True Random Number Generator Embedded in Reconfigurable Hardware

Viktor Fischer[1] and Miloš Drutarovský[2]

[1] Laboratoire Traitement du Signal et Instrumentation,
Unité Mixte de Recherche CNRS 5516, Université Jean Monnet,
Saint-Etienne, France
fischer@univ-st-etienne.fr
[2] Department of Electronics and Multimedia Communications,
Technical University of Košice,
Park Komenského 13, 041 20 Košice, Slovak Republic
Milos.Drutarovsky@tuke.sk

Abstract. This paper presents a new True Random Number Generator (TRNG) based on an analog Phase-Locked Loop (PLL) implemented in a digital Altera Field Programmable Logic Device (FPLD). Starting with an analysis of the one available on chip source of randomness - the PLL synthesized low jitter clock signal, a new simple and reliable method of true randomness extraction is proposed. Basic assumptions about statistical properties of jitter signal are confirmed by testing of mean value of the TRNG output signal. The quality of generated true random numbers is confirmed by passing standard NIST statistical tests. The described TRNG is tailored for embedded System-On-a-Programmable-Chip (SOPC) cryptographic applications and can provide a good quality true random bit-stream with throughput of several tens of kilobits per second. The possibility of including the proposed TRNG into a SOPC design significantly increases the system security of embedded cryptographic hardware.

1 Introduction

Random number generators represent basic cryptographic primitives. They are widely used for example as confidential key generators for symmetric key and public-key crypto-systems (e. g. RSA-moduli) and as password sources. In some algorithms (e.g. DSA) or protocols (e.g. zero-knowledge), random numbers are intrinsic to the computation [1]. In all these applications, security depends greatly on the randomness of the source.

Because security algorithms and protocols rely on the unpredictability of the keys they use, random number generators for cryptographic applications must meet stringent requirements. Unfortunately computers and digital hardware can implement only pseudo-random generators. A Pseudo-Random Number Generator (PRNG) is a deterministic polynomial time algorithm that expands short (hopefully true random and well distributed) seeds into long bit sequences, this

B.S. Kaliski Jr. et al. (Eds.): CHES 2002, LNCS 2523, pp. 415–430, 2003.

distribution is polynomially indistinguishable from the uniform probability distribution. PRNGs rely on complexity and their use in cryptography, for example to generate keys, is very critical. An alternative solution is to get true random numbers, hence true security for crypto-systems, using a True Random Number Generator (TRNG) based on a random physical phenomenon. Even an ideal PRNG relies upon, and is limited by, the quality of its input seed data. Good TRNG is designed to generate high-quality random numbers directly or as a seed for PRNG. Current modern high-density Field Programmable Logic Devices (FPLDs) provide a suitable hardware platform for a complete System-On-a-Programmable-Chip (SOPC). This SOPC can be used for cryptographic applications, even for system-level integration of embedded algorithms. Unfortunately, high quality embedded TRNGs were not realizable in FPLDs. Most hardware TRNGs follow unpredictable natural processes, such as thermal (resistance or shoot) noise or nuclear decay. Such TRNGs are not compatible with modern FPLDs and cannot provide a SOPC solution. The fact that TRNG cannot be implemented inside the FPLD represents significant security and system disadvantages in embedded cryptographic applications.

TRNGs can be produced using any non deterministic process. The fundamental probabilistic phenomena utilized by proposed TRNG is the frequency instability of electronic oscillator. The use of this phenomena to generate truly random numbers is not new and was used e.g. in [2], [3]. These implementations used two free running oscillators with relatively high instability at least one of them.

This paper describes implementation of new analog Phase-Locked Loop (PLL) based TRNG that uses on-chip resources of recent Altera FPLD families (e. g. APEX E [4], APEX II [5], etc.). Described TRNG uses two coupled oscillators that are not free running and originally designed to be as stable as possible. Proposed method reliably extracts intrinsic randomness from low-jitter clock signals synthesized by on-chip analog PLL circuits and to our best knowledge it is the first TRNG implementation that uses only on-chip FPLD resources. This paper extends the description of the proposed method first announced in [6], provides new results of tested output TRNG signals, reveals some deviations from ideal TRNG, and discusses system aspects of proposed TRNG. It is organized as follows: a brief overview of jitter performance of analog PLL circuits embedded in recent FPLDs is given in Sect. 2. In Sect. 3, a proposed new method of reliable true randomness extraction from low jitter on-chip PLL synthesized clock signal is presented. The experimental TRNG hardware used for the testing of the proposed method is described in Sect. 4. In Sect. 5, statistical evaluations of output TRNG signals are made. Finally, concluding remarks are presented in Sect. 6.

2 PLL – Source of Randomness in Recent FPLDs

Recent FPLDs use often on-chip PLLs to increase performance of clock distribution and to provide on-chip clock-frequency synthesis. There are two fundamental

approaches to implement PLL in FPLDs - one uses digital delay lines, or DLL, (e.g. in XILINX Virtex FPLDs [7]) and the second one uses true analog PLL (e.g. in Altera APEX E [4] and APEX II [5] FPLDs). Both approaches have some system advantages and disadvantages but we believe that analog PLL is a better candidate for cryptographic TRNG design since it contains analog source of unpredictable randomness.

2.1 Analog PLL in Altera FPLD

To support high-speed designs, new Altera FPLD devices offer ClockLock, Clock-Boost and ClockShift circuitry containing several integrated on-chip analog PLL circuits. Block diagram of enhanced PLL sub-circuit available in latest versions of APEX E and APEX II FPLDs is depicted in Fig. 1 [4], [5].

