This is the accepted version of the following article: Naoki Fujieda and Sogo Takashima: Enhanced use of mixed-mode clock manager for coherent sampling-based true random number generator, 8th International Symposium on Computing and Networking Workshops (CANDARW 2020), pp. 197–203 (11/2020), which has been published in final form at https://doi.org/10.1109/CANDARW51189.2020.00047.
The article was presented at 12th International Workshop on Parallel and Distributed Algorithms and Applications (PDAA-12), a workshop of CANDAR 2020.

Enhanced use of mixed-mode clock manager for coherent sampling-based true random number generator

Naoki Fujieda† and Sogo Takashima
Department of Electrical and Electronics Engineering, Faculty of Engineering,
Aichi Institute of Technology, Toyota, Aichi, Japan
†nfujieda@aitech.ac.jp

***Abstract*—A True Random Number Generator (TRNG) using a pair of Phase-locked Loops (PLLs) or Digital Clock Managers (DCMs) has an advantage of minimal usage of logic elements of FPGA. This paper focuses on a new clocking element of Xilinx FPGAs called mixed-mode clock manager (MMCM) and presents its effective use for better randomness and higher throughput. According to an evaluation on an Artix-7 FPGA using 132 sets of parameters, 89 sets passed AIS-31 Procedure B with the proposed method, while only 9 sets passed when simply porting an existing DCM-based method. The average throughput was 1.18 Mbit/s, which was 5x faster than an existing DCM-based method.**

I. Introduction

In secure computer systems, a true random number generator (TRNG) is important to obtain unpredictable random numbers. They are used as, for example, an encryption key and nonce of challenge-response protocols. A TRNG utilizes a physical phenomenon as a source of entropy, of which the AIS-31 standard [5] requires an appropriate stochastic model.

There are some types of TRNGs that are suitable for FPGA (field programmable gate array) implementations and compliant with the AIS-31 [10]. They use physical phenomena of internal logic or complementary elements, while thermal noise of resisters [16] or transistors [8] is often used in ASIC (application-specific integrated circuit) implementations. Coherent sampling [1], [6], [7] is one of the operating principles of them.

Coherent sampling-based TRNGs with clocking elements, such as phase-locked loop (PLL) [2] and digital clock manager (DCM) [4], have an advantage of being implementable with a minimal number of logic elements. Coherent sampling requires two clock signals that have slightly different frequencies. Instead of using two ring oscillators, they use frequency synthesized signals from one oscillator or an external clock input. They require two clocking elements but fewer logic elements. In general, an FPGA-based computer system requires a large number of logic elements, while most of clocking elements remain unused. They let precious logic resources use for other parts of the system.

This research focuses on a new clocking element of Xilinx FPGAs called MMCM (mixed-mode clock manager) [14]. It is available for Virtex-6 and 7 series (or newer) FPGAs and a PLL for these FPGAs is its subset [14]. It enables finer multiplying and dividing factors to be set as parameters than earlier DCMs. Since existing methods targeted Intel (Altera) FPGAs or earlier Xilinx FPGAs (such as Virtex-5 [3], [4]), it is not obvious whether they can be adopted to recent Xilinx FPGAs in the same manner.

In this paper, we propose an effective use of MMCMs for coherent sampling-based TRNG. In particular, we present a novel way of selecting parameter sets of MMCMs that maximize entropy and bit rate of generation. We first demonstrate that simply porting an existing DCM-based TRNG [4] does not offer enough entropy. We then show that the proposed method gives random number sequences that pass the statistical tests of AIS-31 in most cases. The most important contribution of this paper is to make TRNGs with clocking elements available for recent Xilinx FPGAs.

The organization of this paper is as follows. A brief explanation of the basis of our research, coherent sampling

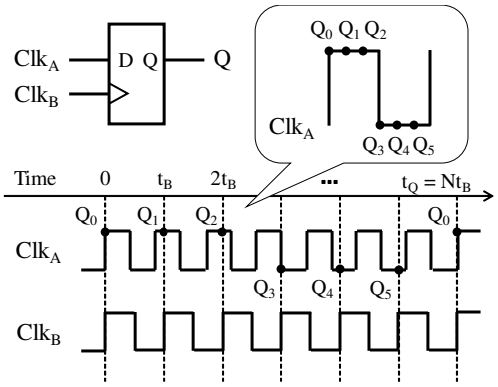

Fig. 1. Example of coherent sampling where the frequency ratio is 7 : 6.

and MMCM, is presented in Section II. Section III describes porting of the existing DCM-based method, while Section IV presents the proposed selection of parameter sets of MMCMs. The quality of random numbers, the generation bit rate, and the amount of hardware of the MMCM-based TRNG are evaluated in Section V. We conclude the paper in Section VI.

