Accurate Prediction of Random Telegraph Noise Effects in SRAMs and DRAMs

Karthik .V. Aadithya*‡, Alper Demir†, Sriramkumar Venugopalan* and Jaijeet Roychowdhury*
*Department of Electrical Engineering and Computer Sciences, The University of California, Berkeley, CA, USA
†Department of Electrical and Electronics Engineering, Koç University, Istanbul, Turkey
‡Contact author. Email: aadithya@berkeley.edu

***Abstract*—With aggressive technology scaling and heightened variability, circuits such as SRAMs and DRAMs have become vulnerable to Random Telegraph Noise (RTN). The bias dependence (i.e., non-stationarity), random temporal nature, bi-directional coupling, and high inter-device variability of RTN present significant challenges to understanding its effect at the circuit level. In this paper, we propose two CAD tools, MUSTARD and SAMURAI, for accurately predicting the impact of non-stationary RTN on SRAMs/DRAMs in the presence of variability. While SAMURAI offers the advantage of easy interoperability with any SPICE-like circuit simulator (no modifications are required to the simulator), MUSTARD offers the capability to analyse bi-directionally coupled RTN using a built-in circuit simulator. Where analytical expressions are available (e.g., constant bias, unidirectionally-coupled RTN), our CAD tools closely match their predictions. And in practical situations where analytical expressions are typically not available (e.g., RTN in an SRAM/DRAM), our tools enable accurate RTN characterisation by generating realistic traces of non-stationary RTN under time-varying bias conditions. In particular, we use MUSTARD to duplicate experimentally observed RTN-induced bit errors in SRAMs, and variable retention times in DRAMs. We are also able to generate statistical characterisations of RTN-induced failures in both SRAMs and DRAMs, in the presence of variability. To our knowledge, no existing CAD tool for RTN analysis offers such capabilities. We also provide comparisons with a recently published technique [1] for RTN generation.**

I. INTRODUCTION

Random Telegraph Noise (RTN) has become an important challenge associated with the design of many different kinds of circuits in deep sub-micron technologies. Indeed, with aggressive CMOS scaling and increased parameter variability, RTN has now emerged as a critical limiting factor that produces transient failures in SRAMs, DRAMs, oscillators, PLLs and many RF circuits [2]–[6].

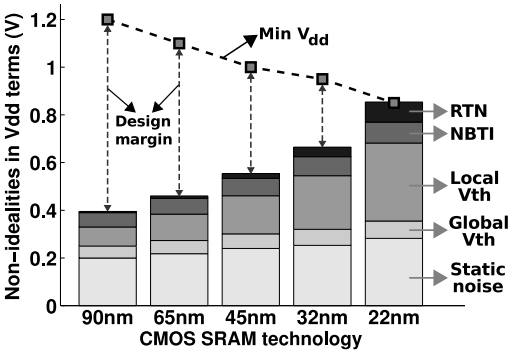

Fig. 1. Impact of RTN on SRAM design margins (data courtesy Yasumasa Tsukamoto, Renesas Electronics Corporation, Japan).

In SRAMs, for example, RTN has been shaving away design margins over the last few CMOS generations. This is readily seen from Fig. 1 [2], which quantifies, in supply voltage terms, the adverse effects of various non-idealities on SRAM design margins, as technology has progressed from 90nm to 22nm. In the figure, each CMOS technology is represented by a stacked bar, onto which the design margin impacts of different non-idealities (including static noise, local and global parameter variation, negative-bias temperature instability, and RTN) are successively added on top of one another. The figure also depicts how SRAM supply voltages have scaled with time (as shown by the downward sloping dashed line at the top). Thus, the vertical distance from the top of each stacked bar to the downward sloping dashed line quantifies the design margin available at each CMOS technology.

From the figure, it is seen that the impact of RTN on SRAM design margin has been steadily increasing under continued CMOS scaling. Indeed, of all the variability/noise sources accounted for in Fig. 1, RTN is the fastest growing factor contributing to design margin deterioration. Even more disturbing is the observation that, at the 22nm node, RTN (coming on top of parameter variability) is large enough to push the stacked bar above the minimum supply voltage, thereby driving design margins negative. In fact, RTN-induced SRAM *failures*, leading to transient read/write bit-errors, have already been experimentally observed and reported [2], [5]. Thus, from an SRAM design perspective, RTN has become a severe limiting factor that requires immediate attention.

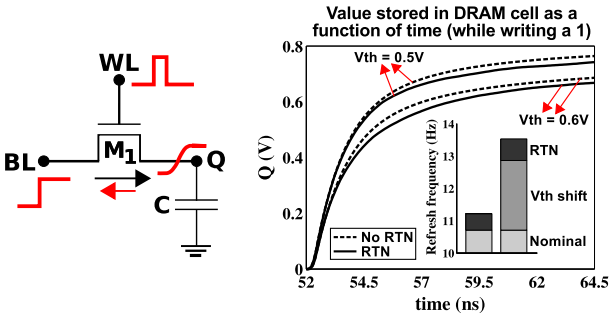

Fig. 2. Left: Writing the bit 1 to a 1T DRAM cell. The black arrow indicates the direction of drain current, while the red arrow indicates the direction of the noise current due to RTN. Right: Effect of RTN on the stored DRAM value for two different threshold voltages.