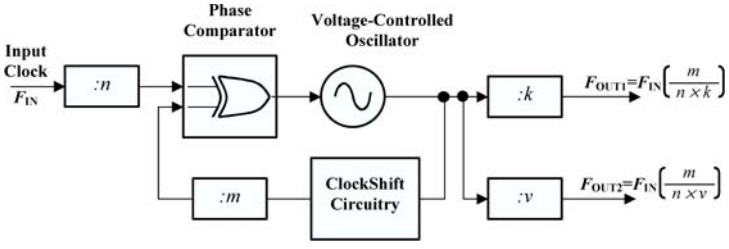


Fig. 1. Block diagram of enhanced Altera PLL circuit

In analog PLLs, various noise sources cause the internal voltage controlled oscillator (VCO) to fluctuate in frequency. The internal control circuitry adjusts the VCO back to the specified frequency and this change is seen as jitter. Under ideal conditions, the jitter is caused only by analog (non-deterministic) internal noise sources. Such jitter is called an intrinsic jitter. Other possible frequency fluctuations are caused by variations of supply voltage, temperature, external interference through the power, ground, and even by the internal noisy environment generated by internal FPLD circuits [7]. From cryptographic point of view, these sources should be regarded as deterministic and the function of TRNG must not be deteriorated by them. In other words, the output TRNG must in any case depend also on the non-deterministic intrinsic jitter. Any additional disturbing deterministic jitter is possible as far as dependency of the output signal on intrinsic jitter is guaranteed.

2.2 Jitter Characteristics of Altera PLL Circuitry

Parameters of the proposed TRNG depend on the jitter characteristics of Altera embedded PLLs. Real measurements of jitter parameters requires the use of special equipment which was not available, so we had to rely on the parameters

given in the Altera data sheets [4] and the application note [8]. Some of these parameters have been independently confirmed by Xilinx and the results are available in [7]. Since these parameters are fundamental for our TRNG design, they are summarized and discussed in this subsection.

Altera tries to minimize the clock jitter[1] by a proper design, for example their typical analog intrinsic PLL jitter in an APEX FPLD has 1-sigma value of $\sigma_{\text{jit}} \approx 15$ ps (under Gaussian approximation, the peak-to-peak jitter value is approximately $t_{\text{JITTER}} = 6\sigma_{\text{jit}}$) for a $F = 66.6$ MHz synthesized clock signal and multiplication factor of $2\times$ [7]. Actual distribution of jitter values is depicted in Fig. 2 [7]. These results were acquired under "ideal conditions", with only a minimal amount of occupied FPLD resources and minimal input/output activities.

Fig. 2. APEX intrinsic jitter performance for 1,000 clock samples (bottom curve, peak-to-peak value 97.0 ps, $\sigma_{\text{jit}} \approx 15.9$ ps) and 1,000,400 clock samples (upper curve, peak-to-peak value 151.4 ps, $\sigma_{\text{jit}} \approx 15.7$ ps)

In [7] it was shown that the clock jitter in APEX FPLD is significantly higher, when internal FPLD flip-flops are switching with different clock frequencies. It was shown that when 35 % of the total available flip-flops were clocked with a 33.3 MHz clock and 35 % of the flip-flops with a 66.6 MHz clock, jitter is much higher than that specified in the data-sheet. These conditions simulated an internal noisy environment generated by internal FPLD circuits and jitter distribution was split into two peaks with a 665 ps total peak-to-peak value [7]. Although overall jitter performance exceeds data sheet specification, true intrinsic jitter is still present and it is clearly visible as two approximated Gaussian peaks have around 150 ps. We can conclude that under real conditions the clock jitter

[1] There are two types of jitter described in [7], [8], period jitter and cycle-to-cycle jitter. Period jitter is the deviation in time of any clock period from the ideal clock period (also known as "edge-to-edge" jitter). Peak-to-peak jitter defines an upper bound on the jitter. Cycle-to-cycle jitter is the deviation in clock period between adjacent or successive clock cycles.

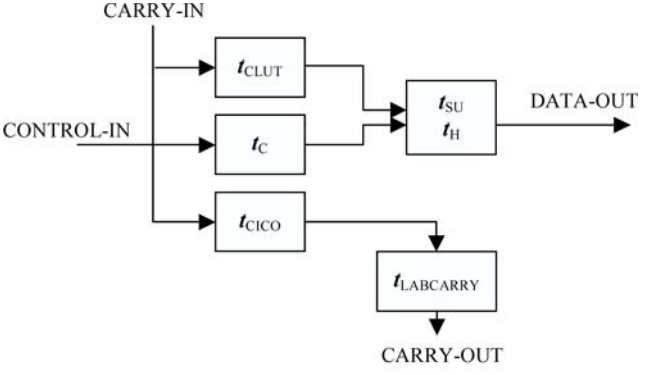
always contains intrinsic jitter and only the overall jitter distribution is changed. Such behavior is expected, since the intrinsic jitter cannot be removed by any interference.

3 Randomness Extraction from an Intrinsic Jitter

The principle behind our method is to extract the randomness from the jitter of the clock signal synthesized in the embedded analog PLL. The jitter is detected by the sampling of a reference (clock) signal using a correlated (clock) signal synthesized in the PLL. The fundamental problem lies in the fact that the reference signal has to be sampled near the edges influenced by the jitter. From the previous section we know that clock edges of a synthesized signal can vary under ideal conditions in the range of several tens of ps. This value is significantly lower than the smallest delay obtainable in APEX FPLDs and our method must overcome this problem.

3.1 Timing Analysis of the Logic Cell in Altera FPLD

The smallest possible delay in Altera FPLDs is obtainable between the carry-in and carry-out of the Logic Cell (LC). A simplified timing model of the logic cell is depicted in Fig. 3.



t_{CLUT} – Look-up-table (LUT) delay for carry-in
t_{SU} – Logic cell (LC) register setup time for data
t_H – LC register hold time for data
t_C – LC register control signal delay
t_{CICO} – Carry-in to carry-out delay
$t_{LABCARRY}$ – Routing delay for the carry-out signal of a LC driving the carry-in signal of a different LC in a different Logic Array Block (LAB).