II. Background

A. Principle of Coherent Sampling

Figure 1 depicts the operating principle of coherent sampling with an example. It requires two clock signals that have slightly different frequencies. These signals, $\mathrm{Clk_A}$ and $\mathrm{Clk_B}$, are given to the data and clock input ports of a D flip-flop (D-FF), respectively. For ease of explanation, we assume that the ratio of the frequencies is $f_A : f_B = (N+1) : N$. The example of Fig. 1 shows the case of the ratio of 7 : 6. The periods of the respective clocks are denoted by t_A and t_B.

Suppose that the both clock signals rise at time zero. The time when the both clocks rise at the same time again will be $t_Q = Nt_B$. As $\mathrm{Clk_B}$ goes slightly slower than $\mathrm{Clk_A}$, it can effectively capture a waveform of a single cycle of $\mathrm{Clk_A}$ with N samples. As a result, the output of the D-FF becomes a series of consecutive '1's and consecutive '0's. The expected value of the number of consecutive '1's is $N/2$ (if the duty cycle of $\mathrm{Clk_A}$ is $1/2$).

TRNGs utilize the jitter of these clock signals. When the vicinity of edge of $\mathrm{Clk_A}$ (such as Q_0 and Q_3 in Fig. 1) is captured, the output of the D-FF may vary with slight time jitter. Also, when the both edges come quite close, D-FF may fall into a metastable state due to timing violation, which results in random output. These phenomena make the actual number of consecutive '1's uncertain. Its LSB (least significant bit) can be used as a source of entropy, which is obtained by a T flip-flop (a D-FF and an XOR gate). To harvest enough entropy, the jitter σ_J should be sufficiently larger than the difference of the periods $t_d = t_B - t_A = t_A/N$. Without considering the effect of metastability, the standard deviation of the number of '1's is proportional to σ_J/t_d.

B. Coherent Sampling-based TRNG

One of the methods to obtain clock signals for coherent sampling-based TRNGs is to use two ring oscillators [6], [9], [12]. Even though the oscillators have the same topology, their oscillation frequencies may slightly differ because of manufacturing variation. If we got a proper period difference t_d, we would generate high-quality random numbers at a fast rate (in an order of Mbit/s). However, improper t_d results in the lack of entropy (too large t_d), or reduction of the bit rate of generation (too small t_d). It was a serious problem for practical use to properly adjust t_d. A solution for this problem has recently been proposed, which uses route-selectable ring oscillators [9]. To improve the generation bit rate, a mutual sampling method [12] was proposed, which captured $\mathrm{Clk_B}$ by $\mathrm{Clk_A}$ in addition to capturing $\mathrm{Clk_A}$ by $\mathrm{Clk_B}$. It was reported, however, that the quality of random numbers was degraded due to the correlation among the output bits [12].

Another method for clock signals is to utilize clocking elements such as PLLs [2] and DCMs [4]. Since factors of multiplication and division can be set as parameters, it is easy to get a proper frequency ratio. When an external clock source is used, no ring oscillators are required. Even when clocks must be generated internally (i.e. an external clock source is unreliable), only one ring oscillator is required. This type of TRNGs have an advantage of minimizing the number of required logic elements in exchange for two additional PLLs or DCMs. A shortcoming of this method is relatively large power comsumption of clocking elements.

Our research is based on a DCM-based TRNG proposed by Johnson et al. [4], whose target is Xilinx Virtex-5. The parameters of Virtex-5 DCM are multiplier M and divisor D. They both must be integer and meet $2 \leq M \leq 33$ and $1 \leq D \leq 32$. Proper range of N is $400 \leq N \leq 1000$ (i.e. $t_A/400 \geq t_d \geq t_A/1000$) [4]. They presented 23 sets of parameters that meets these conditions [4]. For example, from an input clock of $f_{IN} = 100$ MHz, they generated $\mathrm{Clk_A}$ of $(15/31)f_{IN} \sim 48.39$ MHz and $\mathrm{Clk_B}$ of $(14/29)f_{IN} \sim 48.28$ MHz. In this case, the frequency ratio is 435 : 434 (i.e. $N = 434$) and the expected value of the number of '1's is 217. The numbers of '1's are obtained at the rate of $48.28/434 \simeq 0.111$ Msample/s. In the DCM-based TRNG, three LSBs of the number of '1's are extracted and the generated bitstring is post-processed by the von Neumann Corrector [13]. Its bit rate of generation is, theoretically, $0.111 \times 3 \times 1/4 \simeq 0.083$ Mbit/s. Although this rate varies with parameters, the actual rate was 0.210 Mbit/s on average according to an additional evaluation with various parameters [3]. In this paper, in consideration of a requirement of AIS-31 [5] for a random bitstring without post-processing, only one LSB of the number of '1's is extracted and post-processing is not applied unless explicitly stated.

