A High-Speed FPGA-Based True Random Number Generator Using Metastability With Clock Managers

Fabio Frustaci, *Senior Member, IEEE*, Fanny Spagnolo, *Member, IEEE*, Stefania Perri, *Senior Member, IEEE*, and Pasquale Corsonello, *Member, IEEE*

***Abstract*—True random number generators (TRNGs) are fundamentals in many important security applications. Though they exploit randomness sources that are typical of the analog domain, digital-based solutions are strongly required especially when they have to be implemented on Field Programmable Gate Array (FPGA)-based digital systems. This brief describes a novel methodology to easily design a TRNG on FPGA devices. It exploits the runtime capability of the Digital Clock Manager (DCM) hardware primitives to tune the phase shift between two clock signals. The presented auto-tuning strategy automatically sets the phase difference of two clock signals in order to force on one or more flip-flops (FFs) to enter the metastability region, used as a randomness source. Moreover, a novel use of the fast carry-chain hardware primitive is proposed to further increase the randomness of the generated bits. Finally, an effective on-chip post-processing scheme that does not reduce the TRNG throughput is described. The proposed TRNG architecture has been implemented on the Xilinx Zynq XC7Z020 System on Chip (SoC). It passed all the National Institute of Standards and Technology (NIST) SP 800-22 statistical tests with a maximum throughput of 300 × 10^6 bit per second. The latter is considerably higher than the throughput of other previously published DCM-based TRNGs.**

***Index Terms*—True Random Number Generator (TRNG), FPGA, Digital Clock Manager (DCM), metastability.**

I. Introduction

RANDOM numbers are used in many important security applications, such as the generation of secret and public keys, nonces and challenges in authentication protocols and initialization vectors for cryptographic systems. For this reason, in the last few years the importance of hardware-based True Random Number Generators (TRNGs) has become crucial. A TRNG is a hardware system that generates a sequence of random numbers without the need of an initial seed and it is based on the unpredictability of some physical phenomena such as thermal noise, phase noise and quantum phenomena that are typically exploited in analog circuits [1]. On the other hand, a lot of effort has been recently focused towards the design of fully digital TRNGs to be implemented on Field Programmable Gate Arrays (FPGAs) due to their widely adoption [2], [3], [4], [5], [6], [7], [8], [9], [10], [11], [12], [13], [14], [15], [16], [17]. Typically, the existing designs rely on metastability [2], [3], [4] and jitter [5], [6], [7], [8], [9], [10], [11], [12], [13], where the randomness is commonly extracted by Look-up Table (LUT)- based Programmable Delay Lines (PDLs) and free running Ring Oscillators (ROs). The use of two PDLs driving the data and the clock inputs of a Data Flip-Flop (FF) is proposed in [2]. There, the PDLs are designed as series-connected LUTs each one implementing an inverter. The delay of each LUT is modulated by setting its unused inputs to appropriate values, without interfering with the implemented logic. By carefully tuning the delay difference between the two PDLs, the setup or hold time of the FF can be violated, thus forcing the FF to enter its metastable region. Alternative approaches are used in [5] and [10], where the output of several ROs are fed to a XOR gate whose output is sampled by a FF. In the presence of a phase jitter, the FF may sample glitches with a non-deterministic length, thus producing a random bit. The architecture demonstrated in [9] exploits a LUT-based tunable delay chain in conjunction with a time-to-digital converter consisting of 32 8-bit series-connected fast carry chains. The physical implementations of such schemes require a careful analysis of the appropriate number of ROs and inverters within each RO, and need specific tuning of the PDLs delays, that could be significantly affected by the routing. Aside from the used technique, most of the above-mentioned architectures require a post-processing block and a feedback control to achieve adequate randomness.

