AWGN IP Core Design

Rahul Katta

Features:

- Based on Box – Muller algorithm to generate Gaussian distributed noise.
- Produces two highly accurate 16-bit noise samples with bit errors as low as 10^{-12}.
- Quantized noise output with 5 integer bits and 11 fractional bits.
- Fractional bit optimization based on intensive error analysis.

Applications:

In this day and age, communication systems performance plays a critical role in various domains such as marketing, finance, search and rescue, deep-space, telecommunications, etc. Hence to ensure high quality communication, systems are modulated with AWGN to analyze the bit error performance. Therefore, the output of this IP core design can be used for evaluation purposes of communication systems.

AWGN IP Core Facts					
Core Specifics					
Supported Device Family	Xilinx Artix -7 AC701, Altera Cyclone V				
Resources Used by Xilinx	I/O	LUTs	FF/Ls	MUX	DSP
	211	963	32	24	7
Resources Used by Altera	I/O	LUTs	FF/Ls	MUX	DSP
	226	1277	32	23	14
Design Tool Requirements					
Verification	MatlabR2015aSP1				
Simulation	ModelSim – Altera 10.4b				
Synthesis	Vivado 2016.2, Quartus Prime 15.1				

Functional Description:

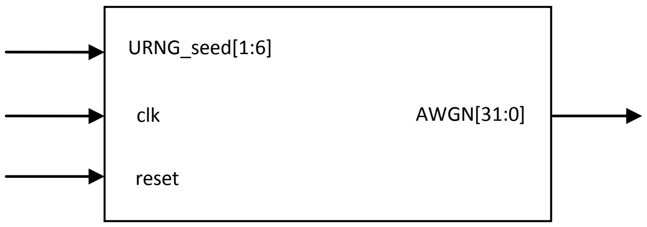

Fig: Core Schematic Symbol

The Box Muller method utilizes two Tausworthe's uniform random noise generators (URNG) with three input seeds each, which generates two 32-bit uniform random numbers. The URNG output is given to three functional blocks – Sin/Cos block, Log block and Square Root block; undergoing polynomial transformation to produce intermediate signals whose product produces two Gaussian-distributed noise samples.

Pinout:

Name	Direction	Description
URNG_seed[1:6]	Input	32-bit input values for Tausworthe's URNG's
Clk	Input	Clock that triggers the sequential elements of the RTL design
Reset	Input	Resets the URNG's to initial values
AWGN[31:0]	Output	Output produces two highly accurate 16-bit noise samples following the Gaussian distribution

Architecture:

The Box Muller architecture is illustrated in the below figure.

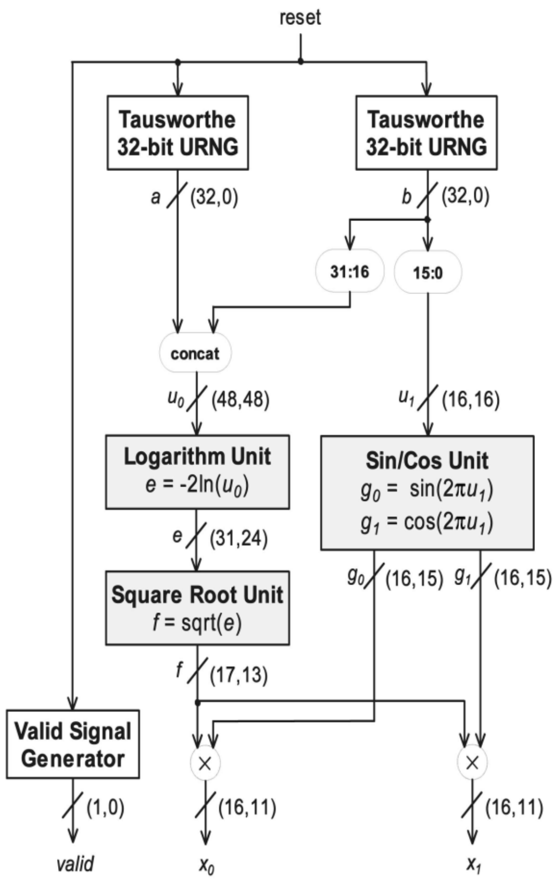

Fig: Process flow involved in the Box Muller method.

In each of the functional blocks i.e, Sin/Cos, Log and Square Root, the incoming signal undergoes polynomial transformation through a series of multiplications and additions with coefficient values stored in tables (i.e, ROM) producing quantized intermediate signals whose product generates a Gaussian distributed random signal.

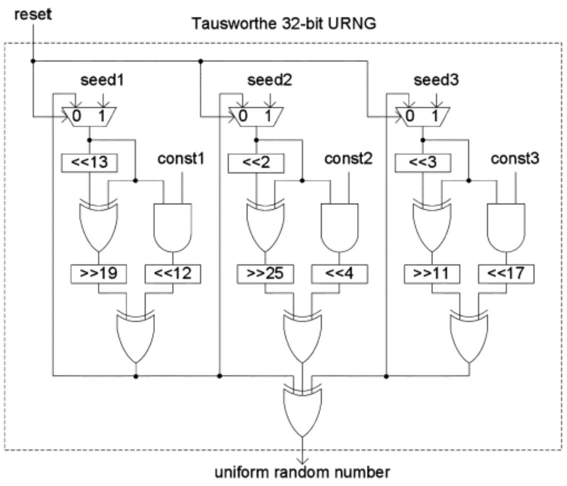

Fig: Architecture of Tausworthe 32-bit URNG

References

[1] Dong-U Lee, John D. Villasenor, Wayne Luk, and Philip H.W. Leong, "*A Hardware Gaussian Noise Generator Using the Box-Muller Method and Its Error Analysis*"

[2] http://www.asic-world.com/verilog

[3] http://stackoverflow.com/questions/tagged/matlab