A Tunable True Random Number Generator with an enhanced DCM for Xilinx FPGA

K. Meghana(1) & D. Raghava Reddy(2) & P. Malleswari(3)

[1]PG Scholar, Dept of ECE, Narsaraopeta Institute of Technology, Yellamanda, Narsaraopeta, Guntur, Andhra Pradesh, India.

[2]HOD, Dept of ECE, Narsaraopeta Institute of Technology, Yellamanda, Narsaraopeta, Guntur, Andhra Pradesh, India.

[3]Assistant Professor, Dept of ECE, Narsaraopeta Institute of Technology, Yellamanda, Narsaraopeta, Guntur, Andhra Pradesh, India.

Received: May 29, 2018 Accepted: July 08, 2018

ABSTRACT *True random number generators (TRNGs) play a very important role in modern cryptographic systems. Fieldprogrammable gate arrays (FPGAs) form an ideal platform for hardware implementations of many of these security algorithms. In this brief, we present a highly efficient and tunable TRNG based on the principle of beat frequency detection, specifically for Xilinx-FPGA-based applications. The main advantages of the proposed TRNG are its on-the-fly tunability through dynamic partial reconfiguration to improve randomness qualities. We describe the mathematical model of the TRNG operations and experimental results for the circuit implemented on a Xilinx Virtex-V FPGA. The proposed TRNG has low hardware footprint and built-in bias elimination capabilities. The random bitstreams generated from it pass all tests in the NIST statistical testsuite.*

Keywords: *Digital clock manager (DCM), dynamic partial reconfiguration (DPR), field-programmable gate arrays (FPGAs), true random number generator (TRNG).*

I.INTRODUCTION

TRUE random number generators (TRNGs) have become an indispensable component in many cryptographic systems, including PIN/password generation, authentication protocols, key generation, random padding, and nonce generation. TRNG circuits utilize a nondeterministic random process, usually in the form of electrical noise, as a basic source of randomness. Along with the noise source, a noise harvesting mechanism to extract the noise and a postprocessing stage to provide a uniform statistical distribution are other important components of the TRNG.

Our focus is to design improved field-programmable gate array (FPGA) based TRNGs, using purely digital components. Using digital building blocks for TRNGs has the advantage that the designs are relatively simple and well suited to the FPGA design flow, as they can suitably leverage the CAD software tools available for FPGA design. However, digital circuits exhibit comparatively limited number of sources of random noise, e.g., metastability of circuit elements, frequency of free-running oscillators, and jitters (random phase shifts) in clock signals. As would be evident, our proposed TRNG circuit utilizes the frequency difference of two oscillators and oscillator jitter as sources of randomness.

Reconfigurable devices have become an integral part of many embedded digital systems, predicted to become the platform of choice for general computing in the near future. From being mainly prototyping devices, reconfigurable systems including FPGAs are being widely employed in cryptographic applications, as they can provide acceptable to high processing rate at much lower cost and faster design cycle time. Hence, many embedded systems in the domain of security require a highquality TRNG implementable on FPGA as a component.

We present a TRNG for Xilinx-FPGA-based applications, which has a tunable jitter control capability based on dynamic partial reconfiguration (DPR) capabilities available on Xilinx FPGAs. The major contribution of this brief is the development of an architecture which allows on-the-fly tunability of statistical qualities of a TRNG by utilizing DPR capabilities of modern FPGAs for varying the digital clock manager (DCM) modeling parameters.

Digital clock manager (DCM):

A digital clock manager (DCM) is an electronic component available on some FPGAs (notably ones produced by Xilinx). A DCM is useful for manipulating clock signals inside the FPGA, and to avoid clock skew which would introduce errors in the circuit. FPGA boards grow in size and capabilities, such capabilities include the Digital Clock Manager- DCM resources.

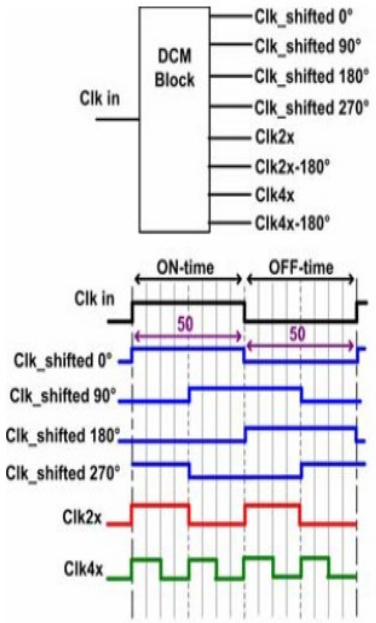


Fig.1. Digital Clock Manager.

