AWGN IP documentation

Features:

- Based on the Box-Muller algorithm and the central limit theorem as described in [1].
- Piecewise polynomial based function approximation units with range reduction [1].
- Guaranteed maximum error up to 1 ulp.
- Probability density function (PDF) deviates less than 0.2 percent from the Gaussian PDF for $|x| < 8.1\sigma$ and is obtained from a closed-form expression.
- Period of noise generator is $10^{15} \sim 2^{50}$
- Noise is quantized to 16 bit samples with 5 bits of integer and 11 bits of fraction.
- 192 bits seed consisting of six 32 bit words to get a different random sequence.
- Bit true MATLAB model included with this IP.
- Uses 28k bits of memory to store the table contents used to evaluate expressions in the Box-Muller algorithm.

Applications:

- The output of the Additive White Gaussian Noise core can be added to signal data to create an AWGN channel useful for measuring the bit error rate performance of a communication system.
- Apart from measuring performance of a communication system, this core can also be used for large scale molecular dynamics simulation, evaluating performance of channel codes like Turbo codes and low density parity check codes(LDPC).

AWGN IP facts:

Core Specifics	
Families Supported	Vertex 7 (all grades)
FPGA Resources: (I/O's,LUT,FF, BRAM)	194 I/O's , 1018 LUT , 826 FF , 5 BRAM
Max frequency	200MHz
Provided with Core	
Documentation	Produce specification
Design File Format	Verilog
Verification	Matlab + Verilog
Instantiation Template	Verilog Wrapper
Additional Notes Included	MATLAB verification and analysis program
Design Tools Requirement	
Verification	MATLAB 2016 or better
Simulation	Modelsim PE edu 10.4a
Xilinx Implementation Tools	Xilinx ISE, Xilinx Vivado

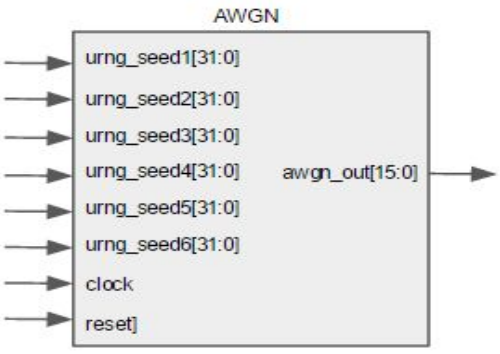

Fig1: Core Schematic Symbol

Santosh Kumar Krishnan

Functional Description:

The AWGN core generates white Gaussian noise using a combination of the Box-Muller algorithm and the central limit theorem. The Box Muller algorithm uses two uniform random numbers and transforms it into a normal distributed random value. The AWGN core has two uniform random number generator (URNG), each of which uses three 32 bit seeds. By changing the values of the seeds, we can get a different sequence of uniform random number. The core starts to generate the normal distributed random by pulling the reset signal down.

Core Description:

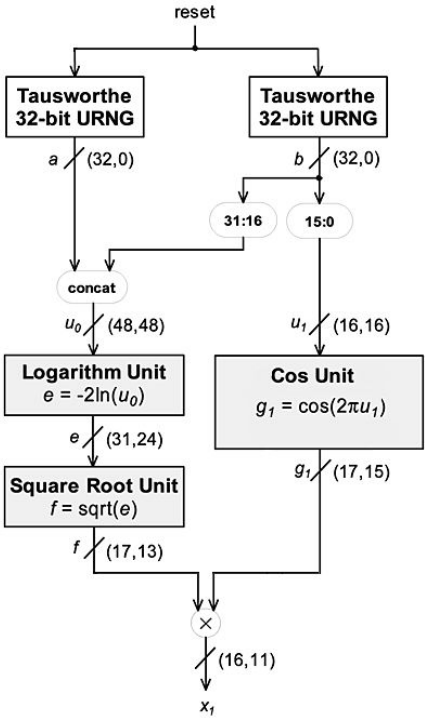

Fig 2: AWGN core architecture [1]

Shown in figure 2, is the AWGN core architecture. The the Tausworthe URNG generators use three 32 bit seeds and use the algorithm presented by L'Ecuyer [2] and generate two uniformly distributed random numbers.. These two random number are then split and combined as shown the figure to generate a 48 bit (U_0)and a 16 bit (U_1) random number. The 48 bit random number is given to the logarithm unit while the 16 bit number is given to the cosine unit. The output of the logarithm unit is given to the square root unit. The bit widths employed are obtained after taking help of the error analysis done by DU Lee [1]. The individual function unit were implemented by obtaining the piecewise polynomial approximation model that minimized the error using the minimax function in MAPLE.

