Kiran Gunnam

ELEN 249, 02/17/2016

Mini Project on Designing an AWGN IP core

Please use the following reference (which is already included in Camino)

Lee, D.-U.; Villasenor, J.D.; Luk, W.; Leong, P.H.W., "A hardware Gaussian noise generator using the Box-Muller method and its error analysis," in Computers, IEEE Transactions on , vol.55, no.6, pp.659-671, June 2006, doi: 10.1109/TC.2006.81

http://ieeexplore.ieee.org/xpl/login.jsp?tp=&arnumber=162895

Read the paper thoroughly with emphasis on being able to implement the Matlab and RTL code for pseudo-code given in Fig 6. A simple block diagram is given in Fig 2.

You can use the specification (including bit widths) from the paper. Though you are not expected to write program on these aspects on arriving at specification, bitwidths and error analysis, you need to understand how these results are obtained.

Deliverables

1) Bit accurate quantized Matlab Model for AWGN based on the provided reference. Your matlab model should be able to generate the test vectors needed for automated verification of your Verilog RTL design.
2) Synthesizable Verilog RTL design for AWGN. The RTL IP core should have the following input interface pins
   Clock (1 bit), reset (1 bit), urng_seed1 (32-bit) and urng_seed2 (32-bit). The output interface pins should be awgn_out (16 bits). urng_seed1 and urng_seed2 are supplied as seeds to two URNGs used at the initialization stage.
3) Make sure to have a Verilog/System Verilog test bench to compare the RTL results and Matlab results. You can choose to generate all the vectors from Matlab model for 10,000 samples and then run a RTL test bench with RTL model to read the test vectors and compare the intermediate values from your RTL model.
   Provide well commented code along with a Model Sim project to simulate the RTL code and read the test vectors generated by Matlab model and do automatic comparsion of key intermediate results. Your simulation should be able to give warnings on any bit-level errors. Functional simulation is required while you are not required to test your design on an FPGA.

Alternative for 1 to 3: You may do a bit accurate system generator model and then then do RTL generation from System Generator. Also please make sure to download the design on a FPGA and do a co-simulation.

4) Prepare a datasheet of your design. Use this as an example:
   http://www.xilinx.com/support/documentation/ip_documentation/awgn.pdf

5) Please upload your developed program via a github. Create a public repository of your code and documentation at https://github.com and submit the link by email for grading.
   [Optional] You may want to add a GNU license according to http://www.gnu.org/licenses/gpl-3.0.en.html, if you would like your code to be used by others for free.