DCM is a clock management system that enables input clock frequency duplication, multiplication and division in addition to generating four different phase shifting clock versions: 0, 90, 180 and 270, maintaining the same 50/50 ON-time/OFF-time relationship as shown in Fig. 1. Where Clk in is the input clock to the DCM block as shown in Fig, which, in turn, produces four different versions of clock phase shifts: Clk_shifted 0 , Clk_shifted 90 , Clk_shifted 180 and Clk_shifted 270°. Also the DCM gives double and four times the input clock, Clk2x and Clk4x, and along with their inverted signals, Clk2x_180 and Clk4x_180 respectively.

Uses of DCM:

DCMs have the following applications:

- Multiplying or dividing an incoming clock (which can come for outside the FPGA or from a Digital Frequency Synthesizer [DFS].
- Making sure the clock has a steady duty cycle.
- Adding a phase shift with the additional use of a Delay-locked loop.
- Eliminating clock skew within an FPGA design.

To the best of our knowledge, this is the first reported work which incorporates tunability in a TRNG. This approach is only applicable for Xilinx FPGAs which provide programmable clock generation mechanism and capability of DPR. DPR is a relatively new enhancement in FPGA technology, whereby modifications to predefined portions of the FPGA logic fabric are possible on-the-fly, without affecting the normal functionality of the FPGA. Xilinx clock management tiles (CMTs) contain a dynamic reconfiguration port (DRP) which allows DPR to be performed through much simpler means [1].

Using DPR, the clock frequencies generated can be changed on-the-fly by adjusting the corresponding DCM parameters. DPR via DRP is an added advantage in FPGAs as it allows the user to tune the clock frequency as per the need.Design techniques exist to prevent any malicious manipulations via DPR which in other ways may detrimentally affect the security of the system [2].

The goal of this brief is the design, analysis, and implementation of an easy-to-design, improved, low-overhead, and tunable TRNG for the FPGA platform. The following are our major contributions.

1) We investigate the limitations of the beat frequency detection (BFD)-TRNG [3] when implemented on an FPGA design platform. To solve the shortcomings, we propose an improved BFD-TRNG architecture suitable for FPGAbased applications. To the best of our knowledge, this is the first reported work which incorporates tunability in a fully digital TRNG.

2) We analyze the modified proposed architecture mathematically and experimentally.

3) Our experimental results strongly support the mathematical model proposed. The proposed TRNG has low hardware overhead, and the random bitstreams derived from the proposed TRNG pass all tests in the NIST statistical testsuite [4].

II. EXISTING METHOD

Single-Phase BFD-TRNG Model:

The BFD-TRNG circuit [3] is a fully digital TRNG, which relies on jitter extraction by the BFD mechanism, originally implemented as a 65-nm CMOS ASIC. The structure and working of the (single phase) BFD-TRNG can be summarized as follows, in conjunction with Fig.2.

1) The circuit consists of two quasi-identical ring oscillators (let us term them as ROSCA and ROSCB), with similar construction and placement. Due to inherent physical randomness originating from process variation effects associated with deep sub micrometer CMOS manufacturing, one of the oscillators (e.g., ROSCA) oscillates slightly faster than the other oscillator (ROSCB).

2) The output of one of the ROs is used to sample the output of the other, using a D flip-flop

(DFF). Without loss of generality, assume that the output of ROSCA is fed to the D-input of the DFF, while the output of ROSCB is connected to the clock input of the DFF.

3) At certain time intervals (determined by the frequency difference of the two ROCs), the faster oscillator signal passes, catches up, and overtakes the slower signal in phase. Due to random jitter, these capturing events happen at random intervals, called "beat frequency intervals." As a result, the DFF outputs a logic-1 at different random instances.

4) A counter controlled by the DFF increments during the beat frequency intervals and gets reset due to the logic-1 output of the DFF. Due to the random jitter, the free running counter output ramps up to different peak values in each of the count-up intervals before getting reset.

