Quantum random flip-flop and its applications in random frequency synthesis and true random number generation

Mario Stipčević[1,*]

[1]Photonics and Quantum Optics Research Unit, Center of Excellence for Advanced Materials and Sensing Devices, Ruđer Bošković Institute, Bijenička 54, 10000 Zagreb, Croatia
*E-mail: mario.stipcevic@irb.hr

In this work a new type of elementary logic circuit, named random flip-flop (RFF), is proposed, experimentally realized and studied. Unlike conventional Boolean logic circuits whose action is deterministic and highly reproducible, the action of a RFF is intentionally made maximally unpredictable and, in the proposed realization, derived from a fundamentally random process of emission and detection of light quanta. We demonstrate novel applications of RFF in randomness preserving frequency division, random frequency synthesis and random number generation. Possible usages of these applications in the information and communication technology (ICT), cryptographic hardware and testing equipment are discussed.

I. INTRODUCTION

Today life is unimaginable without microprocessors that are in the heart of computers, cell phones and other devices which depend on the information processing. Boolean logic circuits are the basis technology for all contemporary microprocessors, computers, mobile phones, communication equipment and many other ICT devices. Logic circuits are divided into non-sequential (e.g. AND, XOR, NOT) whose output state only depends on the current state(s) of input(s), and the sequential (e.g. flip-flop) whose output state depends on both current and past state(s) of the input(s). Their purpose in ICT applications and equipment is information processing and, depending on how data are interpreted by humans, they appear to be performing a variety of tasks like: mathematical calculations, playing music and video, word processing, etc. Logic circuits can also be used for the purpose of analog and digital signal processing: frequency division, multiplication or synthesis, clock restoration, signal delaying, measuring of relative phase shift, detecting coincident events, etc. Boolean logic circuits are explicitly made to behave deterministically that is to give the same output if fed by the same sequence of input signals. This is their great advantage but also a limitation.

Here we propose a new elementary logic building block that behaves *non-deterministically*: the random flip-flop (RFF), and discuss a few of its applications. We show that RFF is a convenient way of "packaging" of randomness in a circuit that can be mixed with conventional logic circuits in order to arrive to a rich set of properties and applications that cannot be achieved by Boolean logic alone.

II. DEFINITION OF THE RANDOM FLIP-FLOP

In this study we only consider the edge-triggered master-slave flip-flops [1]. Henceforth, without loss of generality, we assume that a flip-flop performs its action when clock pulse input (CP) changes state from LOW (logic 0) to HIGH (logic 1) that is upon the "rising edge" of the input pulse. In this study we will predominantly use D-type flip flop (DFF). Conventional edge-triggered DFF, whose symbol is shown in Fig. 1a), is an important building element of logic circuits usually used as a one bit data storage, frequency divider-by-two or a digital counter. With each occurrence of the rising edge at the clock pulse input CP, logic level present at the data input D is copied to the output Q and stays memorized there until the new clock pulse. This operation is described exactly by a truth table Table 1 (3-rd column) and illustrated in Fig. 1b). DFF may optionally feature the inverted output $\overline{Q}$. It can also have either or both Set (S) and Reset (R) inputs which, upon being brought to HIGH unconditionally (asynchronously and irrespective of the CP input) *set* (to HIGH) or *reset* (to LOW) the output Q. DFFs and other logic circuits, are specifically designed to behave deterministically, with a very small probability of error in order to ensure stable and predictable operation of ICT equipment.

		DFF	DRFF
D	Q_n	Q_{n+1}	Q_{n+1}
0	0	0	0
0	1	0	RND
1	0	1	RND
1	1	1	1

Table 1. The truth tables for deterministic D-type flip-flop (DFF) and D-type random flip-flop (DRFF). RND means that upon the clock the corresponding output is set to randomly chosen value 0 or 1 regardless of the previous states.

There are a few technical details of marginal importance for this study which, for simplicity, we do not take into consideration: (1) *setting time* (t_{SET}): D input has to hold a vale for some time *before* the clock; (2) *propagation time* (t_{CPQ}): it is a finite time delay between the clock input CP and setting of the correct value at the output Q; (3) *(re)set time* (t_{RSQ}): it is the time delay between (Reset)Set input and the output Q. In the experimental technique that we will use, all

M. Stipčević (corresponding author) is with Centre of Excellence for Advanced Materials and Sensing Devices, Ruđer Bošković Institute, Bijenička 54, 10000 Zagreb, Croatia (e-mail: mario.stipcevic@irb.hr).

these time delays are on the order of 1 ns and of little or no importance for the effects under study as long as they are constant or small with respect to the average clock rate.

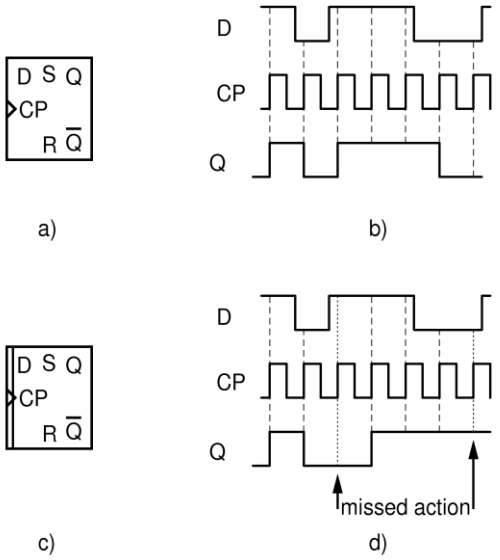

Figure 1. Examples of input/output waveforms for: (a-b) the conventional D-type flip-flop (DFF); (c-d) D-type random flip-flop (DRFF).

By definition, the D-type random flip-flop (DRFF) (symbol shown in Fig. 1c)), operates similar to DFF with only difference that, upon the clock pulse, state of the data input D is transferred randomly to the output Q with probability of ½, which ensures maximal unpredictability. This is described by a truth Table 1 (4-th column) and illustrated in Fig. 1d).