There are two methods to deal with the number of '1's: the number of consecutive '1's [7] and the sum of the number of '1's in N samples [1]. When using two ring oscillators, the latter method is not available because the frequency

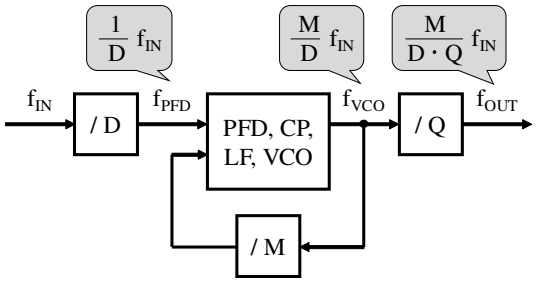

Fig. 2. Simplified block diagram of a mixed-mode clock manager (MMCM).

ratio cannot be exactly determined. With clocking elements, the both methods can be used. The DCM-based TRNG by Johnson et al. adopted the former method [4]. The former method detects the falling edge of output of the D-FF, instead of counting the number of samples, which makes hardware simpler. However, since sampling the vicinity of edge happens consecutively, '0's ('1's) may appear in consecutive '1's ('0's). When counting the number of consecutive '1's in the former method, this causes quite small counter values, which have smaller entropy than counter values near the expected value [3]. This research adopts the latter method to avoid such a negative effect.

C. Mixed-mode Clock Manager

Figure 2 abstracts the operation of the mixed-mode clock manager (MMCM) [14]. Functions unrelated to this research are omitted from this figure. The input clock of frequency of f_{IN} is first divided by D, and then passed phase frequency detector (PFD), charge pump (CP), loop filter (LF), and voltage-controlled oscillator (VCO). The input frequency of the PFD f_{PFD} is

$$f_{PFD} = \frac{1}{D} f_{IN}. \tag{1}$$

The output of the VCO is fed back to the PFD after divided by M. Since this feedback signal has the same frequency as f_{PFD}, the output frequency of the VCO f_{VCO} becomes

$$f_{VCO} = M \cdot f_{PFD} = \frac{M}{D} f_{IN}. \tag{2}$$

The output of the MMCM is obtained by dividing the VCO output by Q, whose frequency f_{OUT} is

$$f_{OUT} = \frac{M}{D \cdot Q} f_{IN}. \tag{3}$$

The parameters of the MMCM, D, M, and Q, have the following constraints:

$$1 \leq D \leq 106, \tag{4}$$

$$2 \leq M \leq 64, \tag{5}$$

$$1 \leq Q \leq 128. \tag{6}$$

D must be integer, while M and Q can be integer or fraction with 1/8 interval. These parameters enable setting of the output frequency to be finer than the earlier DCM.

TABLE I
PARAMETERS OF MMCM TO SIMPLY PORT THE DCM-BASED TRNG [4].

ID	Target	M_A	D_A	Q_A	M_B	D_B	Q_B
J01	DCM	15	31	-	14	29	-
	MMCM	7.50	1	15.50	7.00	1	14.50
J02	DCM	21	22	-	20	21	-
	MMCM	10.50	1	11.00	10.00	1	10.50
J22	DCM	30	31	-	29	30	-
	MMCM	7.50	1	7.75	7.25	1	7.50
J23	DCM	31	32	-	30	31	-
	MMCM	7.75	1	8.00	7.50	1	7.75

These frequencies have constraints due to the characteristics of the PFD and the VCO. They are slightly different with FPGA family and speed grade. The target FPGA of our evaluation, Artix-7 of speed grade -1, has the following constraints [15]:

$$10 \leq f_{PFD} \leq 450 \text{ [MHz]}, \tag{7}$$

$$600 \leq f_{VCO} \leq 1200 \text{ [MHz]}, \tag{8}$$

$$4.68 \leq f_{OUT} \leq 800 \text{ [MHz]}. \tag{9}$$

The input frequency is set to $f_{IN} = 100$ MHz in this research. According to Eqs. (7) and (8) for D and M/D, respectively, the effective constraints of the parameters are as follows:

$$1 \leq D \leq 10, \tag{10}$$

$$6 \leq \frac{M}{D} \leq 12. \tag{11}$$

In addition, the MMCM offers the power down mode [14] to save power consumption when it is not in use. Although it might be a solution of the shortcomings of PLL/DCM-based TRNGs referred to in Section II.B, we do not consider it in this paper.

III. PORTING OF DCM-BASED TRNG

All of our evaluations in this paper use a Digilent Arty FPGA board, which includes an Artix XC7A35T FPGA. Circuits are synthesized by Vivado 2019.2 with the default options unless explicitly stated. The number of '1's or its LSB is measured by a counter and sent to a PC via 3-Mbps UART. The programming file is generated for each set of parameters of MMCMs; dynamic reconfiguration of the parameters is left for future work.

In this section, we simply port the DCM-based TRNG by Johnson et al. [4] to FPGAs with MMCMs. Because of the constraints on M and D, or Eqs. (10) and (11), the existing DCM parameters cannot be directly adopted. To make the parameters comply with MMCMs, Q is used as a dividing factor instead of D, and then both M and Q are divided by two (if $M \leq 24$) or four (if $M > 24$). Table I shows a part of the parameter sets obtained from this strategy. We assigned numbers of J01, J02, ..., and J23 to the parameter sets of the DCM-based TRNG [4]. The parameters M_A, D_A, Q_A are for Clk_A and M_B, D_B, Q_B are for Clk_B. We measured 10^7 counter values (the number of '1's for each N samples) for each parameter set and plotted their distribution.