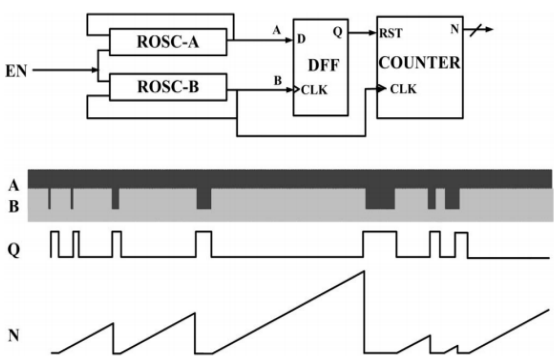


Fig.2.Architecture of the single-phase BFD-TRNG.

5) The output of the counter is sampled by a sampling clock before it reaches its maximum value.

6) The sampled response is then serialized to obtain the random bit stream.

Shortcoming of the BFD-TRNG:

One shortcoming of the previous BFD-TRNG circuit is that its statistical randomness is dependent on the design quality of the ring oscillators. Any design bias in the ring oscillators might adversely affect the statistical randomness of the bit stream generated by the TRNG. Designs with the same number of inverters but different placements resulted in varying counter maximas.

Additionally, the same ring-oscillator-based BFD-TRNG implemented on different FPGAs of the same family shows distinct counter maxima. Unfortunately, since the ring oscillators are free-running, it is difficult to control them to eliminate any design bias. The problem is exacerbated in FPGAs, where it is often difficult to control design bias because of the lack of fine-grained designer control on routing in the FPGA design fabric.

A relatively simple way of tuning clock generator hardware primitives on Xilinx FPGAs, particularly the phase-locked loop (PLL) or the DCM as used in this work, is by enabling dynamic reconfiguration via the DRPs. Once enabled, the clock generators can be tuned to generate clock signals of different frequencies by modifying values at the DRPs [1] on-the-fly, without needing to bring the device offline. We next describe the proposed tunable BFD-TRNG suitable for FPGA platforms.

III. PROPOSED METHOD

Tunable BFD-TRNG for FPGA-based applications:

Design Overview:

Fig.3 shows the overall architecture of the proposed TRNG. In place of two ring oscillators, two DCM modules generate the oscillation waveforms. The DCM primitives are parameterized to generate slightly different frequencies by adjusting two design parameters M (multiplication factor) and D (division factor).

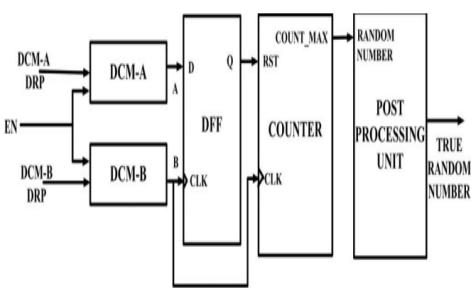


Fig.3.Overall architecture of the proposed DCM-based tunable BFD-TRNG

In the proposed design, the source of randomness is the jitter presented in the DCM circuitry. The DCM modules allow greater designer control over the clock waveforms, and their usage eliminates the need for initial calibration [3]. Tunability is established by setting the DCM parameters on-the-fly using DPR capabilities using DRP ports. This capability provides the design greater flexibility than the ring-oscillator-based BFDTRNG.

The difference in the frequencies of the two generated clock signals is captured using a DFF. The DFF sets when the faster oscillator completes one cycle more than the slower one (at the beat frequency interval). A counter is driven by one of the generated clock signals and is reset when the DFF is set. Effectively, the counter increases the throughput of the generated random numbers. The last three LSBs of the maximum count values reached by the count were found to show good randomness properties.

Additionally, we have a simple post processing unit using a Von Neumann corrector (VNC) [5] to eliminate any biasing in the generated random bits. VNC is a well-known low overhead scheme to eliminate bias from a random bit stream. In this scheme, any input bit "00" or "11" pattern is eliminated; otherwise, if the input bit pattern is "01" or "10," only the first bit is retained. The last three LSBs of the generated random number are passed through the VNC. The VNC improves the statistical qualities at the cost of slight decrease in throughput.

