ASIC IMPLEMENTATION OF A HIGH SPEED WGNG FOR COMMUNICATION CHANNEL EMULATION

Edmund Fung, Kaston Leung, Nitin Parimi, Madhura Purnaprajna, Vincent C. Gaudet

Department of Electrical and Computer Engineering,
University of Alberta
ECERF, 9107-116 Street, Edmonton, Alberta, Canada, T6G 2V4.
{eyfung, kaston, parimi, madhura, vgaudet}@ece.ualberta.ca

ABSTRACT

A design for a White Gaussian Noise Generator (WGNG) is modified and implemented as a 0.18-µm CMOS digital ASIC for high-speed communication channel emulation. The original design was implemented using an FPGA. The goal of the work presented is to enhance the performance of the WGNG in order to achieve emulation of high-speed communication standards unattainable by the FPGA implementation. This is accomplished by pipelining the original design and implementing it using an ASIC. A layout is generated based on a standard digital design flow provided by Canadian Microelectronics Corporation (CMC). This implementation achieves an output rate of 182 Msamples/sec, which exceeds the speed of the original FPGA implementation by more than seven times.

1. INTRODUCTION

In order to simulate a communication system, it is necessary to emulate three key components: the transmitter, the channel, and the receiver. Emulation of such a system can be done using either a software or hardware approach. However, due to the high number of parameters inherent in modern communication systems, and the number of iterations required for a Monte-Carlo simulation with an accurate Bit Error Rate, the software emulation approach can be very time consuming [1]. As such, it is desirable to use hardware emulation of the communication system in order to speed up the simulation and design process.

This paper deals with implementation of a communication channel emulator and tester. The type of channel chosen is the Additive White Gaussian Noise (AWGN) channel. To emulate this kind of channel, a White Gaussian Noise Generator (WGNG) is required. Figure 1 illustrates the role that a WGNG plays in the simulation of a complete communication system. The purpose of the WGNG is to simulate the effect of noise on digital data as it is transmitted from sender to receiver. Four well-known methods of generating random Gaussian variables are described in [2]. A research group composed of collaborators from ENST-Paris (France), SUP'COM (Tunisia), LESTER (France), and the University of Toronto (Canada) designed a WGNG based on the central limit theorem and the Box-Muller method [3].

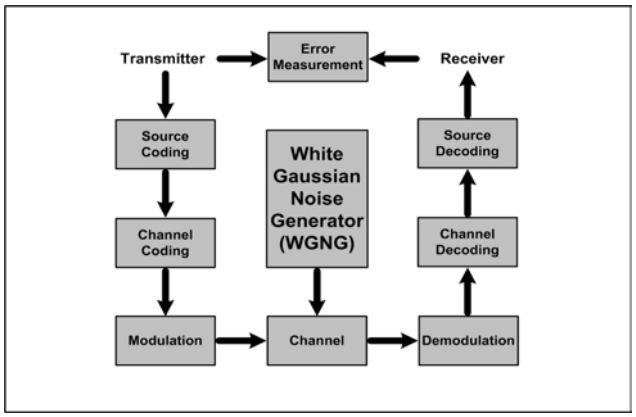

Figure 1: Simulation and Testing of a Communication System

The WGNG was optimized and implemented using an Altera FLEX10K100EQC240-1 FPGA. Using this implementation, a maximum clock rate of 98 MHz and a maximum output rate of 24.5 Msamples/sec were achieved [3]. In general, the speed attainable by realizing a given design using an ASIC is higher than that of an FPGA. Table 1 details the throughputs of various network communication standards. As evidenced by this table, the output rate achieved using the aforementioned FPGA implementation can only support at-speed channel emulation and testing for a limited number of available standards.

In order to attain superior clock and output rates, we have made modifications to the original design and implemented it using a digital ASIC. This enables channel emulation for a wider range of communication standards than what is possible with an FPGA implementation.

