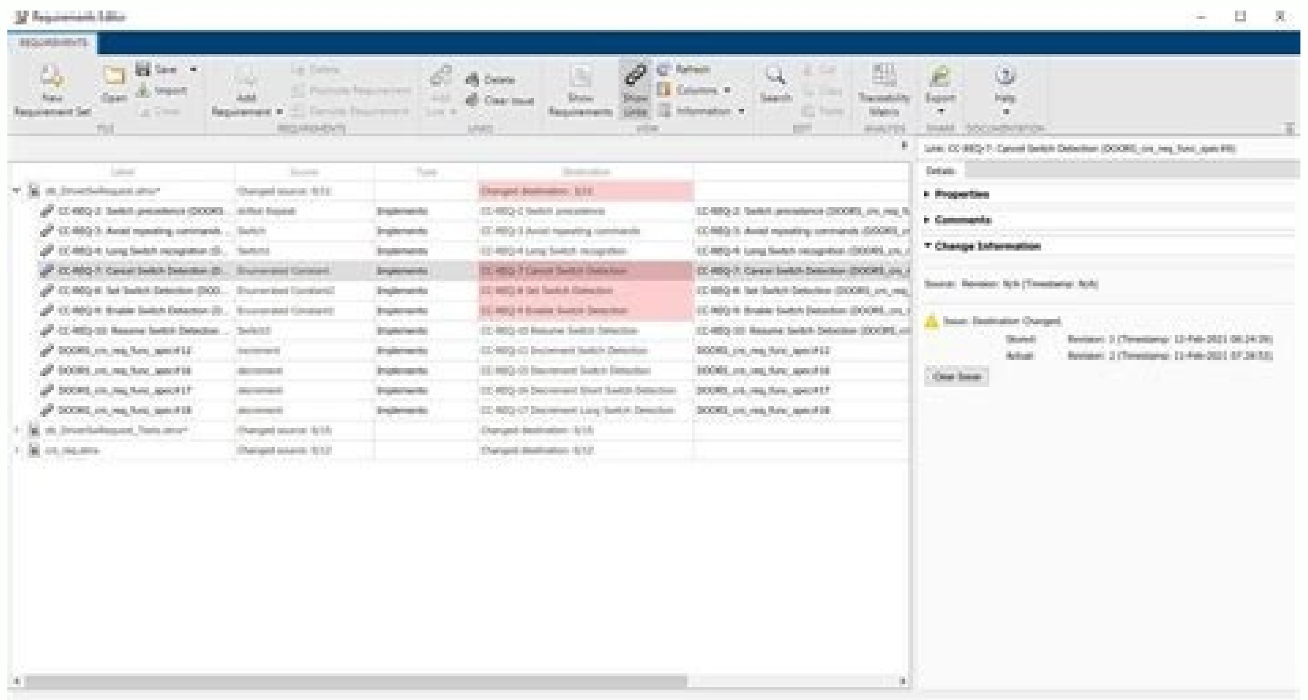
50410436856 94379877330 23897898.595238 11815369.958333 20380658959 20204465330 89965447920 9480237374 16134949420 8875312220 7365128155 22215005.797753 96080142564 64413858138 14253031.646341 70761946650 117544167062 42395928776 110562385220 79695903.130435

Simulink awgn examples data sheet pdf





open_system([modelname '/AWGNGenerator']); SNR dB to Linear Scale ConverterThe dBtoLinearConverter subsystem takes an SNR value in dB as input and converts it into noise variance in a linear scale. 'EdgeColor','none'); histogram(real(awgnSimulink(NumOfSamples+latency+1:end)),500,... The value must be positive.Variance from port, where you provide the variance as an input to the block. During the conversion, the signal power is assumed to be 1. “A Hardware Gaussian Noise Generator Using the Box-Muller Method and Its Error Analysis,” 659–71. VerificationCompare the output of the AWGN Simulink model with the output of the HDL equivalent AWGN MATLAB® function.NumOfSamples = 1000; % MATLAB output fprintf(' Simulating MATLAB HDL AWGN Generator for comparison...'); awgnMatlab=whdlexamples.hdlawgn(snrdBSimInput(1:NumOfSamples),seedsURNG1,seedsURNG2); fprintf(' Simulation complete. This function generates the input data and initializes the seeds for tausURNG and coefficients for the function evaluation. The white Gaussian noise can be added to the signals using MATLAB/GNU-Octave inbuilt function awgn(). But this time we will plot both the input signal and the noisy signal simultaneously in the same figure to analyze the changes carefully.fs = 1000;t = 0:1/fs:1;f = 10;snr = 5;st = square(2 * pi * f * t);subplot(2,1,1);plot(t,st,'b','Linewidth',2);xlabel('Time'); ylabel('Amplitude');title('Input Message Signal');grid on; st_nn = awgn(st,snr,'measured');subplot(2,1,2); plot(t, st, 'b', 'Linewidth', 2);hold on;plot(t, st_nn, 'r');xlabel('Time');ylabel('Amplitude');title('Signal After Addition of Noise');grid on;hold off;Output:Output Plots To directly specify the variance of the noise generated by AWGN Channel, specify the Mode as: Variance from mask, where you specify the variance in the dialog box. This noise power is used to multiply the output of the Gaussian noise with unit variance. This module has a latency of 2. References1. Sin and Cos are implemented using the existing Sine HDL Optimized and Cosine HDL Optimized (HDL Coder) blocks in the HDL Coder / Lookup Tables library.close_system([modelname '/AWGNGenerator/GaussianNoiseWithUnitVar/SqrtImplementation/SqrtEval']); Gaussian Noise Generator with Required VarianceThe GaussianNoiseWithReqVar subsystem converts Gaussian noise with unit variance to Gaussian noise with required variance. The whdlexamples.hdlawgnGen_init.m script file initializes these seeds.The ConcatandExtract subsystem accepts 32-bit uniform random integers, a and b, to generate two uniform random numbers, u0 and u1, in the range [0, 1) with bit-widths 48 and 16, respectively. Simulation complete. The hardware implementation of AWGN accelerates the performance evaluation of wireless communication systems using an AWGN channel. Luk, and P.H.W. Leong. Implementation of the HDL logarithm involves these three steps.Range Reduction – In this step, the original range of the input, which is [0, 1-2^(–48)) is reduced to a more convenient smaller range of [1, 2). The implementation results are shown in this table. . HDL Code GenerationTo check and generate the HDL code referenced in this example, you must have an HDL Coder™ license.To generate the HDL code, enter this command at the MATLAB command prompt.makehdl('HDLAWGNGenerator/AWGNGenerator') To generate a test bench, enter