Additive White Gaussian Noise (AWGN) Core v1.0

DS210 October 30, 2002 **Product Specification**

Features

- Designed for Virtex™-II and Virtex-II Pro™ using structural VHDL
- Probability density function (PDF) deviates less than 0.2 percent from the Gaussian PDF for $|x| < 4.8\sigma$ and is obtained from a closed-form expression
- Based on the Box-Muller algorithm and the central limit theorem as described in [1]
- Period of generated noise sequence is ~ $2^{190} = 1.57\times10^{57}$ samples
- Power spectral density is flat
- SNR input ranges from 0.0 to 15.9 dB in steps of 0.1 dB and provides scaling to obtain desired variance
- Noise is quantized to 16 bits with 5 bits of integer and 11 bits of fraction
- 760-bit internal seed selectable through top-level generics
- Core returns to its initial state upon reset
- Bit-true Simulink model and MATLAB programs included
- Uses relationally placed macro (RPM) mapping and placement technology, for maximum and predictable performance
- Requires 480 slices (40 rows, 12 columns), five block RAMs and five hardware multipliers
- Maximum clock rate and output sample rate of 245 MHz in Virtex-II
- Maximum clock rate and output sample rate of 300 MHz in Virtex-II Pro

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 (BER) performance of a communication system.

LogiCORE™ Facts

Core Specifics				
Supported Device Family	Virtex-II, Virtex-II Pro			
Resources Used	**I/O**	**LUTs**	**FFs**	**Block RAMs**
	28	733	902	5
Special Features	RPM Core			
Provided with Core				
Documentation	Product Specification			
Design File Formats	VHDL			
Constraints File	UCF			
Verification	MATLAB + ModelSim + Hardware			
Instantiation Template	VHDL Wrapper			
Reference Designs & application notes	BER Measurement of Uncoded BPSK in Hardware			
Additional Items	Bit-True Simulink Model; MATLAB Analysis programs			
Design Tool Requirements				
Xilinx Implementation Tools	ISE 4.2.03i or later			
Verification	MATLAB R12			
Simulation	ModelSim PE 5.4e			
Synthesis	Synplify Pro 7.1			
Support				
Support provided by Xilinx, Inc.				

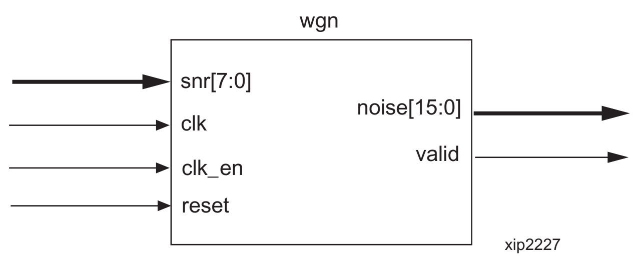

Figure 1: **Core Schematic Symbol**

Functional Description

The AWGN core generates white Gaussian noise using a combination of the Box-Muller algorithm and the central limit theorem, following the general approach described in [1]. The Box-Muller algorithm generates a unit normal random variable via a transformation of two independent random variables that are uniformly distributed over [0,1]. The uniform random variables internal to the core are generated using multiple-bit leap-forward LFSRs [2]. The outputs of the LFSRs are used to index the contents of ROMs that store the function values used in the Box-Muller algorithm. The outputs of multiple parallel Box-Muller designs are then averaged to obtain a PDF that is Gaussian to within 0.2% out to 4.8σ. Finally, scaling is performed based on the value of the SNR input to achieve the desired noise variance.

The SNR input is defined as (E_b/N_0) in dB for uncoded BPSK with unit symbol energy (E_s = 1). The valid range for the SNR input is from 0.0 to 15.9 in steps of 0.1 dB. The SNR is quantized to 8 bits with 4 bits of integer (valid range is 0 to 15) and 4 bits for a decimal fraction (valid range is 0 to 9). For example, entering an SNR of 5.9 dB is accomplished by setting the 4 MSBs equal to "0101" and the 4 LSBS equal to "1001."