Similarly, we define other types of RFF such as T-type random flip-flop (TRFF). While Boolean T-type flip-flop (TFF) toggles output state with each clock if the input T is held HIGH, TRFF toggles with probability of ½. In both TFF and TRFF clock input is disabled while T=0, as exactly described in Table 2. Any of those RFFs (deterministic or random) may optionally feature the inverted output $\overline{Q}$ and Set (S) and Reset (R) inputs which unconditionally *set* (to HIGH) or *reset* (to LOW) the output Q.

		TFF	TRFF
T	Q_n	Q_{n+1}	Q_{n+1}
0	0	0	0
0	1	1	1
1	0	1	RND
1	1	0	RND

Table 2. The truth tables for deterministic T-type flip-flop (TFF) and T-type random flip-flop (TRFF). RND means that upon the clock the corresponding output is set to randomly chosen value 0 or 1 regardless of previous states.

Interestingly, the random flip-flops can emulate each other in exactly the same way as their deterministic counterparts, as shown in Fig. 2, but as we shall see, the analogy between random and deterministic FFs generally does not exist.

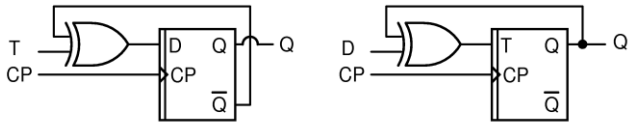

Figure 2. The RFFs can emulate each other in exactly the same way as their deterministic counterparts do.

III. PRACTICAL REALIZATION OF AN OPTICAL QUANTUM RANDOM FLIP-FLOP

In this study all experiments were performed using logic gates and flip-flops available in the Altera MAX7000 family reconfigurable PLD chip. The flip-flops therein conform with the above description and truth tables. The setting time t_{SET} is about 1 ns, both propagation delays t_{CPQ} and t_{RSQ} are about 2 ns and maximum counting frequency exceeds 250 MHz. Output rise and fall times are below 1 ns.

Before addressing practical realization of RFF, let us introduce notion of random pulse train (RPT), an important concept in this presentation. By definition RPT is a sequence of electrical logic pulses of constant amplitude, typically but not necessarily of constant duration, wherein time intervals t_i between subsequent pulses (also known as "waiting times") follow the exponential probability density function. It is best to picture a RPT as being a result of an underlying stationary and memoryless physical process generating random events and that each event generates one logic electrical pulse whose leading (i.e. positive-going) edge corresponds to the time of the event. A RPT, illustrated in Fig. 3, is characterized by a single parameter: the average pulse rate henceforth referred to as "random frequency" or just "frequency" and denoted by f.

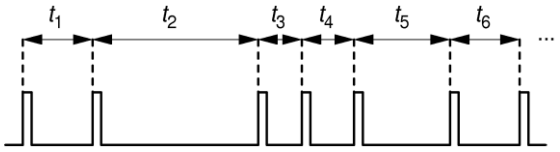

Figure 3. A random pulse train (RPT) is a sequence logic pulses usually of constant duration. By definition, pulse waiting times t_i (i=1,2,3,...) between (rising edges of subsequent pulses follow exponential probability density function.

Quite generally, we can define the average frequency f of a digital signal as:

$$f = \lim_{n\to\infty} \frac{1}{\frac{1}{n}\sum_{i=1}^{n} t_i} \qquad (1)$$

that is as an average number of rising edges appearing in a very long time interval divided by the length of the interval. This definition holds quite generally: for periodic and non-periodic signals, including RPTs. Of course, for periodic signals it is enough to take $n = 1$ because all intervals t_i are equal and the averaging over n periods in unnecessary.

A precise RPT (in terms of obeying exponential p.d.f.) is important for realization of the RFF and in some other applications. A particularly precise RPT can be realized by use of quantum randomness. Of all possible sources of randomness, those relying on quantum processes of photon

emission and detection are probably the best understood from the first principles and offers theoretically perfect randomness.

In this study we use a stationary optical random pulse train generator (ORPTG), similar to those described in [2-3], shown in Fig. 4. It consists of InGaAs red light emitting diode (LED) which shines light attenuated by a dispersive neutral density filter (NDF) upon a single-photon detector (SPD). Upon each successful detection of a photon SPD generates a logic pulse and in that way a train of pulses which appear randomly in time is generated.

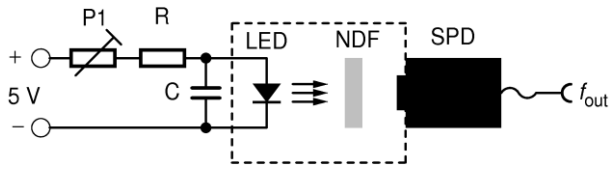

Figure 4. An optical random pulse train generator (ORPTG) consists of InGaAs red light emitting diode (LED) operated in constant-power mode that shines light upon a single-photon detector (SPD) through a dispersive attenuating filter (DF). The light power, and thus the mean pulse rate, can be adjusted via potentiometer P1.

The LED (Hamamatsu L7868, λ=670 nm) is operated in a constant-current mode. Such a source is well-known to yield photons emitted at random times due to quantum effects [2], [4]. Filter NDF is conveniently chosen such that the resulting rate of photon detections falls in the desired range. Light power can be fine-tuned by potentiometer P1. The detector SPD is home-made photon counting module based on a Geiger-mode avalanche photodiode (Laser Components, SAP500) and the active quenching circuit described in Ref. [5]. Upon each detection SPD generates a TTL pulse of duration τ_p = 12 ns. No two output pulses can come closer than the dead time of the detector, τ_{dead} = 24 ns. Maximum photon detection rate of random light is over 38 Mcps. Given the spectral width of the emitted light, $\Delta\lambda \approx 20$ nm, one can estimate the coherence time $\tau_{cohr} = \lambda^2/(c\Delta\lambda) \approx 75$ fs. As long as τ_{cohr} is much smaller than the smallest period between successive photon detections, waiting times between successive photon detections obey exponential probability density function. Main correlations among events come from imperfections of the SPD notably the afterpulsing and the dead time. Afterpulsing of SAP500 is relatively high (3-5%) at "usual" operating conditions that are optimized for low noise and high detection efficiency (temperature in the range -25 to -10 C, overvoltage 15 V) [6]. In order to improve on afterpulsing we operated SAP500 at an unusually high junction temperature of +35 C and overvoltage of 9 V which resulted in the afterpulsing probability of only 0.28% and an estimated afterpulsing lifetime of about 10.5 ns. The near-exponential empirical distribution of photon waiting times of the ORPTG built with this detector is shown in Fig. 5. The exponential p.d.f. hypothesis probed by the chi-square test, using statistics of 10^6 recorded intervals, is accepted with probability greater than 0.999999 confirming that the measured distribution conforms with what one would expect from a random source.

