Available online at www.sciencedirect.com

SciVerse ScienceDirect

Procedia Engineering 29 (2012) 2432 – 2437

Procedia Engineering

www.elsevier.com/locate/procedia

2012 International Workshop on Information and Electronics Engineering (IWIEE)

Research of True Random Number Generator Based on PLL at FPGA

Li Dejun[a*] , Pei Zhen[a]

[a]*Wuhan Textile University, Hongshan qu fangzhilu No.1,WUHAN 430073, CHINA*

Abstract

It's very difficult to use resistance thermal noise for pseudo-random number in encryption algorithm at increasingly powerful personal computing functionality now. So the situation we need the true random number more and more. The Direct amplification, Oscillation sampling, Discrete-time chaos, Metastable sampling [1]. This paper use PLL in Altera NIOS develops a True Random Number generator (TRNG). It's simpler, faster and cheaper than previous system.

*Keywords:*TRNG, PLL, NIOS, FPGA

1. Introduction

To pseudo-random number, if the attacker has enough computing power, it could be forecasted the random number sequence. So we hope the attacker cannot obtain the next random number when it has max computing power, and get all random number sequence [2].

There are many ways to produce a variety of real random numbers, such as the use of chaotic systems, the use of noise ADC sampling, the use of the quantum effects, etc.. This design is the use of random number generator of electronic components caused by noise in digital logic random jitter to generate. Affect the performance of those who have TRNG: Entropy Source, Harvesting Mechanism) and Post-processing, as shown in Figure 1 [1]. Thus, both in the classic field of information security or in the field of quantum information, are required to have a True Random Number Generator(TRNG).

* * Corresponding author. Tel.: +86-027-59367409

E-mail address: 397894014@qq.com.

doi:10.1016/j.proeng.2012.01.327

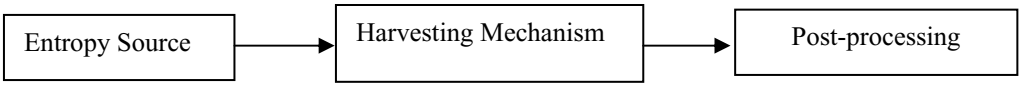

Fig.1 Diagram of a ture random number generator

Unfortunately stand processors (Altera NIOS processors include) can't generate TRNG, because it is a deterministic system. Pseudo-random number generate is too complexity to cryptography, for example to generate keys is inadvisable. Therefore, absolute security password system to obtain random numbers should be built on a physical system based on randomly generated.

This paper is based on the Altera CPLD chip analog phase-lock circuit, describes the Altera APEX FPLD custom built on a TRNG of the most commonly used method. Proposed method can be use a reliable on-chip analog PLL circuit extraction APEX low-jitter clock generator of random signals. TRNG relative NIOS processor was developed as an IP core system module, embedded encryption for the entire SOPC system designed to provide a higher level of system security.

2. The basic principles of PLL generates TRNG

Under ideal conditions, the noise output of the random bit generator sampling appears '0 'and '1' exact probability equal to 1 / 2. But in fact, due to the internal circuitry other non-Gaussian noise as well as external factors, ambient temperature changes the impact on system stability, making the resulting random bit cannot be strictly equal probability and this will affect the sequence of random numbers to form the distribution uniformity.

This paper is the use of the phase detector output signal at the edge of the heat generation time jitter. This jitter has nothing to do with the edge of the noise, can be regarded as an ideal, noise-jitter. It is equivalent to Δ t seconds fixed input clock jitter root mean square. Depending on the design, one or more of the input waveform edge triggered independently of each input device, in order to provide comparison. In between these edges at different time's necessary to convert the current pulse, its width is sufficient to lock the phase difference. By superposition, each generation of timing jitter can be converted to an equivalent phase jitter, pulse width adjustable charge pump, into the rest of the noise synthesizer, on f_s, the frequency and phase detector phase jitter is equivalent:

$$\Delta\phi_{in} = 2\pi f_s \Delta t\left[rad(rms)\right] \tag{1}$$

In actual application, this is a picoseconds jitter meter, but the thermal noise has a sampling frequency much higher than the bandwidth. PLL at a very low duty cycle pulse output pulse sequence. Using this principle of edge-triggered sampling device is a good approximation of the pulse sampler. The equivalent noise bandwidth with half the sampling frequency, the spectral density consistent with the case of frequency conversion within the scope of this part. Therefore, the equivalent input-sided spectral density of phase fluctuations is:

$$S\phi_{IN}(f)\frac{(\Delta\phi_{in})^2}{f_s/2} = 8\pi^2 f_s \Delta t^2\left[\frac{rad^2}{Hz}\right] \tag{2}$$

This indicates that the increase in phase noise provided by the phase detector frequency. For a typical phase-locked loop synthesizer, only one feedback path divider output phase fluctuations produced by two-sided spectral density of the gain of the gain equal to the ratio of voltage divider. It`s means: $N = f_o / f_s$ (f_o is the output frequency synthesizer).