To use the AWGN core in a system with coding and/or to use the AWGN core with different modulation formats, it is necessary to adjust the SNR value to accommodate the difference in spectral efficiency. For example, if we have BPSK modulation with rate ½ coding and keep E_s = 1 and N_0 constant, then E_b = 2 and E_b/N_0 = SNR + 3 dB. If we have uncoded QPSK modulation with I = +/-1 and Q = +/-1 and add independent noise sequences, then each channel looks like an independent BPSK channel and the E_b/N_0 = SNR. If we then add rate ½ coding to the QPSK case, we have E_b/N_0 = SNR + 3 dB.

In order to achieve an E_b/N_0 outside of the built-in dynamic range of the AWGN core, it may be necessary to scale the signal by an additional factor of 2. For example, to obtain an E_b/N_0 of 0 dB for a rate ½ QPSK system, one can simply perform a right-shift of the signal data by one bit before adding the noise. Equivalently, one can perform a left-shift of the noise data before adding the signal. This factor of 2 in voltage results in a -6 dB change in E_b/N_0. So, obtaining an E_b/N_0 of 0 dB for rate ½ QPSK is as simple as performing a left-shift of the noise by one bit and setting the SNR equal to 3 dB.

Pinout

Signal names for the core are shown in Figure 1 and described in Table 1.

Table 1: **Core Signal Pinout**

Name	Direction	Description
snr[7:0]	Input	Signal-to-Noise Ratio in dB assuming uncoded BPSK snr[7:4] = 0-15 specifies integer portion snr[3:0] = 0-9 specifies decimal fraction
clk	Input	Clock that triggers all sequential elements on its rising edge
clk_en	Input	Active high clock enable
reset	Input	Active high synchronous reset
noise[15:0]	Output	White Gaussian noise output
valid	Output	High when noise is valid

Core Architecture

A complete description of the core architecture is provided upon purchase of the core. This includes an overview of the design algorithm, a description of the analytical measures of performance and a description of the bit-true Simulink model.

Core Parameters

The core has seven generic parameters: Target, bxloc, byloc, init1, init2, init3, and init4. The target parameter must be set to Virtex-II. The bxloc and byloc parameters control the placement of the BRAM and MULT18X18s used in the core. Because the design is an RPM, the placement of the slice elements are controlled through the use of either an RLOC or RLOC_ORIGIN attribute on the core. The (bxloc, byloc) pair have the same functionality as an RLOC_ORIGIN attribute, but applied to the BRAM and MULT18X18 blocks.

The init1, init2, init3, and init4 generics are used to specify the initial state of the core. One of the four generics controls one of the four Box-Muller designs. Each LFSR in a given Box-Muller design is initialized to a subvector of the top-level generic.

Core Resource Utilization

The core requires an area of 40 slices tall by 12 slices wide and utilizes 5 BRAMs and 5 MULT18X18s. The layout of the core as viewed in the Xilinx Floorplanner is shown in Figure 3. Note that only 455 slices of the 480 slice rectangle are utilized, leaving 25 slices available for additional logic.

Hardware Implementation

The core was designed using structural VHDL targeting a Virtex-II device and mirrors the architecture of the bit-true Simulink model. The code has embedded relative location constraint (RLOC) attributes for the purpose of creating a relatively placed macro (RPM). The RPM can be relocated to various places on the die with minimal impact to the core's performance.

The design utilizes many of the advanced features of the Virtex-II architecture. For example, the LFSRs contained in the core are all based on SRL16s. The multiplication required in the Box-Muller algorithm and the SNR scaling utilize the embedded hardware multiplier. Many of the ROMs used in the core are implemented with block RAM; others utilize the Virtex-II distributed RAM capability.

Initialization of the core is accomplished in two ways. First, top-level generics are used to specify the initial states of all LFSRs in the core. The LFSRs are programmed during configuration so that the core is initialized immediately to the desired state. After configuration, it is possible to return the core to its initial state by activating the synchronous reset. During this second type of initialization, the LFSRs in each Box-Muller subdesign are chained together to form a single 190-bit shift register. The contents of the 190-bit shift register are then filled with an initialization chain that is read from a BRAM-based ROM. The reset-based initialization of the core requires 194 clock cycles from the time the reset is released to the time the initial state is valid. There is a latency of 14 clocks from the time clk_en goes High to the time the valid output goes High. So, with clk_en tied High the latency from reset to valid is 208 clocks.

