A System for Generating Non-Uniform Random Variates using Graphene Field-Effect Transistors

Nathaniel J. Tye  
Department of Engineering,  
Cambridge Graphene Centre  
University of Cambridge  
njt48@cam.ac.uk

James T. Meech  
Department of Engineering  
University of Cambridge  
jtm45@cam.ac.uk

Bilgesu A. Bilgin  
Department of Engineering  
University of Cambridge  
bab46@cam.ac.uk

Phillip Stanley-Marbell  
Department of Engineering  
University of Cambridge  
phillip.stanley-marbell@eng.cam.ac.uk

Abstract—We introduce a new method for hardware non-uniform random number generation based on the transfer characteristics of graphene field-effect transistors (GFETs) which requires as few as two transistors and a resistor (or transimpedance amplifier). The method could be integrated into a custom computing system to provide samples from arbitrary univariate distributions. We also demonstrate the use of wavelet decomposition of the target distribution to determine GFET bias voltages in a multi-GFET array.

We implement the method by fabricating multiple GFETs and experimentally validating that their transfer characteristics exhibit the nonlinearity on which our method depends. We use the characterization data in simulations of a proposed architecture for generating samples from dynamically-selectable non-uniform probability distributions.

Using a combination of experimental measurements of GFETs under a range of biasing conditions and simulation of the GFET-based non-uniform random variate generator architecture, we demonstrate a speedup of Monte Carlo integration by a factor of up to 2×. This speedup assumes the analog-to-digital converters reading the outputs from the circuit can produce samples in the same amount of time that it takes to perform memory accesses.

Index Terms—Monte Carlo Accelerator, Non-Uniform Random Variates, Graphene, Graphene Transistors

I. INTRODUCTION

Hardware uniform random number generators exist in both research and commercial computer architectures, with generation rates of up to 6.4 Gb/s [12]. Uniform random numbers are widely used in applications such as cryptography, where the objective is to generate bit vectors (e.g., 256-bit vectors) that are uniformly distributed over some range and are therefore difficult to guess. In contrast, this article focuses on non-uniform random number generators.

A. Applications of non-uniform random variates

Many important applications in science and engineering depend not on uniform random samples, but instead require non-uniform random variates (random samples chosen from a non-uniform probability distribution), from a wide range of distributions. Examples of these applications of non-uniform random variates range from Monte Carlo simulations [27], to quantitative finance [34] to particle filter localization for driverless cars [36]. Non-uniform random variates are also important in Bayesian machine learning applications [17], which involve the computation of a marginal probability which goes into the denominator of the expression of Bayesâ€™s rule. Computing these marginal probabilities in turn requires evaluating an integral of a probability distribution. Because the distributions in question are typically high-dimensional and have no known analytic equational form, their integration often requires the use of Monte Carlo integration methods where one samples repeatedly from the corresponding distribution.

B. Challenges

Because generating samples from distributions whose inverse cumulative distribution function (CDF) does not have a closed form requires the use of time-consuming rejection sampling [6], generating random samples from non-uniform distributions is typically an order of magnitude slower and less energy-efficient than generating uniformly distributed random samples [35]. One promising direction for efficiently generating non-uniform random variates is to sample from a physical process whose evolution in time [42] or noise characteristics [25] follow some known and (ideally) controllable probability distribution.

C. Contributions

This article presents the first demonstration of generating non-uniform random variates by exploiting properties of GFETs previously considered to be undesirable: their ambipolar transfer characteristics and their lack of a bandgap. We provide a tutorial overview of the properties of GFETs (Section III) and introduce a circuit topology for using a chain of GFETs together with a uniform random variate generator to generate dynamically-controllable non-uniform distributions (Section III). We present the methodology we used for fabricating an array of GFETs and empirically characterizing their transfer characteristics (Section IV) and we use those empirically-measured GFET transfer characteristics to demonstrate the proposed method in a simulated combined circuit topology (Section V). We then use the generated non-uniform random variates in an end-to-end system example, where we evaluate their benefit to speeding up Monte Carlo integration, as well as their benefit to reducing the error in the Monte Carlo integration process (Section VI). We propose this system as a component/unit in a more general computing system.
Fermi Energy Level, \( \varepsilon_F \), is the Fermi level. In a undoped/unbiased graphene, it is located at the Dirac point. There is no band gap: GFETs have low on- to off-current ratios making them a poor choice for traditional digital logic applications.

II. PROPERTIES OF GRAPHENE FIELD-EFFECT TRANSISTORS

GFETs have a channel made of single- or multi-layer graphene, rather than a semiconducting material such as silicon or germanium [32]. Unlike traditional semiconducting materials, graphene is a semi-metal: it lacks a band gap and its conduction and valence bands don’t overlap. Instead, the conduction and valence bands meet at a single point, known as the Dirac point (Figure 1).

