International Research Journal of Engineering and Technology (IRJET) e-ISSN: 2395 -0056
Volume: 02 Issue: 04 | July-2015 www.irjet.net p-ISSN: 2395-0072

Implementation of PLL based TRNG using Hardware Software Co-designing

Sushil C. Bagal[1], Vilas V. Deotare[2], Dinesh V. Padole[3], Swapnil C. Bagal[4]

[1] *PG Student, Electronics and Telecommunication, SIT Lonavala, Maharashtra, India*
[2] *Head of the Department, Electronics & Telecommunication, SIT, Lonavala, Maharashtra, India*
[3] *Professors, Electronics & Telecommunication, GHRCE, Nagpur Maharashtra, India*
[4] *UG Student, Mechanical engineering, COEP, Maharashtra, India*

Abstract: ***In hazardous environment, if attacker has enough computing power then it becomes difficult to use pseudo-random number in encryption algorithm. The solution to the problem is true random numbers which prevent attacker from predicting next random number sequence. This paper presents a True Random Number Generator [TRNG] using analog Phase-Locked Loop (PLL) on a digital Altera FPGA-based soft processor. It is built on the estimation of cost, power and performance metrics using hardware software co-designing. It optimizes the FPGA architecture to achieve mid - point result between hardware implementation and software implementation. TRNG embedded in SOPC is faster, cheaper and simpler than previous system and increases the security of the system in hostile environment.***

Key Words: *Pseudo-random number, Encryption algorithm, True Random Number Generator, Phase-Locked Loop, FPGA, Soft processor, Co-designing, SOPC*

1. INTRODUCTION

Mainly there are two types of random numbers, Pseudo-random numbers [PRN] and true random numbers [TRN]. There are numerous applications like Cryptography, key generation, statistics, as a password source, art, algorithms and protocols where random numbers are extensively used. Pseudo random number generation [PRNG] is deterministic approach and can be generated using computer and digital hardware. On the other hand true random number generation [TRNG] is nondeterministic approach which requires some real world phenomenon. Sometimes Pseudo-random numbers appears to be truly random and almost indistinguishable from true random numbers. This is due to 'seeding' PRN with TRN still PRNG has some sort of repetition in their pattern.

In hostile environment security is major concern and as randomness increases security increases. If unauthorized personnel have strong computing power then next PRN sequence can be forecasted. In order to prevent strongest advisory from predicting next sequence, TRN are used. The basic requirement for generating TRN is shown in figure 1.

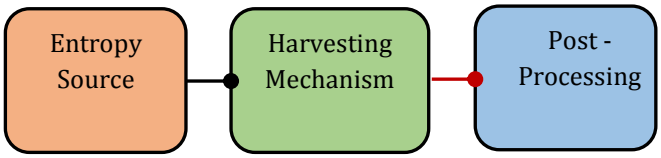

Fig -1: Block diagram of TRNG.

1.1 Entropy Source:

TRNG is non deterministic polynomial and hence generated from some real world and real time phenomenon such as Brownian motion, movement of computer mouse, nuclear decay, thermal and shot noise in circuits, etc. These all are known as entropy sources.

1.2 Harvesting Mechanism:

It is the process of collecting entropy as much as possible without affecting entropy source. This component is as important as entropy source.

1.3 Post - Processing:

It is an optional step. Imperfections at entropy level or harvesting level are removed at this stage in order to increase randomness and strengthen the design.

2. LITERATURE SERVEY

There are different methods proposed in literature to produce TRNG. In [1] detailed analysis of jitter characteristics using [4, 5, 6 and 7] is done. The jitters generated by Phase locked loop (PLL) as an analog component is sampled in order to produce TRNG. In [2] different methods of TRNG generation and there drawback are explained in detail. Discrete-time chaos method- problem- inaccuracies in circuit which limit A/D resolution, Oscillation sampling method- problem- jitter generated in oscillator are not sufficient enough to

produce desired randomness, Direct amplification method- problem- shielding problem between substrate signal and power supply. In [3] TRNGs are generated by random jitter generated by noise caused due to electronic components