Recently, the use of clock managers has been investigated as an effective way to more easily design TRNGs on FPGA devices [14], [15]. These hardware primitives, which aim at producing one or more output clock signals with a programmable frequency, are typically available in modern FPGAs: the Xilinx devices provide the Digital Clock Manager (DCM) [18], whereas the Clock Manager (CM) is available within Intel [19] chips. In [14], [15], the Dynamic Partial Reconfiguration port (DRP) of DCMs is used to configure their parameters on the fly, thus tuning the frequency of signals driving the data and the clock input of a FF. Unfortunately, such strategies lead to a relatively slow random bit production rate, since hundreds of clock cycles elapse between two consecutive samplings.

Manuscript received 29 July 2022; revised 21 September 2022; accepted 29 September 2022. Date of publication 3 October 2022; date of current version 9 February 2023. The work of Fanny Spagnolo was supported by "PON Ricerca & Innovazione—Ministero dell'Università e della Ricerca under Grant 1062_R24_INNOVAZIONE. This brief was recommended by Associate Editor M. E. Yalcin. *(Corresponding author: Pasquale Corsonello.)*

Fabio Frustaci, Fanny Spagnolo, and Pasquale Corsonello are with the Department of Informatics, Modelling, Electronics and Systems Engineering, University of Calabria, 87036 Rende, Italy (e-mail: f.frustaci@dimes.unical.it; f.spagnolo@dimes.unical.it; p.corsonello@unical.it).

Stefania Perri is with the Department of Mechanical, Energy and Management Engineering, University of Calabria, 87036 Rende, Italy (e-mail: s.perri@unical.it).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TCSII.2022.3211278.

Digital Object Identifier 10.1109/TCSII.2022.3211278

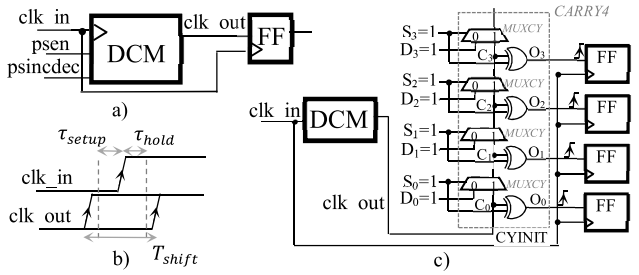

Fig. 1. a) The proposed DCM-based metastability generator; b) timing example; c) the joint proposed use of the CARRY4 and DCM.

This brief describes an alternative use of the clock manager hardware primitive to design a high-speed TRNG. In place of tuning the frequency of the output clocks, the proposed design exploits its dynamic phase shifting (DPS) capability to force one or more FFs to enter the metastability region. The proposed solution has the ability to automatically tune the phase shifting of the DCM so that the random sequence generation automatically starts when this condition occurs. To enhance the randomness, we also propose an unconventional utilization of the carry-chain primitive included in the FPGAs slice with a configurable feedback scheme. Finally, a simple on-chip post processing scheme is proposed that exploits only one Digital Processing Signal (DSP) slice without reducing the bit production rate. When implemented on the Xilinx Zynq XC7Z020 System on Chip (SoC), equipped with a 28nm Artix®-7 based programmable logic, the proposed TRNG passes all the National Institute of Standards and Technology (NIST) SP 800-22 [20] and AIS statistical tests [21] with a throughput up to three orders of magnitude higher than the DCM-based solutions [14], [15].

II. The Proposed Use of the Xilinx DCM

The Xilinx DCM offers the opportunity to dynamically tune the phase of the output clock at run time, without reconfiguring the FPGA device [18]. To this aim, the primitive uses two input signals, *psen* and *psincdec*, as described in Fig. 1a. When *psen* is asserted high, the dynamic configuration starts and the phase of the output clock is incremented/decremented if the value of *psincdec* is high/low. The phase shift resolution T_{shift} only depends on the frequency F_{VCO} of the voltage-controlled oscillator (VCO) internal to the DCM and it is equal to $\frac{1}{56F_{VCO}}$ [18]. Fig. 1a also describes the proposed idea of using the DCM with the purpose to ingenerate metastability in a FF. The system clock, i.e., the DCM input clock *clk_in*, drives the clock input of a FF. The DCM produces the output clock *clk_out* with the same frequency that drives the data input of the FF. To force the FF entering the metastable region, *clk_in* and *clk_out* should arrive almost simultaneously at the FF inputs, thus violating the FF setup/hold timing constraints. However, due to their different routing paths, the arrival times of data and clock edges may significantly differ, thus preventing the metastability occurrence. The DCM phase shift is exploited to compensate this difference in routing delays.