Performance Characteristics

Using the Xilinx static timing analysis tool (TRACE), the core achieves a clock rate of 245 MHz in an XC2V1000-6 device (ADVANCED 1.111 2002-07-11 speed files) and a clock rate of 300 MHz in an XC2VP20-7 device (ADVANCED 1.62 2002-06-12 speed files). These speeds were obtained using a single instance of the core with an RLOC_ORIGIN of X0Y0 and BRAM+MULT18X18 placement as shown in Figure 3.

Design Verification

The core was tested in the lab using a Virtex-II prototyping board populated with an XC2V3000-6 FG676 device. A hardware testbench was created to test the core for uncoded BPSK in an AWGN channel. The testbench consists of an LFSR-based data generator, the AWGN core, an adder, a bit counter, and an error counter. The hardware testbench was also designed using RPM techniques to allow at-speed testing of the core. Additional routing congestion, however, caused the speed of the core to decrease from 245 MHz to 243 MHz when embedded in the hardware testbench compared with being instantiated alone.

Table 2 lists the measurements taken while running the core in hardware at speed (243 MHz). These results are plotted in Figure 2. In terms of SNR, the measured data set has a maximum deviation of only 0.03 dB from expected, confirming that the core is highly accurate. Note that it took only 77 minutes to generate the point at 1.29×10^{-10} BER, illustrating the astronomical advantage of running at-speed in hardware over traditional simulation methods.