Electrons at the Dirac point are effectively massless and so have unusually high electron mobilities. As a result, the phonon-limited carrier mobility (the highest possible mobility limited by interactions between carriers and vibrations of the channel’s crystal lattice) of graphene on SiO_2 is predicted to be as high as 200,000 cm^2 V^{-1} s^{-1} [4]. Although high electron mobilities result in more efficient flow of charge, the lack of a band gap means GFETs have low on- to off-current ratios and can never completely turn off, making them unsuitable for digital logic applications [15].

The poor on- to off-current ratios of GFETs in digital logic applications do not preclude their use in other areas of computing. Because it is possible to tune the Fermi level in graphene (which typically lies at the Dirac point) by biasing the channel, it is possible to control device characteristics, e.g., using multi-gate structures, in a manner not equally possible in typical metal-oxide-semiconductor field-effect transistors (MOSFETs). GFETs also have unique transfer characteristics: as the gate voltage is swept, the drain current exhibits a v-shaped characteristic curve, with the conductance increasing until it reaches a minimum value before increasing again.

III. A GFET NON-UNIFORM RANDOM VARIATE GENERATOR

If the signal at the gate of a GFET is a uniform random voltage distribution, then the distribution of the drain current will be modified by the GFET’s transfer characteristics. The exact shape of the transfer characteristics varies with the source-drain voltage \( V_{DS} \). Thus, for a uniform random voltage distribution at the gate, varying \( V_{DS} \) for a GFET changes the distribution of drain currents. If these drain currents are converted to a voltage and passed through additional GFETs, it is possible to combine the transfer characteristics and biasing of multiple GFETs to achieve a range of drain current (and hence voltage) distributions.

Figure 2 shows a possible circuit to implement generation of a controllable non-uniform voltage distribution using GFET properties. Each GFET in Figure 2 has a bias voltage, \( V_{DS} \), that controls its transfer characteristics. The first GFET has a uniformly-distributed random voltage across its gate and a corresponding distribution of drain currents, with the values of the drain currents for each input voltage determined by the transfer characteristics of the GFET at its bias voltage \( V_{DS} \). The circuit in Figure 2 converts the drain current of the first GFET into a voltage input to the gate of the second GFET, using a resistor, \( R_1 \). In practice, using a transimpedance amplifier (TIA) to perform this current-to-voltage conversion would result in less Johnson-Nyquist noise in the generated voltages, though the presence of such noise may not be detrimental given our goal of generating random variates. The analyses that follow in Section V therefore use a resistor for converting the drain currents to voltages to control the second stage in the circuit.

The second GFET in Figure 2, operating at a bias voltage of \( V_{DS} \) further shapes the distribution of the output signal. By selectively connecting multiple GFETs in the manner of Figure 2 (and possibly using multi-gate GFETs), this method in principle permits generation of a final output \( V_{out} \) with a range of selectable distributions, controlled by the combination of \( R_1 \), \( V_{DS} \), and \( V_{DS} \).

A. Integration with an Existing Computer Architecture

For integration into a larger system, we propose the use of a programmable analog switching matrix, e.g. a MAX11300 [24], such as that illustrated in (Figure 3). This is used as an interface between an external microcontroller or processor and the GFET die and allows for dynamic reconfiguration of the GFET circuit. We present a hardware prototype of this analog switching matrix in Section VI.

The GFET random number generator could be integrated into a package with an existing CPU and ADC using bond wires to connect the two separate dies. Figure 4 shows a block diagram of the arrangement. The maximum frequency, \( f \), and energy cost, \( E \), for a signal traveling across the bond wires are \( 2.55 \times 10^{17} \) b/s and \( 1.82 \times 10^{14} \) J/b respectively using the values in Table I. The maximum frequency at which a bit on the bond wire could change state is calculated using the capacitance between the bond wires, \( C \), and the resistance...
across a bond wire, R. Hughes et al. [11] show that \( f \) is given by:

\[
f = \frac{1}{5RC}.
\]

The energy cost of changing the value of a bit on the bond wire is calculated using the logic high voltage \( V \). \( E \) is then given by [11]:

\[
E = CV^2.
\]

The capacitance \( C \) used in Equations 1 and 2 is calculated using \( \epsilon_0 \) as the permittivity of free space, \( L \) as the length of the bond wire, \( a \) as the radius of the bond wire, and \( d \) as the distance between them. Grigsby [8] shows that \( C \) is given by:

\[
C = \frac{\pi \epsilon_0 L}{\ln \frac{d-a}{a}}.
\]

The resistance \( R \) used in Equation 3 is calculated using \( \rho \) as the resistivity of a bond wire and \( A \) as the cross-sectional area of a bond wire. Grigsby [8] shows that \( R \) is given by:

\[
R = \frac{\rho L}{A}.
\]

These constraints are negligible, the overall speed is limited by the ADC sample rate and the overall power consumption is determined by the ADC and GFET random number generator.