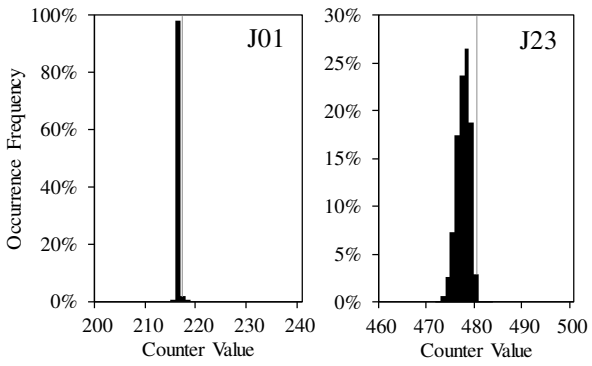

Fig. 3. Distribution of counter values using the MMCM parameters shown in Table I.

TABLE II
FREQUENCY AND PEAK-TO-PEAK JITTER OF THE OUTPUT OF MMCM.

M	D	Q	Freq. [MHz]	Jitter [ps]
7.75	1	8.00	96.875	141.837
15.50	2	8.00		184.566
23.25	3	8.00		229.787
31.00	4	8.00		273.577
38.75	5	8.00		305.392
46.50	6	8.00		343.210
54.25	7	8.00		383.515
62.00	8	8.00		427.425

Figure 3 depicts the distributions of counter values with J01 and J23 parameter sets. The X-axis represents the counter value and the Y-axis represents the occurrence frequency for each value. Gray vertical lines mean the expect value of the counter, $N/2$. The variance of the counter values was much smaller than the results of the previous experiments with Virtex-5 DCMs [3]. In particular with J01 parameter set, 98% of the values were the same: 216. As we will evaluate in more detail in Section V, TRNGs constructed from these parameters do not give enough entropy and, as a result, they fail statistical tests in most cases. It might be because logic and clocking elements of the newer FPGAs become more fault tolerant: jitter of MMCMs, setup time, and hold time of D-FFs might get smaller. Also, their bit rate of generation are about 0.1 Mbit/s, which is an order of magnitude slower than other type of TRNGs suitable for FPGAs [10]. The conclusion of this section is that a simple porting of the DCM-based TRNG is not enough in either randomness or throughput.

IV. ENHANCED PARAMETER SELECTION

A. Method for Larger Entropy

As we have overviewed in Section II.A, there are two ways to increase the variance of the number of '1's: decreasing the difference of the periods t_d or increase the jitter σ_J. Decreasing the period difference should be avoided because it has a tradeoff with the generation bit rate. Instead, we adjust the parameters in order to increase the jitter while keeping the frequency unchanged.

TABLE III
PARAMETERS OF MMCM TO OBTAIN LARGER JITTER.

ID	Method	M_A	D_A	Q_A	M_B	D_B	Q_B
J01	NM	7.50	1	15.50	7.00	1	14.50
	JT	**60.00**	**8**	15.50	**63.00**	**9**	14.50
J02	NM	10.50	1	11.00	10.00	1	10.50
	JT	**63.00**	**6**	11.00	**60.00**	**6**	10.50
J22	NM	7.50	1	7.75	7.25	1	7.50
	JT	**60.00**	**8**	7.75	**58.00**	**8**	7.50
J23	NM	7.75	1	8.00	7.50	1	7.75
	JT	**62.00**	**8**	8.00	**60.00**	**8**	7.75

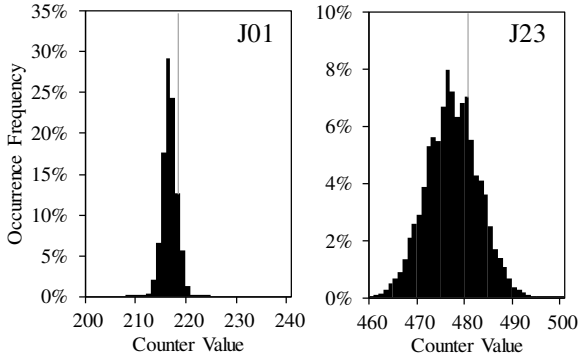

Fig. 4. Distribution of counter values using the MMCM parameters shown in Table III.

Concretely speaking, we multiply both M and D by the same integer. When multiplying and dividing factors are large, the effect of internal noise becomes large and the jitter becomes increased. For example, Tab. II summarizes the peak-to-peak jitter of $\mathrm{Clk_A}$ of J23 parameter set, when M and D are multiplied by 2, 3, ..., and 8. These values can be found from the clocking wizard of Vivado [14]. The worst case jitter becomes three times larger. The maximum multiplier, say D_{max}, is constrained by Eq. (5) as follows:

$$D_{max} = \left\lfloor \frac{64}{M} \right\rfloor, \tag{12}$$

where $\lfloor x \rfloor$ is the maximum integer that is not more than x. Since M/D remains unchanged, no additional constraints come from Eq. (11).