It is worth noting that if T_{shift} is too large, as depicted in Fig. 1b, the rising edge of *clk_out* arrives either too earlier or too later than the rising edge of *clk_in*. That means that the phase shift resolution achieved by the DCM could be not as fine as required. For this reason, similarly to [9], we have also used a carry chain primitive and four FF replicas. This allows a much finer phase shift resolution control at the destination to be achieved and prevents the metastability region is sometimes skipped. Fig. 1c depicts the application of this principle with the CARRY4 hardware primitive within the slice of the Xilinx devices. Four FFs receive as data input the sum outputs ($O_{[3:0]}$) of a CARRY4 chain. The signal *clk_out* drives the CARRY4 input signal C_{YINIT}. The generic signal O_{i+1} is delayed with respect to O_i by the propagation delay τ_{MUX} of the multiplexer in-between the output positions i and $i+1$, with $i = 0 \ldots 2$. In the event that T_{shift} is too large to ingenerate metastability, it is reasonable to assume that $\tau_{MUX} < T_{shift}$, hence a finer phase shifting of the FF data inputs is obtained.[1] As a consequence, the probability that at least one FF enters the metastability region increases. The propagation of the signal *clk_out* through the four bit positions of the carry chain is assured by setting the CARRY4 input signal $S_{[3:0]}$ to logic 1 (the signals $D_{[3:0]}$ do not affect the carry propagation and they can be set, as instance, to logic 1). Finally, it is worth noting that the CARRY4 primitive and the four FFs can be placed within the same slice, so that the differences of the propagation delays of the signals $O_{[3:0]}$ can be considered independent of the routing between the XOR gates and the FFs but only affected by the propagation delays of the multiplexers in the carry chain.

III. The Proposed TRNG Architecture

Fig. 2 illustrates the complete TRNG core and its integration within a specialized testing architecture realized on a Heterogeneous FPGA System on Chip (SoC). From a top-level perspective, it can be seen that the random sequence generated by the TRNG core is transferred towards the external DRAM memory by means of AXI-Stream transactions managed by a Direct Memory Access (DMA) module. The latter is configured by the ARM®-based processing system via dedicated AXI-Lite interfaces.

The DCM-based scheme discussed in Section II is completed with a XOR gate that merges the output of the four FFs. Thus, it does not matter which of them became metastable. The output of the XOR is then sampled by an FF (FF_{XOR}) clocked by the system clock (i.e., *ck_in*). If the phase difference between *clk_in* and *clk_out* is high enough to avoid metastability (due to either the initial DCM state or an unpredictable difference of the signals routing), the four FFs of the randomness generator sample the same stable value and the output T of FF_{XOR} is 0. The signal T is the input of a Finite State Machine (FSM) that controls the configuration signals of the DCM in a feedback fashion. If T is 0, the FSM starts a phase shifting transaction with the DCM by asserting *psen* to 1 for one clock cycle (*psincdec* can be constantly set to 0). Then, the FSM waits for a fixed number N of clock cycles (here $N = 20$) still monitoring the value of the signal T. If T continues to be 0 during the N clock cycles, a new phase shifting transaction is activated. This auto-calibration procedure of the phase difference between *clk_in* and *clk_out* ends when

[1] As an example, in the Xilinx Kintex-7 FPGAs, τ_{MUX} is about 15ps [22].

