2019 32nd International Conference on VLSI Design and 2019 18th International Conference on Embedded Systems (VLSID)

Design and Implementation of Low-power High-throughput PRNGs for Security Applications

Bikram Paul* Apratim Khobragade** Javvaji Sai Soumith**
Sushree Sila P. Goswami* Sunil Dutt* Gaurav Trivedi*

*Department of Electronics and Electrical Engineering, Indian Institute of Technology Guwahati, (bikram, sushree, sunil.dutt, trivedi)@iitg.ac.in
**Department of Electronics and Communication Engineering, National Institute of Technology Tiruchirapalli, (apratim.khobragade, soumith98)@gmail.com

***Abstract*—Pseudo-Random Number Generators (PRNGs) are an integral part of cryptographic applications, such as key generations, digital signatures, *Internet-of-Things* (IoT) security, etc. These applications require low-power and high-throughput PRNGs along with statistically secure random numbers generation capability. In this paper, we propose two PRNG methods based on *Blum-Blum-Shub* (BBS), Xorshift and Permuted Congruential PRNGs. The first PRNG is preferred for general purpose applications while the second is preferred for low-power IoT applications. The proposed PRNG methods are implemented on Xilinx FPGA ZedBoard Zynq™−7000 and generate 4.83×10^7 and 4.29×10^7 random numbers per-second, respectively. The total dynamic power consumption of the proposed PRNGs is $17mW$ at $48.31Mhz$ and $16mW$ at $42.90Mhz$ with a maximum throughput of $184.288MBps$ and $163.651MBps$, respectively. The proposed PRNGs are tested on Diehard battery and US *National Institute of Standard and Technology* (NIST) SP $800-22$ suites for analyzing the randomness quality.**

***Keywords*-PRNG, FPGA, Cryptography, Blum-Blum-Shub, Xorshift, Permuted Congruential Generator.**

I. Introduction

Pseudo-Random Number Generators (PRNGs) are deterministic and reproducible algorithms which are used to generate the random number sequences for cryptographic applications, $e.g.$, key generation, digital signature, etc. Data and network security domains require PRNGs having various statistical properties, such as low autocorrelation values, secure, irreproachable and high linear span with non-repeatability. The key design challenge is to develop secure PRNGs which can generate random numbers with high throughput as well as can stand against malicious attacks. In recent years, several random number generators, especially hardware *True Random Number Generators* (TRNGs) [1] have been proposed for the cryptographic applications. As compared to PRNGs, TRNGs have good quality of randomness, but due to the lower speed and non-reproducible with unknown period, it is difficult to employ them in most of the cryptographic application. On the other hand, PRNGs are widely used in encryption schemes and security-related systems due to higher speed, less complexity and fewer hardware requirements.

In this paper, we propose two PRNGs (BluXor and *Modified PCG* (MPCG)) by combining and modifying *Blum-Blum-Shub* (BBS) [2], Xorshift [3] and *Permuted Congruential Generator* (PCG) [4] algorithms. Note that BBS, Xorshift and PCG are based on modular exponent, *Liner Feedback Shift Register* (LFSR) and *Linear Congruential Generator* (LCG) approaches, respectively. The key feature of the proposed PRNGs is that they are highly secure and exhibit low-power as well as high-throughput.

II. Implementation of BluXor and MPCG PRNG

BluXor and MPCG are designed and implemented using Verilog HDL and Xilinx Vivado on ZedBoard Zynq™−7000 FPGA platform to generate random numbers. The binary outputs generated by FPGA are converted to single precision floating point value according to $IEEE-754$ format.

A. Implementation of BluXor

One 32-bit BBS module and three 64-bit Xorshift modules are proposed to design *BluXor*, where BBS module is employed to generate the initial seed values and its output is fed to Xorshift modules. Initial seed value of BBS module is provided in the beginning, whereas in Xorshift modules, the initial values are fed from the BBS module. Based on the precondition, Xorshift modules right shift their outputs fixed number of times continuously in each iteration. BluXor takes a few clock cycles for initialization and then at every clock cycle, it produces 32-bit random number. The sequence predictability problem of Xorshift is resolved by feeding new seeds to Xorshift module through BBS. Five LSB bits of BBS act as switches for Xorshift outputs which are checked at every iteration to determine the next state of Xorshift. If these bits are set, then the higher 32-bit of Xorshift output becomes the new state, otherwise, the lower 32-bit of Xorshift output are considered as the new state. In this way, the limitations of BBS and Xorshift are resolved in BluXor.