Table III illustrates some parameter sets modified by the above strategy. The method to obtain parameter sets shown in Section III is represented as NM (Normal). The method proposed here is represented as JT (Jittery). Both M and D are multiplied by D_{max} to maximize the jitter while Q is kept unchanged. This modification is applied to both $\mathrm{Clk_A}$ and $\mathrm{Clk_B}$.

Figure 4 the distribution of counter values where the JT method is applied. Note that the scale of the Y-axis is not the same as Fig. 3. The standard deviation of the counter values became 10 times and 4 times larger then the NM method with J01 and J23 parameter sets, respectively. Considering the effect of quantization (that counter values must be integer), this result basically matches the increase of jitter.

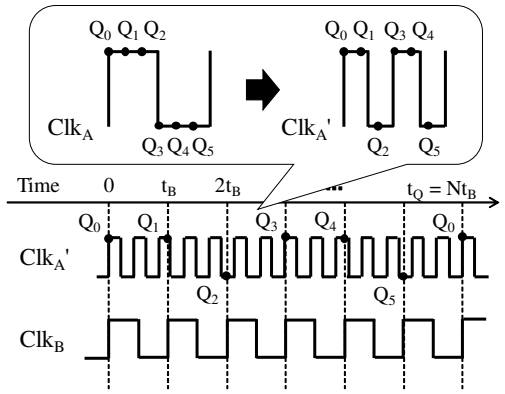

Fig. 5. Example of coherent sampling where the frequency ratio is 14 : 6.

TABLE IV
PARAMETERS OF MMCM TO OBTAIN HIGHER THROUGHPUT.

ID	Method	M_A	D_A	Q_A	M_B	D_B	Q_B
J01	JT	60.00	8	15.50	63.00	9	14.50
	CB	60.00	8	**7.75**	63.00	9	14.50
J02	JT	63.00	6	11.00	60.00	6	10.50
	CB	63.00	6	**5.50**	60.00	6	10.50
J21	JT	58.00	8	7.50	63.00	9	7.25
	CB	58.00	8	**3.75**	63.00	9	7.25
J23	JT	62.00	8	8.00	60.00	8	7.75
	CB	62.00	8	**4.00**	60.00	8	7.75

* J22 is not available because its N is odd.

B. Method for Higher Throughput

To increase the generation bit rate, we consider doubling the frequency of Clk_{A} and halving the number of samples per count (N). Figure 5 illustrates this idea by a similar example to Fig. 1, where the frequency ratio is changed from 7 : 6 to 14 : 6. From the principle of coherent sampling, in this case the waveform of two cycles of Clk_{A} is captured by N samples. If each of the counter values of the first half (Q_0, Q_1, and Q_2 in the example) and the last half (Q_3, Q_4, and Q_5) is counted independently, the generation bit rate can be doubled while avoiding the effect on the entropy (the variance of the counter values). From a lesson from mutual sampling [12], the count should be independent from each other; otherwise, unwanted correlation may occur. In this method, this means that N must be an even number to sample the logically same waveform between the first and the last halves.

This idea can be extended to multiplier of arbitrary natural number. The multiplier, denoted as K, has the following constraints: Q/K must be a permitted value as a parameter of Q and N must be divisible by K.

Table IV shows some parameter sets to double the frequency of Clk_{A} while keeping the JT method applied. We denote this combination of the proposed methods as CB (Combined). We applied the CB method to 13 (out of 23) parameter sets where N is even. Only Q_A is halved and the other parameters are left unchanged.

According to an evaluation, the distribution of the counter values was almost unchanged. The generation bit rate was exactly doubled because the circuit to count the number of samples is deterministic. We will conduct a detailed evaluation in Section V.

TABLE V
PARAMETERS OF MMCM THAT CAN MAXIMIZE JITTER AND THROUGHPUT.

ID	Method	M_A	D_A	Q_A	M_B	D_B	Q_B
N001	NM	6.000	1	7.000	7.375	1	8.625
	JT	**60.000**	**10**	7.000	**59.000**	**8**	8.625
	CB	60.000	10	**1.000**	59.000	8	8.625
N002	NM	6.375	1	7.000	6.250	1	6.875
	JT	**63.750**	**10**	7.000	**62.500**	**10**	6.875
	CB	63.750	10	**1.000**	62.500	10	6.875
N003	NM	6.375	1	7.000	7.500	1	8.250
	JT	**63.750**	**10**	7.000	**60.000**	**8**	8.250
	CB	63.750	10	**1.000**	60.000	8	8.250
N130	NM	8.000	1	15.000	6.125	1	11.500
	JT	**64.000**	**8**	15.000	**61.250**	**10**	11.500
	CB	64.000	8	**1.000**	61.250	10	11.500
N131	NM	8.000	1	15.000	7.125	1	13.375
	JT	**64.000**	**8**	15.000	**57.000**	**8**	13.375
	CB	64.000	8	**1.000**	57.000	8	13.375
N132	NM	8.000	1	15.000	7.250	1	13.625
	JT	**64.000**	**8**	15.000	**58.000**	**8**	13.625
	CB	64.000	8	**1.000**	58.000	8	13.625