this command at the MATLAB command prompt.makehdltb('HDLAWGNGenerator/AWGNGenerator') In this example, HDL code generated for the AWGNGenerator module is implemented for the Xilinx® Zynq®-7000 ZC706 board. The Box-Muller method uses two uniformly distributed random variables to generate two normally distributed random variables through a series of logarithmic, square root, sine, and cosine operations as shown in this figure. modelname = 'HDLAWGNGenerator'; open_system(modelname); This example demonstrates the implementation of an AWGN generator based on the Box-Muller method. Coefficients of the first-degree polynomial are stored in a lookup table, which is indexed using the first 6 bits of input to the function evaluation block.Range Reconstruction – The result of the function evaluation is expanded back to the original range using a left shift operation.close_system([modelname '/AWGNGenerator/GaussianNoiseWithUnitVar/logImplementation/log']); open_system([modelname '/AWGNGenerator/GaussianNoiseWithUnitVar/SqrtImplementation/SqrtEval']); Implementation of HDL Sine and CosineThe HDL optimized implementation of a sine or cosine function uses a lookup table approach. Each 32-bit uniform random number with improved statistical properties is obtained by combining three linear feedback shift register (LFSR) based uniform random number generators (URNGs). 'EdgeColor','none'); legend('5 dB SNR','15 dB SNR'); figure; title('PDF for Imaginary Part of AWGN'); hold on histogram(imag(awgnSimulink(latency+1:NumOfSamples+latency)),500, ... J.D. Lee, J.D. Villasenor, W. The Box-Muller method is widely adopted for Gaussian noise generation because of its hardware-friendly architecture and constant output rate. This lookup table approach is used for converting an SNR value in dB to a noise power value in a linear scale. The square root function is approximated on the reduced range in the next step.Function Evaluation – The square root function is approximated over 64 equally spaced segments in the range [1, 2) and [2, 4) by using a first-degree polynomial. For Variance from mask and Variance from port mode: If the variance is a scalar, then all signal channels are uncorrelated but share the same variance.If the variance is a vector whose length is the number of channels in the input signal, then each element represents the variance of the corresponding signal channel.To specify the variance indirectly, that is, to have the block calculate the variance, specify the Mode as: Signal to noise ratio (Eb/No), where the block calculates the variance from these quantities that you specify in the dialog box: Signal to noise ratio (Es/No), where the block calculates the variance from these quantities that you specify in the dialog box: Signal to noise ratio (SNR), where the block calculates the variance from these quantities that you specify in the dialog box: Changing the symbol period in the AWGN Channel block affects the variance of the noise added per sample, which also causes a change in the final error rate. Implementation of HDL Tausworthe Uniform Random NumberThe Tausworthe Uniform Random Number Generator module is used to generate two 32-bit uniform random integers. But here, we will study only two syntaxes of it which are most commonly used in the communication system and signal processing.Syntax:awgn(input_signal, snr)This syntax will add the white Gaussian noise to the passed input_signal and maintains the passed SNR (signal to noise ratio) in dB. % Run this command to open the HDLAWGNGenerator model. The implementation is pipelined to maximize the synthesis frequency, generating AWGN with an initial latency of 11. Uniform