Fig. 3. Altera simplified logic cell timing model

We have taken the parameters obtained from the Quartus II [9] version 2.0 timing analyzer as the basis for our method. From the result of this analysis we can conclude that the smallest obtainable delay in APEX FPLDs is $\tau \approx t_{\mathrm{CICO}} = 500\,\mathrm{ps}$. The delay τ is only a statistical value and its real size can vary with time, temperature and supply voltage.

3.2 Basic Principle of Randomness Extraction

The basic principle of the proposed randomness extraction is illustrated in Fig. 4 [6].

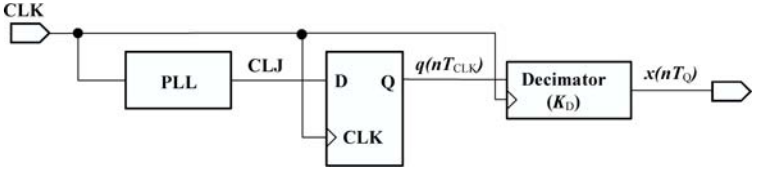


Fig. 4. Basic principle of randomness extraction from low-jitter clock signal

Let CLK be a system clock signal with the frequency F_{CLK}. In an actual implementation CLK can be either an external signal or it can be internally synthesized by an additional on-chip PLL. Let CLJ be an on-chip PLL-synthesized rectangular waveform with the frequency $F_{\mathrm{CLJ}} = F_{\mathrm{CLK}} K_{\mathrm{M}}/K_{\mathrm{D}}$. Let values of multiplication factor K_{M} and division factor K_{D} be relative primes, so

$$\mathrm{GCD}\,(K_{\mathrm{M}}, K_{\mathrm{D}}) = 1 \tag{1}$$

where GCD is an abbreviation for Greatest Common Divisor. Equation (1) ensures that the maximum guaranteed distance between the closest edges of CLK and CLJ (denoted as $\mathrm{MAX}\,(\Delta T_{\mathrm{min}})$) is minimized. As it is discussed in Sect. 2, signal CLJ certainly includes intrinsic analog PLL jitter σ_{jit} and it can also contain other "deterministic" jitter components from external or internal environment. This signal is sampled into the D flip-flop using a clock signal with frequency F_{CLK}. The sampled signal $q\,(nT_{\mathrm{CLK}})$ contains certain random values. Their exact position is not known and potentially it can vary in time. Random values can be easily extracted by a standard XOR decimator [10], [11]. In the proposed design the decimator produces one output bit per K_{D} input values $q\,(nT_{\mathrm{CLK}})$ (one period T_{Q}). The next paragraphs analyze more deeply the functionality of the proposed circuit.

Let us consider the output signal in two different conditions: ideal conditions without jitter and real conditions when jitter is included in the synthesized clock signal. Under ideal conditions when the jitter is zero ($\sigma_{\mathrm{jit}} = 0$), signal $q(nT_{\mathrm{CLK}}),\ n = 0, 1, \ldots$ is deterministic and under condition (1) periodic with the period

$$T_{\mathrm{Q}} = K_{\mathrm{D}} T_{\mathrm{CLK}} = K_{\mathrm{M}} T_{\mathrm{CLJ}}\ . \tag{2}$$

Therefore decimated output signal $x(nT_{\mathrm{Q}})$, $n = 0, 1, \ldots$

$$x(nT_{\mathrm{Q}}) = q(nT_{\mathrm{Q}}) \oplus q(nT_{\mathrm{Q}} - T_{\mathrm{CLK}}) \quad \oplus \ldots \oplus q(nT_{\mathrm{Q}} - (K_{\mathrm{D}} - 1)\, T_{\mathrm{CLK}}) \quad (3)$$

which represents bit-wise addition modulo 2 of K_{D} input samples, is also deterministic. The situation is completely different under real conditions when the jitter is nonzero ($\sigma_{\mathrm{jit}} > 0$). If K_{D} is chosen so, that the jitter σ_{jit} is comparable with the maximum distance $\mathrm{MAX}(\Delta T_{\mathrm{min}})$ between the two closest edges of CLK and CLJ, we can guarantee that during T_{Q} the rising edge of CLK will fall at least once into edge zone of CLJ (edge zone means the time interval around the edge including jitter[2]). The value $\mathrm{MAX}(\Delta T_{\mathrm{min}})$ can be computed as

$$\mathrm{MAX}(\Delta T_{\mathrm{min}}) = T_{\mathrm{CLK}} \frac{\mathrm{GCD}(2K_{\mathrm{M}}, K_{\mathrm{D}})}{4K_{\mathrm{M}}} = T_{\mathrm{CLJ}} \frac{\mathrm{GCD}(2K_{\mathrm{M}}, K_{\mathrm{D}})}{4K_{\mathrm{D}}}\,. \quad (4)$$

The during current period of T_{Q}, K_{D} values of CLJ will be sampled into D flip-flop and at least one of them will depend on the random jitter. The decimated signal $x(nT_{\mathrm{Q}})$ will not be deterministic anymore and its value will depend on this jitter. In Fig. 5-7 we analyze different possibilities for small values of K_{M} and K_{D} that demonstrate the validity of (4).