B. Wavelet Decomposition and Reconstruction of Distributions

In signal processing, the Fourier transform decomposes a time-varying signal into its constituent frequency components and the Fourier series allows for the construction of an arbitrary signal from a sum of sines and cosines. This is a special case of wavelet analysis, which allows for any function to be described by a set of orthonormal basis functions.

In wavelet analysis, an analysing wavelet is used with a scaling function to generate a set of basis functions. These basis functions are simply scaled and shifted versions of the analysing wavelet [7] and the inner product of the scaling and wavelet functions, which are necessarily orthogonal, gives a matrix of wavelet coefficients.

The discrete wavelet transform (DWT) uses known scaling and wavelet functions to generate a known set of basis functions. When applied to a discrete signal, those basis functions give an approximation of the signal and the signal is characterised by it’s wavelet coefficients [5]. The inverse transform is simply the linear combination (i.e. the sum) of these basis functions, and thus allows for reconstruction of the original signal with a desired level of accuracy dependent on the number of coefficients used.

In the DWT, the set of coefficients can be considered as a transformation matrix or filter which is applied to a data vector. The coefficients are ordered using two patterns: the first acts as a smoothing filter and the second brings out details in the data. A pyramidal algorithm is used to apply the matrix, with coefficients arranged such that odd rows contain coefficients acting as a smoothing filter and even rows contain those acting as those which bring out the data’s detail. Each pair of rows can be thought of as a level of analysis; as the number of levels increases, the total number of inner products is divided by two [7]. Thus, each step smooths the data and so information is lost.

We demonstrate wavelet decomposition and reconstruction of a distribution in the following example. We show the distributions reconstructed from inverse DWTs with different numbers of coefficients, corresponding to a given level of accuracy for the lognormal distribution in Figure 5. We used a second-order Coiflet wavelet for both the DWT and the inverse-DWT. We chose this wavelet arbitrarily as a proof-of-concept for the proposed method, but it appeared to give reasonable results. We calculated the Kullback-Leibler (KL) divergence...
[16], a measure of the closeness of two distributions, by calculating the peak positions of the generated distribution and the reconstructed distributions and comparing them. Figure 5(b) uses the most coefficients and was thus the most accurate reconstruction, with a calculated KL divergence of 0, an identical reconstruction. Figure 5(c) used less than half the coefficients of (b) and had a KL divergence of 1.19. Figure 5(d), which used less than a quarter of the coefficients of (b) was actually closer than (c), with a KL divergence of 0.45.

Fig. 5. (a) A software-generated lognormal distribution; (b) reconstructed distribution using 55 coefficients; (c) reconstruction using 22 coefficients; (d) reconstruction using 12 coefficients. In each case, the x-axis is simply a number.

Several approaches to implementing wavelet transforms have been developed: Stephane Mallat proposed a Fast Wavelet Transform algorithm [21] and DWT algorithms have been implemented using FPGAs [2]. The output of such an approach would form part of the digital input in Figure 4. Figure 5 is a block diagram of a proposed complete system. Section IV and Figure 7(c) present an early prototype of this system.

We have shown that a distribution can be reconstructed from a set of wavelet coefficients determined by a wavelet transform. If we consider these coefficients to be bias voltages for GFETs, which have a tuneable characteristic transfer function with a certain distribution, then the characteristic of a GFET can be considered as a mother wavelet, with the bias voltages being the scaling parameters. Summing the output of each GFET (or combination of GFETs), with each representing a basis function, in principle, allows for the reconstruction of any arbitrary function, with the accuracy dependent on the number of GFETs used.

IV. GFET Fabrication and Electrical Characterization

GFETs in the results presented here consist of an Si substrate onto which we patterned a gold back gate. We grow a 60 nm alumina (Al₂O₃) layer by atomic layer deposition (ALD), onto which we transfer a monolayer of graphene using a wet transfer process. This graphene monolayer was grown by chemical vapor deposition (CVD) and purchased from Graphenea. After patterning the graphene, we deposit the gold source and drain contacts onto the graphene to create a 40 μm × 40 μm GFET channel that lies exactly above the back gate. The channel is insulated by another layer of alumina, onto which we pattern gold top gates aligned with the GFET channels. We electrically passivate the whole device by a final ALD deposition of alumina. Finally, to facilitate electrical access to the GFETs, we etch away the alumina on top of the contacts that are electrically linked to the source, drain, back, and top gates of the GFETs.