There are three important considerations when conceiving a practical RFF:

1. the action of a RFF should be truly random and independent of all other RFFs used in the same circuit;
2. the action probability of RFF should be as close as possible to ½, for applications in this work within $\pm 10^{-4}$;
3. the propagation delay between clock pulse and the output(s) should either be negligible with respect to the average clock rate or be constant, in order to allow for synchronization of operation of several RFFs. This is important in some applications like the random frequency synthesis (Sect. VI) and the Bell test [7].

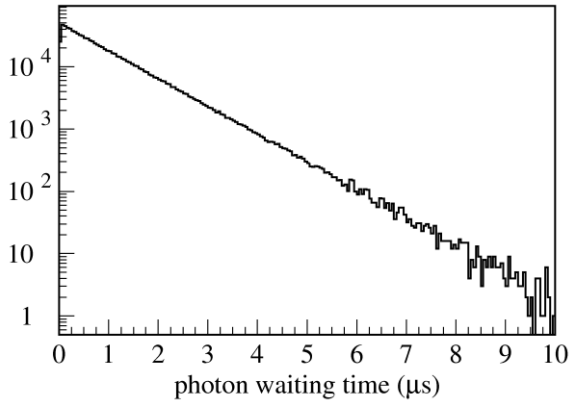

Figure 5. Histogram of time intervals between subsequent pulses (waiting times) generated by the optical random pulse train generator (ORPTG) at a frequency of 1 Mcps.

It is not known to us whether all three conditions can be met exactly, even in theory. However, the three conditions can be met asymptotically by the use of the Random Pulse Train (RPT).

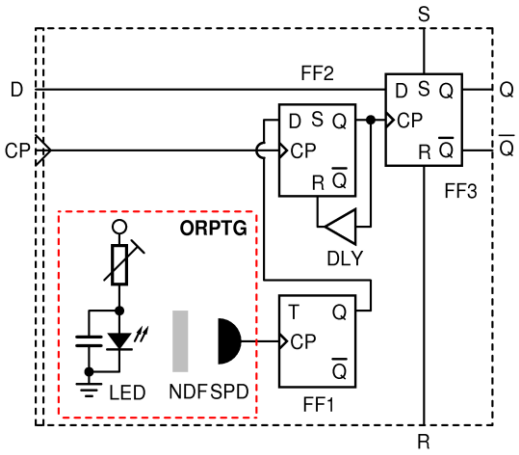

Figure 6. Realization of the D-type random flip-flop by using an optical quantum random pulse train generator (ORPTG) made of: 1) an LED light source (LED); 2) neutral density filter (NDF); and 3) a single-photon detector (SPD).

D-type RFF can now be constructed as shown in Fig. 6. Under the control of an optical random pulse train generator (ORPTG) the output Q of the T-type flip-flop FF1 toggles upon each received pulse. The ORPGT is made such that its average frequency (in the sense of Eq. (1)) stays unchanged over time. Because of that, FF1 spends exactly equal time in

HIGH and LOW state while changes happen at random (unpredictable) times. If a logic pulse appears at the global clock pulse input CP, FF2 captures momentary logic state of the FF1 by its input D. If D is LOW nothing happens. However, if D is HIGH then output Q of the FF2 goes HIGH and then back to LOW after a short period of time due to the feedback between its output Q and reset input R, generating a short pulse of duration equal to the delay through buffer DLY. We assume that duration of this pulse is shorter than the time to the next global clock pulse. Since probability of finding FF1 in HIGH state is exactly equal to ½, FF2 generates the pulse with probability of ½. Thus FF3, and in fact the whole DRFF, will act as an ordinary DFF but with clock action probability of exactly ½ which, by definition, is the action of a DRFF. As explained in [3], any short-lived correlations among pulses of the ORPTG, as well as theoretical imperfections of this method will have a vanishingly small effect on randomness of the sampled state of FF1 as long as the sampling period is sufficiently long with respect to lifetimes of imperfections (dead time, afterpulsing, ...) and the average toggling period of the FF1. In order to ensure good randomness at sampling rates of up to 3 MHz, the frequency of the ORPTG (i.e. the photon detection frequency) was set to about 24 Mcps. We emphasize that the action probability if the RFF does not depend on the frequency of the QRPTG which is one of the strengths of this design. Consequently its choice is not critical. An alternative approach would be to use a fast-response random bit generator described in Ref. [7].

The construction laid out in Fig. 6 presents the basic DRFF used in this work. All logic circuits required for the RFF as well as data acquisition are made by programming a single Altera MAX7000 family reconfigurable PLD chip complemented with a Cypress CY7C68013 communication chip for transfer of data to the PC computer via USB2 link.

IV. RANDOMNESS PRESERVING FREQUENCY DIVIDERS

In this section we are concerned with frequency division of RPTs.

Dividing frequency of periodic signals, by an integer or fractional number, can be performed by various well known techniques [8-10]. For example, a single TFF will divide by two frequency of its clock signal while division by *n* can be made by a counter-to-*n*. If a RPT is divided by such a divider, the resulting signal will have a correct frequency but the distribution of waiting times will not be exponential anymore and will be less random. In fact, the direct division of frequency of random pulse trains (RPTs) in such a way that the resulting signal is again a RPT wasn't feasible so far. Now, this can be made by "random frequency dividers" made with random flip-flops. A few simple random frequency dividers are shown in Fig. 7.