C. Selection Strategy

Based on the examinations shown in Section IV.A and Section IV.B, we consider selection of parameter sets in order to maximize the effectiveness of the proposed CB method. First, to maximize the generation bit rate,

$$Q_A \in \mathbb{N} \tag{13}$$

in the NM and JT methods and the multiplier $K = Q_A$. This means $Q_A = 1$ in the CB method. From the constraints of the frequencies of the VCO and the output, (8), (9), the range of M_A becomes

$$6 \leq M_A \leq 8. \tag{14}$$

As a result from Eqs. (12) and (14), D_A in the JT and CB methods becomes either 8, 9, or 10, which maximizes the jitter. For $\mathrm{Clk_B}$, although arbitrary parameters can be set as long as Eq. (11) is met, we constrain the range of M_B as

$$6 \leq M_B \leq 8, \tag{15}$$

to make D_B be also either 8, 9, or 10. Finally, parameters that meets constraints on the output frequency are selected: the integer ratio of the frequencies is $(N+1) : N$, $400 \leq N \leq 1000$ [4], and N is divisible by K. The number of possible sets of parameters is at most an order of hundreds of thousands. We conducted full search to find promising parameter sets.

We found 132 sets of parameters in a range of output frequency between 50 and 100 MHz. Table V summarizes a part of them. We assigned numbers of N001, N002, ..., N132 to these parameter sets. There are two parameter sets which have also enumerated in the DCM-based TRNG [4]: N009 and N029 are identical to J19 and J23 (except for the CB method), respectively.

TABLE VI
MIN-ENTROPY OF GENERATED RANDOM BITSTRINGS.

ID	NM	JT	CB
J01	**0.0294**	0.9964	0.7988
J02	0.9254	0.6097	**0.6157**
J03	0.2614	0.6742	-
J04	0.4915	**0.5647**	-
J05	0.9117	0.9191	0.7953
J06	0.4927	0.7687	-
J07	0.9800	0.6926	-
J08	0.4113	0.7566	-
J09	0.6232	0.6316	0.6782
J10	0.7167	0.8034	-
J11	0.9621	0.6798	0.6442
J12	0.3369	0.8610	0.8002
J13	0.7646	0.8534	-
J14	0.9749	0.6827	0.6623
J15	0.9762	0.5669	0.7551
J16	0.4921	0.6435	0.8879
J17	0.8709	0.9839	0.9935
J18	0.9895	0.7425	-
J19	0.7684	0.9936	1.0000
J20	0.8910	0.9917	-
J21	0.8384	0.8969	0.8906
J22	0.6670	0.7579	-
J23	0.9863	0.9963	0.9959
Avg.	0.7114 (0.7458)	0.7855 (0.8047)	0.8090

TABLE VII
GROUPING OF THE PARAMETER SETS.

Name	Condition	# of Sets
A	$400 \leq N < 500$	35
B	$500 \leq N < 600$	32
C	$600 \leq N < 700$	22
D	$700 \leq N < 800$	17
E	$800 \leq N < 900$	14
F	$900 \leq N \leq 1000$	12
All		132

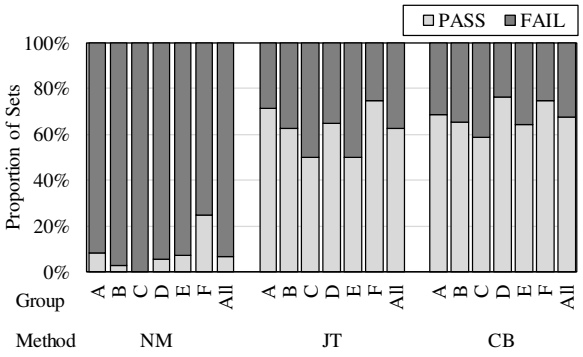

Fig. 6. Proportion of parameter sets that passed AIS-31 Procedure B.

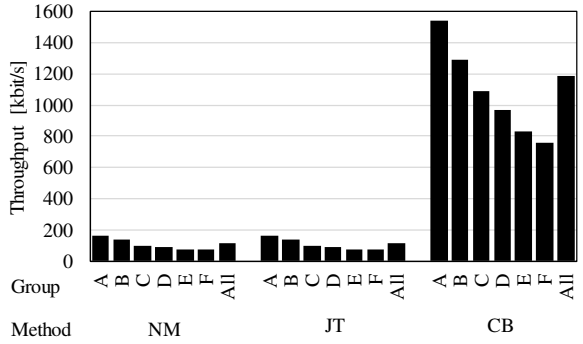

Fig. 7. Generation rate of random bits.