In addition to the above unfavourable impact on SRAMs, RTN is also thought to be responsible for variable retention times in DRAMs [7], [8]. Fig. 2 (left) shows a standard 1T DRAM cell to which the bit 1 is written. Fig. 2 (right) depicts how the stored value Q of the DRAM cell evolves as a function of time, both in the absence and in the presence of RTN, for two different threshold voltages. To compensate for RTN, it becomes necessary to refresh the DRAM cell more frequently, which in turn leads to increased power consumption and reduced speed of operation. The bar chart of Fig. 2 (right) quantifies the increase in refresh frequency necessitated by RTN, both in the presence and in the absence of threshold voltage shifts due to parameter variation.

Worse yet, the magnitude of RTN grows at the rate $1/WL$[1], which is much faster than many other sources of variability (e.g., Random Dopant Fluctuation grows only as $1/\sqrt{WL}$) [9]. In future, therefore, it is expected that RTN will affect more circuits adversely, and affect them more adversely. Thus, in order to ensure continued CMOS scaling, there is a need to develop new CAD tools/techniques that enable RTN-aware circuit design, while also simultaneously accounting for other sources of variability.

However, the very nature of RTN – from how it originates at the device level to how it impacts performance at the circuit level – poses several technical challenges for CAD tools:

Non-stationarity: At the device level, RTN is produced by a stochastic process involving the capture and release of charge carriers by dangling bonds (known as "traps") located in the MOS oxide layer [3], [4], [10]. The statistical parameters governing this stochastic process are strongly (and non-trivially) dependent on time-varying bias conditions (such as the transistor gate voltage). As a result, there is an inherent *non-stationarity* associated with RTN, i.e., the statistics of RTN are rapidly and continuously changing with time. Most existing techniques for noise analysis/characterisation work only when the underlying stochastic processes are stationary, i.e., their statistics remain constant with time. Therefore, existing noise analysis approaches do not apply to RTN; it is necessary to develop more powerful techniques.

High inter-device variability: Analytical approaches to RTN characterisation rely on the assumption that a large number of active traps exist in the dielectric. Under this assumption, statistical averaging (based on the law of large numbers) shows that RTN, in *every* device, obeys the *same* $1/f$ (or more generally, $1/f^{\alpha}$) characteristics [10], [11]. However, with increasingly small device sizes, this assumption is no longer valid [3], [12], [13]. Indeed, detailed models for trap profiling (corroborated by measured data) suggest that in deeply scaled CMOS technologies, only about 5-10 traps are active at any given bias point [3], [12], which results in significant inter-device variations.

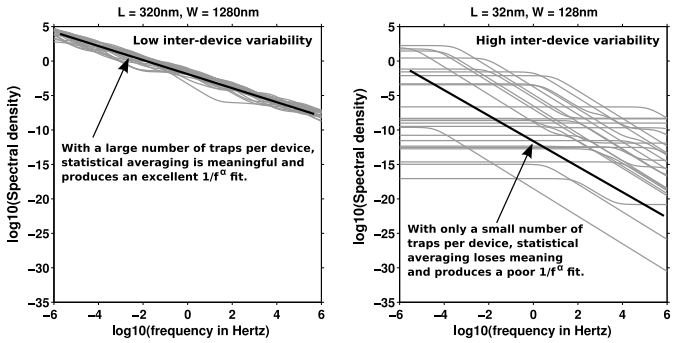

Fig. 3. Spectral density plots for 25 randomly sampled devices in two CMOS technologies.

Fig. 3 illustrates this point using spectral density plots for 25 device instances (randomly sampled using the trap profiling model of [3], and held at constant bias) from two CMOS technologies. While inter-device variation is negligible for the older technology (left), it is highly significant for devices in the newer technology (right). Thus, statistical averaging over trap profiles can no longer be used for characterising RTN in today's devices, which creates additional challenges for CAD tool development.

[1]Here, W and L denote the physical dimensions (width and length, respectively) of minimum-sized transistors in the latest generation CMOS technology.

Bi-directional coupling: The statistics of RTN exhibit a complicated bias dependence. Therefore, the probability that RTN will cause a given circuit to fail depends strongly on the time-varying bias patterns exhibited by the circuit. This leads to a bi-directionally coupled interaction between RTN and the rest of the circuit, i.e., the time-varying biasses in the circuit affect the statistical parameters of RTN, while RTN simultaneously produces changes in these very biasses. Thus, strictly speaking, RTN is not an effect that can be considered in isolation from the rest of the circuit. Rather, the circuit and its RTN evolve together as a coupled system – a feature that makes RTN characterisation especially difficult.

The challenges imposed by the above three innate characteristics of RTN, namely, *non-stationarity*, *high inter-device variability* and *bi-directional coupling*, have hampered the development of CAD tools for RTN characterisation. Indeed, even though detailed, device-level equations for RTN generation from first principles have been available for decades [3], [10], there is still no CAD tool that incorporates these models to achieve circuit level characterisation of non-stationary RTN. Instead, existing CAD techniques for RTN analysis tend to be limited in scope. For example, Tian and El Gamal [4] derive analytical expressions for RTN in the case of constant gate bias and switched gate bias. Roy and Enz [14] extend these results to the case of periodic gate bias. However, these techniques cannot be applied to circuits such as SRAMs and DRAMs, which are typically characterised by large, rapid and non-periodic swings in gate bias. For such applications, Ye et. al. [1] have recently proposed a simulation-based approach involving the generation of RTN traces by a 2-stage comparator circuit driven by white noise. However, this approach applies only to constant-bias, stationary RTN; it does not take into account the non-stationary statistics of RTN when the gate bias varies with time. Moreover, even for the stationary case, the RTN traces produced by this method often do not match the statistical characteristics predicted by analytical expressions (please see §VI for details). Furthermore, this method is time-consuming (as shown in §VI) because it requires SPICE-like simulation of white noise sources (one for each trap).