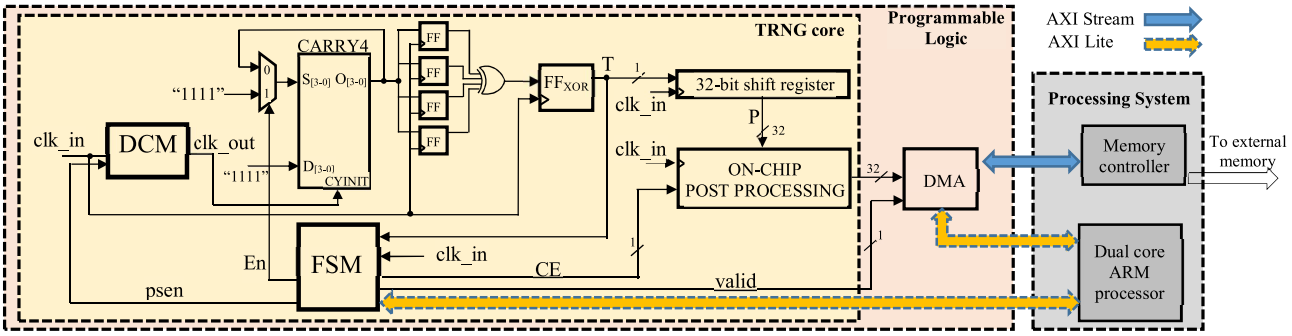

Fig. 2. The proposed TRNG core and the adopted testing architecture.

T becomes 1, as a reasonable sign that one FF has entered its metastable region. In such a case, the FSM stops shifting the phase of *clk_out* and a signal *En* is set to 0 starting the acquisition of the random bit-stream. A preliminary analysis, on different placement sites, demonstrated that after the auto-calibration which takes 160 clock cycles on average, at least one of the fours FFs actually enters the metastable region. Indeed, over a 10Mb sequence outputted by the XOR gate, the percentage of 0's and 1's is close to 50%. In the proposed scheme, the signal T represents the raw random bit that is generated with a throughput equal to the system clock frequency. With the goal of increasing the randomness of the signal T, a further technique is here adopted in conjunction with the use of the DCM. As visible in Fig. 2, the signals $O_{[3:0]}$ are in a feedback loop to drive the selectors $S_{[3:0]}$ of the multiplexers of the carry chain. When the signal *En* is 1, the auto-calibration phase is still running and $S_{[3:0]}$ is set to "1111", as explained above. Once metastability has been ingenerated, *En* is set to 0 and $S_{[3:0]}$ is set to $O_{[3:0]}$. Such a selection is performed by four multiplexers, controlled by *En*, whose logic can be implemented by the four Look-up Tables (LUTs) within the same slice of the CARRY4. The purpose of the proposed scheme is to force the XOR gates of the CARRY4 in a race condition. The generic XORi gate in the carry chain, with $i = 1 \ldots 3$, receives as inputs the output signal O_i and the carry C_i propagating from the previous position, whose value depends on O_{i-1}: if $O_{i-1} = 0$ then $C_i = D_{i-1}$ (i.e., logic 1), otherwise $C_i = C_{i-1}$. The first XOR gate in the chain, i.e., the one having O_0 as output, has *clk_out* and the same O_0 as inputs. It is worth noting that, differently from the conventional ring oscillators [5], each XOR gate oscillates with different phase due to the delay path composed by MUXCY. Furthermore, the state of each oscillator depends on the state of the previous one.

The random bit T is then inputted to a shift register that aggregates 32 consecutive bits into a 32-bit single packet P. The latter is then elaborated by the on-chip post-processing module to eliminate any bias in the generated random sequence [2], [7], [14], [15]. To avoid the detrimental effects on the generation rate caused by traditional Von Neumann corrector, we adopted the simple post-processing methodology detailed in Fig. 3. The packet P is inputted to a 32-bit accumulator. With P_i being the i-th 32-bit packet, the accumulation result is $P_{i,ACC} = (P_i + P_{i-1,ACC})(mod\ 2^{32})$. This logic can be easily accommodated into one of the Digital Signal Processing (DSP) slices also available in modern FPGAs. The

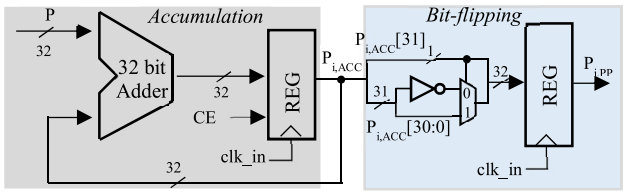

Fig. 3. The proposed DSP-based post-processing circuit.