We fabricate four identical GFETs on each silicon die (Figure 7(a)) and we electrically connect the GFETs via wire-bonding the die from its gold contacts (Figure 7(b)) to a custom printed circuit board (PCB) (Figure 7(c)). The PCB comprises an array of DACs, ADCs, and analog switches, all of which allow for dynamic and in-situ (re)configuration of a given
Fig. 8. (a) Plot of the drain current, $I_D$, against the top gate voltage, $V_{GS}$, for different bias voltages of $V_{DS}$; (b) plot of the drain current, $I_D$, against the source-drain voltage, $V_{DS}$, for a stepped gate-source voltage $V_{GS}$.

circuit. Because graphene is sensitive to atmospheric effects\footnote{In principle, atmospheric effects can lead to doping of the channel. These effects should however not occur even in the absence of the sealed glass protective cover, as we fabricated the devices in a cleanroom environment and encased the graphene in alumina as described above. We however cannot rule out inadvertent doping as a result of the fabrication process.}, and also to protect the wire-bonding, we place a glass protective cover over each die once bonded to the PCB, sealed with hot glue. The hardware prototype in Figure 7(c) implements the GFETs required to realise the circuit in Figure 2, as well as the analog switching matrix described in Figure 3 and Figure 4.

We performed electrical characterization of the GFETs using two Keithley 2450 source-measure units (SMUs) synchronized using TSP-link. Figure 8(a) shows the transfer characteristic for a constant source-drain bias voltage ($V_{DS}$), which we updated for each measurement.

The Dirac points in Figure 8(a) lie to the left of 0.0 V, which suggests an n-type doping of the graphene channel. The deepening of the valley with increasing bias voltage $V_{DS}$ is a commonly-observed characteristic in GFETs\footnote{In principle, atmospheric effects can lead to doping of the channel. These effects should however not occur even in the absence of the sealed glass protective cover, as we fabricated the devices in a cleanroom environment and encased the graphene in alumina as described above. We however cannot rule out inadvertent doping as a result of the fabrication process.}, as is the hysteresis in the transfer characteristics, which the measurements of Figure 8(a) show for all applied bias voltages. This hysteresis is a result of multiple phenomena: charge trapping between the graphene channel and the insulating layers are a major cause\footnote{In principle, atmospheric effects can lead to doping of the channel. These effects should however not occur even in the absence of the sealed glass protective cover, as we fabricated the devices in a cleanroom environment and encased the graphene in alumina as described above. We however cannot rule out inadvertent doping as a result of the fabrication process.}, however, additional factors include capacitive gating causing a negative shift and charge transfer causing a positive shift\footnote{In principle, atmospheric effects can lead to doping of the channel. These effects should however not occur even in the absence of the sealed glass protective cover, as we fabricated the devices in a cleanroom environment and encased the graphene in alumina as described above. We however cannot rule out inadvertent doping as a result of the fabrication process.} in the conductance with respect to gate voltage.

We also measured the $I_{DS}$ versus $V_{DS}$ characteristics of the GFETs while varying the gate voltage $V_{GS}$ between 0.2 V and 1.0 V, for source-drain voltages $V_{DS}$ over the range 0.0 V to 10.0 V. Traditional MOSFETs exhibit saturation of their $I_{DS}$ versus $V_{DS}$ characteristics, with the characteristics separated into two main operating regions: linear and nonlinear. In GFETs however, this saturation does not appear, due to a combination

of graphene’s lack of a bandgap and Klein tunneling\footnote{In principle, atmospheric effects can lead to doping of the channel. These effects should however not occur even in the absence of the sealed glass protective cover, as we fabricated the devices in a cleanroom environment and encased the graphene in alumina as described above. We however cannot rule out inadvertent doping as a result of the fabrication process.}. Figure 8(b) shows the measured characteristic curves for the GFET and Figure 8(a) shows the transfer characteristics.

The characterization data in Figure 8(b) indicate that the devices switch from a relatively linear region of conductance to a nonlinear region at a bias voltage of around 3.5 V. This is in line with previous results which show that GFETs, in comparison to MOSFETs, often have a second linear region; there is a point of inflection\footnote{In principle, atmospheric effects can lead to doping of the channel. These effects should however not occur even in the absence of the sealed glass protective cover, as we fabricated the devices in a cleanroom environment and encased the graphene in alumina as described above. We however cannot rule out inadvertent doping as a result of the fabrication process.} which appears to be the case in the plots here, at a $V_{DS}$ of approximately 3.5 V.

As we show in Section V the nonlinearity of the GFET transfer characteristics, combined with the tunability of the characteristic shape by controlling $V_{DS}$ allows us to use one or more GFETs to trans form uniform distributions of $V_{GS}$ into non-uniform distributions of $I_{DS}$.