Throughput	Communication Standard
2Mbps	3G wireless[15]
10 Mbps	10Base-T Ethernet[11]
11 Mbps	IEEE 802.11b wireless Wi-Fi (2.4 GHz band)[13]
12 Mbps	Universal Serial Bus (USB)[7]
25.6-155.52 Mbps	Asynchronous Transfer Mode (ATM)[9],[10]
54 Mbps	IEEE 802.11a wireless WLAN (5 G Hz band)[14]
54 Mbps / 11 Mbps	IEEE 802.11g wireless WLAN (2.4 GHz band)[12]
100 Mbps	100Base-T Ethernet (Fast Ethernet)[11]
400 Mbps	FireWire (IEEE 1394)[8]
480 Mbps	USB 2.0 [7]

Table 1: Network communication speeds

Our paper is organized as follows. Section 2 describes the architecture of the WGNG design used. Section 3 presents the modifications made to the original design in order to enhance performance, and explains the design flow used to obtain an ASIC layout from VHDL code. In Section 4, the performance of various WGNG implementations are analyzed and compared. Section 5 describes the provisions made for testing. Finally, Section 6 summarizes our accomplishments and states future work to be done.

2. WGNG ARCHITECTURE

The WGNG design models noise that affects data transmitted between the sender and receiver in a communication system. This noise is simulated by generating a random variable that has a Gaussian distribution [3]. A block diagram of the architecture of the WGNG design is shown in Figure 2.

Various hardware implementations of random number generators are presented in [4]. The WGNG design described in this paper is based on the multiple-bit LFSR generator.

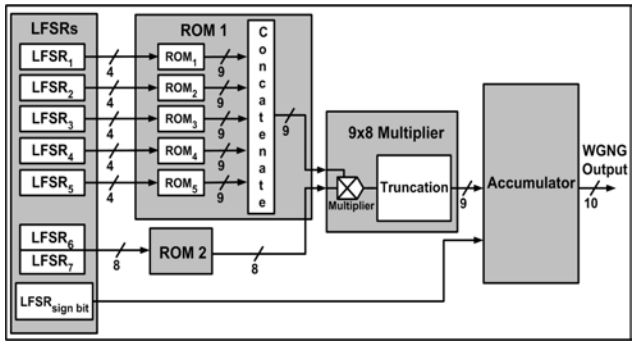

Figure 2: Architecture of the WGNG

The first stage required in the WGNG is a group of *linear feedback shift registers* (LFSR) that generates a 29-bit output. Traditionally, this is accomplished by using 29 LFSRs, each of which has an architecture similar to the one shown in Figure 3. Each LFSR is composed of a chain of D flip-flops (DFF) where the output of the last DFF in the chain feeds the input of the first DFF. XOR gates are placed along the chain of DFFs. The outputs of these XOR gates are connected to the inputs of certain flip-flops. The two inputs of each XOR gate are supplied by the output of the previous DFF and the output of the last DFF in the chain. The location of these XOR gates is determined by the generator polynomial of the LFSR. The XOR gates cause the outputs of the DFFs to change on every clock cycle in a pseudorandom manner, thus making the LFSR a pseudorandom sequence generator.

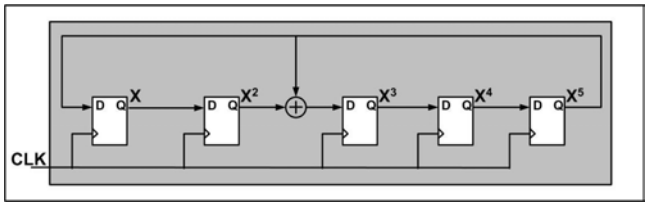

Figure 3: LFSR with Single-bit Output

However, the single-bit LFSR in Figure 3 can be reorganized so that it generates a 4-bit output, as shown in Figure 4. This greatly reduces the amount of hardware required to generate pseudorandom patterns [3].

The 29-bit output required from the first stage of the WGNG is generated by eight LFSRs. Seven of these produce 4-bit outputs and one produces a 1-bit output. 28 bits of the 29-bit output are used to feed two ROM lookup tables, which require 8-bit and 20-bit inputs. ROM 1 in Figure 2 uses the 20-bit input to generate a 9-bit output as defined in the following function [3]:

$$f(x_1) = \sqrt{-\ln(x_1)} \quad (1)$$

ROM 2 in Figure 2 uses the 8-bit input to generate an 8-bit output as defined by the following function [3]:

$$g(x_2) = \sqrt{2}\cos(2\pi x_2) \quad (2)$$

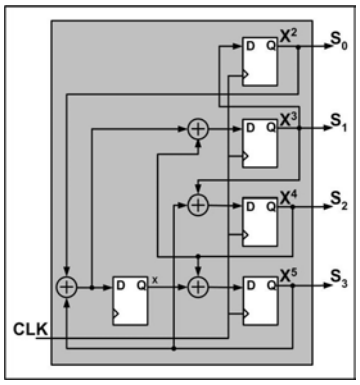

Figure 4: LFSR with 4-bit Output

Pre-computed values of functions (1) and (2) are stored in ROM 1 and ROM 2 respectively. The two outputs of the ROMs are then multiplied and the most significant 10 bits of the resulting 17-bit product are fed to an accumulator. The remaining bit from the 29 bits generated by the LFSRs is also fed to the accumulator. The accumulator sums every 4 samples generated by the multiplier and outputs a 10-bit binary number. Thus, since the multiplier generates a product once per clock cycle, the WGNG generates a 10-bit pseudorandom number once for every four clock cycles.

3. IMPLEMENTATION PROCESS

3.1. Design Modifications

The original design of the WGNG is specified in VHDL code and is provided by the aforementioned research group. In addition to using a faster implementation, the performance of the WGNG is further increased by making modifications to the original design.

Upon analysis of the WGNG architecture, it was realized that applying pipelining to portions of the design involving combinational logic could increase the output rate. Pipelining involves breaking a large block of combinational logic into smaller blocks and placing memory elements between them. This serves to increase the throughput of the entire block of combinational logic. Pipelining was applied to the multiplier in the WGNG design in order to increase the output rate of the entire design.

The multiplier is implemented using the Synopsys DesignWare DW02 5-stage multiplier [5]. Figure 5 illustrates the basic structure of this multiplier. With the use of this pipelined multiplier, the achievable data output rate of the WGNG is greatly increased compared to that of the original design.

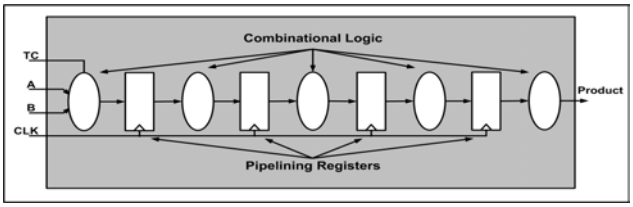

Figure 5: Block Diagram of 5-stage Pipelined Multiplier

3.2. Design Flow

Using a design flow provided by the *Canadian Microelectronics Corporation* (CMC) for a TSMC six metal layer 0.18-µm CMOS technology [6], the design is converted from VHDL code to an ASIC layout. The design kit provided by CMC includes the necessary standard cell libraries for our implementation.

The implementation strategy is primarily composed of the following seven major stages: synthesis, simulation, scan insertion, floorplanning, placement, routing, and *Layout Versus Schematic* (LVS) and *Design Rule Check* (DRC) verification.

Constraints for the design such as output load-capacitance, clock period, clock skew, and scan style are defined in Synopsys Design Analyzer. For both the original and modified designs, clock constraints are set to meet a target clock rate of 500 MHz.

The Cadence Physical Design Planner (PDP) is used for floor planning and optimized placement of the physical layout of the design. Routing is carried out using the Cadence Silicon Ensemble tool. The *Cadence Clock Tree Generation* (CTGen) tool is used to synthesize a clock tree to implement a single-phase clock.

Once timing goals are satisfied, the design is exported into the Cadence *Design Framework II* (DFII) environment.

The placed and routed version of the design is verified to ensure that it contains the same instances, nets, and connectivity as the original netlist using the LVS and

DRC capabilities in Cadence DFII. This is a good indication that the fabricated design will function properly.

The functional validation for the behavioral, gate-level and the post-placed netlist is done using Mentor Graphics ModelSim and Cadence NCVerilog simulation tools.

4. PERFORMANCE ANALYSIS

Two circuit layouts are generated using the aforementioned digital design flow. One layout is produced for the implementation of the original unaltered design, and another for the design that is modified to include the 5-stage pipeline multiplier in order to improve the output rate. A plot of the layout for the latter is shown in Figure 6.