3. The basic design principles of TRNG

Altera FPLD using reconfigurable system-on-chip PLL to improve performance and enhance the on-chip clock synthesizer frequency. The theoretical basis of this issue is the use of embedded analog PLL circuit extracts the clock signal synthesizer random jitter [3], as shown:

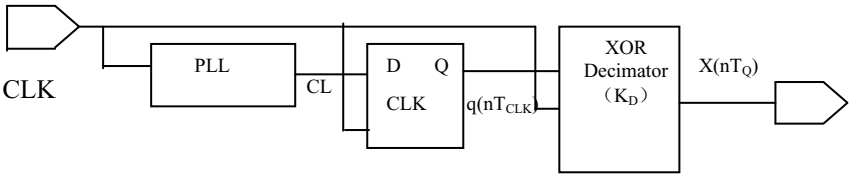

Fig.2 Basic principle of randomness extraction from low-jitter clock signal

Jitter is related to the PLL clock synthesizer (CLJ) from the clock signal (CLK) in the sample detector out. Make CLK to system clock frequency of F_{CLK} and CLJ on-chip synthesis. [4]

$$F_{EXT}\frac{80\times 14}{11\times 101}=33.570MHz \tag{3}$$

$$F_{CLJ}=F_{CLK}\frac{K_M}{K_D} \tag{4}$$

Extraction multiplication factor K_M and Frequency factor K_D:

$$GCD(K_M,K_D)=1 \tag{5}$$

where GCD is an abbreviation of Greatest Common Divisor. Equation (5) ensures that the maximum guaranteed distance between the closest edges of CLK and CLJ (denoted as $\max \Delta T_{\min}$) over the period.

$$T_Q=K_D T_{CLK}=K_M T_{CLJ} \tag{6}$$

Is equal to(5)

$$MAX(\Delta T_{\min})=T_{CLK}\frac{GCD(2K_M,K_D)}{4K_M}=T_{CLJ}\frac{GCD(2K_M,K_D)}{4K_D} \tag{7}$$

By proper choosing of K_M , K_D , T_{CLK} it is possible to guarantee that $MAX(\Delta T_{\min})<\sigma_{jit}$。 According to [5], an intrinsic jitter can be approximated by Gaussian distribution and $\sigma_{jit}\geq 15ps$. Moreover proposed method is insensitive to an overall jitter characteristic as far as an intrinsic PLL jitter is included. Use XOR gate extracted with several independent signal synthesis delay line, in order to ensure that (6) of the cycle signal TQ generated by random distribution of random bits.

4. Hardware design Alter FPLD

To measure the real performance of our proposed TRNG, an Altera NIOS development board was selected. This development board was chosen to eliminate concerns about proper board layout technique. The same board was also used in [6] for reference PLL measurements so we can expect that jitter characteristics presented in [6] can be directly applied to our design. The board features a PLL capable APEX EP20K200-2X with four on-chip analog PLLs. In order to use as large output data rate as possible, the two on-chip PLLs shown in Fig.3 were used for generating CLJ and CLK signals.

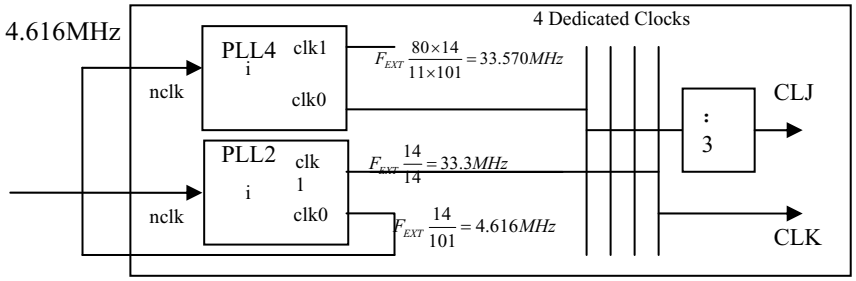

Fig.3. Actual PLL configuration used in proposed TRNG IP block

In Fig.3.external clock source is $F_{EXT}=33.3MHz$, on-chip synthesized clocks is $F_{CLK}=33.3MHz$ (this frequency is the NIOS processor system clock frequency). $F_{CLJ}=F_{CLK}(1120/3333)\approx33.6MHz$. From formula (7) to ensure parameters $MAX(\Delta T_{\min})\approx6.7ps<\sigma_{jit}$. The output bit-rate of TRNG is $1/T_Q\approx10000bit/s$. The TRNG is written as parameterized VHDL code using standard Altera mega function for embedded PLL configuration and LPM_ADD_SUB (adder/subtract) mega function.

TRNG peripherals to the NIOS processor through the data register (read only) or control / status register (read and write) is mapped to visit two storage units, shown in Figure 4. TRNG standard memory pool can be enabled to access the application, access as easy as using interrupts. TRNG the base address and interrupt request can be SOPC compiler exact configuration.