Against this background, in this paper:

[1] We present SAMURAI[2], a computational technique for accurate, trap-level, non-stationary analysis of RTN in circuits such as SRAMs/DRAMs. The core of SAMURAI is a technique called Markov uniformisation, which extends stochastic simulation ideas from the biological community and applies them to generate realistic traces of non-stationary RTN under arbitrarily time-varying bias conditions. To the best of our knowledge, SAMURAI is the first computational approach that employs detailed, trap-level stochastic RTN generation models to accurately analyse the impact of non-stationary RTN at the circuit level. Being a stochastic simulation technique, SAMURAI is also well suited to analyse RTN under high inter-device variability. Another key advantage of SAMURAI is that it can be implemented as a stand-alone, separate module outside of the SPICE-level circuit simulator. Thus, SAMURAI can easily be used in conjunction with any existing circuit simulator, without modifying the simulator. The only aspect of RTN that SAMURAI does not address is bi-directional coupling; for this, we have developed a new tool called MUSTARD.

[2]SAMURAI stands for SRAM Analysis via Markov Uniformisation with RTN-Awareness Incorporated. A preliminary version of SAMURAI [15] is scheduled to appear at the Design, Automation, and Test conference in Europe (DATE 2011). The present work elaborates on [15], providing more detailed explanations, additional experimental results and comparisons with a previously published method [1].

[2] We present MUSTARD[3], a second technique for predicting the impact of RTN on SRAMs and DRAMs in the presence of variability. In addition to the features already provided by SAMURAI, MUSTARD offers an extra capability; it can generate RTN that is bi-directionally coupled with the rest of the circuit. Thus, MUSTARD enables accurate, non-stationary, bi-directionally coupled, discrete stochastic RTN trace generation seamlessly integrated with deterministic, continuous circuit simulation. The main drawback of MUSTARD is that, unlike SAMURAI, it requires modifying the underlying circuit simulator. However, once the simulator is modified, MUSTARD provides greater accuracy than SAMURAI because it bi-directionally couples circuit simulation with RTN generation. Thus, MUSTARD and SAMURAI each have their own strengths, and together they overcome all the challenges associated with circuit level RTN analysis. Moreover, both MUSTARD and SAMURAI are *generic techniques*; they apply equally well to SRAMs, DRAMs, ring oscillators, etc. – even though RTN affects these circuits in markedly different ways.

[3] We validate SAMURAI against analytical expressions known for constant-bias, uni-directionally coupled RTN. We show that the statistical properties of RTN traces generated by SAMURAI are in excellent agreement with those predicted by analytical expressions. In effect, this also validates MUSTARD because, for the constant-bias uni-directionally coupled case, MUSTARD reduces to SAMURAI.

[4] We compare SAMURAI with the 2-stage RTN generation method recently proposed by Ye et. al. [1], from both an accuracy and an efficiency perspective. We demonstrate that SAMURAI is not only more accurate, but also significantly more efficient than 2-stage RTN generation.

[5] We apply MUSTARD to duplicate experimentally observed RTN-induced failures in SRAMs. Using MUSTARD, we also generate statistical characterisations of SRAM bit errors, in the presence of parameter variation.

[6] We present MUSTARD-generated results showing the effect of RTN on DRAM retention times, in the presence of parameter variation.

II. EXISTING METHODS FOR RTN CHARACTERISATION

Before describing SAMURAI and MUSTARD, we first provide an overview of existing RTN characterisation techniques.

Three different approaches to RTN characterisation may be distinguished from the literature: (a) analytical expressions for RTN at the device level, (b) measurement-based RTN characterisation at the circuit level, and (c) simulation-based RTN characterisation at both levels.

Analytical expressions: Much work has been done on the stochastic modelling of individual trap activity (as mentioned in §I, trap activity is the principal mechanism behind RTN generation) at the device level. Physics-based equations [3], [10] have been developed to describe how the statistical parameters governing individual trap activity depend on time-varying bias conditions. Detailed device models [14], [17] are also available for translating individual trap activity to RTN noise currents (by modulating the trap activity with the drain current). Several models [3] have also been proposed for obtaining realistic trap-profiles in today's deep sub-micron technologies. Therefore, at the device level, it is possible to construct realistic, non-stationary models for RTN from first principles.

However, at the circuit level, analytical expressions are available only for the simplest of cases, namely, the constant-bias (stationary) case [3], [10] and the periodic-bias (cyclo-stationary) case [4], [14] (neither of which applies to circuits such as SRAMs/DRAMs). Analytical expressions are not available for more general cases, chiefly because mathematical techniques are not known for precisely analysing the non-stationary random process that produces RTN.

Measurement-based characterisations: This is the primary mode of RTN analysis available today for circuits such as SRAMs and DRAMs. Measurement-based characterisations are based on subjecting post-fabrication SRAM/DRAM arrays to a large number of measurement tests in the hope of detecting any existing vulnerabilities to RTN [2]. For this purpose, accelerated RTN testing techniques have also been developed [5].

However, measurement-based characterisations suffer from several drawbacks. Firstly, measurements are made so late in the manufacturing process that it becomes very costly to correct any errors discovered at that stage. Secondly, while measurements can provide a series of success/failure outcomes, they *usually do not provide insight into why the failures occurred*. For instance, a measurement can only indicate that an SRAM cell has been compromised due to RTN; it cannot pinpoint which trap-level RTN events triggered the failure, or even how likely it is for such a failure to occur under normal operating conditions. Thirdly, unlike other sources of variability, RTN is a temporal effect that cannot be completely characterised by measurement data alone. For example, identical tests carried out on the same SRAM/DRAM chip at different time points can yield completely different success/failure outcomes [2]. Thus, even if a large number of measurements indicate that a chip is robust to RTN, it may in fact not be so.