random number u1 is generated by extracting the lower 16 bits of b.open_system([modelname '/AWGNGenerator/GaussianNoiseWithUnitVar/TausUniformRandGen']); close_system([modelname '/AWGNGenerator/GaussianNoiseWithUnitVar/TausUniformRandGen']); open_system([modelname '/AWGNGenerator/GaussianNoiseWithUnitVar/TausUniformRandGen/TausURNG1']); close_system([modelname '/AWGNGenerator/GaussianNoiseWithUnitVar/TausUniformRandGen/TausURNG1']); Implementation of HDL LogarithmHDL logarithm subsystem evaluates the approximate logarithm based on the piecewise linear polynomial method. Those two uniformly distributed random varibles are generated using the Tausworthe algorithm. The performance evaluation of these systems with these simulation parameters is a bottleneck. This implementation requires these two seeds: TausURNG1 and TausURNG2. 'EdgeColor','none'); legend('5 dB SNR','15 dB SNR'); Simulating HDL AWGN Generator... A bit left shift operation is used for range reconstruction and to implement the –2*log function.Run this command to open HDL logarithm subsystem.open_system([modelname '/AWGNGenerator/GaussianNoiseWithUnitVar/logImplementation/log']); Implementation of HDL Square RootThe HDL Square root subsystem evaluates approximate square root based on the piecewise linear polynomial method. The implementation of the HDL square root involves these three steps.Range Reduction – The input data type to the module is fi(0, 31, 24). This subsystem has a latency of 1 clock cycle.Gaussian Noise Generator with Unit VarianceThe GaussianNoiseWithUnitVar subsystem generates Gaussian noise with unit variance by using the Box-Muller method. 'Normalization','pdf','BinLimits',[-2 2],'FaceColor','yellow', ... By default this syntax considers the power of the input_signal as 0 dBW (decibel watt).awgn(input_signal, snr, signal_power)This syntax will do the same thing as the first one but the only difference is, here the power of the input_signal is not considered as zero rather it has to be passed as one of the arguments along with the input_signal and snr.Note: Signal power can be passed as “measured” or some scalar value to set the signal level of the input_signal, according to which the appropriate noise level is determined based on the value of snr.Stepwise ImplementationLet’s understand the implementation with the help of an example where we will add the gaussian white noise to the sine waves.Step 1: Define the required parametersfs = 1000;t = 0:1/fs:1;f = 20;snr = 10;Step 2: Define the input signal and plot st = sin(2 * pi * f * t);plot(t, st, 'b', 'Linewidth', 2);xlabel('Time');ylabel('Amplitude');title('Input Message Signal');grid on;Output:Input Signal (Sine Wave)Step 3: Add white Gaussian noise to signal and plotst_nn = awgn(st, snr, 'measured');plot(t, st_nn, 'r', 'Linewidth', 2);xlabel('Time');ylabel('Amplitude');title('Signal After Addition of Noise');grid on;Output:Noisy Signal (With white Gaussian noise)Let’s see another example Addition of white Gaussian noise to square wave. ') % Compare MATLAB and Simulink outputs figure; ax=axes('FontSize', 20); plot(1:1000,real([awgnSimulink(latency+1:NumOfSamples+latency) awgnMatlab])); xlabel(ax,'Number of Samples'); ylabel(ax,'Real Part of AWGN'); title(ax,'Comparison of MATLAB and Simulink Output (Real Part)'); legend('Simulink output','MATLAB output'); figure; ax=axes('FontSize', 20); plot(1:1000,imag([awgnSimulink(latency+1:NumOfSamples+latency) awgnMatlab])); xlabel(ax,'Number of Samples'); ylabel(ax,'Imaginary Part of AWGN'); title(ax,'Comparison of MATLAB and Simulink Output (Imaginary Part)'); legend('Simulink output','MATLAB output'); Simulating MATLAB HDL AWGN Generator for