Tuning Circuitry:

The architecture of the tuning circuitry is shown in Fig.4. The target clock frequency is determined by the set of parameter values actually selected. The random values reached by the counter as well as the jitter are related to the chosen parameters M and D. This makes it possible to tune the proposed TRNG using the predetermined stored M and D values. As unrestricted DPR has been shown to be a potential threat to the circuit [6], the safe operational value combinations of the D and M parameters for each DCM are predetermined during the design time and stored on an on-chip block RAM (BRAM) memory block in the FPGA. There are actually two different options for the clock generators—one can use the PLL hard macros available on Xilinx FPGAs or the DCMs.

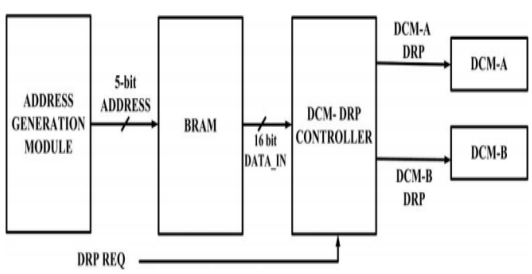


Fig.4.Architecture of tuning circuitry.

We next describe analytical and experimental results which compelled us to choose DCM in favor of the PLL modules for clock waveform generation. Mathematical model of the proposed TRNG: Circuit Behavior with PLL as Clock Generator: We first consider the operational principle for the PLL and its feasibility as a component of the proposed TRNG. The Xilinx PLL synthesizes a clock signal whose frequency is given by

$$F_{\mathrm{CLKFX}} = F_{\mathrm{CLKIN}} \cdot \frac{M}{D} \qquad (1)$$

Where FCLKIN is the frequency of the input clock signal and M and D are the multiplication and division factors previously mentioned. The values of M and D can be varied to generate the required clock frequency. The two PLLs can be parameterized with the necessary set of (M, D) values to generate two slightly different clock frequencies. Without loss of generality, assume that PLLA is set up to be slightly faster than PLLB, i.e., the time periods are related by TA < TB. On reaching the beat frequency interval (e.g., n clock cycles), by definition, PLLA completes one cycle more than the slower one.

The following equation depicts this simple model:

$$\frac{T_A}{T_B} = \frac{N}{N+1}. \qquad (2)$$

N = 2 · n, where n is the estimated maximum counter value. For the first n clock cycles, the counter does not increment and then increments by one for each of the next n clock cycles. Hence, the maximum counter values reached are n. Then, (2) leads to

$$n = \left\lfloor \frac{T_B}{2(T_B - T_A)} \right\rfloor. \qquad (3)$$

Using design configuration parameters (M and D), one of the oscillators is made to run faster than the other. This is done in order to limit the range of counter values produced. If both of the oscillators were configured to run at the same frequency, we may get random numbers, but the maximum counter value produced will be very high (theoretically infinite) as per (3).

In other words, the latency of the circuit will be very high, since the counter sets and resets only after reaching a very large count value. When the Xilinx PLLs are used as clock generators, the predicted and observed counter values for all combinations of (M, D) values remain the same. This confirms that the Xilinx PLL instances demonstrate close-toideal behavior and are quasi-identical, and have negligible jitter between the waveforms generated by them.

Since the BFDTRNG is critically dependent on the presence of jitter between the two generated clock waveforms, PLLs seem unsuitable as components of the proposed TRNG. Hence, next we examine the DCM as clock generators.

Circuit Behavior with DCM as Clock Generator:

Without loss of generality, the clock signals produced by one of the DCM (e.g., DCMA)

are slightly faster than the other (DCMB), implying TA < TB. This is ensured by assigning the design parameters M and D as in (7). Timing diagrams of the DCM clock outputs and the resultant DFF response are shown in Fig.5.