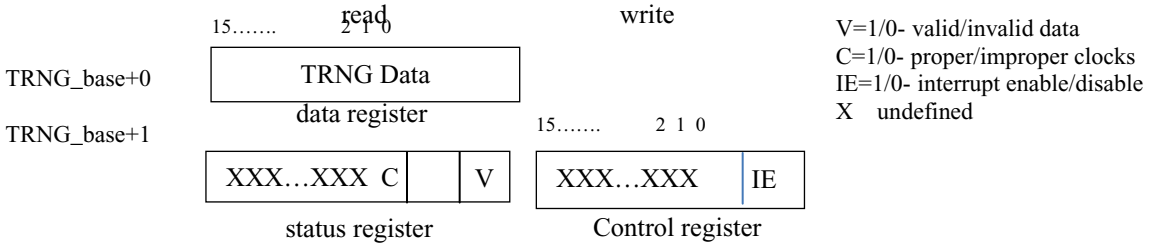

Fig.4 Data and Control/Status registers of 16-bit TRNG

Table 1 shows examples of 16-bit / 32-bit NIOS resources required and the corresponding 16-bit / 32 TRNG tools (the original logic that Altera FPLD of structural units). Shows the results after the Altera Quartus II, Altera IOS 2.2 and Leonardo Spectrum simulation.

Table 1. simulation results in Altera APEX EP20K200-2X

BLPCK	LE	% of total capacity
NIOS-16/NIOS-32	140/1480	14/18
UART	170	2
TRNG-16/TRNG-32	150/200	2/2.5
ALL & interface logic	1669/2143	20/26

5. Data test of TRNG output

There are many way to find pros and cons of TNRG. Currently, the most common test system was developed by the U.S. National Institute of Standards and Technology (NIST) released statistics test suite [7]. This test is used in August 2011 sts-2.1.1 version. This reference to the Institute of Electro-Optical Engineering, Taiyuan University of Technology Li pu's way of testing the application.

In this suite α (Significance level) can use straightforward. If P-value$\geqslant\alpha$, The assumption holds, that the sequence is random. If P-value$\leqslant\alpha$, The assumption does not hold, both non-random sequence. The α is the probability of error means. Usually α value in the range of [0.001,0.01].

We use continuous TRNG output of 1Gbit experimental hardware test strategy, setting TRNG have 1Mbit serial. Evaluate P-value get m=1024， α =0.01. This value is consistent with the test specification.

Table 2. NIST test results(in 1Gigabit P-value)

Project	T1	T2	T3	T4	T5	T6	T7	T8	T9	T10	P-value
Frequency	112	103	114	95	98	105	91	95	104	107	0.827418
Block frequency	111	103	103	91	104	110	101	108	93	100	0.920212
FFT	83	110	116	110	112	108	120	87	79	99	0.027813
Cycle template	117	107	90	95	108	98	102	99	105	103	0.830876
Universal	130	95	111	112	99	91	97	92	111	86	0.072399
Serial	95	107	105	126	99	94	94	96	104	104	0.510619
Linear-Complexity	105	108	96	96	103	114	106	87	108	101	0.807953

6. Conclusion and outlook

This paper describes the development of custom in FPLD hardware encryption system the possibility of TRNG. Shows the FPLD in engineering design of new applications. CPLD has a similar user-defined ASIC advantage, but to avoid the ASIC's high development costs, and avoid design changes in production after the inconvenience. Test results show that the TRNG can be achieved within the FPGA, and significantly increased the level of security for embedded systems, and expanded its application in the encryption system. Future development of TRNG IP core will include the FIPS and AIS-based online TRNG hardware encryption standard test system.

References

[1]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):p1152-1156；

[2]ZHANG Hongfei WANG Jian LUO Chunli CUI Ke YAO Zhiming LIANG Hao JIN Ge. Development of a high speed true random number generator based on jitter. *Nuclear Techniques*，2011,7:p556-560；

[3]ZHANG hongbo, DAI zizhu, SUN wanzhong. research and design of NIOS embedded processor development system. *IC and Component*，2004,1:p9-10,18；

[4]V. Fischer, M. Drutarovský, True Random Number Generator Embedded in ReconfigurableHardware, *Proceedings of the Workshop on Cryptographic Hardware and Embedded Systems -CHES'2002*, Redwood Shores, California, USA, August 2002:p415-430；

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

[6]HU jianyong, SU jinghai. Scheme for Real Time Test of Randomness, *Computer Engineering*，2009,35(9):p136-138;

[7] NIST. NIST Random Number Generation and Testing [OL], 2011, http://csrc.nist.gov/rng;
[8]LI pu, NIST Random test,, 2011, http://wenku.baidu.com/view/48531eda6f1aff00bed51e47.html?from=rec&pos=0