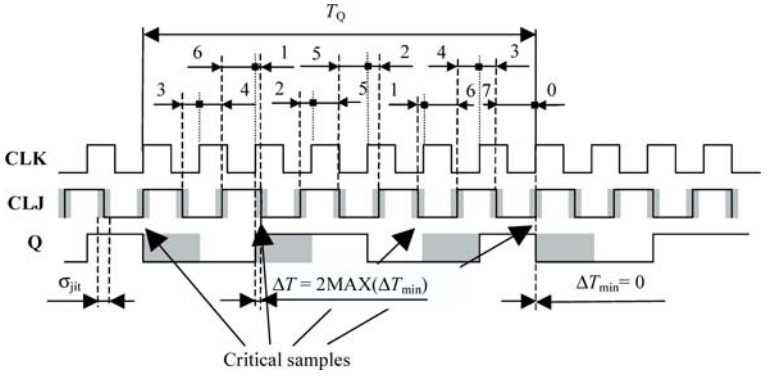


Fig. 5. Clock relation for $K_{\mathrm{M}} = 5$, $K_{\mathrm{D}} = 7$ ($F_{\mathrm{CLJ}} < F_{\mathrm{CLK}}$)

Figure 5 shows the case when $\mathrm{GCD}(2K_{\mathrm{M}}, K_{\mathrm{D}}) = 1$ and $F_{\mathrm{CLJ}} < F_{\mathrm{CLK}}$. In real implementation it is not possible to guarantee the position of CLJ in relationship to CLK. In this example the minimum distance ΔT_{min} is 0 (the last sample of the period T_{Q}). The worst case (maximum value of the minimum distance - $\mathrm{MAX}(\Delta T_{\mathrm{min}})$) will be the event when CLJ will be shifted by a half

[2] For qualitative analysis we can assume that the width of the edge zone is for example $6\sigma_{\mathrm{jit}}$. Therefore there is some non-zero probability that the jitter will influence the sampled signal value.

step (the step is equal to[3] $T_{\mathrm{CLJ}}/2K_{\mathrm{D}}$) to the left or to the right. In that case the minimum difference will be the half step in at least one of critical samples (they are indicated by arrows) and the output value Q will be nondeterministic during one period T_{CLK} (gray zones in Q output signal). Conclusion: the jitter should be comparable with the half step, therefore $\sigma_{\mathrm{jit}} \approx T_{\mathrm{CLJ}}/4K_{\mathrm{D}}$ and so

$$\mathrm{MAX}\left(\Delta T_{\mathrm{min}}\right) = \frac{T_{\mathrm{CLK}}}{4K_{\mathrm{M}}} = \frac{T_{\mathrm{CLJ}}}{4K_{\mathrm{D}}} = \frac{T_{\mathrm{CLJ}}}{28} \tag{5}$$

Figure 6 shows the case when $\mathrm{GCD}\left(2K_{\mathrm{M}}, K_{\mathrm{D}}\right) = 1$ and $F_{\mathrm{CLJ}} > F_{\mathrm{CLK}}$. Following the previous study it can be found that $\mathrm{MAX}\left(\Delta T_{\mathrm{min}}\right)$ can be expressed in the same way as in (5), so (4) is valid, too.

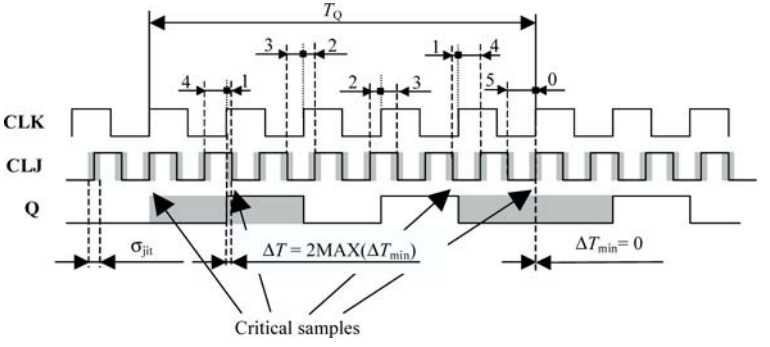


Fig. 6. Clock relation for $K_{\mathrm{M}} = 7$, $K_{\mathrm{D}} = 5$ ($F_{\mathrm{CLJ}} > F_{\mathrm{CLK}}$)

Figure 7 shows the case when K_{D} is even so $\mathrm{GCD}\left(2K_{\mathrm{M}}, K_{\mathrm{D}}\right) = 2$ and $F_{\mathrm{CLJ}} < F_{\mathrm{CLK}}$. It can be found that $\mathrm{MAX}\left(\Delta T_{\mathrm{min}}\right)$ can be expressed as

$$\mathrm{MAX}\left(\Delta T_{\mathrm{min}}\right) = \frac{T_{\mathrm{CLJ}}\mathrm{GCD}\left(2K_{\mathrm{M}}, K_{\mathrm{D}}\right)}{4K_{\mathrm{M}}} = \frac{2T_{\mathrm{CLJ}}}{4K_{\mathrm{D}}} = \frac{T_{\mathrm{CLJ}}}{2K_{\mathrm{D}}} = \frac{T_{\mathrm{CLJ}}}{16} \tag{6}$$

and (4) is valid also in this case.

Following this analysis we can conclude that according to (4) it is better (if it is possible from system point of view) to choose relative primes K_{M}, K_{D} in such a way that K_{D} is odd. This choice will decrease $\mathrm{MAX}\left(\Delta T_{\mathrm{min}}\right)$ by the factor of 2.

[3] Note that there are $K_{\mathrm{D}} = 7$ clock periods in interval T_{Q}. The longest distance ΔT is 7Δ. 7Δ is a half period of CLJ. So the longest distance ΔT is the half period of CLJ. The worst case of the largest distance $\mathrm{MAX}\left(\Delta T_{\mathrm{min}}\right)$ is $0.5\Delta = 1/14$ of the half period of CLJ. That means $1/28 = 1/\left(4K_D\right)$ of the full period of CLJ.