FSM activates the accumulator register through the *CE* signal only once 32 consecutive random bits have been collected by the shift register. To reduce the number of any repetitive patterns, a dynamic bit-flipping is performed. Towards this aim, the final post-processed 32-bit packet $P_{i,PP}$ is generated as given in (1):

$$P_{i,PP}[31] = P_{i,ACC}[31]$$
$$\begin{cases} P_{i,PP}[30:0] = P_{i,ACC}[30:0] & \text{if } P_{i,ACC}[31] = 1 \\ P_{i,PP}[30:0] = not\big(P_{i,ACC}[30:0]\big) & \text{if } P_{i,ACC}[31] = 0 \end{cases} \quad (1)$$

It is worth noting that the proposed bit-flipping mechanism does not introduce any periodic feature in the generated bit-stream since it depends on the value of the random bit $P_{i,ACC}$ [31]. The FSM asserts the signal *valid*, thus enabling the AXI-Stream-based transfer, only when a new 32-bit packet P_{PP} is produced.

IV. Preliminary Analysis

All the physical experiments discussed in the following have been performed on the Xilinx Zynq XC7Z020 SoC of a Zedboard™ board, equipped with a 28nm Artix®-7 based programmable logic and a general-purpose ARM®-based processor. First of all, we separately analyze the impacts of the various TRNG components (i.e., the DCM phase shifting, the race condition on CARRY4, the Post Processing) on the sequence randomness, by selectively enabling/disabling them and analyzing 10M bits random sequences with the AIS-T8 entropy test. Results collected in Table I show that the effect of each component is additive: the bit entropy obtained by the joint application of two or more techniques is always higher than that obtained by each one when applied separately. The DCM shifting functionality has the highest impact on the sequence randomness compared to the race condition on CARRY4, which itself alone is not enough to achieve a significant entropy. As expected, the proposed Post Processing scheme notably increases the bit entropy; however, it is able

TABLE I
EFFECT OF THE PROPOSED TRNG ENTROPY SOURCES

Technique	Bit Entropy
DCM	0.717
DCM + Post Processing	0.917
Race condition on CARRY4	0.220
Race condition + Post Processing	0.945
DCM + race condition	0.917
DCM + race condition + Post Processing	0.99941

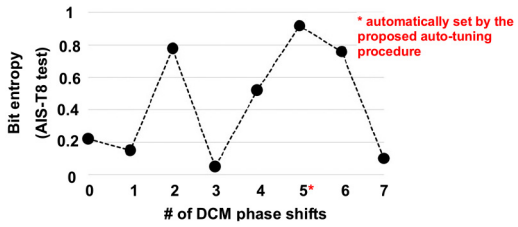

Fig. 4. Bit entropy for the signal T for different number of DCM phase shifts.

to let the TRNG exceed the minimum pass threshold (i.e., 0.987 dictated by the T8 test) only when the bit entropy of the raw sequence is already significantly high (i.e., when the DCM and race condition on CARRY4 are both enabled).

Fig. 4 depicts the bit entropy of the raw bitstream collected at the node T for different number of phase shifts of the DCM. Interestingly, the entropy of such a sequence shows a maximum value corresponding exactly to the number of phase shifts performed by the DCM when the proposed auto-tuning procedure is controlled by the FSM, thus confirming the validity of our approach.

To investigate the start-up behavior of the proposed TRNG, the autocorrelation of 200 1Mbit different start-up sequences and the correlation between all the possible sequences pairs have been calculated. Fig. 5 shows three examples of auto-correlation histograms for different lags and the obtained correlation matrix. The maximum absolute value of all the autocorrelation coefficients (correlation matrix) has been found to be merely 0.014 (0.02), thus proving the correct start-up behavior of the proposed TRNG.