B. Implementation of Modified PCG

MPCG algorithm employed tuple based permutation technique to generate more secure pseudo-random sequences of 32-bit length. MPCG involves several transformations (bitwise XOR and shift) and parameters to generate random numbers. The algorithm initiates its execution with a seed value (k_0). The first random number (k_1) is generated using (k_0); the next random number (k_2) is generated using (k_1) and this process continues till $i = n-1$, where n is the desired random number sequence. At the end of MPCG algorithm, a stream of $(n-1)$, 32-bit random numbers is generated. One important aspect of MPCG is that the initial seed value need to be good enough to generate statistically secure random numbers.

2380-6923/19/$31.00 ©2019 IEEE
DOI 10.1109/VLSID.2019.00123

Table I
HARDWARE IMPLEMENTATION RESULTS FOR THE PROPOSED PRNGS.

Parameters			Proposed PRNGs	
			BluXor	MPCG
Resources	LUTs		1267 (2.38%)	1371 (2.58%)
	Registers		730 (0.69%)	430 (0.40%)
	DSP Blocks		3 (2.38%)	none
Power	Dynamic	Logic	6 mW	7 mW
		Other	11 mW	9 mW
	Static		120 mW	120 mW
Delay	Logic Path		13.409 nS	12.974 nS
	Network Delay		7.291 nS	10.337 nS
	Critical Path		10.407 nS	11.7125 nS
Speed in Mhz			48.31 MhZ	42.90 MhZ
Throughput	Generation		$4.83 \times 10^7/sec$	$4.29 \times 10^7/sec$
	Memory		184.288 MBps	163.651 MBps

Table II
COMPARISON OF THE PROPOSED PRNGS WITH EXISTING PRNGS BASED ON DIEHARD BATTERY TEST.

Diehard Battery Tests	p-value (acceptable range $0.01 \leq p\text{-}value < 1$)			
	Existing PRNGs		Proposed PRNGs	
	AES(OFB)	*MT19937*	*BluXor*	*MPCG*
Birthdays	0.09141	0.83079	0.89915	0.64816
Operm5	0.31162	0.64989	0.37516	0.83809
Rank 32×32	0.71908	0.75294	0.96346	0.90656
Rank 6×8	0.07287	0.52856	0.44333	0.04604
Bitstream	0.14959	0.61691	0.98372	0.64737
DNA	0.79221	0.58120	0.38034	0.16069
Parking Lot	0.24201	0.96263	0.69672	0.06102
2-d Sphere	**0.99533***	0.31825	0.21653	0.36845
3-d Sphere	0.26868	0.55299	0.93923	0.99020
Squeeze	0.79345	0.87338	0.12832	0.82152
Diehard Sums	0.03846	0.01877	0.44894	**0.00340***
Diehard Runs	0.45078	0.13221	0.35698	0.59319
Marsaglia GCD	0.61111	0.04368	0.89999	0.36776
STS Monobit	0.08752	0.10336	0.12999	0.85297
STS Runs	0.45966	0.60370	0.97132	0.06399
Min. Distance	0.92905	0.90109	0.13589	0.24214
Permutations	0.24814	0.68669	0.01729	0.24297
DAB Monobit-2	0.37664	0.71992	0.24738	0.74744

Here, * signifies that a particular PRNG fails to pass a perticular test.

III. RESULTS

Table I shows the hardware implementation results of the proposed BluXor and MPCG. In order to evaluate and compare the randomness quality, the proposed PRNGs are tested using the standard Diehard battery and US *National Institute of Standard and Technology* (NIST) SP $800-22$ suites. Our test results are tabulated in Table II and Table III. As shown in Table II, in case of Diehard battery test (compared with *Advance Encryption System* (AES) [5] and *Mersenne Twister* (MT) [6]), BluXor passes all the 18 tests and MPCG passes 17 tests. Similar results are observed (see Table III) while performing the testes with NIST SP $800-22$ suite (compared with LCG [7], BBS [8] and Xorshift [8]) where both BluXor and MPCG passes all the 14 NIST tests.