A random frequency divider by 2 is shown in Fig. 7a. For each clock pulse the DRFF decides randomly whether to set output Q to HIGH. If it does, the bootstrap between Q and R resets Q to LOW essentially after a propagation time of the non-inverting delay buffer DLY, thus generating a short pulse. Since the random decisions are not correlated to input pulses, the output pulses are memoryless (exponentially distributed) but only a half of them will yield an output pulse. In another words, if a RPT of random frequency f is brought to the input CP, a RPT of random frequency $f/2_R$ will appear at the output Q. The subscript "R" denotes "random division" action. Random division preserves exponential distribution of pulse waiting times. Stacking two DRFFs as shown in Fig. 7b one achieves division by 4_R etc. Note that these particular dividers always produce a sequence of short pulses, that is an RPT.

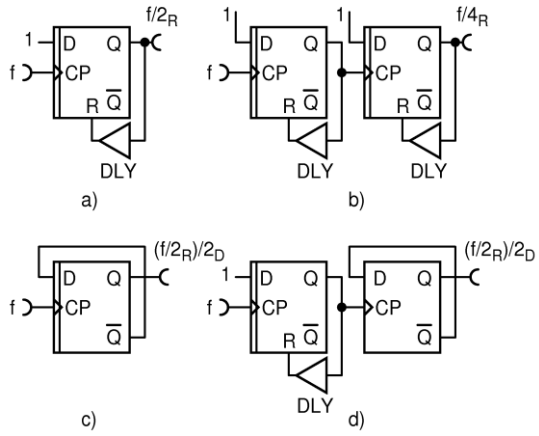

Figure 7. B asic dividers with RFFs: a) random divider by 2; b) random divider by 4; c) divider which performs random division by 2 followed by deterministic division by 2 which has the same functionality as d).

A surprising possibility of dividing frequency by 4 with a single DRFF is shown in Fig. 7c. This division is composed of first dividing by 2 randomly and then by 2 deterministically, which we denote as $1/2_R/2_D$ where subscript "D" stands for "deterministic". Result of such a division is a signal with variable pulse width and bell-shaped waiting time p.d.f. shown in Fig. 8. Note that random and deterministic divisions do not commute: $1/2_R/2_D \neq 1/2_D/2_R$. This circuit is functionally equivalent to the circuit in Fig. 7d.

In comparison to the random division, deterministic division of a RPT by n results in a pulse train with frequency of f/n too, however its pulse waiting times get modified and follow Erlang distribution with the shape parameter n:

$$\mathrm{Erlang}(t, f, n) = \frac{f^n t^{n-1} e^{-ft}}{\Gamma(n)} \qquad (2)$$

which for large n tends to the Normal distribution due to the Central Limit Theorem. Note that as n goes to infinity, the divided signal becomes increasingly periodic and eventually its randomness vanishes. Thus, the deterministic division drives a random input signal *away* from randomness. Interestingly, by virtue of the Poisson limit theorem (see e.g. Ref. [11]), random division brings a pulse train obeying *any* initial time interval distribution asymptotically *towards* exponential distribution, that is towards maximal randomness.

This is illustrated in Fig. 8 by experimentally measured data, where an initial Erlang distributed pulse train is repeatedly divided by 2_R until it almost resembles exponential p.d.f. In that respect, random frequency division and synthesis of RPTs (Section VI) are stable, self-healing processes which tend to eliminate effects of imperfections in hardware.

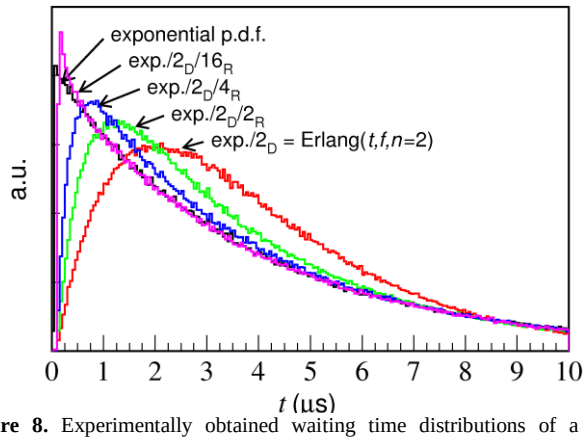

Figure 8. Experimentally obtained waiting time distributions of a RPT (exponential) and of a RPT divided by factors: 2_D (Erlang p.d.f.), 2_D2_R, 2_D4_R and 2_D16_R.

Note that in the special case of dividing a RPT by 2_D (for example by means of a deterministic flip-flop, as shown in Fig. 9) no information is lost: every edge (rising or falling) corresponds to one pulse from the original RPT and therefore it is possible to restore the original RPT form the divided one.

The restoration can be done by a deterministic multiplier by 2_D, such as the one shown in Fig. 9, which upon every edge (rising or falling) generates one short pulse of width equal to the propagation delay of the non-inverting buffer (DLY). On the contrary, when a RPT is randomly divided by 2_R then half of the information (ever second pulse, on average) is lost and there is no way to recover the original RPT.

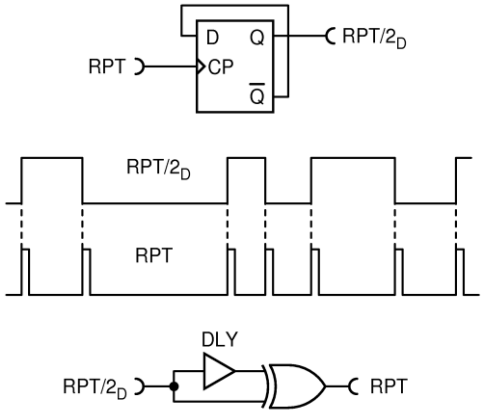

Figure 9. Deterministic divider by 2 (top) and deterministic multiplier by two (bottom). The input and output waveforms are shown in the middle. The output pulse width of the multiplier equals to propagation delay of the non-inverting buffer.

Random frequency dividers are the basis for the random counters and random frequency synthesis that are described in the next two sections.

V. RANDOM COUNTER

Counters are important building blocks, made of flip-flops, that can be used for counting events or for frequency division. To illustrate the complexity of circuits built with RFFs in this section we analyze the "random synchronous counter", defined as a synchronous up-counter in which all flip-flops are replaced by their random counterparts.