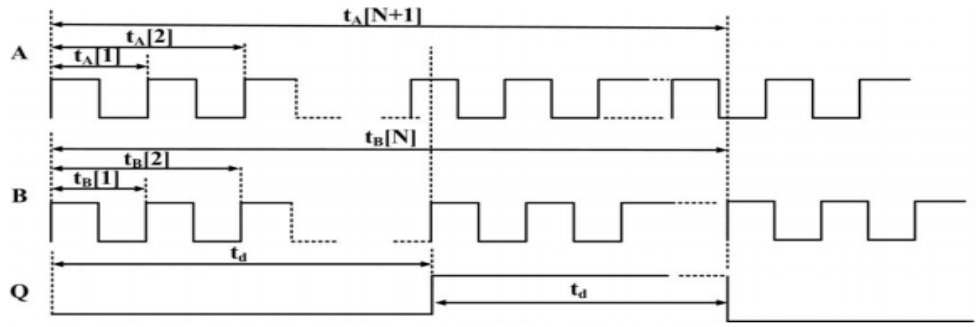


Fig.5.Timing diagram of DCM output waveforms and the corresponding and DFF response.

Let N be the number of clock cycles of the slower clock signal in which the faster clock signal completes exactly one cycle more. Then

$$t_A[N+1] = (N+1)T_A + \epsilon_A \quad (4)$$

$$t_B[N] = NT_B + \epsilon_B \quad (5)$$

Where eA and eB are the uncertainties due to jitter in DCMA and DCMB, respectively. The uncertainties due to jitter in DCMA and DCMB are different; this is because the DCMs are designed with distinct modeling parameters M and D.. In this case, DCMA is configured with M = 15 and D = 31, and DCMB is configured with M = 14 and D = 29. These results in peak-to-peak jitter values of 0.600 and 0.568 ns for DCMA and DCMB, respectively. Of course, we also have the following: tA[N + 1] = tB[N]. Assuming that there is no metastability for the DFF if signal transitions occur in the setup-hold timing window around its driving clock edge (the metastability issue can be avoided by cascaded DFF combination), the transition time (td) of the DFF, the time interval after which it sets (i.e., the counter driven by the DFF resets), is estimated by

$$t_d = \frac{t_A[N+1] + t_B[N]}{2} = \frac{(N+1)T_A + NT_B + \epsilon_A + \epsilon_B}{2}. \quad (6)$$

From (6), the transition time of DFF is a random process. The output of the DFF, i.e., the time interval (td) after which the counter resets, is thus a random function. As a result, the count value obtained when the counter resets is also a random quantity. The counter resets automatically when the DFF sets, and the operation continues. The DFF resets approximately n cycles after it sets, and the counter starts counting again.

Tuning Parameter Value Ranges:

Equations (1) and (2) also hold true for DCM-based BFD. Hence, we have the following relationships:

$$\frac{D_1 \cdot M_2}{D_2 \cdot M_1} = \frac{N}{N+1} \begin{cases} 2 \le M_i \le 33, \\ 1 \le D_i \le 32, \\ 400 \le N \le 1000, \\ M_i, D_i,\ N \in \mathbb{Z} \end{cases} \quad (7)$$

Where M and D values are as per the Xilinx DCM specification [1]. The count value to be sampled was set to be between 200 and 500; hence, the values of N are as per (7). Higher value of count is not desired as it leads to higher power dissipation.

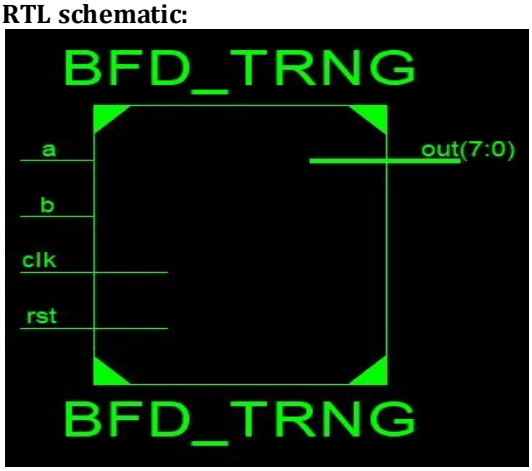
As per (7), there are 23 sets of (M, D) value combinations for the two DCMs, which satisfy the required count range. These values are stored in a BRAM, and for 23 distinct pairs, we require 5-bit address line for selecting one of the combinations of M and D values, and if the BRAM is configured to hold 16-bit words, we require 46 B of memory. The address increments to the required BRAM location where the corresponding values of the DCMB are stored on demand, using a simple address generation module. .