Pin Out:

The signal names for the core is shown in figure 1. and described in the table below.

Pin	Dir	Description
clock	Input	Clock signal.
Reset	Input	The reset signal is used to initialize the registers, memory tables.
urng_seed1[31:0]	Input	The first 32 bit word that is used by the first URNG.
urng_seed2[31:0]	Input	The second 32 bit word that is used by the first URNG.
urng_seed3[31:0]	Input	The third 32 bit word that is used by the first URNG.
urng_seed4[31:0]	Input	The first 32 bit word that is used by the second URNG.
urng_seed5[31:0]	Input	The second 32 bit word that is used by the second URNG.
urng_seed6[31:0]	Input	The third 32 bit word that is used by the second URNG.
awgn_out[15:0]	Output	16 bit normally distributed random number.

Core Parameters

The core doesn't have any generic parameter.

Core Resource Utilization:

The AWGN core uses 5 BRAMS to store the coefficients of the piecewise polynomial for the different functions in the Box-Muller algorithm. The core also uses 13 DSP48 for multiplication and addition of inputs of varying sizes. The placement is shown in figure 3.

Design Verification:

A testbench was written in verilog to not only verify the output but also to analyse the deviation from the 'golden' MATLAB model. Initially the MATLAB verification program is run and the intermediate golden values given in figure 2 are stored in a text file in the hexadecimal format. The verilog testbench reads the files containing the golden values and on each clock cycle after a sufficiently long warmup period, compares the instantaneous values with the golden values. If the instantaneous values of the intermediate signals deviate from the golden values by the theoretical error values obtained during error analysis, an error counter is incremented. This tells us whether the output is accurate to 1 ulp and if not then which operation are the contributors to the error.

Hardware Implementation

The core was designed using structural verilog targeting a Vertex 7 device mirroring a bit-true simulink model . The design uses many of the advanced features on the Vertex 7 architecture. For example the DSP48 have highly optimized multipliers and adders and so are used for the computation of the piecewise polynomial. All the ROM's used are implemented on the inbuilt block

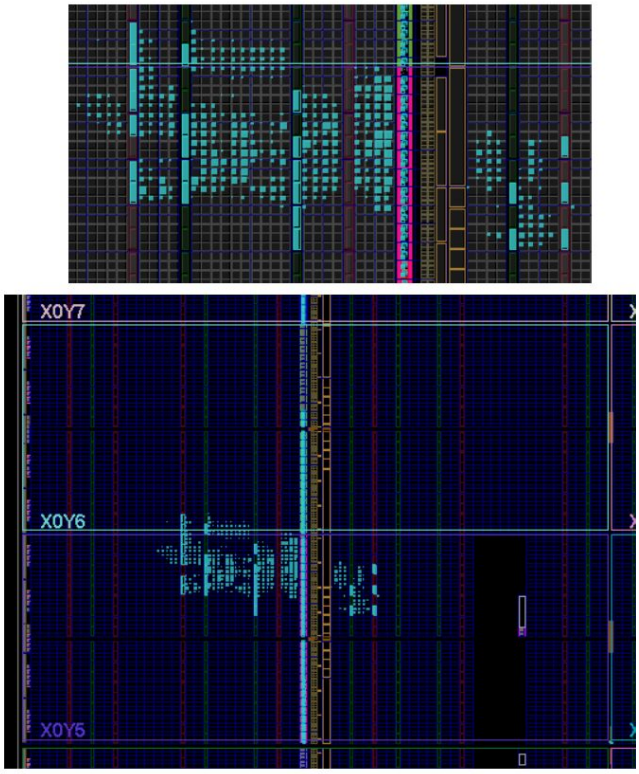

Fig 3 Placement of core with a close up screenshot

When reset is pulled down , the core starts to produce a value every clock cycle after a latency of 16 clock cycles.

References:

[1] DU.lee, "A Hardware Gaussian Noise Generator Using the Box-Muller Method and its Error Analysis", IEEE transactions on Computers, June 2006.

[2]P. L'Ecuyer, "Maximally Equidistributed Combined Tausworthe Generators", Math. Computation, vol. 65, no.213, pp.203-213, 1996

RAMS and distributed RAMS on the FPGA. Use the core, the 6 keys must be present when reset is high