V. SIMULATIONS OF GFET CIRCUITS

We use the GFET characterization data from Section IV to simulate possible topologies for the GFET-based non-uniform random variate generator of Figure 2 using a custom-built simulation model of the circuit, implemented in Mathematica. We use an interpolating function to model the measured device characteristics of each GFET and stimulate the gate of the first GFET in the circuit using a uniform random distribution between $-8$ and $+8$, effectively a $V_{GS}$ voltage in the range $-8$ V to $+8$ V. Figure 2 shows the time series of the uniform random voltages and their corresponding histogram distribution. We pass the output of the model of the first GFET, which is its drain current, through a modeled resistor which converts the drain current into a voltage. We then feed this voltage to the model of the second GFET of Figure 2 which we again model by encapsulating our experimental measurement data in another interpolating function. We apply the drain current of the second GFET to another resistor to obtain the output voltage.

We used the characteristics of the GFET in Figure 8 for simulations here. For the first GFET, we used the characteristic for a 0.8 V bias and for the second GFET, we used the characteristic for a 1 V bias. We use a resistance of 2.2 kΩ for the first resistor and of 1 kΩ for the output resistance. Figure 10(a) shows the distribution of currents from the first GFET, and Figure 10(b) shows the distribution for the second GFET.
Algorithm 1: Monte Carlo integration.

Result: Error $E$ and time $t$

Timer start
Generate $N$ random samples from distribution $D$
Sort $N$ random samples

for All pairs of samples do
  $b = \text{Samples}[i] - \text{Samples}[i - 1]$
  $h = (f(\text{Samples}[i]) + f(\text{Samples}[i - 1]))/2$
  $A^+ = b \ast h$
end

$E = \text{abs}(1 - A)$
Timer stop
t = stop - start
RETURN $E, t$

A. Results

Figure 12(a) shows that it is on average $1.05 \times$ faster to use a C++ lognormal random number generator than a C++ uniform random number generator. Running the program assuming that the lognormal samples are generated by the hardware random number generator in the same amount of time required for a memory access is up to $1.99 \times$ faster and always at least $1.26 \times$ faster than using the C++ lognormal random number generator. Figure 12(b) shows that the error reduction for the Monte Carlo integration using the $[0, 3]$ C++ uniform random number generator plateaus at around $10^4$ samples but for the hardware lognormal the error continues to decrease. The lognormal and $[0, 3]$ C++ uniform lines intersect at between $10^5$ and $10^6$ samples and the intersection point shifts to the

where $F(x)$ is the output distribution of the circuit. To compare the similarity of the generated distribution to a genuine lognormal, we calculated the mean ($\mu$) and standard deviation ($\sigma$) of the logarithm of the simulation output and used these as parameters to generate a lognormal distribution over the same input space. Figure 11(b) shows histograms of the simulation output and the reference distribution.

VI. END-TO-END EXAMPLE: MONTE CARLO INTEGRATION

In Monte Carlo simulations it is common to need to integrate various un-normalized non-uniform density functions to convert them to valid probability density functions [34]. Monte Carlo integration is a convenient way of doing this. The normalization could require the integration of a lognormal distribution $f(x)$ where $A$ is an unknown normalizing constant, $\mu = 0$ is the mean and $\sigma = 0.25$ is the standard deviation:

$$f(x) = \frac{A}{\sigma \sqrt{2\pi}} e^{-\frac{(\ln(x) - \mu)^2}{2\sigma^2}}. \quad (6)$$

Let $E$ be the error of a Monte Carlo integration and $t$ be the time taken by the integration. Let $N$ be the number of random samples used in the integration and $D$ be the distribution that we sample from. Let $A$ be the area of each rectangle used in the integration and $b$ and $h$ be the corresponding rectangle base and height. Algorithm 1 shows the integration scheme that we used.

We repeated the integration with $D$ as: 1) a hardware generated lognormal distribution and 2) a lognormal distribution generated with the C++ standard library’s utility for generating lognormal variates, with the same $\mu$ and $\sigma$ as $f$. We also performed the integration with $D$ as a uniform distribution generated with the C++ standard library’s random number generator, with various ranges. We assume that samples from the hardware lognormal generator can be generated in the time required for one memory access. We ran all simulations on an 2.8 GHz Intel Core i7 CPU using OpenMP parallelization to utilize all eight processor threads.
right as we increase the range of the C++ uniform random number generator.

B. Insights from Monte Carlo Integration

Figure 12b) shows that increasing the range of the C++ uniform random number generator decreases the minimum error that the integration plateaus at. Unfortunately increasing the range of the C++ uniform random number generator also increases the number of samples required for the estimate of the area to approach the true value. The proportion of the uniform probability density function overlapping the lognormal probability density function decreases as the range of the uniform distribution is increased. For a given function we cannot know beforehand which range of uniform random numbers will produce a sufficiently small bound on the error of integration. We can avoid this problem by sampling from the exact lognormal density that we want to integrate. When performing the Monte Carlo integration of any non-uniform distribution we should sample from the probability density function of that distribution with the same parameters to minimize the error and number of samples required to get a reasonable estimation. This is not possible when sampling from a bounded uniform distribution.