Simulation-based characterisations: These are very recent developments in RTN analysis. The main idea is that: because the number of active traps in today's devices is quite small (in the ballpark of 5-10 for all realistic bias conditions), the activity of each individual trap can be stochastically simulated (for example, alongside a SPICE run). Whenever such a simulation predicts an RTN-induced failure, it is possible to examine the simulation trace to investigate the RTN events that triggered the failure. This may even yield design intuition on how to build circuits that are more robust to RTN. Moreover, because such a simulation-driven design methodology accounts for RTN right in the design phase, it greatly reduces the risk of unpleasant findings during the (much later) measurement/testing phase.

The state-of-the-art simulation-driven strategy (excluding SAMURAI and MUSTARD) is the one proposed by Ye et. al. [1], which uses a 2-stage comparator circuit for generating stationary RTN under constant bias conditions, starting from an independent white noise source for every trap in the circuit. This method has been applied to (stationary) analysis of RTN due to a single trap in an SRAM cell [1]. However, this method cannot truly reproduce the non-stationary RTN patterns produced under the large and rapid bias variations that a switching SRAM cell typically undergoes. Indeed, we show in §VI that this method often does not generate even stationary RTN accurately. So we believe that the predictive utility of this method for determining the robustness of an SRAM cell to non-stationary RTN is not clear.

[3]MUSTARD stands for Markov Uniformisation based Simulation of Trap Activity for RTN-Aware Design. We have submitted a preliminary version of MUSTARD [16] to the Design Automation Conference (DAC 2011), where it has been accepted for publication. The present work includes more detailed explanations, and comparisons with a previously proposed RTN generation technique ([1]).

III. UNI-DIRECTIONALLY AND BI-DIRECTIONALLY COUPLED MODELS FOR RTN

We now describe the RTN models used by SAMURAI and MUSTARD. SAMURAI uses a non-stationary RTN model that is uni-directionally coupled with the rest of the circuit; on the other hand, MUSTARD's RTN model takes into account bi-directional coupling (in addition to all the features supported by SAMURAI).

As mentioned in §I, RTN is produced by the random capture and release of electrons by dangling bonds located in a MOS transistor's oxide layer [10]. As depicted in Fig. 4 (left), the oxide layer (especially at the oxide-semiconductor interface) contains silicon (Si) atoms with unsatisfied valences, called dangling bonds or "traps". While some traps are passivated by hydrogen, others are not. And when the transistor is on, each un-passivated trap has a propensity to randomly (a) capture an electron from the inversion layer, and (b) release the captured electron back into the inversion layer [10].

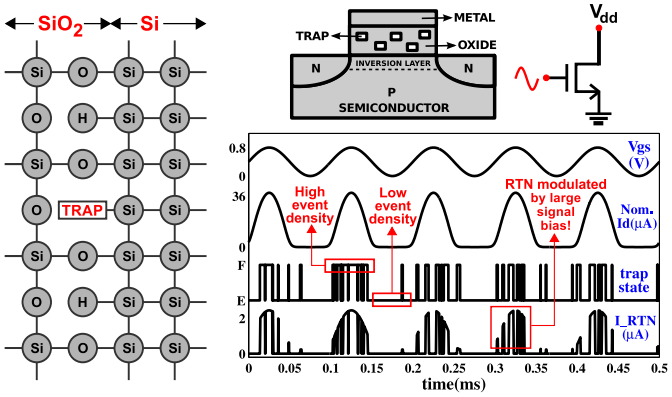

Fig. 4. Left: Dangling bonds at the Si/SiO_2 interface. Right: Trap activity under sinusoidal gate bias, and its modulation by large signal drain current.

Thus, at any given time, each un-passivated trap can be in one of two possible states: either *filled* (with an electron) or *empty*. An *empty* trap can become *filled* by capturing an electron, while a *filled* trap can become *empty* by releasing its captured electron.

Also, whenever a trap becomes *filled*, its captured electron modifies (a) the electric field inside the transistor, (b) the mobility of electrons in the inversion layer, and (c) the number density of charge carriers contributing to the transistor current [18]. As a result, every electron capture/release event by a trap brings about a change in the transistor current, which is observed as a random waveform $I_{RTN}(t)$ that opposes the nominal drain current $I_d(t)$.

Furthermore, the propensity of a trap to capture/release an electron is not constant, but depends on the instantaneous gate bias $V_{gs}(t)$ [3], [4], [14]. Mathematically speaking, given that a trap *tr* is *empty* at time t, the probability it changes state to become *filled*, within a short time interval dt, is given by $\lambda_{c,tr}(t)dt$, where $\lambda_{c,tr}(t)$ is called the *capture propensity* of the trap *tr* at time *t*. Similarly, given that a trap *tr* is *filled* at time t, the probability it changes state to become *empty*, within a short time interval dt, is given by $\lambda_{e,tr}(t)dt$, where $\lambda_{e,tr}(t)$ is called the *emission propensity* of the trap *tr* at time *t*.