A few of its characteristics are investigated: counting, random number generation and frequency division.

Figure 10 shows the well-known synchronous counter topology, however, realized with TRFFs instead of TFFs. In our experiment it has been built using four independent TRFFs, each realized by an independent DRFF, built according to Fig. 6, via the emulation circuit shown in Fig 2. Both the deterministic counter (made with TFFs) and the random counter (made with TRFFs) feature 16 possible (output) states which can be represented by a binary number $\{Q_3, Q_2, Q_1, Q_0\}$ whose value is in the range from 0 to 15. The deterministic version of this counter advances by one upon each (rising edge of the) clock pulse, that is it counts from 0 to 15, then falls back to 0 etc. Thus the counter can only advance in steps of 1 or -15. Starting from 0 the counter advances by 1 in 15 consecutive steps and then advances by -15 in a single step, thus exhibiting an average advance of 0 in the long run.

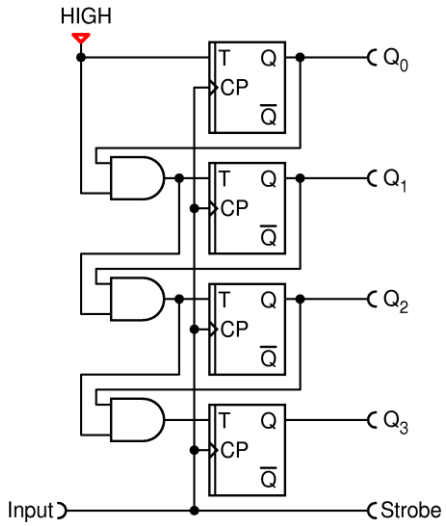

Figure 10. Synchronous 4-bit random counter. With each clock pulse the counter randomly advances up similar to random walker on a line.

Regarding counting, unlike its deterministic counterpart, upon a clock pulse random counter can go: up, down or stay unchanged, that is the output numbers appear "random". On a long run, probabilities of all 16 possible output states (numbers) are equal for both deterministic and random counter.

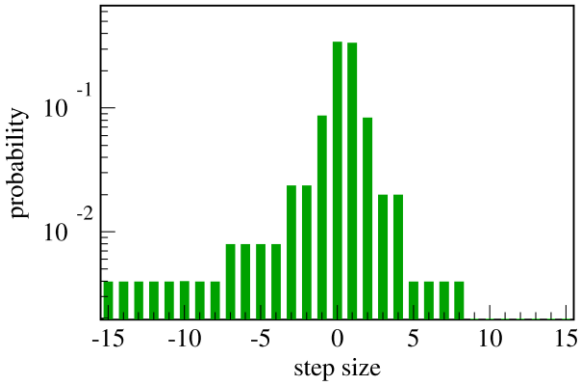

Figure 11. Probabilities of step sizes of the 4-bit synchronous random counter.

The difference is in the variety of possible step sizes by which the counter advances and their probabilities. Possible advancement steps are in the range from -15 to +8 since advances greater than 8 (the most significant bit) are impossible. Namely in order to achieve step of 8 or greater it is necessary that Q3 goes from LOW to HIGH. But to enable this, considering the three AND gates we see that all other outputs must be HIGH. Therefore, the step of 8 is the greatest possible. A histogram of probabilities of advancement step upon a clock pulse of the random counter, obtained by computer simulation, is shown in Fig. 11. In order to determine these probabilities theoretically, one would need to determine probabilities of 24 possible steps for each of the 16 possible states, in total $M = 384$ values. The distribution is skewed and grouping of values in clusters of 1, 2, 4 and 8 steps is apparent.

By far the most probable step sizes are 0 (no change) and 1, each with roughly equal probabilities. This could be expected from a forward counter topology constructed such that the counter mostly advances by 1, where deterministic flip-flops are replaced by random ones which have the probability of action equal to ½, thus the most probable steps are indeed 0 and 1. However, interestingly, the *average* step size is 0, as is for the deterministic counter. The number of probabilities (M) that need to be determined in order to calculate probabilities of steps rises faster than exponentially with the number of flip-flops (N): $M = (2^N)^2 + 2^{2N-1}$, indicating high complexity of this seemingly simple circuit.

Regarding random number generation, state of the random counter changes unpredictably in such a manner that the 16 possible outcomes are distributed uniformly, but numbers so generated are strongly correlated and therefore not completely random. Namely, when considered individually, each of the outputs Q_i (taken in synchronization with the Strobe output) yields a random bit sequence with zero bias but with a non-zero serial autocorrelation coefficients of lag k, defined for example in [12]:

$$a_k = \frac{\sum_{i=1}^{N-k}(x_i - \bar{x})(x_{i+k} - \bar{x})}{\sum_{i=1}^{N-k}(x_i - \bar{x})^2} \qquad (3)$$

Experimentally determined, autocorrelation coefficients obey geometric law with respect to k: $a_k(Q_i) = (1 - 2^i)^k$ indicating Markov process. Indeed, one expects Markovian behavior since sequentiality of RFF dictates that the next state of the counter depends solely (albeit non-deterministically) on the previous state. Of course, autocorrelation among individual bits causes autocorrelation of the sequence of 4-bit binary numbers $\{Q_3, Q_2, Q_1, Q_0\}$ generated by the random counter. Triangular dots (blue) in Fig. 12 represent first 36 autocorrelation coefficients $(1 \le k \le 36)$ experimentally obtained by the random counter (statistics $3 \cdot 10^8$ numbers, error are bars smaller than the dot size) whereas round dots (red) correspond to the deterministic counter and square dots (black) correspond to completely random numbers. Of course, stream of random values does not show any autocorrelation.

Regarding usability of the random counter for frequency division, we note that if the Input is fed by a RPT, each output Q_i gives another RPT. The frequency of i-th output (Q_i) is equal to the input frequency divided randomly by 2^{i+1} and then deterministically by 2 (in our notation: $f(Q_i) = f_{in}/(2^{i+1})_R/2_D$). Unfortunately, RPTs appearing at outputs $Q_0 - Q_3$ are strongly correlated: any RPT contains all pulses from any lower frequency RPT therefore they cannot be combined to obtain RPTs of frequencies other than those. Thus only divisions by 2^i are possible.