3. Design principal of TRNG

Any undesired displacement of clock from its desired position is known as jitter. This jitters are used as source of randomness (Entropy source). The detailed description of fig.2 is as follows:

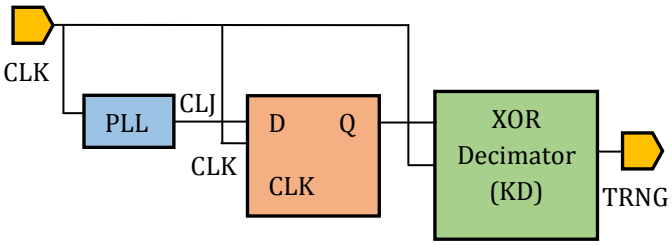

Fig -2: Design Principal of TRNG.

3.1 PLL: It generates jittery clock due to different reasons such as interference due to power, ground, internal noise of circuitry, thermal agitation of charge carriers etc. Jitters generated due to internal noise of VCO of PLL is of more interest as its non-deterministic source of jitter while other are deterministic source jitter.

3.2 D-flip flop: It samples the output clock (CLJ) generated by PLL with respect to reference clock (CLK). When both CLJ and CLK have same initial phase and frequency maximum entropy can be achieved on the contrary when CLJ and CLK have different initial phase and frequency minimum entropy is achieved.

3.3 XOR Decimator: this is the last step in the process of generating TRNG and mainly used for post processing to strengthening the design.

4. MATHEMATICAL ANALYSIS

PLL can be used for frequency synthesis. Equation-1 shows how CLJ is generated from CLK by proper selection of dividing K_D and multiplying factor K_M.

$$CLJ = CLK * K_M / K_D \quad (1)$$

The values of K_M and K_D are selected in such a way that, following conditions are satisfied:

GCD ($2 K_M, K_D$) =1 and
K_D = Must be an odd value. (2)

GCD is nothing but Greatest Common Divisor and above equation insures that the maximum guaranteed distance between the closest edges of CLJ and CLK (denoted by T_{JIT}).

$$T_{JIT} = [1/ F_{REF}]*[GCD (2 K_M, K_D) / (4* K_M)] \quad (3)$$

With proper selection of K_M, K_D and F_{REF} it's possible to satisfy below conditions:

Also $T_{JIT} \leq \sigma_{JIT}$
Where $\sigma_{JIT} \geq 15$ ps (4)

If all above conditions are satisfied the proposed method becomes insensitive to deterministic PLL jitter and completely depends on non-deterministic PLL jitter.

5. HARDWARE IMPLEMENTATION AND RESULTS

The proposed TRNG is realized in Altera cyclone-III development board by writing parameterized VHDL code for it in Altera Quartus-II software. The FPGA development board has four on chip PLLs out of which two are used to generate CLJ and CLK signals. Fig.3 shows the complete hardware set up of the TRNG.

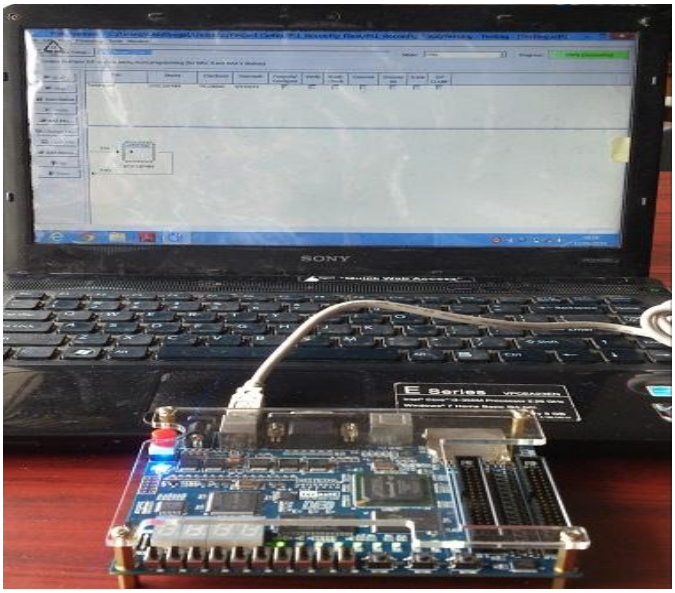

Fig -3: Hardware setup for TRNG.