For any given trap, the capture and the emission propensities (i.e., λ_c and λ_e) are non-trivial functions of the time-varying gate bias $V_{gs}(t)$. The exact form of these functions is specified in [3], from which we obtain:

$$\lambda_{c,tr}(t)+\lambda_{e,tr}(t)=\frac{1}{\tau_0 e^{\gamma y_{tr}}} \tag{1}$$

$$\text{and } \beta_{tr}(t)=\lambda_{e,tr}(t)/\lambda_{c,tr}(t)=g\ e^{\frac{(E_T-E_F)_{tr}}{kT}} \tag{2}$$

$$[\text{where } (E_T-E_F)|_{tr,t}=\text{function}(E_{tr},y_{tr},V_{gs}|_t,\text{device parms})]$$

From the above equations, it is seen that for a given trap, the *sum* $\lambda_c+\lambda_e$ is constant with time, depending only on the position of the trap y_{tr}, the time constant τ_0 for traps at the silicon interface, and the tunnelling coefficient γ. However, the *ratio* $\beta=\lambda_e/\lambda_c$ at time t is a complicated function (whose exact form is given in [3]) of the instantaneous gate bias $V_{gs}|_t$. It is this bias dependence that causes the electron capture/release process to be non-stationary [4]. Thus, for a given trap in a given transistor in a given circuit such as an SRAM/DRAM, the statistical parameters of the capture/release process are continuously changing, depending on how the gate bias of the transistor is evolving with time. (Here g is the trap degeneracy factor, k is the Boltzmann constant and T is the temperature.)

Finally, the formula that relates the states of individual traps (i.e., *filled* or *empty*) to the collective noise current $I_{RTN}(t)$, exhibits a complicated dependence on the biasses $I_d(t)$ and $V_{gs}(t)$. For instance, one model that has been proposed for NMOS RTN is the following equation [17]:

$$I_{RTN}(t)=\frac{N_{filled}(t)\ q}{WL\ C_{ox}(V_{gs}(t)-V_{th})}\ I_d(t), \tag{3}$$

where N_{filled} denotes the number of *filled* traps in the transistor, q is the electronic charge ($\sim 1.6\times10^{-19}$ Coulomb), W and L are the transistor dimensions, C_{ox} is the oxide capacitance per unit area and V_{th} is the transistor threshold voltage. More elaborate models have also been suggested in the literature [19].

Therefore, the net effect $I_{RTN}(t)$ is that of a non-stationary electron capture/release process, whose resulting *trap occupancy function* $N_{filled}(t)$ is modulated by a bias-dependent large signal waveform.

Fig. 4 (right) illustrates the above for a single trap in an NMOS transistor whose gate is driven by a sinusoidal voltage source. The figure depicts the nominal biasses $V_{gs}(t)$ and $I_d(t)$, in addition to the trap occupancy function (which moves between the states *empty* (E) and *filled* (F)) and the noise current $I_{RTN}(t)$. The plot brings out an important feature of RTN: because the gate bias is time-varying, the capture and emission propensities are not constant. As a result, the density of capture/release events is non-uniform; hence it is common to observe some time intervals with a low event density and others with a high event density, which provides a visual cue that the underlying capture/release process is non-stationary.

To develop a CAD tool that enables accurate, circuit level, non-stationary RTN analysis, it is necessary to formulate a mathematical abstraction that captures all the concepts described above. In the case of SAMURAI and MUSTARD, this mathematical abstraction is a time-inhomogeneous Markov chain with a hypercube state transition graph, which, as we explain below, is capable of incorporating all the above aspects of RTN without loss of generality.

Consider a circuit with N active traps, with each trap contributing to the I_{RTN} of a specific transistor (multiple traps may belong to the same transistor). Each trap has two possible states; so the system of N traps has 2^N possible states, which can be encoded using N bits (with one bit per trap, where 0 denotes *empty* and 1 denotes *filled*). Every time the k^{th} trap changes state, the k^{th} bit of the N-bit system state is changed to reflect the RTN event. Mathematically, this corresponds to a *state transition graph* that is an N-dimensional hypercube (as shown in Fig. 5 for $N=1,\ 2$ and 3).

Each dimension in the hypercube corresponds to a unique trap. Thus,

all hypercube edges along a given dimension represent capture/release events involving a unique trap. Therefore, for every trap *tr*, all edges along the *tr*-dimension are annotated with the propensities $\lambda_{c,tr}(t)$ and $\lambda_{e,tr}(t)$ (Fig. 5), which results in a time-inhomogeneous Markov chain.

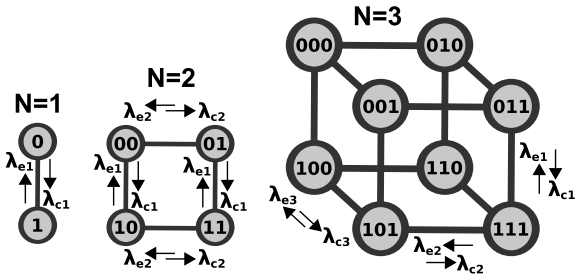

Fig. 5. Hypercube Markov state transition graphs that model the RTN produced by 1, 2, and 3 trap systems. For the last case, although the figure does not explicitly show it, all parallel edges are assigned identical propensities.

In the context of a circuit, however, the N-trap system is not isolated; rather, its (discrete) state evolves simultaneously with the underlying circuit, which has its own (continuous) state vector $\vec{x}$ of voltages and currents. These voltages and currents follow a set of circuit equations (based on Kirchoff's laws and branch constitutive relationships). Without loss of generality, these equations can be cast in the form of a Differential Algebraic Equation (DAE) system D [20], given by:

$$D: \quad \frac{d}{dt}\vec{q}(\vec{x}) + \vec{f}(\vec{x}) + \vec{b}(t) = \vec{0}$$

Putting it all together, RTN is produced by an N-dimensional hypercube Markov chain, whose time-varying propensities are determined by a state vector $\vec{x}$, which itself evolves according to a DAE system D, whose $\vec{q}$ and $\vec{f}$ functions are in turn determined by the Markov chain's state. This bi-directionally coupled RTN model is summarised by Fig. 6.