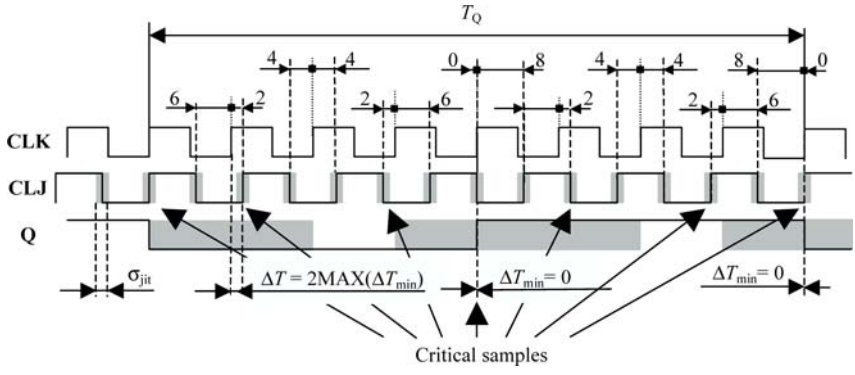


Fig. 7. Clock relation for $K_M = 7$, $K_D = 8$ ($F_{CLJ} < F_{CLK}$)

3.3 TRNG Realization

Under real conditions D flip-flop in Fig. 4 produces signal $q(nT_{CLK})$ that is sampled K_D times during the time interval T_Q. Based on the analysis in Sect. 3.2 it is possible to express the decimated output signal $x(nT_Q)$ as

$$\begin{aligned} x(nT_Q) = [&q(nT_Q) \oplus q(nT_Q - T_{CLK}) \oplus \dots \\ &\dots \oplus q(nT_Q - (J-1)T_{CLK}) \oplus q(nT_Q - (J+1)T_{CLK}) \oplus \dots \\ &\dots \oplus q(nT_Q - (K_D - 1)T_{CLK})] \quad \oplus \quad q(nT_Q - JT_{CLK}) \end{aligned} \tag{7}$$

where the first term in (7) contains all values not influenced by the jitter (therefore they are deterministic) and the second term[4] $q(nT_Q - JT_{CLK})$ is influenced by the jitter. In general, values $q(nT_Q - JT_{CLK})$ are statistically biased random bits that have expectation (long run average) $p = \mathrm{E}[q(nT_Q - JT_{CLK})]$ different from the ideal value of $1/2$ by a bias $b = p - 1/2$. Under Gaussian approximation the bias for intrinsic jitter can be computed by

$$|b| = \left| \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\frac{\Delta T_{min}}{\sigma_{jit}}} e^{\frac{-x^2}{2}} dx - \frac{1}{2} \right| \leq \frac{1}{2} \mathrm{erf}\left(\frac{\mathrm{MAX}(\Delta T_{min})}{\sigma_{jit}\sqrt{2}} \right) < \frac{1}{2} \tag{8}$$

where erf() is the Error function [15]. Since the exact value of J is not known (it can be influenced by a non-deterministic jitter described in Sect. 2 or by the temperature and supply voltage variations that influence delays in the D flip-flop) it is necessary to use a decimator that produces $x(nT_Q)$ according to (7) with the output sample rate T_{CLK}/K_D. This ensures that randomness included in the value $q(nT_Q - JT_{CLK})$ is also included in $x(nT_Q)$ without precise knowledge of

[4] In principle more terms could be influenced by the jitter but according to the previous analysis, choosing proper values K_M and K_D we can guarantee that at least one sample will be influenced by the jitter.

the actual value J (position of the sample influenced by jitter in the frame of one period T_{Q}).

Good TRNG should produce binary outputs with equal probability, so $b \to 0$. Signal $x\,(nT_{\mathrm{Q}})$ generally does not fulfill this requirement. One common way to reduce statistical bias is to use a XOR corrector [10], [11]. The simplest XOR corrector takes non-overlapped pairs of bits[5] from the input stream and XORs them to produce an output stream with the half bit-rate of the input stream. If input stream bits are statistically independent then the bias at the output (decimated) stream is $b_{\mathrm{out}} = -2b_{\mathrm{in}}^2$ and $|b_{\mathrm{out}}| < |b_{\mathrm{in}}|$ since $|b_{\mathrm{in}}| < 1/2$ [10]. There are two XOR operators needed in the complete TRNG realization (see Fig. 8): the XOR decimator implied by the basic principle of the method (described above) and a XOR corrector of N_{d} samples

$$q_i\,(nT_{\mathrm{CLK}}) = q\left(nT_{\mathrm{CLK}} - \sum_{j=0}^{i} j\tau_j\right),\ i = 0, 1, \ldots, N_{\mathrm{d}} - 1,\ \tau_j \approx \tau\ . \qquad (9)$$

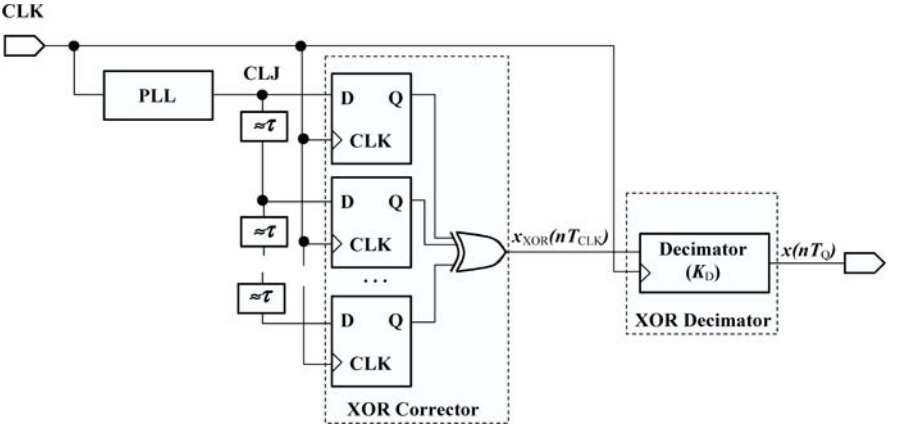


Fig. 8. Simplified block diagram of complete TRNG

To increase the probability of overlapping CLK and CLJ edge zones during the T_{Q} period, the signal CLJ is delayed in $N_{\mathrm{d}} - 1$ delay elements. Outputs of these elements are synchronously sampled with the frequency F_{CLK} and XOR-ed together to produce signal

$$x_{\mathrm{XOR}}\,(nT_{\mathrm{CLK}}) = q_0\,(nT_{\mathrm{CLK}}) \oplus q_1\,(nT_{\mathrm{CLK}}) \oplus\ \ldots\ \oplus q_{N_{\mathrm{d}}-1}\,(nT_{\mathrm{CLK}})\ . \qquad (10)$$