IV. CONCLUSION

In this paper, we proposed two PRNGs: (i) BluXor by combining the BBS and Xorshift; and (ii) MPCG by modifying the conventional PCG. Our experimental results (on ZedBoard Zynq™-7000 FPGA) showed that BluXor generates 4.83×10^7 random numbers with $17mW$ at $48.31Mhz$ and MPCG generates 4.29×10^7 random numbers with $16mW$ at $42.90Mhz$. The maximum throughput of BluXor and MPCG is reported as $184.288MBps$ and $163.651MBps$, respectively. We also evaluated and compared the randomness quality with existing PRNGs using the standard Diehard battery and NIST SP $800-22$ suites. Our results showed that BluXor passes all the tests, whereas MPCG fails only in one Diehard battery test. Although MPCG fails marginally in one test, it does not show any significant decline in the security aspects. Consequently, MPCG can be used in low-power and area-constrained applications, such as IoT security applications. On the other hand, BluXor is recommended for the general purpose cryptographic applications.

Table III
COMPARISON OF THE PROPOSED PRNGS WITH EXISTING PRNGS BASED ON NIST $SP800-22$ TEST.

NIST $SP800-22$ Tests	p-value (acceptable range $0.01 \leq p\text{-}value < 1$)				
	Existing PRNGs			Proposed PRNGs	
	LCG	*BBS*	*Xorshift*	*BluXor*	*MPCG*
Frequency	0.59035	0.32435	0.14533	0.50650	0.10663
Block Frequency	0.49274	**0.00000***	0.02882	0.48455	0.04156
Cumulative Sums	0.52526	**0.00000***	0.07440	0.55293	0.06733
Runs	0.57225	**0.00000***	0.73992	0.48909	0.28754
Longest 1's	0.28407	**0.00000***	0.55442	0.44052	0.05784
Rank test	0.53949	**0.00000***	0.23681	0.44333	0.15037
DFT test	0.34448	**0.00000***	0.51412	0.46412	0.41735
Nonperiodic Template Matchings	0.49353	**0.00000***	0.51236	0.79626	0.01231
Overlapping Template Matchings	0.49444	**0.00000***	0.59550	0.55384	0.31532
Universal Stat.	0.53203	**0.00000***	0.12232	0.45476	0.24316
Rand. Excursions	0.05238	**0.00000***	0.50781	0.28161	0.41754
Random Excursions Variant	0.03655	**0.00000***	0.28959	0.39831	0.38923
Serial test	0.48996	0.04336	0.49585	0.49831	0.02652
Linear Complexity	0.27462	**0.00000***	0.24928	0.54371	0.21839

Here, * signifies that a particular PRNG fails to pass a perticular test.

REFERENCES

[1] C. Li, Q. Wang, J. Jiang, and N. Guan, "A metastability-based true random number generator on fpga," in *2017 IEEE 12th International Conference on ASIC*, Oct 2017, pp. 738–741.

[2] L. Blum, M. Blum, and M. Shub, "A simple unpredictable pseudo-random number generator," *SIAM Journal on Computing*, vol. 15, no. 2, pp. 364–383, 1986. [Online]. Available: https://doi.org/10.1137/0215025

[3] G. Marsaglia, "Xorshift rngs," *Journal of Statistical Software, Articles*, vol. 8, no. 14, pp. 1–6, 2003. [Online]. Available: https://www.jstatsoft.org/v008/i14

[4] R. S. Katti, R. G. Kavasseri, and V. Sai, "Pseudorandom bit generation using coupled congruential generators," *IEEE Transactions on Circuits and Systems II: Express Briefs*, vol. 57, no. 3, pp. 203–207, March 2010.

[5] AES_CFB Mode cipher feedback (2017, December). [Online]. Available: https://www.cryptopp.com/wiki/CFB_Mode

[6] Al-Khaffaf, H. S. M., A. Z. Talib, and R. Abdul Salam, "A study on the effects of noise level, cleaning method, and vectorization software on the quality of vector data." Springer, 2008, pp. 299–309.

[7] J. Boyar, "Inferring sequences produced by a linear congruential generator missing low-order bits," *Journal of Cryptology*, vol. 1, no. 3, pp. 177–184, Oct 1989.

[8] M. B. Jacques, F. Xiaole, G. Christophe, and L. Laurent, "Fpga design for PRNG based on chaotic iteration used in information hiding application," August 2016, pp. 1–24.