VII. RELATED RESEARCH

A hardware random number generator, integrated in a CPU capable of producing samples from arbitrary distributions does not currently exist. Table II shows the state-of-the-art of hardware non-uniform random number generators. The prior work on non-uniform random number generation in Table II is fundamentally different to prior work on uniform random number generation. The publications in Table II characterize the non-uniform distribution of the physical process used to obtain the random samples. The prior work on uniform random number generators does not produce or refer to a non-uniform distribution of random numbers [18], [30], [33], [39]. No comparison can be made between the GFET and uniform random number generators. The uniform random number generation efforts excluded from Table II produce single bit samples where the result is either 0 or 1. The non-uniform random number generation efforts included in Table II produce multiple (usually 6 or greater) bit samples with a given non-uniform distribution. Prior work that is capable of producing samples from arbitrary non-uniform distributions exists [28]. Their method is not well suited for integration with current CPU architectures as it requires large optical components, it is therefore unclear how it could be miniaturized and integrated into a CPU [28]. In contrast Section III describes how the GFET random number generator would interface with an existing CPU.

Currently no architecture exists with a Gb/s generation rate for arbitrary non-uniform distributions. As the method we present is analog, it should be possible to use existing technology to drive and sample from it. This will allow us to produce samples from arbitrary distributions at Gb/s sample rates. Popular statistical tests such as Dieharder are designed for samples from a [0, 1] uniform distribution and are therefore not compatible with the non-uniformly distributed random numbers produced in this work [3].

VIII. SUMMARY AND INSIGHTS

This article demonstrates a novel circuit-level approach to generating samples from non-uniform probability distributions, exploiting the transfer characteristics and ambipolarity of graphene field-effect transistors (GFETs).

We describe the fabrication of arrays of GFETs on a silicon substrate and wire bonding of the fabricated devices to a custom PCB. We experimentally characterize the GFET transfer and output characteristics at a range of GFET $V_{DS}$ bias voltage configurations. Using the obtained characterization data, we simulate possible circuit designs for non-uniform random number generators comprising circuits requiring just two transistors and a resistor (or transimpedance amplifier). The results demonstrate that a circuit comprising a chain of two GFETs transforms a uniformly-distributed random input voltage into a non-uniformly distributed output. In the first

### Table II

| Architecture | Speed | Distribution(s) | Year | Paper |
|--------------|-------|-----------------|------|-------|
| Memristor    | 6.00kb/s | Unnamed         | 2017 | [13]  |
| Photon Detection | 1.77Gb/s | Exponential     | 2017 | [17]  |
| FRET         | 2.89Gb/s | Exponential     | 2018 | [17]  |
| Photo Diode  | 17.4Gb/s | Husumi          | 2018 | [17]  |
| Photon Detection | 66.0Mb/s | Arbitrary       | 2018 | [17]  |
| Photon Detection | 200Mb/s | Normal          | 2018 | [17]  |
| Photon Detection | 320Mb/s | Exponential     | 2018 | [17]  |
| Electronic Noise | 6.40Gb/s | Normal          | 2019 | [17]  |
| Photon Detection | 6.80Mb/s | Exponential     | 2019 | [17]  |
| Photon Detection | 63.5Mb/s | Exponential     | 2019 | [17]  |
| Photon Detection | 8.25Mb/s | Normal          | 2019 | [17]  |
| Photon Detection | 1.00Mb/s | Exponential     | 2019 | [17]  |
| Electronic Noise | 13.8kb/s | Normal          | 2020 | [25]  |
demonstration, biasing the first GFET at 0.8 V, outputting current through a 2.2kΩ resistor and inputting the resultant voltage to the gate of the second GFET, biased at 1 V, produces an output voltage distribution through a 1kΩ output resistor resembling a Burr-type XII distribution. Varying the GFET bias voltages and the resistances permits the circuit to generate other dynamically-chosen distributions. We generated an approximation of a lognormal distribution by biasing the first GFET in the simulation to 1 V and the second to 0.8 V, setting the first resistor to 1.2kΩ and keeping a 1kΩ output resistance, and then taking the complementary cumulative distribution function.

We evaluate the end-to-end use of the GFET-circuit-generated distributions in an application performing Monte Carlo integration. The results show that, using the GFET-circuit-generated non-uniform distributions instead of uniform random samples for sampling locations in the lognormal distribution improves the speed of Monte Carlo integration by a factor of up to 2×. This speedup is based on the assumption that the analog-to-voltage to the gate of the second GFET, biased at 1 V, produces an output voltage distribution through a 1kΩ output resistor resembling a Burr-type XII distribution. Varying the GFET bias voltages and the resistances permits the circuit to generate other dynamically-chosen distributions. We generated an approximation of a lognormal distribution by biasing the first GFET in the simulation to 1 V and the second to 0.8 V, setting the first resistor to 1.2kΩ and keeping a 1kΩ output resistance, and then taking the complementary cumulative distribution function.