In this way, using a restricted DPR methodology, the designer has control over the DCM configuration to choose the best combination generating random numbers with the best statistical quality. In order to avoid malicious modifications via DPR, we have enabled DPR restrictively by storing the allowable modeling parameters. In order to implement this secure tunable design, slightly higher hardware overhead and power dissipation are required. The DCM-DRP controller initiates DPR in DCMA and DCMB using standard Xilinx design methodology [1].

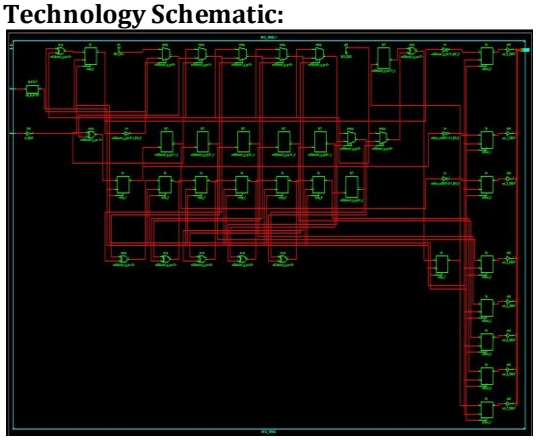
IV.RESULTS

Simulation results:

RTL schematic:



Technology Schematic:



Design summary:

Device Utilization Summary (estimated values)			
Logic Utilization	Used	Available	Utilization
Number of Slices	9	4656	0%
Number of Slice Flip Flops	16	9312	0%
Number of 4 input LUTs	11	9312	0%
Number of bonded IOBs	10	190	5%
Number of GCLKs	1	24	4%

Timing Report:

```
Data Path: m5/out_7 to out<7>
                                   Gate     Net
  Cell:in->out      fanout   Delay   Delay  Logical Name (Net Name)
  ----------------------------------------  ------------
   FDR:C->Q             1   0.514   0.357  m5/out_7 (m5/out_7)
   OBUF:I->O                3.169          out_7_OBUF (out<7>)
  ----------------------------------------
  Total                     4.040ns (3.683ns logic, 0.357ns route)
                                    (91.2% logic, 8.8% route)
```

VI. CONCLUSION

In this paper, we have presented an improved fully digital tunable TRNG for FPGA-based applications, based on the principle of BFDand clock jitter, and with built-in error-correction capabilities. The TRNG utilizes this tunability feature for determining the degree of randomness, thus providing a high degree of flexibility for various applications. The proposed design successfully passes all NIST statistical tests.

REFERENCES

1. Virtex-5 FPGA Configuration User Guide UG 191 (v3.11) Xilinx Inc., San Jose, CA, USA, Accessed:May2016.[Online].Available:www.xilinx.com/support/documentation/user_guides/ug191.pdf .
2. A. P. Johnson, R. S. Chakraborty, and D. Mukhopadhyay, "A PUF-enabled secure architecture for FPGA-based IoT applications," IEEE Trans. MultiScaleComput. Syst., vol. 1, no. 2, pp. 110–122, Apr.–Jun. 1, 2015.
3. Q. Tang, B. Kim, Y. Lao, K. K. Parhi, and C. H. Kim, "True random number generator circuits based on single- and multi-phase beat frequency detection," in Proc. IEEE Custom Integr. Circuits Conf., Sep. 2014, pp. 1–4.
4. A. Rukhin, J. Soto, J. Nechvatal, M. Smid, and E. Barker, "A statistical test suite for random and pseudorandom number generators for cryptographic applications," Nat. Inst. Standards Technol. (NIST), Gaithersburg, MD, USA, DTIC Document, Tech. Rep., 2001.
5. J. Von Neumann, "Various techniques used in connection with random digits," Nat. Bureau Standards Appl. Math. Ser., vol. 12, pp. 36–38, 1951.
6. A. P. Johnson, S. Saha, R. S. Chakraborty, D. Mukhopadyay, and S. Gören, "Fault attack on AES via hardware Trojan insertion by dynamic partial reconfiguration of FPGA over Ethernet," in Proc. 9th WESS, Oct. 2014, pp. 1–8.
7. A. P. Johnson, R. S. Chakraborty, and D. Mukhopadhyay, "A novel attack on a FPGA based true random number generator," in Proc. 10th WESS, Oct. 2015, pp. 1–6.5.