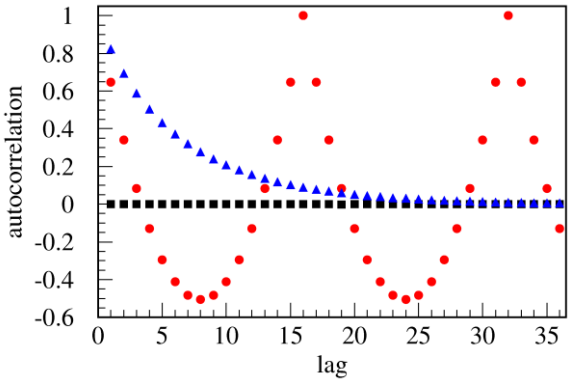

Figure 12. Autocorrelation coefficients with lag 1-36 of 4-bit numbers generated by 3 devices: random counter (blue triangles), periodic up counter (red round dots) and random number generator (black squares). Color online.

In this example of a random counter we see that behavior of circuits containing only a few RFFs may exhibit a very rich behavior. In particular, the random synchronous counter counts erratically up and down, can be used for the generation of correlated random numbers and for the randomness-preserving frequency division by a power of two. However, frequency division by an integer number n is yet unknown. Instead, random frequency synthesis can be used.

VI. RANDOM FREQUENCY SYNTHESIS

Frequency synthesis of periodic signals can be accomplished by well known techniques, for example phase-locked loop (PLL) or direct digital synthesis (DDS). These enable generation of a set of frequencies separated by an arbitrary small, constant increment in a desired frequency interval.

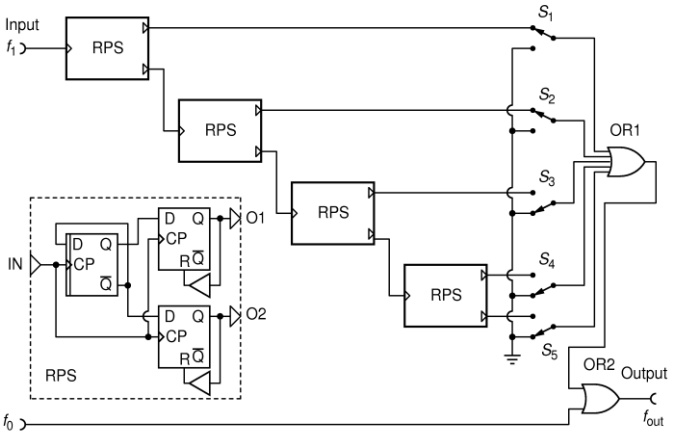

Figure 13. A full-featured random frequency synthesizer in the range $[f_0, f_0 + f_1]$ in steps of $f_1/2^N$. Output frequency f_{out} can be set via switches $S_1 - S_5$.

Using RFF, one is for the first time able to build a full-featured random frequency synthesizer (RFS) with digitally controlled rate. In this example the output signal is a RPT even though this is not the only possibility. A RSF that makes possible synthesis of RPTs having frequency in an arbitrary interval and with arbitrary small steps, is shown in Fig. 13. A random pulse splitter (RPS) conveys a pulse from the input IN randomly to one and only one of its two outputs, O1 or O2. Pulses of frequency f_1 from the Input are randomly picked up and directed towards switches S_i with mutually independent probabilities of $1/2^i$. Depending on the setting of switches, none, some or all pulses from the Input reach the input of the OR1 gate which sums their frequencies by simply interleaving the incoming pulses.

Frequency of the resulting RPT is equal to $(\frac{k}{2^N})f_1$ where k can take on any integer value in the range $0 \le k \le 2^N$. Parameter k depends on setting of the switches:

$$k = S_{N+1} + \sum_{i=1}^{N} 2^{N-i} S_i \quad . \tag{4}$$

Switch setting S_i refers to i-th switch and can take on either value of 0 (switch open) or 1 (switch closed) thus enabling the frequency of RPT coming out of OR1 to take on a value between 0 and f_1 in 2^N equally sized steps. The same summing property of OR gate can be used once again to add an arbitrary frequency f_0. With addition of base frequency f_0 by means of the gate OR2, it is possible to generate a RPT of frequency in the range $[f_0, f_0 + f_1]$ in steps of $f_1/2^N$. Both the starting frequency (f_0), range (f_1) are arbitrary and step size ($f_1/2^N$) can be made as small as desired. Note that when switches are re-set, the random frequency jumps virtually instantaneously to the new value (after combined propagation delay of OR1 and OR2) which is in contrast to other methods that would have finite settling time [13-14].

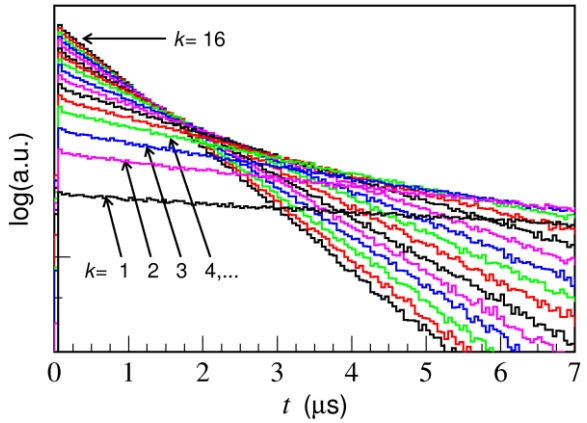

Figure 14. Pulse waiting time distributions for 16 possible output frequencies. For $k = 0$ output frequency is 0.

Using four DRFFs and a single ORPTG of frequency $f_1 = 1$ Mcps (and $f_0 = 0$), the circuit in Fig. 13 was built and tested. It allows generation of 16 frequencies in the range 0-1 Mcps in steps of 0.0625 Mcps. Waiting time distributions, shown in logarithmic scale in Fig. 14, indicate exponential p.d.f. as expected. The largest measured deviation of any output frequency from the expected theoretical value was 230 cps.