Output of the complete TRNG can be written in the form:

[5] This principle can be applied also to more non-overlapped bits that are XORed together.

$$x\left(nT_{\mathrm{Q}}\right) = x_{\mathrm{XOR}}\left(nT_{\mathrm{Q}}\right) \oplus x_{\mathrm{XOR}}\left(nT_{\mathrm{Q}} - T_{\mathrm{CLK}}\right) \oplus \ldots \\ \ldots \oplus x_{\mathrm{XOR}}\left(nT_{\mathrm{Q}} - \left(K_{\mathrm{D}} - 1\right) T_{\mathrm{CLK}}\right) \ . \tag{11}$$

The minimal signal delay obtainable in Altera APEX family is $\tau \approx 500$ ps and actual values $\tau_i,\ i = 0, 1, \ldots, N_{\mathrm{d}} - 1$ fluctuate around this value and are influenced by the supply voltage and the temperature. This mechanism causes fluctuation of biases b_i of individual outputs of the delay line (since corresponding values $\Delta T^i_{\mathrm{min}},\ i = 0, 1, \ldots, N_{\mathrm{d}} - 1$ influence b_i according to (8)). In order to decrease the output bias it is necessary to use a N_{d} which is as large as possible. We propose to use about N_{d} delay elements, where the maximal value of N_{d} is limited by

$$N_{\mathrm{d}} \leq T_{\mathrm{CLK}}/\tau = 1/(F_{\mathrm{CLK}}\tau) \ . \tag{12}$$

The sum of the delays thus spans one period T_{CLK} and ensures (for $T_{\mathrm{CLK}} < T_{\mathrm{CLJ}}$) that if the edge of the CLJ signal is in the current T_{CLK} window, the edge zone is sampled only once with the probability $\sigma_{\mathrm{jit}}/\tau$. Larger values of N_{d} are not recommended, since sampling one edge of CLJ signal two or more times can create an undesired statistical dependency. Therefore at each output-sampling interval nT_{Q}, the signal $x\left(nT_{\mathrm{Q}}\right)$ is the result of XOR-ing

$$N_{\mathrm{XOR}} = K_{\mathrm{D}} N_{\mathrm{d}} \tag{13}$$

individual bits. There are $2K_{\mathrm{M}}$ edges of CLJ signal over T_{Q} period, so approximately N_{bit} bits, N_{bit} being calculated by

$$N_{\mathrm{bit}} \approx 2K_{\mathrm{M}}\sigma_{\mathrm{jit}}/\tau, \tag{14}$$

are influenced by the intrinsic jitter and these bits are used by XOR corrector for a bias reduction. Although value (14) is just statistical estimation, it provides information about the applicability of some statistical rules.

If the input bits were statistically independent, the decimated output sequence $x\left(nT_{\mathrm{Q}}\right)$ would quickly converge to an unbiased binary sequence that is uncorrelated. Since the binary stream $x_{\mathrm{XOR}}\left(nT_{\mathrm{CLK}}\right)$ is influenced by an analog part of the PLL, we can expect that obtained values will be statistically independent. This hypothesis is tested in Sect. 5.

4 Experimental Hardware Implementation

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 [7] for the reference PLL measurements so we can expect that jitter characteristics presented in Sect. 2 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[6]

[6] It is possible to create TRNG based only on one PLL, but it requires a different crystal than the NIOS board actualy uses.

on-chip PLLs shown in Fig. 9 were used for generating CLJ and CLK signals. The external clock source was 33.3 MHz, on-chip synthesized clocks were $F_{\mathrm{CLK}} = 33.3 \times 159/60 = 88.245$ MHz and $F_{\mathrm{CLJ}} = F_{\mathrm{CLK}}\,(785/1272) \approx 54.459$ MHz, so $K_{\mathrm{M}} = 785$ and $K_{\mathrm{D}} = 1272$. These values were chosen as a compromise of minimal $\mathrm{MAX}\,(\Delta T_{\mathrm{min}})$ for actual NIOS board constraints. According to (4) they ensure that $\mathrm{MAX}\,(\Delta T_{\mathrm{min}}) \approx 7.2$ ps $< \sigma_{\mathrm{jit}}$. The TRNG was implemented for $N_{\mathrm{d}} = 22$ in VHDL using standard Altera megafunction for embedded PLL configuration.