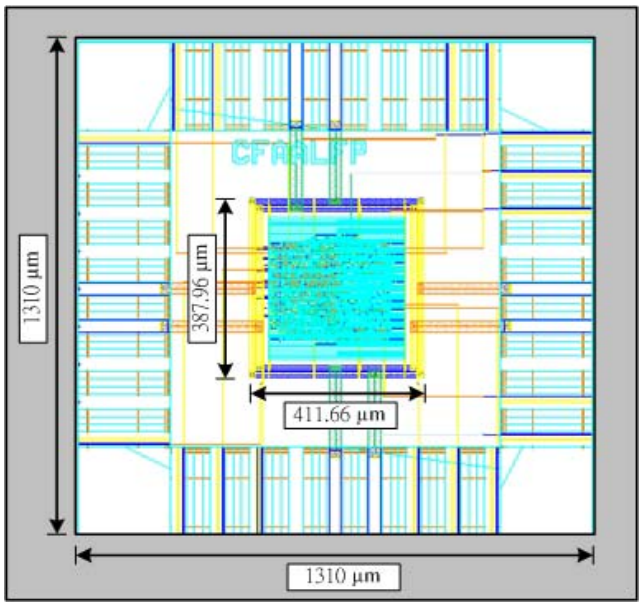

Figure 6: ASIC Layout

Various performance parameters are then obtained for each of these two designs. These are listed in Table 2. For comparison, the maximum clock rate and maximum output rate of the FPGA implementation are also listed. The total area and core area of the layouts are obtained from Cadence DFII. The critical path delay is found to be 1.37 ns, which is 31.5% lower than our target clock period of 2 ns, indicating that maximum clock speed is approximately 730 MHz. This exceeds our original target clock speed of 500 MHz. This is verified by simulating the netlist, created upon completion of the routing for the design at this clock rate, and ensuring that no timing violations from Synopsys Design Analyzer are encountered. Simulation waveforms verify that the WGNG generates one output for every four clock cycles as expected.

Dynamic power consumption is the power consumed as a result of switching in the circuit. Cell leakage power consumption is the power that is consumed due to the leakage current of the CMOS gates in the static phase. Both of these are determined by analyzing the report files generated using Synopsys Design Analyzer.

	Altera FPGA	**IC Implementation of Original Design**	**IC Implementation of Modified Design with Pipelined Multiplier**
Total Area Occupied	-	1.232×10^6 µm^2	1.716×10^6 µm^2
Core Area	-	1.328×10^5 µm^2	1.597×10^5 µm^2
Maximum clock rate	98 MHz	125 MHz	730 MHz
Maximum output rate	24.5 Msamples/s	31.25 Msamples/s	182 Msamples/s
Dynamic Power Consumption	-	32.4050 mW	54.8010 mW
Cell Leakage Power Consumption	-	1.9457 µW	2.8225 µW

Table 2: Performance Comparison

The performance of the two IC implementations is highlighted by the significant increase in output rate over the FPGA implementation. As shown in Table 2, the output rate achieved by the IC implementation of the modified design is more than seven times that of the FPGA implementation of the original design. Comparison of the output rates in Table 2 with the theoretical throughputs in Table 1 show that this implementation is able to support channel emulation for a wide range of communication standards. The achievable output rate of 182 Msamples/s can support the ATM, IEEE 802.11a/g, baseband and Fast Ethernet standards, which cannot be supported using the FPGA implementation. These results offer a positive indication that further optimized ICs can provide channel emulation

for next generation communication standards that FPGA implementations cannot.

The enhanced performance of the modified design over the original design comes at the expense of increased core area and power consumption.

The random binary numbers generated by the modified WGNG design are plotted in a histogram using MATLAB. This is shown in Figure 7. 24880 samples of the 10-bit numbers have been placed in 10000 equally spaced bins of the possible output range to produce the histogram. This plot illustrates the Gaussian distribution of the WGNG outputs, which satisfies the requirements of the desired communication channel emulator.

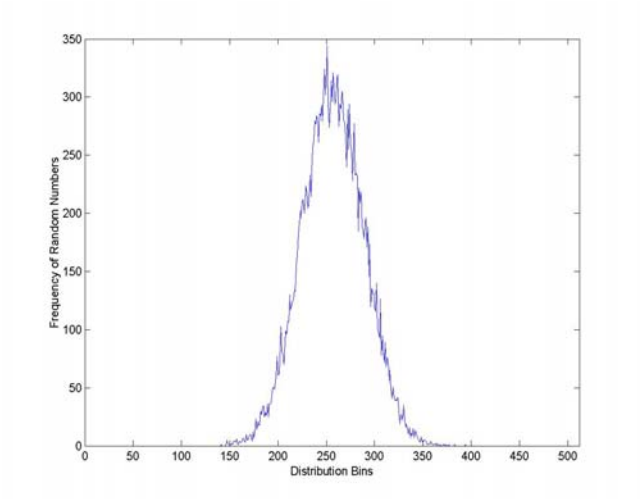

Figure 7: Histogram of WGNG Output

5. TESTING STRATEGY

A scan-based testing strategy is employed to detect faults initiated during the manufacture and packing stage. The Test Compiler tool from the Synopsys Design Analyzer tool-set replaces all DFFs with scannable DFFs. Subsequently, they are all connected together to form a scan chain. The Test Compiler then creates fault-detecting test vectors.