The frequency of reference clock source is 80 MHz, K_M and K_D values are 700 and 1121 respectively. These values ensures that all above equations (From equation 1-4) get satisfied resulting in values $T_{JIT} \approx 4.46$ps $\leq \sigma_{JIT}$, Bit rate ≈ 71 kb/sec and Sensitivity ≈ $251*10^{12}$.

As compared to [1] and [3] bit rate has increased by approximately 2kb/sec and 61kb/sec. respectively. Fig.4 shows the resource utilization of proposed TRNG.

Flow Summary	
Flow Status	Successful - Sun Jun 21 09:54:58 2015
Quartus II Version	10.1 Build 153 11/29/2010 SJ Web Edition
Revision Name	Testing
Top-level Entity Name	ReconfigWithPllSignals
Family	Cyclone III
Device	EP3C16F484C6
Timing Models	Final
Total logic elements	768
Total combinational functions	756
Dedicated logic registers	436
Total registers	436
Total pins	17
Total virtual pins	0
Total memory bits	0
Embedded Multiplier 9-bit elements	0

Fig -4: Resource utilization of TRNG.

The advantage of hardware software co-designing (i.e. using on chip hardware PLL and reconfiguring it using Altera quartus-II software) can be well understood from fig.5.
It can be seen that software implementation is more cost efficient and flexible but results in less performance as compared to hardware and hardware software co-designing. While hardware implementation is more costly and result in highest performance but has less flexibility as compared to software and hardware software co-designing. In comparison with software and hardware implementation, an optimum result can be achieved with hardware software co-designing as per as cost, performance and flexibility is concern and that can be seen from fig. below:

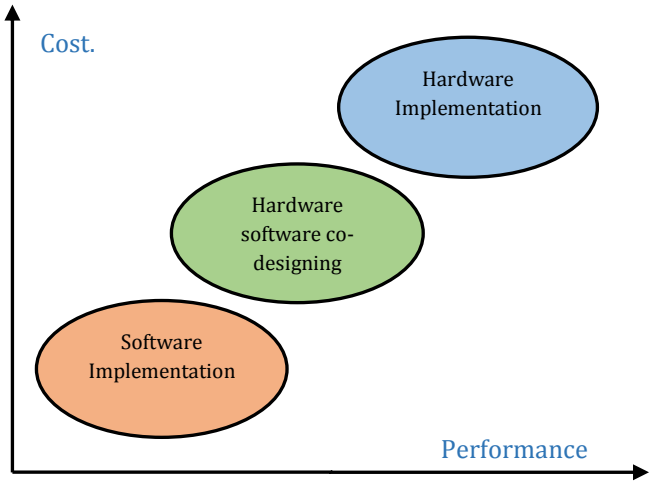

Fig -5: Comparison of software, hardware and hardware software co-designing implementation methods.

6. TESTING AND VALIDATION

There are many tests which can be used to check the quality of random numbers generated. One of the most promising test among them is NIST statistical test. It is developed by National Institute of Standards and Technology (NIST) U.S. and publically available. This test consists of one significance value (∝) which is probability of error mean and lies in the range of 0.001 to 0.01. If P-Value calculated by test suit is greater than ∝ then sequence is considered as random else non-random.
We have performed this test on 1-Gb output generated by PLL based TRNG Generator by selecting a set of 1-Mb (m=1024) sequence. The NIST test results are summarized in table -1.