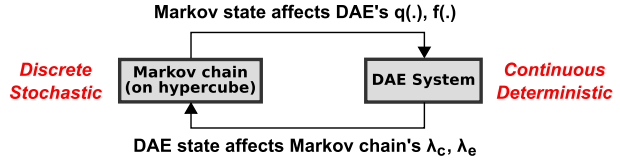

Fig. 6. The bi-directionally coupled model for RTN: A non-stationary Markov chain that drives, and is in turn driven by, a DAE system.

Therefore, in order to truly generate non-stationary RTN at the circuit level, one needs to develop the capability to stochastically simulate the above bi-directionally coupled Markov/DAE system. In fact, an exact algorithm is known for carrying out such simulations; this algorithm, which is based on a technique called Markov uniformisation, forms the core of MUSTARD (described in §VII).

We note that a commercial, SPICE-like circuit simulator does not have the capability to simulate Markov/DAE systems. Rather, it can only simulate stand-alone DAE systems (i.e., circuit equations). To use MUSTARD, therefore, it is necessary to either modify one's existing circuit simulator, (or) switch to the circuit simulator that we have already built into MUSTARD. Sometimes, however, neither of these choices may be acceptable to a designer, who may require a separate RTN analysis module that is completely independent of the circuit simulator. To address such a requirement, we have developed SAMURAI.

SAMURAI works by approximating the above bi-directionally coupled Markov/DAE system, by a uni-directionally coupled Markov/DAE system. In SAMURAI, the time-varying effect of the DAE state on the Markov propensities (i.e., non-stationarity) is fully taken into account; however, the DAE itself is not changed as and when individual RTN events occur. Instead, a new DAE is constructed at the end of the RTN simulation; this new DAE is augmented with I_{RTN} current sources between the source and drain of each transistor, which are in turn obtained by stochastic simulation of the non-stationary Markov chain whose propensities are obtained from the trajectory of the original DAE. Thus, during a SAMURAI run, the effects of each RTN event are recorded and SPICE-simulated only at the end of the run, whereas in reality, these effects are felt by the circuit as soon as the RTN events occur. Henceforth, we call this approximation the uni-directionally coupled RTN model.

IV. SAMURAI: A CAD TOOL FOR UNI-DIRECTIONALLY COUPLED RTN SIMULATION BY MARKOV UNIFORMISATION

Having described the non-stationary, uni-directionally coupled RTN model, we now discuss how to simulate this model using Markov uniformisation, which is the core technique behind SAMURAI.

Algorithm 1: Non-stationary RTN generation in SAMURAI

Input: Trap profile, Bias $\{V_{gs}(t),\ I_d(t)\ \ldots\}$, t_0, t_f
Output: Realistic $I_{RTN}(t)$ trace in time interval $[t_0, t_f]$

```
foreach trap tr in the device do
    compute λc(t), λe(t), t ∈ [t0, tf], for tr (use Eq. (1), (2));
    λ* = λc(t0) + λe(t0);
    curr_time = t0; curr_state = tr.init_state;
    times = [curr_time]; states = [curr_state];
    while curr_time < tf do
        next_cand_time = curr_time + exprand(1/λ*);
        curr_time = next_cand_time;
        if curr_time > tf then break;
        if curr_state == 1 then
            λnext = λe(curr_time)
        else
            λnext = λc(curr_time)
        end
        bool change_the_state = rand() < λnext/λ*;
        if change_the_state then
            times.append(curr_time);
            states.append(curr_state);
            curr_state = (curr_state == 1) ? 0 : 1;
            times.append(curr_time);
            states.append(curr_state);
        end
    end
    trap_occupancy[tr] = [times, states];
end
compute I_RTN(t) from trap_occupancy[tr] (use, e.g., Eq. (3))
```

Specifically, given a circuit netlist (e.g., an SRAM/DRAM), and a time interval $[t_0, t_f]$ during which the voltages and currents in the circuit can evolve continuously, the goal is to construct a realistic, non-stationary RTN trace (taking into account the continually changing capture/emission propensities of each trap) for each transistor in the circuit, over the given time interval. For a single transistor, this goal is accomplished by Algorithm 1 (which is run once for each transistor in the circuit).