After scan insertion, functional test vectors are generated to serve two purposes. The first is to ensure that data can be shifted in and out of the scan chain as expected, and the second is to estimate fault coverage. The fault coverage estimated by the Test Compiler tool is 99%.

6. CONCLUSION

A WGNG has been implemented for the purpose of high-speed communication channel emulation. The WGNG design is implemented using an ASIC to obtain an increase in speed over a previous FPGA implementation. Modifications are subsequently made to the original WGNG design to include a 5-stage pipelined multiplier in order to increase output rate. The ASIC implementation of the modified design demonstrates a further significant improvement in output rate over the ASIC implementation of the original design. This improvement comes at the cost of additional die area and power consumption. However, the achieved output rate of 182 Msamples/second allows for channel emulation of several communication standards that cannot be supported by the FPGA and ASIC implementations of the original design.

The design could be further improved by implementing modifications that would increase the number of inputs to the accumulator. This would allow the WGNG to produce one output per clock cycle instead of one output per four clock cycles. This would likely require an increase in hardware, but would result in a higher maximum clock rate. Future work would also include fabrication of the design and testing for proper functionality to validate the design on silicon.

7. ACKNOWLEDGMENTS

The authors would like to thank Dr. Emmanuel Boutillon for providing us with the VHDL code of the original WGNG design, and CMC for providing access to the aforementioned CAD tools and technology.

8. REFERENCES

[1] J.L Danger, A. Ghazel, E. Boutillon H. Laamari, “Efficient FPGA Implementation of Gaussian Noise Generator for Communication Channel Emulation,” *The 7th IEEE International Conference on Electronicsm Circuits & Systemes (ICECS'2K)*, Kaslik, Lebanon, Dec 2000

[2] M. F. Schollmeyer and W. H. Tranter, “Noise Generators for the Simulation of Digital Communications Systems”, *Proc. 24th Ann. Simulation Symp.*, 1991.

[3] A. Ghazel, E. Boutillon, J. Danger, G. Gulak, H. Laamari, “Design and Performance analysis of high speed AWGN Communication channel Emulator,” *PACRIM'01*, Victoria, British Columbia, Canada, August 2001

[4] P. P. Chu and R. E. Jones, “Design Techniques of FPGA “Based Random Number Generator”, *Proc. Military and Aerospace Applications of Prog. Devices and Tech. Conf.*, 1999.

[5] Five-Stage Pipelined Multiplier DW02_mult_5_stage, *DesignWare Building Block IP* http://www.synopsys.com/products/designware/docs/doc/dwf/datasheets/dw02_mult_5_stage.pdf

[6] Tutorial on CMC's Digital IC Design Flow, Developed by Canadian Microelectronics Corporation, October 7th 2002

[7] Universal Serial Bus Specification, Compaq, Hewlett Packard, Intel, Lucent, Microsoft, NEC, Philips, Revision 2.0, April 2000

[8] IEEE Standard for High Performance Serial Bus, IEEE Standards Board, 1995

[9] Physical Layer Specification for 25.6 Mb/s Twisted pair Cable, The ATM Forum, af-phy-0040.00, November 1995

[10] ATM Physical Medium Dependent Interface Specification for 155 Mb/s over Twisted Pair Cable, The ATM Forum,af-phy-0015.000,September 1994

[11] 802.3 Part 3: Carrier sense multiple access with collision detection (CSMA/CD) access method and physical layer specifications, March 2002

[12] 802.11g Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications. Amendment 4:Further higher Data Rate Extension in 2.4 GHz Band, June 2003

[13] 802.11b Part 11:Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: Higher-Speed Physical Layer Extension in the 2.4 GHz Band, September 1999

[14] 802.11a Part 11:Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: High-Speed Physical Layer in 5 GHz Band, September 1999

[15] 3G Today – Technology, www.3gtoday.com/technology/