ACKNOWLEDGEMENTS

This research is supported by an Alan Turing Institute award TU/B/000096 under EPSRC grant EP/N510129/1, by EPSRC grant R022534/1, and by EPSRC grant EP/V004654/1. N.J. Tye acknowledges funding from EPSRC grant EP/L016087/1. J.T. Meech acknowledges funding from EPSRC grant EP/L015899/1.

REFERENCES

[1] M. Avesani, D. G. Marangon, G. Vallone, and P. Villocresi. Source-device-independent heterostructure-based quantum random number generator at 17 gbps. Nature communications, 9(1):1–7, 2018.
[2] M. Bahoura and H. Ezzaidi. Fpga-implementation of discrete wavelet transform with application to signal denoising. Circuits, Systems, and Signal Processing, 31(3):987–1015, 2012.
[3] R. G. Brown. Dieharder. Available at: http://webhome.phy.duke.edu/~rgb/Dieharder.php
Accessed 17/04/2020.
[4] J.-H. Chen, C. Jang, S. Xiao, M. Ishigani, and M. S. Fuhrer. Intrinsic and extrinsic performance limits of graphene devices on sio2. Nature nanotechnology, 3(4):206–209, 2008.
[5] I. Daubechies. Ten Lectures on Wavelets, pages 53–55. Society for Industrial and Applied Mathematics, Jan. 1992. ISBN: 0898712742.
[6] L. Devroye. Non-Uniform Random Variate Generation. page 42. Springer-Verlag, McGill University Montreal H3A 2K6 Canada, 1986. ISBN: 1461386454.
[7] A. Graps. An introduction to wavelets. IEEE Computational Science and Engineering, 2(2):50–61, 1995.
[8] L. L. Grigsby. Electric power generation, transmission, and distribution, pages 144,158. CRC press, 2016. ISBN: 1439856281.
[9] X. Guo, C. Cheng, M. Wu, Q. Gao, P. Li, and Y. Guo. Parallel real-time quantum random number generator. Optics letters, 44(22):5566–5569, 2019.
[10] Y. Hu, Y. Wu, Y. Chen, G. C. Wan, and M. S. Tong. Gaussian random number generator: Implemented in fpga for quantum key distribution. International Journal of Numerical Modelling: Electronic Networks, Devices and Fields, 32(3):e2554, 2019.
[11] E. Hughes, J. Hiley, K. Brown, and I. McKenzie-Smith. Hughes electrical and electronic technology, chapter 5, page 119 and 123. Pearson Education, 2012. ISBN: 0273755102.
[12] Intel. Intel® digital random number generator (drg). 2018. Available at: https://software.intel.com/sites/default/files/managed/98/4a/DRNG_Software_Implementation.pdf
Accessed 17/04/2020.
[13] H. Jiang, D. Belkin, S. E. Savelyev, S. Lin, Z. Wang, Y. Li, S. Joshi, R. Midya, C. Li, M. Rao, et al. A novel true random number generator based on a stochastic diffusive memristor. Nature communications, 8(1):1–9, 2017.
[14] J. Jedzierski, P. Hsu, P. Healey, P. W. Wyatt, C. L. Keast, M. Sprinkle, C. Berger, and W. A. de Heer. Epitaxial graphene transistors on sic substrates. IEEE Transactions on Electron Devices, 55(8):2078–2085, Aug 2008.
[15] R. W. Keyes. What makes a good computer device? Science, 230(4722):138–144, 1985.
[16] S. Kullback and R. A. Leibler. On information and sufficiency. Ann. Math. Statist., 22(1):79–86, 03 1951.
[17] B. Lambert. A Student’s Guide to Bayesian Statistics, pages 23–50. SAGE, 2018. ISBN: 1473916364.
[18] K. Lee and M. Lee. True random number generator (t rng) utilizing fm radio signals for mobile and embedded devices in multi-access edge computing. Sensors, 19(19):4310, 2019.
[19] M. Lemme. Current status of graphene transistors. In Getting and Defect Engineering in Semiconductor Technology XIII, volume 156 of Solid State Phenomena, pages 499–509. Trans Tech Publications Ltd, 1 2010.
[20] J. Lin, Y. Wang, Q. Cao, J. Kuang, and L. Wang. True random number generation based on arrival time and position of dark counts in a multichannel silicon photomultiplier. Review of Scientific Instruments, 90(11):114704, 2019.
[21] S. G. Mallat. A theory for multi-resolution signal decomposition: the wavelet representation. IEEE Transactions on Pattern Analysis and Machine Intelligence, 11(7):674–693, 1989.