One of the advantages of this approach is possibility of generating RPTs with a very low yet precisely known frequency. The current art uses negative feedback loop to set a frequency of an adjustable RPT generator [15-16]. The minimum time required to stabilize loop to a given relative precision rises quadratically with the mean period of the RPT. For example, to stabilize a variable RPTG to 1 Hz within precision 1% it would take at least 10,000 seconds because it takes that long time to measure the random frequency of 1 Hz to 1% precision. To stabilize random frequency of 0.1 Hz it would take 100 times longer, etc. On the other hand, governed by the same law, high random frequency can be stabilized much faster. For example 10 MHz RPTG can be stabilized to 1% in only ~0.001 s. Then, dividing such signal with a randomness preserving divider one can quickly obtain lower frequency with the same relative precision regardless of the division factor.

Frequency synthesis is useful in a wide variety of applications from telecommunications to laboratory equipment. RTPs of precisely known frequency could find application in spread spectrum communication, test equipment, signal generators, research of random number generators, simulation and calibration of nuclear and radiation detectors, in secure authentication methods, spurless fractional frequency division [17] and in computing with random pulse trains [18].

VII. RANDOM NUMBER GENERATOR

Probably the simplest application of the random flip-flop is the random number generator (RNG), a physical device that produces one or more random bits upon a request. Two equivalent circuits, shown in Fig. 15, would generate one random bit upon each rising edge of the Request signal in synchronization with the Strobe signal. If one bit per request is not enough, a random number generator that produces multiple random bits per single request can be obtained by parallelizing a desired number of mutually independent bit generators that share the same Request signal. For example. parallelizing 8 RFFs one could generate one random byte (random integer number between 0 and 255) per a single request pulse.

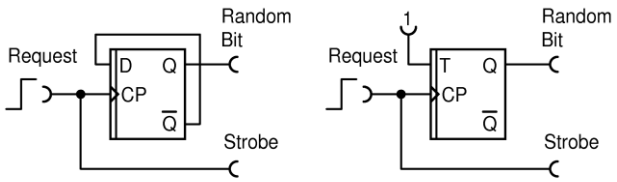

Figure 15. Two equivalent random bit generators: by each rising edge at the Request input circuit generates a fresh new random bit synchronously with the Strobe pulse.

To test the quality of such a RNG we constructed four independent DRFFs (each according to Fig. 6). Internal ORPTGs (each according to Fig. 4) were running at a frequency of ≈ 24 Mcps each. Since all four RNGs share the same Request line, each Request signal results in simultaneous

generation of 4 bits. By applying a periodic request signal of 3 MHz, bits were generated at a total rate of $1.2 \cdot 10^7$ bits/sec and sent to a PC computer via a USB2 interface. Several random sequences, each 10^9 bits long, have been generated this way, stored in a PC computer and their randomness tested with well-known NIST's Statistical Test Suite, version 2.1.1 [19], which they passed without any post-processing. The NIST STS suite is a collection of stringent statistical tests that are used as a standard randomness assessing tool. Passing of the tests is important because it indicates good randomness of the four DRFFs that have been used in experimental demonstrations in this study.

Physically generated random numbers find use in cryptography (securing mobile communications, online payments, e-banking, etc.), Monte Carlo numerical simulations, scientific research etc.

VIII. AN EXAMPLE OF APPLICATION TO HAZARD GAMES: THE ELECTRONIC DICE

Global gambling market is in steady rise since 2003 and has reached $32 billion revenue for online games and over $310 billion in total hazard games [20]. While not all games require random numbers, most lucrative ones do. Fig. 16 shows an elegant construction of electronic dice made of three random flip-flops, that could be perfectly suited as a substitute for 6-sided dice in online games.

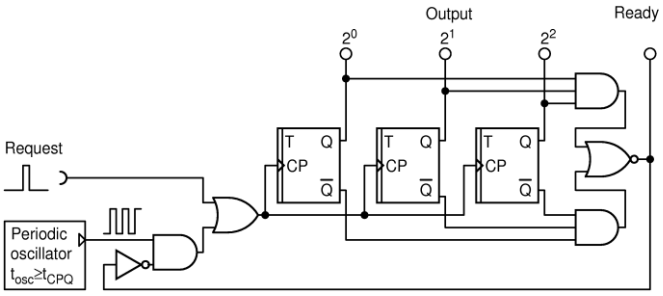

Figure 16. Electronic dice generates random numbers in the range 1-6 upon a request. Completion of a throw is signified by appearance of logic 1 at the output Ready.

It functions in the following way. A signal at the Request input triggers three independent TRFFs to generate a binary random number in the range 0-7 at its Output. If 0 or 7 get generated, then the periodic oscillator keeps sending additional requests until a random number in the range 1-6 is generated by a chance. When this happens, further requests from the periodic oscillator are suppressed and logic "1" appears at the output Ready signifying completion of the "throw". We performed an experiment where the electronic dice was connected to a PC computer via USB2 interface and "thrown" at a rate of 1.5 million throws per second to generate a sample of $N = 10^7$ numbers.

One cannot use NIST STS or similar tests that operate on long strings of random bits to test this sample. Namely, because 6 is not an integer power of 2, convert string of random throws into a string of random binary bits is quite a demanding mathematical task. But even if such conversion would be made, testing the resulting binary string would miss the point. What we need to assess is any un-equalness of probabilities of outcomes and any correlations among successive outcomes that a player could use to his advantage to win the game more frequently that if the outcomes were perfectly random. Therefore in order to test randomness of the dice, since there are no readily available standard randomness tests for this case, two ad-hoc tests were devised. Randomness of the sequence was tested by evaluating normalized Shannon entropy for $M = 6$ throws and for $M = 36$ combinations of two consecutive throws, that is by evaluating:

$$H(M) = -\frac{1}{\log(M)} \sum_{i=1}^{M} p_i \log(p_i)\,. \qquad (5)$$

where p_i are empirical probabilities of each outcome. The entropy $H(M)$ is equal to 1 for an infinite sequence of perfectly fair dice. However for a sequence of finite length (in our case $N = 10^7$) the entropy is smaller than unity and there is a statistical error associated with it. In our analysis we follow the theory developed in [21].