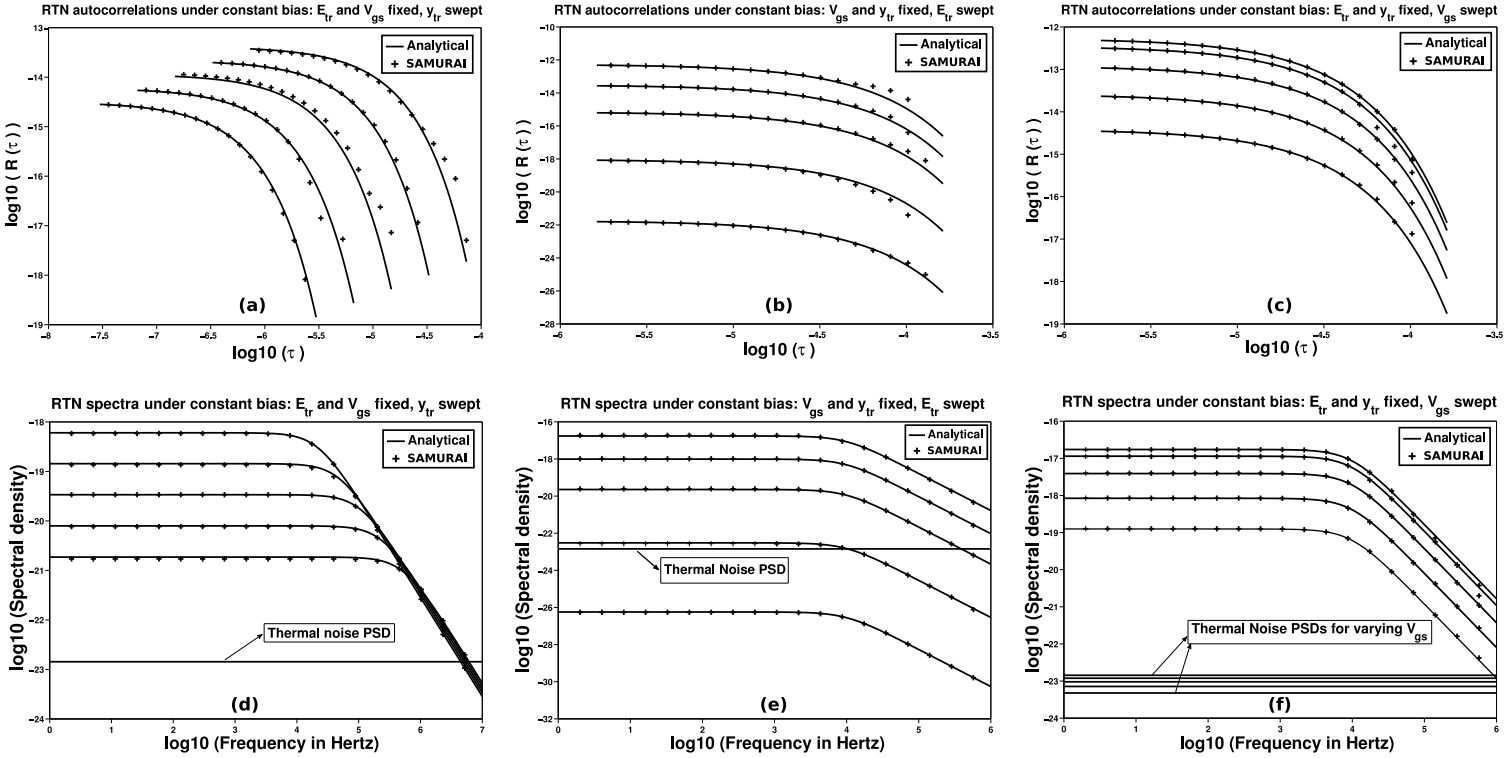

Fig. 7. Plots showing that the RTN traces generated by SAMURAI closely match analytical predictions in both the time domain (autocorrelation plots (a)-(c)) and the frequency domain (spectral density plots (d)-(f)).

Algorithm 1 takes as input, (a) the *trap profile* of the device (i.e., the position y_{tr} and energy E_{tr} of each trap), and (b) the time-varying bias conditions (such as the gate bias $V_{gs}(t)$). The former is obtained either from measured data, or by using technology-specific trap profile models proposed in the literature (e.g., [3]). The latter is obtained by SPICE simulation of the circuit DAE over time interval $[t_0, t_f]$. The output produced by Algorithm 1 is an $I_{RTN}(t)$ trace for the device, whose statistics are guaranteed to be *exactly identical* to the time-varying RTN statistics under the uni-directionally coupled, non-stationary model of §III.

Briefly, the algorithm works by generating *more* trap activity than necessary, and then *discarding* some of the generated activity so that the time-varying trap statistics are exactly preserved. Line 3 computes λ^*, which is an upper bound on the functions $\lambda_{c,tr}(t)$ and $\lambda_{e,tr}(t)$. In each iteration of the while loop (line 6), a *candidate capture/release event* is generated (line 7) corresponding to a *stationary* two-state Markov chain (with a 2D hypercube state transition graph, similar to the left most Markov chain in Fig. 5) with both propensities set to λ^*. Thus, the original non-stationary Markov chain is first *uniformised* into a stationary (but high rate) Markov chain. In line 15, a probabilistic decision is made to either *keep* or *discard* the generated RTN event, which exactly restores the non-stationarity of the original Markov chain. That this algorithm exactly preserves the original Markov chain's non-stationarity is proved in [21]–[23].

For analysing circuits with multiple transistors (e.g., SRAMs), we apply Algorithm 1 individually to each transistor, thus obtaining one noise current source $I_{RTN}(t)$ per transistor. A new circuit is now constructed, by connecting these noise current sources between the source and drain of the corresponding transistors. The new circuit, which now incorporates RTN, can now be analysed using SPICE-level simulation. Thus, SAMURAI's non-stationary RTN analysis involves two separate circuit simulations, (a) simulation of the original circuit to obtain the time-varying propensities required by Algorithm 1, and (b) simulation of a new circuit derived by augmenting the original circuit with RTN sources produced by Algorithm 1. Each of these circuit simulations can be performed by any SPICE-like circuit simulator, without any modification to the simulator. Our paper [15] illustrates all the above steps using a 90nm SRAM cell and SpiceOpus [24] as the simulator.