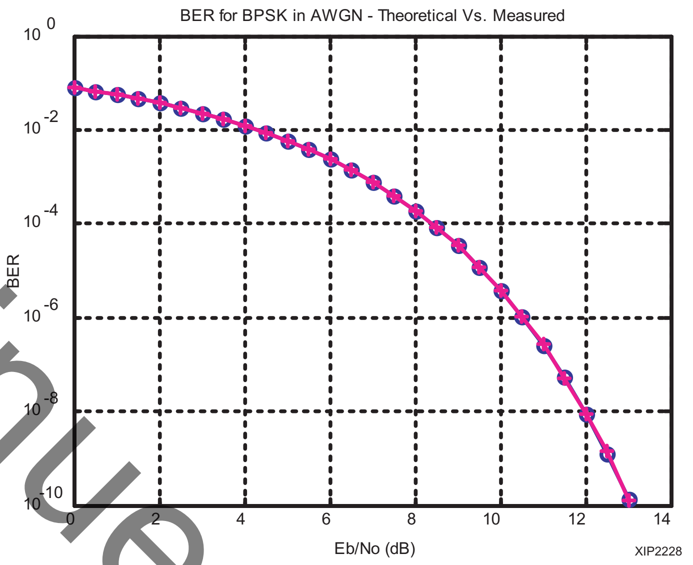

Figure 2: **Plot of Measured Vs. Theoretical BER for the Uncoded BPSK Test**

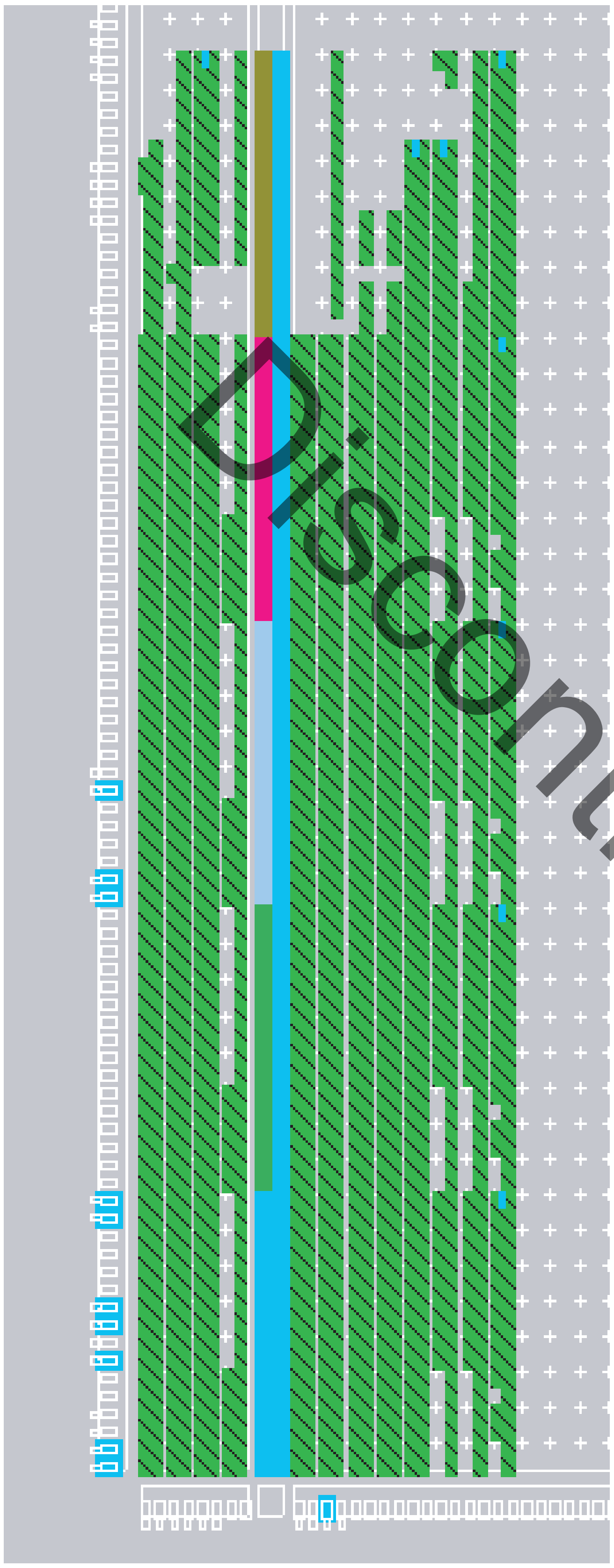

Figure 3: **Core Layout in Xilinx Floorplanner**

Table 2: **BER Measurements for the Uncoded BPSK Test**

SNR (dB)	Log_2 (Bit Count)	Error Count	BER
0.0	18	513D	7.93e-002
0.5	18	44B8	6.71e-002
1.0	19	72C4	5.60e-002
1.5	19	5ECA	4.63e-002
2.0	20	984F	3.72e-002
2.5	20	797D	2.97e-002
3.0	21	BA5B	2.27e-002
3.5	21	8CCC	1.72e-002
4.0	22	CBC1	1.24e-002
4.5	22	8E00	8.67e-003
5.0	23	C0BA	5.88e-003
5.5	23	7C92	3.80e-003
6.0	24	9B73	2.37e-003
6.5	24	5C17	1.41e-003
7.0	25	6469	7.66e-004
7.5	25	3404	3.97e-004
8.0	27	648F	1.92e-004
8.5	27	2C45	8.44e-005
9.0	28	237A	3.38e-005
9.5	28	0C89	1.20e-005
10.0	31	2068	3.86e-006
10.5	31	08F3	1.07e-006
11.0	32	048C	2.71e-007
11.5	34	034C	4.91e-008
12.0	37	04DB	9.04e-009
12.5	38	017D	1.39e-009
13.0	40	008E	1.29e-010

References

1. A. Ghazel, E. Boutillon, J. L. Danger, G. Gulak and H. Laamari, “Design and Performance Analysis of a High Speed AWGN Communication Channel Emulator,” IEEE PACRIM Conference, Victoria, B. C., August 2001.

2. P. Chu and R. Jones, “Design Techniques of FPGA-Based Random Number Generator,” Military and Aerospace Applications of Programmable Devices and Technologies Conference, 1999.

Ordering Information

This Additive White Gaussian Noise LogiCORE product, sold as source code, is provided under the LogiCORE Source License Agreement. A free evaluation form is available from Xilinx DSP marketing or through your Xilinx sales representative.

For part number information, please refer to the product page for this core on the Xilinx IP Center, http://www.xilinx.com/ipcenter/index.htm. To purchase this core, contact your local Xilinx sales representative.

Information on additional Xilinx LogiCORE modules is available on the Xilinx IP Center.