V. EVALUATION

A. Min-entropy

In this section, we evaluate the effect of the proposed parameter selection on the randomness, the bit rate of generation, and the amount of hardware of an MMCM-based TRNG. We first evaluate the entropy based on the parameter sets from the DCM-based TRNG (i.e. J01–J23). We measured the occurrence frequency of the LSB of 10^7 counter values. We calculated the min-entropy of the sequence of LSBs as $H_\infty = -\log_2\{\max(p_0, p_1))\}$, where the occurrence frequencies of the LSB of '0' and '1' are denoted as p_0 and p_1, respectively.

Table VI enumerates the calculated min-entropy. The row Avg. corresponds to the arithmetic mean of the evaluated sets. The arithmetic mean of the 13 sets to which the CB method is applicable is noted in parentheses. The minimum value for each method is shown in boldface type. The results indicates that the min-entropy is basically increased by applying the JT method. In particular, any case that only one counter value frequently appeared, as shown in Fig. 3, were not observed in the JT and CB methods. No significant differences were observed between JT and CB.

B. AIS-31 Statistical Tests

We then evaluate the quality of random bitstrings, generated by concatenating the LSBs of the counters, using AIS-31 [5] Procedure B statistical test suite. We used all of the 132 parameter sets found in Section IV.C. This test assumes that the input bitstring is no less than 7 Mbit long and not post-processed. If the decision is not reliable (i.e. failing in only one of the tests), the tests are conducted again with another bitstring. We thus generated 16 Mbits of bitstring for each parameter set. We also calculated the generation bit rate by measuring the time to obtain the bitstring.

The number of samples per count, N, affects the tradeoff between the generation bit rate and the quality of random numbers. We grouped the parameter sets according to N as shown in Tab. VII.

Figure 6 summarizes the result of the statistical tests, while Fig. 7 plots the average of generation bit rate. The X-axis represents a kind of method and a group of parameter sets, while Y-axis is the proportion of sets passed or failed (Fig. 6) or the generation bit rate (Fig. 7). Only 9 sets (out of 132) passed when the existing method is simply ported (NM), while 83 sets and 89 sets passed in JT and CB, respectively. An expected usage in actual applications is to find an appropriate parameter set through testing some possible sets using

TABLE VIII
RESULT OF THE NIST SP 800-22 TEST SUITE. A SIMPLE POST-PROCESSING WITH A 4-BIT LFSR WAS APPLIED.

name	p-value	proportion
Frequency	0.37287	99.35%
BlockFrequency	0.72770	98.97%
CumulativeSumsUp	0.85956	99.35%
CumulativeSumsDown	0.38881	99.44%
Runs	0.05050	98.97%
LongestRun	0.21278	99.25%
Rank	0.24976	98.97%
FFT	0.82609	99.16%
NonOverlappingTemplate	0.16313	99.00%
OverlappingTemplate	0.06071	98.60%
Universal	0.02751	98.79%
ApproximateEntropy	0.60990	98.97%
RandomExcursions	0.97985	98.98%
RandomExcursionsVariant	0.01551	99.36%
Serial1	0.93685	98.70%
Serial2	0.57125	98.60%
LinearComplexity	0.01405	99.07%

dynamic reconfiguration. Increase of the number of passing parameter sets means the reduction of the time spent for such an advance preparation.

Regarding the generation bit rate, CB achieved 1.18 Mbit/s on average while both NM and JT was 0.118 Mbit/s. This means that the proposed TRNG was ten times faster than the simple porting of the existing method by Johnson et al. [4]. Also, it was even five times faster than their implementation on a Virtex-5 FPGA [3]. In particular, 1.54 Mbit/s of generation bit rate was achieved with Group A, the group with the smallest N. This result is comparable with other kinds of TRNGs [10]. Any significant difference in the result of statistical tests was not observed among the groups. In actual applications, it may be reasonable to try parameter with small N at first. Parameter sets with smaller N might still have sufficient randomness, though we leave it for future work.

C. NIST SP 800-22 Test Suite

We conduct a more detailed statistical test, NIST SP 800-22 test suite [11], to random numbers generated by the proposed method with a simple post-processing. We chose N029 parameter set (basically equivalent to J23) with the CB method. As a post-processing method, the random bitstring was XOR-ed with an output sequence of 4-bit linear feedback shift register (LFSR) by software. This corresponds to a quite simple debiasing. As recommended in AIS-31 [5], we obtained 1,073 1-Mbit bitstrings and conducted the tests to each of them. The generation bit rate of the bitstrings was 0.808 Mbit/s.

Table VIII summarizes the result of the NIST test suite. For each test, a p-value and the proportion of passed bitstrings are shown. The test is considered as pass if the p-value is no less than 10^{-4} and the proportion is within 3σ range from 99%. The post-processed bitstrings passed the NIST test suite, for all of the tests meets these conditions. Although the hardware for post-processing is not included in the following evaluation in Section V.D, it will be almost negligible.

TABLE IX
NUMBER OF LOGIC ELEMENTS FOR TRNG.