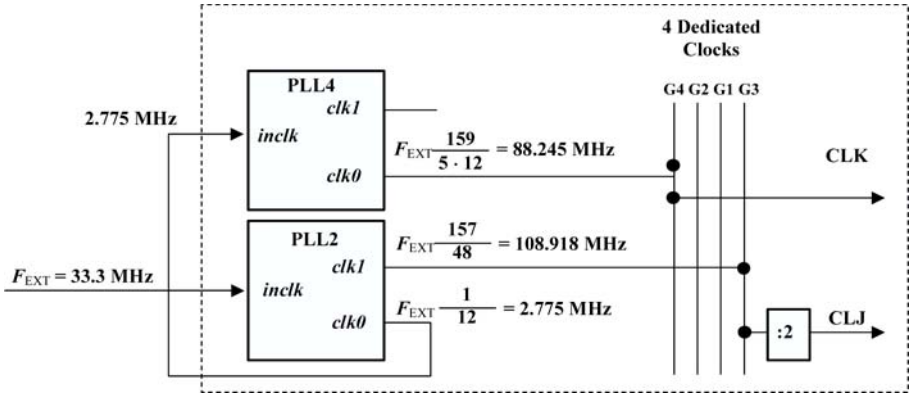


Fig. 9. Actual PLL configuration used in experimental hardware

There are two problems related to the random number generator implementation in an FPLD:

- the function of the generator cannot be verified using simulation (jitter is not simulated),
- since detection of the jitter is based on a repetition of a small signal delay using a carry chain, placement and routing has a significant impact on the generator operation (for example, to guarantee the correct operation of the generator, D flip-flops have to be implemented in the same logic array block as the carry chain delay).

For a proper operation the design must use resource locking (assignments) and the design must be verified and tested on a real hardware. Generator blocks have been designed using both Altera Hardware Description Language (AHDL) and VHDL. Since its implementation is hardware-specific, it seems to be more practical to use AHDL instead of VHDL (at least for the jitter detector block), because AHDL is closer to the hardware and the implementation can be better controlled on a low level basis (assignments of hardware elements).

The FPLD resource requirements of the proposed TRNG block as well as the supporting logic (FIFO, control logic) of the experimental hardware implementation is shown in Table 1. The first four columns show resource requirements

(in Logic Cells and Embedded System Blocks (ESB)) of the generator, as it is presented in Fig. 8. The second four columns give resource requirements of the complete TRNG circuit including 8 bits wide 1024-byte FIFO and a data bus controller. Presented results have been obtained using Altera Quartus II v. 2.0 [9]. Values $x\left(nT_{\mathrm{Q}}\right)$ generated with bit-rate $1/T_{\mathrm{Q}} \approx$ 69375 bits/s were saved on the hard disk for further analysis.

Table 1. APEX FPLD resource requirements

Device	TRNG only				TRNG + FIFO			
	LCs #	LCs %	ESBs #	ESBs %	LCs #	LCs %	ESBs #	ESBs %
EP20K200EFC484-2X	48	0.6	0	0	121	1.5	4	7.7

5 Statistical Evaluation of TRNG

Testing a hardware random number generator differs from testing a PRNG [12]. In particular, if we know the design of the generator we can tailor some of the tests. However, the random number generator (either random or pseudorandom) might pass the test and still not be a good generator. There are some well documented general statistical tests that can be used to look for deviations from an ideal TRNG [13], [14], [15]. A good TRNG should pass all kinds of tests.

5.1 Testing of Basic Statistical Assumption

A potential problem of using XOR decimation technique for bias removing is that XOR decimation should be used only with statistically independent bits. Our XOR corrector performs XOR-ing of $N_{\mathrm{XOR}} = 1272 * 22 = 27984$ input bits. According to (14) there are about $N_{\mathrm{bit}} \approx 47$ input bits per one output bit that are influenced by a non-deterministic jitter. Under ideal assumption (statistically independent biased jitter values) the output signal must converge to an almost unbiased binary sequence ($B \to 0$) with probability of 1's and 0's equal to $1/2 \pm B$, where the total bias B can be computed as

$$B = E\left[x\left(nT_{\mathrm{Q}}\right)\right] = E\left[f\left(q_0\left(nT_{\mathrm{Q}}\right), \ldots q_{N_{\mathrm{d}}-1}\left(nT_{\mathrm{Q}}\right), b_0, b_1 \ldots b_{N_{\mathrm{d}}-1}\right)\right] + \\ + (-2)^{N_{\mathrm{d}}-1} \prod_{i=0}^{N_{\mathrm{d}}-1} b_i \,. \tag{15}$$

For statistically independent values the first term of (15) is zero and the second term of (15) very quickly converges to a low value since $|b|_i < 1/2$, $i = 0, 1, \ldots, N_{\mathrm{d}} - 1$. Table 2 shows the results of the mean value computation for several 1-Gigabit TRNG output records acquired from two available NIOS boards.

It is clear that there is a certain small difference from an ideal TRNG. This difference is caused by a certain small non-zero statistic dependency in the first term of (15). This is the first[7] detected difference between our TRNG and ideal one.

Table 2. Mean values computed for several 1-Gigabit records

Record	1	2	3	4	5
NIOS	(Board A)	(Board B)	(Board B)	(Board B)	(Board B)
Mean	0.500109	0.499917	0.499911	0.499896	0.499872

5.2 The NIST Statistical Tests

A large number of generalized statistical tests for randomness have been proposed. It seems that the NIST statistical test suite [15] is currently the most comprehensive tool publicly available. Our NIST statistical tests were performed on 1-Gigabit of continuous TRNG output records and followed the testing strategy, general recommendations and result interpretation described in [15]. We have used a set of $m = 1024$ 1-Megabit sequences produced by the generator and we have evaluated the set of P-values (some typical values are shown in Table 3 [6]) at a significance level $\alpha = 0.01$. The total number of acceptable sequences was within the expected confidence intervals [15] for all performed tests and P-values were uniformly distributed over the $(0, 1)$ interval.

We have performed the same tests for several 1-Gigabit records and have uncovered certain deviations in the FFT statistical test results. For ideal TRNG the distribution of P-values is uniform in the interval $(0, 1)$. For tested TRNG this uniformity is checked by using a χ^2 test distribution of P-values in subintervals C1-C10. If the P-value shown in Table 4 (more precisely a P-value of the P-values [15]) is lower than 0.0001 the test fails and indicates a detectable difference from the ideal TRNG.