[22] D. G. Marangon, G. Vallone, and P. Villocresi. Source-device-independent ultrafast quantum random number generator. Physical review letters, 118(6):060503, 2017.
[23] MatWeb. Gold, Au. Available at: http://www.matweb.com/search/datasheet.aspx?matguid=222119a08904a0affe70e59408cd8bd88&ckc=1
Accessed 17/04/2020.
[24] Maxim Integrated. PIXI, 20-Port Programmable Mixed-Signal I/O with 12-Bit ADC, 12-Bit DAC, Analog Switches, and GPIO. 2016. Rev. 3.
[25] J. T. Meech and P. Stanley-Marbell. Efficient programmable random variate generation accelerator from sensor noise. 2020. arXiv:2001.05400.
[26] I. Meric, M. Y. Han, A. F. Young, B. Ozylilmaz, P. Kim, and K. L. Shepard. Current saturation in zero-bandgap, top-gated graphene field-effect transistors. Nature nanotechnology, 3(1):654–659, 2008.
[27] N. Metropolis and S. Ulam. The monte carlo method. Journal of the American statistical association, 44(247):335–341, 1949.
[28] L. Nguyen, P. Rehain, Y. M. Sua, and Y.-P. Huang. Programmable quantum random number generator without postprocessing. Optics letters, 43(4):631–634, 2018.
[29] B. K. Park, H. Park, Y.-S. Kim, J.-S. Kang, Y. Yeom, C. Ye, S. Moon, and S.-W. Han. Practical true random number generator using cmos image sensor dark noise. IEEE Access, 7:91407–91413, 2019.
[30] B. Perach et al. An asynchronous and low-power true random number generator using stt-mtj. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 27(11):2473–2484, 2019.
[31] F. Raffaelli, G. Ferranti, D. H. Mahler, P. Sibson, J. E. Kennard, A. Santamato, G. Sinclair, D. Bonneau, M. G. Thompson, and J. C. Matthews. A homodyne detector integrated onto a photonic chip for measuring quantum states and generating random numbers. Quantum Science and Technology, 3(2):025003, 2018.
[32] F. Schwierz. Graphene transistors. Nature nanotechnology, 5(7):487–496, 2010.
[33] S. Srinivasan, S. Mathew, R. Ramanaranayan, F. Sheikh, M. Anders, H. Kaul, V. Errangutla, R. Krishnamurthy, and G. Taylor. 2.4 ghz 7mw all-digital pvt-variation tolerant true random number generator in 45nm cmos. In 2010 Symposium on VLSI Circuits, pages 203–204. IEEE, 2010.
[34] D. B. Thomas. Acceleration of financial monte-carlo simulations using fpgas. In 2010 IEEE Workshop on High Performance Computational Finance, pages 1–6. IEEE, 2010.
[35] D. B. Thomas, L. Howes, and W. Luk. A comparison of cpus, gpus, fpgas, and massively parallel processor arrays for random number generation. In Proceedings of the ACM SIGDA international symposium on Field programmable gate arrays, pages 63–72, 2009.
[36] S. Thrun. Toward robotic cars. Communications of the ACM, 53(4):99–106, 2010.
[37] A. Tomasi, A. Meneghetti, N. Massari, L. Gasparini, D. Rucatti, and H. Xu. Model, validation, and characterization of a robust quantum random number generator based on photon arrival time comparison. Journal of Lightwave Technology, 36(18):3843–3854, 2018.
[38] D. Vivek, D. Heidi, and C. Mel. Wire bonding considerations. Available at: https://www.ece.ubc.ca/~robertor/Links_files/Files/MaxtekWireBondingArticle0706.pdf Accessed 17/04/2020.
[39] K. Wallace, K. Moran, E. Novak, G. Zhou, and K. Sun. Toward sensor-based random number generation for mobile and iot devices. IEEE Internet of Things Journal, 3(6):1189–1201, 2016.
[40] H. Wang, Y. Wu, C. Cong, J. Shang, and T. Yu. Hysteresis of electronic transport in graphene transistors. ACS Nano, 4(12):7221–7228, 2010. PMID: 21047068.
[41] H. Xu, N. Massari, L. Gasparini, A. Meneghetti, and A. Tomasi. A spad-based random number generator pixel based on the arrival time of photons. Integration, 64:22–28, 2019.
[42] X. Zhang, R. Bashizade, C. LaBoda, C. Dwyer, and A. R. Lebeck. Architecting a stochastic computing unit with molecular optical devices. In 2018 ACM/IEEE 45th Annual International Symposium on Computer Architecture (ISCA), pages 301–314. IEEE, 2018.