Element	NM	JT	CB
LUT	19	19	17
D Flip-flop	18	18	18

D. Amount of Hardware

Finally, we evaluate the amount of hardware after synthesized, placed and routed. In the evaluated circuit, only the LSB of the number of '1's is counted (by a T flip-flop), instead of the whole number (by a counter). A circuit for UART communication is not included. To avoid packing with other circuits, we add a `-flatten_hierarchy none` synthesis option. We used the N029 parameter set in the same way as Section V.C.

Table IX shows the summary of the implementation results. The number of required LUTs (look-up tables) or flip-flops was comparable to the DCM-based TRNG [4]. The number of LUTs was reduced only in the CB method because the number of samples per count was reduced from N to N/K. The bit width of the counter can be further reduced if the use of only the CB method is supposed.

VI. CONCLUSION

In this paper, we proposed a novel method of parameter selection of MMCMs to apply a coherent sampling-based TRNG with clocking elements to recent Xilinx FPGAs. The number of parameter sets that pass the AIS-31 Procedure B statistical tests became about 10 times larger than simple porting of the existing method. The bit rate of generation also became 10 times larger than the simple porting and 5 times larger than an implementation on an earlier FPGA.

As the configurability of the parameters is important for real applications, we are going to integrate our MMCM-based TRNG into a system, in order that an appropriate parameter could be set automatically.

REFERENCES

[1] F. Bernard, V. Fischer, and B. Valtchanov, "Mathematical Model of Physical RNGs Based on Coherent Sampling," *Tarta Mountains Mathematical Publications*, vol. 45, no. 1, pp. 1–14, 2010.

[2] V. Fischer and M. Drutarovsky, "True Random Number Generator Embedded in Reconfigurable Hardware," in *Proc. 3rd Workshop on Cryptographic Hardware and Embedded Systems*, 2002, pp. 415–430.

[3] N. Fujieda, M. Takeda, and S. Ichikawa, "An Analysis of DCM-based True Random Number Generator," *IEEE Transaction on Circuits and Systems II: Express Briefs*, vol. 67, no. 6, pp. 1109–1113, 2020.

[4] A. P. Johnson, R. S. Chakraborty, and D. Mukhopadyay, "An Improved DCM-Based Tunable True Random Number Generator for Xilinx FPGA," *IEEE Transaction on Circuits and Systems II: Express Briefs*, vol. 64, no. 4, pp. 452–456, 2017.

[5] W. Killmann and W. Schindler, *A proposal for: Functionality classes for random number generators, version 2.0*, Federal Office for Information Security, 2011.

[6] P. Kohlbrenner and K. Gaj, "An embedded true random number generator for FPGAs," in *Proc. 12th International Symposium on Field Programmable Gate Arrays*, 2004, pp. 71–78.

[7] Y. Lao, Q. Tang, C. H. Kim, and K. K. Parhi, "Beat Frequency Detector–Based High-Speed True Random Number Generators: Statistical Modeling and Analysis," *ACM Journal on Emerging Technologies in Computing Systems*, vol. 13, no. 1, pp. 1–25, 2016.

[8] N. C. Laurenciu and S. D. Cotofana, "Low cost and energy, thermal noise driven, probability modulated random number generator," in *Proc. 2015 IEEE International Symposium on Circuits and Systems*, 2015, pp. 2724–2727.

[9] A. Peetermans, V. Rožić, and I. Verbauwhede, "A Highly-Portable True Random Number Generator based on Coherent Sampling," in *Proc. 29th International Conference on Field Programmable Logic and Applications*, 2019, pp. 218–224.

[10] O. Petura, U. Mureddu, N. Bochard, V. Fischer, and L. Bossuet, "A survey of AIS-20/31 compliant TRNG cores suitable for FPGA devices," in *Proc. 26th International Conference on Field Programmable Logic and Applications*, 2016, pp. 1–10.

[11] A. Rukhin, J. Soto, J. Nechvatal, M. Smid, E. Barker, S. Leigh, M. Levenson, M. Vangel, D. Banks, A. Heckert, J. Dray, and S. Vo, *A statistical test suite for random and pseudorandom number generators for cryptographic applications*, NIST Special Publication 800–22, Rev. 1a, 2010.

[12] B. Valtchanov, V. Fischer, and A. Aubert, "Enhanced TRNG based on the coherent sampling," in *Proc. 3rd International Conference on Signals, Circuits and Systems*, 2009, pp. 1–6.

[13] J. von Neumann, "Various techniques used in connection with random digits," *Monte Carlo Method, National Bureau of Standards Applied Mathematics Series 12*, pp. 36–38, 1951.

[14] Xilinx Inc., *7 Series FPGAs Clocking Resources*, User Guide UG472 v1.14, 2018.

[15] ——, *Artix-7 FPGAs Data Sheet: DC and AC Switching Characteristics*, Data Sheet DS181 v1.25, 2018.

[16] H. Zhun and C. Hongyi, "A truly random number generator based on thermal noise," in *Proc. 4th International Conference of ASIC*, 2001, pp. 862–864.