6 Conclusions

In this paper we have evaluated a new method of true random numbers generated in SOPC based on a reconfigurable hardware. The randomness of the sequence of numbers has been extensively tested and only small differences from an ideal TRNG have been detected. We believe that intrinsic analog PLL noise is a good source of true randomness and at least for typical cryptographic keys with the length from hundreds to several thousands bits, our TRNG is not distinguishable from the ideal TRNG. For very critical cryptographic applications the proposed

[7] Note that this difference is really detectable only for long streams and we believe that proposed TRNG can be used for key generation in typical cryptographic applications.

Table 3. NIST test results (uniformity of P-values and proportion of passing sequence) for 1-Gigabit record that passed all tests

C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	P-value	Proportion	Statistical Test
112	103	114	95	98	105	91	95	104	107	0.827418	0.9873	Frequency
111	103	103	91	104	110	101	108	93	100	0.920212	0.9922	Block-Freq.
103	113	89	100	92	104	107	97	92	127	0.242375	0.9873	Cusum
97	81	97	117	114	91	93	106	115	113	0.144842	0.9941	Runs
86	108	102	92	93	94	122	99	125	103	0.106543	0.9922	Long-Run
99	92	116	110	90	115	103	93	104	102	0.582174	0.9902	Rank
83	110	116	110	112	108	120	87	79	99	0.027813	0.9951	FFT
117	107	90	95	108	98	102	99	105	103	0.830876	0.9824	Periodic-Template
130	95	111	112	99	91	97	92	111	86	0.072399	0.9863	Universal
91	114	118	102	85	94	108	96	112	104	0.327204	0.9951	Apen
95	107	105	126	99	94	94	96	104	104	0.510619	0.9932	Serial
110	90	104	127	94	96	78	107	114	104	0.056616	0.9863	Lempel-Ziv
105	108	96	96	103	114	106	87	108	101	0.807953	0.9893	Linear-Complexity

Table 4. NIST FFT test results (uniformity of P-values and proportion of passing sequence) for all tested 1-Gigabit records

#	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10	P-value	Proportion	Statistical Test
1	83	110	116	110	112	108	120	87	79	99	0.027813	0.9951	FFT
2	105	136	100	113	111	99	101	79	88	92	0.010138	0.9932	FFT
3	**96**	**113**	**125**	**143**	**96**	**96**	**118**	**86**	**82**	**69**	**0.000002**	***0.9951**	**FFT**
4	**107**	**132**	**133**	**110**	**117**	**95**	**71**	**93**	**86**	**80**	**0.000010**	***0.9971**	**FFT**
5	91	132	115	128	101	93	99	109	78	78	0.000301	0.9941	FFT

TRNG can be used at least as an useful internal source of entropy or efficiently combined with one-way hash functions or PRNGs.

The proposed solution is very cheap. It uses very small amounts of FPLD resources and it is fast enough for typical embedded cryptographic applications. The advantage of our solution lies in the fact that the proposed TRNG block together with symmetrical and asymmetrical algorithms can fit into one FPLD chip and significantly increase the system security of an embedded cryptographic SOPC system.

The bias reduction of the TRNG can be further improved by a proper choice of parameters K_{M} and K_{D} and using more sophisticated XOR corrector. This solution is currently in development and will be presented in a future paper.

References

1. Menezes, J.A., Oorschot, P.C., Vanstone, S.A.: Handbook of Applied Cryptography, CRC Press, New York (1997)
2. Faifield, R.C., Mortenson, R.L., Coulthart, K.B.: An LSI Random Number Generator (RNG). Lecture Notes in Computer Science, Vol. 0196. Springer-Verlag, Berlin Heidelberg New York (1984) 203–230
3. Jun, B., Kocher, P.: The INTEL Random Number Generator. Cryptography Research, Inc., White Paper prepared for Intel Corporation, April 1999, 1–8, http://www.intel.com
4. APEX 20K Programmable Logic Device Family. Data Sheet, February 2002, ver. 4.3, 1–116, http://www.altera.com
5. APEX II Programmable Logic Device Family. Data Sheet, December 2001, ver. 1.3, 1-96, http://www.altera.com
6. Fischer, V., Drutarovský, M.: True Random Number Generator in Field Programmable Logic Devices. Submitted to Electronic Letters, Paper Number ELL 32365, April 2002
7. Superior Jitter management with DLLs. Virtex Tech Topic VTT013 (v1.2), January 21, 2002, 1–6, http://www.xilinx.com
8. Jitter comparison analysis: APEX 20KE PLL vs. Virtex-E DLL. Technical Brief 70, January 2001, ver.1.1, 1–7, http://www.altera.com
9. Quartus II - Programmable Logic Design Software. January 2002, ver.2.0, 1–45, http://www.altera.com
10. Davies, R.B.: Exclusive OR (XOR) and Hardware Random Number Generators. February 28, 2002, 1–11, http://webnz.com/robert/
11. Eastlake, D., Crocker, S., Schiller, J.: Randomness Recommendations for Security. Request for Comments 1750, December 1994, http://www.ietf.org/rfc/rfc1750.txt
12. Davies, R.: Hardware Random Number Generators. Paper presented to the 15th Australian Statistics Conference, July 2000, 1–13, http://statsresearch.co.nz
13. Marsaglia, G.: DIEHARD: A Battery of Tests of Randomness. http://stat.fsu.edu/ geo/diehard.html
14. Security Requirements for Cryptographic Modules. Federal Information Processing Standards Publication 140-2, U.S. Department of Commerce/NIST, 1999, http://www.nist.gov
15. Rukhin, A., Soto, J., Nechvatal, J., Smid, M., Barker, E., Leigh, S., Levenson, M., Vangel, M., Banks, D., Heckert, A., Dray, J., Vo, S.: A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications. NIST Special Publication 800-22, May 15, 2001, 1–153, http://www.nist.gov