V. EXPERIMENTAL RESULTS

One thousand consecutive 1M-bit random sequences have been generated and acquired for a total of 1G random bits at different temperatures, by means of a temperature chamber. In first experiments, the maximum clock frequency achieved by the DMA (i.e., 100MHz) has been used for the entire architecture of Fig. 2. The sequences have been then analyzed with the NIST SP 800-22 and AIS statistical tests. All tests have been successfully passed[2] at all the operating conditions. Most relevant results are collected in Fig. 6. It shows that the proposed TRNG design is not systematically influenced by the temperature. Tests performed with the NIST SP800-90B suite also shown that our TRNG achieves the minimum (maximum) H_∞ entropy of 0.937 for the *ttuple* test (0.998 for the *multi mmc prediction* test), while its mean H_∞ entropy over all the tests is 0.979.

[2]One thousand sequences globally pass a test if the percentage of 1M-bit sequences passing the test is$\geq$ 98% and if the $P-\text{value}_\text{T}$ is $\geq$0.0001.

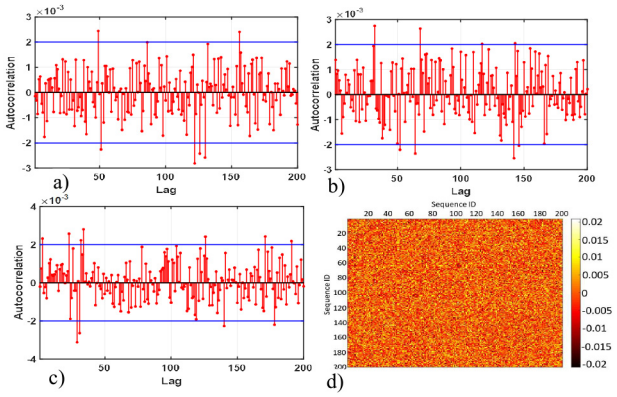

Fig. 5. (a)-(c) Autocorrelation histograms; (d) correlation matrix for 200 start-up sequences.

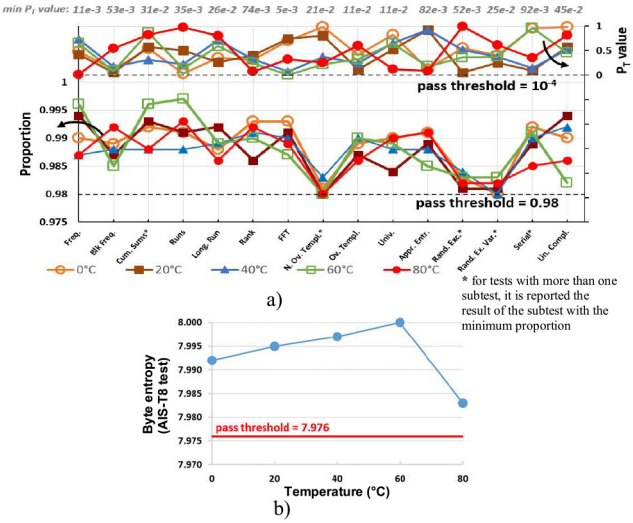

Fig. 6. (a) NIST SP800-22 and (b) AIS-T8 tests temperature analysis.

Then, to prove the correct running of the proposed TRNG also at its 300MHz maximum frequency, the architecture of Fig. 2 has been provided with two separate clock domains to guarantee that, while the TRNG core is clocked at 300 MHz, the DMA of the testing infrastructure operates at 100MHz. Also in this case, all the NIST SP 200-22 tests have been passed with a minimum proportion of 98.1%.

Table II collects results, obtained for several competitors, in terms of the resources requirements, the maximum operating frequency (O.F.), the metric $(Thr \cdot /(Slice \cdot OF)$ introduced in [4], the worst proportion value and the entropy obtained from NIST SP 800-22 tests and AIS T8, respectively. First of all, it is worth noting that among the compared designs only [5], [7], [8] and the proposed solution pass all the NIST SP 800-22 tests. Moreover, our TRNG achieves the highest throughput and one of the highest $(Thr \cdot /(Slice \cdot OF)$ value, which is more than one magnitude order higher than those shown by the other DCM-based technique [14]. Only [12] shows a higher value of the comparison metric (+13%), but its throughput is considerably lower than the proposed TRNG (−96%). The bit entropy of the proposed TRNG is comparable to that achieved by the competitors, and even better at the 100MHz clock frequency. The TRNG described in [13] has a comparable maximum throughput but its proportion is only 70%.