Table -1: NIST test results

Project	F	BF	FFT	CT	U	S	LC
T_1	112	111	83	117	130	95	105
T_2	103	103	110	107	95	107	108
T_3	114	103	116	90	111	105	96
T_4	95	91	110	95	112	126	96
T_5	98	104	112	108	99	99	103
T_6	105	110	108	98	91	94	114
T_7	91	101	120	102	97	94	106
T_8	95	108	87	99	92	96	87
T_9	104	93	79	105	111	104	108
T_{10}	107	100	99	103	86	104	101
P - Value	0.8274	0.9202	0.0278	0.8308	0.0723	0.5106	0.8079

Where, **F**= Frequency, **BF**= Block Frequency, **FFT**= Fast Fourier Transform, **CT**= Cycle Template, **U**= Universal, **S**= Serial, **LC**= Linear Complexity.

7. CONCLUSION

In this paper we proposed a novel method of TRNG generation using hardware software co-designing. Its advantage lies in the fact that it uses non deterministic source of jitter generated by analog on chip PLL which is being configured using software quartus-II justifying the name hardware software co-designing. Proposed TRNG generation method closely resemble ideal TRNG by passing NIST test and hence can be used in different cryptographic application where higher bit rate and long bit sequences are desirable. As our proposed method is almost indistinguishable from ideal TRNG it can at least be used as true source of entropy or ''seeding'' for PRNGs. Above results shows that TRNG can be realized in FPGA by significantly increasing the level of security for cryptographic applications. The future work might be proceeded in the direction of optimizing the design by proper selection of K_M, K_D and F_{REF} values using different advanced engineering optimization algorithms and there comparative performance evaluation.

ACKNOWLEDGEMENT

All that goes with this project is a compromise between theoretical consideration and practical limitation. It involves all technical and non-technical experience from various sources. We wish our thanks to all those whose experience in electronic helped us in completing in this project. I am glad to express my sincere thanks to my project guide and **H.O.D. Prof. V. V. Deotare** who offered me valuable tips to complete my project his cheerful encouragement, invaluable suggestion & technical support of vital importance has made us to complete this project successfully. Finally I am thankful to all the technical staff that extended their cooperation towards me as and when needed

REFERENCES

[1] V. Fischer and M. Drutarovsk´y. True Random Number Generator Embedded in Reconfigurable Hardware In B. S. Kaliski Jr., C¸. K. Ko¸c, C. Paar, editors, Workshop on Cryptographic Hardware and Embedded Systems - CHES 2002, pages 415 – 430, Berlin, Germany, LNCS 2523 2003. Springer-Verlag Berlin Heidelberg.

[2] Zhou Gan-min Yang Sheng-guang Jiang Zhao-yu GAO Ming-lun. A True Random Number Generator Based on PLL. Journal of Electronics and Information Technology, 2005, 27(7): p 1152-1156；

[3] Li Dejun, Pei Zhen. Research of True Random Number Generator Based on PLL at FPGA. International Workshop on Information and Electronics Engineering (IWIEE), 2012, 29: p 2432 – 2437.

[4] APEX 20K Programmable Logic Device Family. Data Sheet, February 2002, ver. 4.3, 1-116, http://www.altera.com ;

[5] Quartus II - Programmable Logic Design Software. January 2002, ver.2.0, 1-45, http://www.altera.com ;

[6] Jitter comparison analysis: APEX 20KE PLL vs. Virtex-E DLL. Technical Brief 70, January 2001, ver.1.1, 1-7, http://www.altera.com ;

[7] Superior Jitter management with DLLs. Virtex Tech Topic VTT013 (v1.2), January 21, 2002, 1-6, http://www.xilinx.com.

BIOGRAPHIES

Sushil C. Bagal is pursuing his M.E. in VLSI and Embedded System in Electronics & telecommunication department of SIT Lonavala, Savitribai Phule Pune University Maharashtra, India.

Vilas V. Deotare is H.O.D. of Electronics & telecommunication department of SIT Lonavala, Savitribai Phule Pune University Maharashtra, India

Dinesh V. Padole is working as Professor in Electronics & Telecommunication department GHRCE, Nagpur Maharashtra, India

Swapnil C. Bagal is UG student in Mechanical Engineering department at COEP Pune, Maharashtra, India.