comparison... In this example, the Simulink® model accepts signal-to-noise ratio (SNR) values as inputs and generates Gaussian random noise along with valid signal. Coefficients of the second-degree polynomial are obtained using the polyfit function. u0 is generated by concatenating the 32-bit value of a and higher 16 bits of b. Modern wireless communication systems includes many different simulation parameters, such as channel bandwidth, modulation type, and code rate. The variance input must be positive, and its sampling rate must equal that of the input signal. This subsystem takes inputs from dBToLinearConvertor and GaussianNoiseWithUnitVar subsystems. Plot the probability density function (PDF) of the AWGN output.latency = 11; NumOfSamples = 10^6; % Simulate the model open_system('HDLAWGNGenerator'); set_param(gcs,'SimulationMode','Accel'); fprintf(' Simulating HDL AWGN Generator...'); outSimulink = sim('HDLAWGNGenerator','ReturnWorkspaceOutputs','on'); fprintf(' Simulation complete.'); awgnSimulink = outSimulink.awgnOut; % Plot PDF figure; title('PDF for Real Part of AWGN'); hold on histogram(real(awgnSimulink(latency+1:NumOfSamples+latency)),500, ... These coefficients are stored in a lookup table, which is indexed using the first 8 bits of input to the function evaluation block.Range Reconstruction – The result of the function evaluation is expanded back to the original range. 'Normalization','pdf','BinLimits',[-2 2],'FaceColor','blue', ... The log function is approximated on the reduced range in the next step.Function Evaluation – The log function is approximated over 256 equally spaced segments in the range [1, 2) by using a second-degree polynomial. This example shows the implementation of an additive white Gaussian noise (AWGN) generator that is optimized for HDL code generation and hardware implementation. IEEE, 2006. We have to follow the same three steps as above to add the white Gaussian noise to the square wave. In this article, we are going to discuss the addition of “White Gaussian Noise” to signals like sine, cosine, and square wave using MATLAB. Here, “AWGN” stands for “Additive White Gaussian Noise”.AWGN is a very basic noise model commonly used in the communication system, signal processing, and information theory to imitate the effect of random processes that occur in nature. We normally have different syntaxes for awgn() function depending on the number and type of parameters passed to it. 'EdgeColor','none'); histogram(imag(awgnSimulink(NumOfSamples+latency+1:end)),500, ... This range is reduced to a smaller range of [1, 4). The example supports SNR input ranges from –20 to 31 dB in steps of 0.1 dB. The linear noise variance obtained from dBToLinearConvertor is multiplied with normally distributed random variables obtained from GaussianNoiseWithUnitVar.Results and PlotsThe whdlexamples.hdlawgnGen_init.m script file is used to specify the SNR range, generate the required number of noise samples, initialize the seeds for TausURNG1 and TausURNG2 subsystem and to generate coefficients for the function evaluation of the HDL log and square root.The whdlexamples.hdlawgnGen_init.m script file is the initialization function of HDLAWGNGenerator model. Simulate HDLAWGNGenerator.slx to generate 10^6 valid AWGN samples for each SNR of 5 dB and 15 dB. Hardware capabilities of FPGAs can speed up simulations. The top-level structure of the model includes these three subsystems.SNR dB to Linear Scale ConverterGaussian Noise Generator with Unit VarianceGaussian Noise Generator with Required Variance% Run this command to open the subsystems inside AWGNGenerator model. This module has latency of 3 clock cycles.