TABLE II
COMPARISON RESULTS

	Area			Thr.	O.F.	Thr.	Prop.	Bit entr.	Platform
	Lut	FF	Slice	[10^6 bps]	[MHz]	Slice · OF		(AIS T8)	
[2]	128	-	32	2	16.4	3.81e-3	90%	-	Virtex 5
[5]	528	177	270	6	24	9.26e-4	98.7%	0.9993	Spartan 3A
[15]	18	17	-	0.209	100	-	7.5%	-	Virtex 5
[14]	26	19	14	0.419	75.8	3.95e-4	80%	-	Virtex 5
[6]	32	48	-	4	1	-	97.5%	0.9999	Virtex 5
[7]	56	19	-	100	400	-	98%	-	Virtex 6
[11]	-	-	-	100	400	-	-	0.9900	Virtex 5
[13]	24	2	-	290	290	-	70%	-	Virtex 6
[4]	4	3	1	0.8	50	1.6e-2	97%	0.9998	Spartan 6
[12]	32	55	33	12.5	12.5	3.03e-2	-	0.9995	Xilinx Zynq-7000
[8]	866	-	-	7.3	80	-	98.1%	-	
This work	38	121	38	300	300	2.63e-2	98.1%	0.9986	
				100	100	2.63e-2	98%	0.9994	

TABLE III
POWER COMPARISON

Technique	O.F. [MHz]	Power [mW]	Power/Thr. [nJ/bit]
[14]	75.8	238	568
[12]	12.5	9.5	0.76
This work	300	119	0.4

Table III reports a comparison in terms of power dissipation. A direct comparison is possible only with the architecture in [12], and results show that the energy per generated bit consumed by the proposed TRNG is about 47% lower than [12].

As a case study, we applied our TRNG output for the generation of random one-time passwords (OTPs), each one composed by 10 ASCII symbols. We collected 40 sequences of 80 bits (i.e., 40 OTPs) to test their safety on the website [23]. All the OTPs were classified as "very strong password" with a claimed mean hacking time of 14M years with a home computer.

VI. CONCLUSION

A new design of a DCM-based TRNG for an easy implementation on FPGA devices has been presented. It exploits the dynamic capability of the DCMs hardware primitives to fine tune the phase difference between two clock signals. The metastability ingenerated by the latter signals is used as a randomness source. The required phase difference is automatically set by a simple FSM. A smart use of the CARRY4 hardware primitive further increases the randomness of the generated bits. Finally, a low-latency on-chip post-processing scheme is also presented. The proposed TRNG architecture has been implemented on the Xilinx Zynq XC7Z020 System on Chip (SoC). It passed all the NIST SP 800-22 and AIS tests, showing a throughout that is considerable higher than those of previously published DCM-based TRNGs.

REFERENCES

[1] M. Drutarovsky and P. Galajda, "A robust chaos-based true random number generator embedded in reconfigurable switched-capacitor hardware," in *Proc. 17th Int. Conf. Radioelektronika*, Apr. 2007, pp. 1–6.

[2] M. Majzoobi, F. Koushanfar, and S. Devadas, "FPGA-based true random number generation using circuit metastability with adaptive feedback control," in *Proc. Crypt. Hard. Embedded Syst*, 2011, pp. 17–32.

[3] H. Hata and S. Ichikawa, "FPGA implementation of metastability-based true random number generator," *IEICE Trans. Inf. Syst.*, vol. E95-D, no. 2, pp. 426–436, Feb. 2012.

[4] R. D. Sala, D. Bellizia, and G. Scotti, "A novel ultra-compact FPGA-compatible TRNG architecture exploiting latched ring oscillators," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 69, no. 3, pp. 1672–1676, Mar. 2022.