The first test ($M = 6$) runs over probabilities of individual throws and is sensitive to evenness of the probabilities of the 6 possible outcomes (generalized bias), while the second test ($M = 36$) runs over probabilities of all combinations of two successive throws and is sensitive to correlations between successive throws (serial autocorrelation), if any. The obtained results are $H(6) = 0.99999977(15)$ for singles and $H(36) = 0.99999916(30)$ for pairs. They agree with expected theoretical values of 0.99999986 and 0.99999902 within statistical errors. One sigma statistical errors and expected values for the measured set were estimated according to [21]. The first result shows that the 6 outcomes appear with equal probabilities on average while the second shows that knowing outcome of (any) one throw will not help narrowing down the probability for the outcome of the next. Since this system has no intentional memory it is reasonable to assume that any higher-order correlations are even smaller than those tested and that consequently this electronic dice pass test of randomness. In order to cross-check this assumption we evaluate autocorrelation coefficients with lags from 1 to 36 on an experimentally obtained set of 10^8 throws. The average value of the 36 coefficients is $+0.08 \cdot 10^{-4}$ with an expected error of $1.00 \cdot 10^{-4}$ and RMS of $0.99 \cdot 10^{-4}$ confirming that the autocorrelation coefficients up to lag 36 are consistent with zero.

The sustained throughput of 1.5 million throws per second should be more than enough to feed a whole server serving web-based hazard games. In a similar fashion one could envisage a system for mixing of play cards, drawing lottery numbers etc.

CONCLUSION

A new sequential logic element, the random flip-flop (RFF), is defined, built exploiting quantum randomness present in light emission and detection, and tested. Some entirely new applications have been identified and experimentally realized such as: randomness preserving frequency division, random frequency synthesis and simultaneous multi-bit random

number generation. Further novel applications are under investigation. We believe RFF will find many more useful applications in the future and if put on chip could become a potent enabler of new technologies.

ACKNOWLEDGMENT

This work was in part financed by MoSES 533-19-14-0002.

REFERENCES

[1] 74F74 Dual D-type flip-flop, Philips Product specification, http://pdf.datasheetcatalog.com/datasheet/philips/74f74.pdf, last accessed 27.06.2015.

[2] T. F. d M. Wahl, M. Leifgen, M. Berlin, T. Roehlicke, H.J. Rahn, O. Benson, "An ultrafast quantum random number generator with provably bounded output bias based on photon arrival time measurements", Appl. Phys. Lett. **98**, 171105 (2011).

[3] M. Stipčević, "Fast nondeterministic random bit generator based on weakly correlated physical events", Rev. Sci. Instrum. **75,** 4442-4449 (2004).

[4] M. Stipčević, B. Medved Rogina, "Quantum random number generator based on photonic emission in semiconductors", Rev. Sci. Instrum. **78,** 045104:1-7 (2007).

[5] M. Stipčević, "A novel active quenching circuit for single photon detection with Geiger mode avalanche photodiodes", Applied Optics **48,** 1705-1714 (2009).

[6] M. Stipčević, D. Wang, and R. Ursin, "Characterization of a commercially available large area, high detection efficiency single-photon avalanche diode", IEEE J. Lightwave Technol. **31**, 3591-3596 (2013).

[7] M. Stipčević, R. Ursin, "An On-Demand Optical Quantum Random Number Generator with In-Future Action and Ultra-Fast Response", Scientific Reports **5**, 10214:1-8 (2015).

[8] V. B. Minuhin, "3/2 frequency divider", patent US4866741, priority date Sep. 12, 1989.

[9] N. Regimbal, F. Badets, D. Bordeaux, J. B. Begueret, inventors; CNRS, Stmicroelectronics S.A., applicants; "Fractional frequency divider", US patent US2012/0001665, 2011 Jun 29.

[10] R. E. Best, "Phase-locked Loops: Design, Simulation and Applications", McGraw-Hill 2003, ISBN 0-07-141201-8

[11] A. Papoulis, S. U. Pillai, Probability, Random Variables, and Stochastic Processes, 4th Edition, McGraw-Hill Europe 2002, ISBN-13: 978-0071226615.

[12] D. E. Knuth, The art of computer programming, Vol. 2, Third edition, (Addison-Wesley, Reading, 1997).

[13] A.S. Robinson, "Pulse rate function generator", patent US3217148 priority date June 28, 1960.

[14] J. A. Hanby, S. J. Redman, "Random Pulse Train Generator with Linear Voltage Control of Average Rate", Rev. Sci. Instrum. **42**, 657-662 (1971).

[15] G. White, "The generation of random-time pulses at an accurately known mean rate and having a nearly perfect Poisson distribution", J. Sci. Instrum. **41**, 361-364 (1964).

[16] Noriyoshi, Y. Hiroyasu, "Random pulse generator and random number generation device and probability generation device utilizing the random number generator", EP1094603A1, 20. Oct 1999

[17] V .S. Reinhardt, R. P. Verdes, I. Shahriari, inventors; Hughes Aircraft Company, asignee; "Spurless fractional divider direct digital frequency synthesizer and method", United States Patent US4815018, 1989, Mar 21.

[18] R.C. Lawlor, "Computer ulitizing random pulse trains", US patent US3612845

[19] A. Rukhin, J. Soto, J. Nechvatal, M. Smid, E. Barker, S. Leigh, M. Levenson, M. Vangel, D. Banks, A. Heckert, J. Dray, S. Vo, "A Statistical Test Suite for the Validation of Random Number Generators and Pseudo Random Number Generators for Cryptographic Applications", NIST Special Publication 800-22rev1a (dated April 2010). URL: http://csrc.nist.gov/groups/ST/toolkit/rng/documents/sts-2.1.1.zip

[20] http://www.statista.com/markets/409/topic/438/gambling/ , last accessed 27.06.2015.

[21] M. S. Roulston, "Estimating the errors on measured entropy and mutual information", Physica D **125,** 285–294 (1999).