[5] N. N. Anandakumar, S. K. Sanadhya, and M. S. Hasmi, "FPGA-based true random number generation using programmable delays in oscillator-rings," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 67, no. 3, pp. 570–574, Mar. 2020.

[6] H. Martin, P. Peris-Lopez, J. E. Tapiador, and E. San Millan, "A new TRNG based on coherent sampling with self-timed rings," *IEEE Trans. Ind. Informat.*, vol. 12, no. 1, pp. 91–100, Feb. 2016.

[7] X. Wang *et al.*, "High-throughput portable true random number generator based on jitter-latch structure," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 68, no. 2, pp. 741–750, Feb. 2021.

[8] K. Demir and S. Ergün, "Random number generators based on irregular sampling and Fibonacci–Galois ring oscillators," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 66, no. 10, pp. 1718–1722, Oct. 2019.

[9] X. Li, P. Stanwicks, G. Provelengios, R. Tessier, and D. Holcomb, "Jitter-based adaptive true random number generation for FPGAs in the cloud," in *Proc. Int. Conf. Field-Programmable Technol. (ICFPT)*, 2020, pp. 112–119.

[10] D. Schellekens, B. Preneel, and I. Verbauwhede, "FPGA vendor agnostic true random number generator," in *Proc. Int. Conf. Field Program. Logic Appl.*, Aug. 2006, pp. 1–6.

[11] A. Cherkaoui, V. Fischer, L. Fesquet, and A. Aubert, "A very high speed true random number generator with entropy assessment," in *Proc. Cryptogr. Hardw. Embedded Syst.*, 2013, pp. 1–18.

[12] M. Grujić and I. Verbauwhede, "TROT: A three-edge ring oscillator based true random number generator with time-to-digital conversion," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 69, no. 6, pp. 2435–2448, Jun. 2022.

[13] J. Cui et al., "Design of true random number generator based on multi-stage feedback ring oscillator," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 69, no. 3, pp. 1752–1756, Mar. 2022.

[14] A. P. Johnson, R. S. Chakraborty, and D. Mukhopadyay, "An improved DCM-based tunable true random number generator for Xilinx FPGA," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 64, no. 4, pp. 452–456, Apr. 2017.

[15] N. Fujieda, M. Takeda, and S. Ichikawa, "An analysis of DCM-based true random number generator," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 67, no. 6, pp. 1109–1113, Jun. 2020.

[16] Y. Liu, R. C. C. Cheung, and H. Wong, "A bias-bounded digital true random number generator architecture," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 64, no. 1, pp. 133–144, Jan. 2017.

[17] L. B. Carreira, P. Danielson, A. A. Rahimi, M. Luppe, and S. Gupta, "Low-latency reconfigurable entropy digital true random number generator with bias detection and correction," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 67, no. 5, pp. 1562–1575, May 2020.

[18] *UG472: 7 Series FPGAs Clocking Resources User Guide*, Xilinx, San Jose, CA, USA, Jul. 2018.

[19] *Cyclone IV Device Handbook*, vol. 1, Altera Corp., San Jose, CA, USA, Mar. 2016.

[20] L. E. Bassham et al., "A statistical test suite for random and pseudorandom number generators for cryptographic applications, Rev. 1a," U.S. Dept. Commerce, Nat. Inst. Stand. Technol., Gaithersburg, MD, USA, Rep. SP 800–22, 2010.

[21] W. Killmann and W. Schindler, *A Proposal for: Functionality Classes for Random Number Generators, Version 2.0; Mathematical-Technical Reference of AIS 20/31*, NIST, Gaithersburg, MA, USA, 2011.

[22] "Kintex-7 FPGAs Data Sheet: DC and switching characteristics," Data Sheet DS182, Xilinx, San Jose, CA, USA, Apr. 2011.

[23] "How secure is your password?" Accessed: Nov. 7, 2022. [Online]. Available: https://www.passwordmonster.com

Open Access funding provided by 'Universitá della Calabria' within the CRUI CARE Agreement