V. SAMURAI: VALIDATION AGAINST ANALYTICAL EXPRESSIONS

As mentioned before, SAMURAI is capable of non-stationary RTN trace generation under arbitrarily time-varying bias conditions. Although analytical expressions are not available for such a general case, they are known for the restricted case of constant gate bias [4], [10]. Here we validate SAMURAI against these expressions for a wide range of trap configurations, as follows:

- We run three validation experiments, using typical values for the 3 parameters that affect the trap capture/release statistics, namely, V_{gs}, E_{tr}, and y_{tr}. In each experiment, we fix two of these parameters and sweep the third over an appropriate range. Hence we generate a variety of trap configurations, which are then simulated under constant gate bias using Algorithm 1.
- Algorithm 1 returns a trace $I_{RTN}(t)$. From this trace, we numerically estimate the autocorrelation function $R(\tau) = E[I_{RTN}(t) I_{RTN}(t+\tau)]$.
- We also translate the above time-domain results into the frequency domain, by computing the stationary power spectral density (PSD) $S(f)$ numerically from $R(\tau)$.
- We plot the above waveforms $R(\tau)$ and $S(f)$ alongside analytical expressions obtained from [4], [10]. To get an idea of the

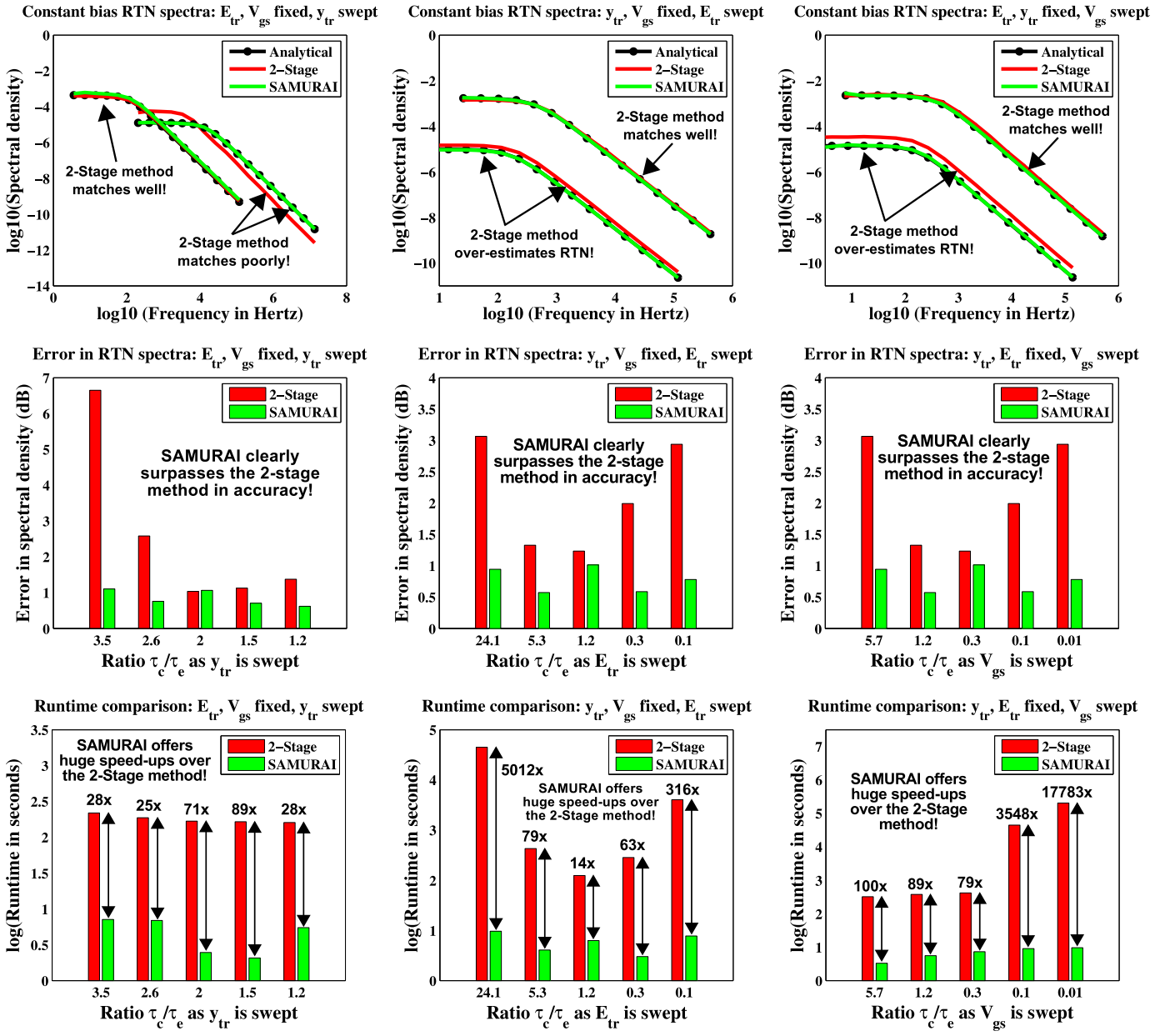

Fig. 8. Comparison of SAMURAI with 2-stage RTN generation. The top three plots demonstrate that even for the single-trap, constant-bias, stationary RTN case, the 2-stage method does not exactly match analytical expressions. On the other hand, SAMURAI demonstrates excellent agreement with analytical expressions over a wide range of trap configurations. The middle three plots further drive home this point; they show that the 2-stage method can mis-predict RTN by several dB, whereas SAMURAI is always within 1dB of the true RTN spectral density. The bottom three plots show that SAMURAI is always much faster than 2-stage RTN generation. (Note that the run-times are plotted on a logarithmic scale).

relative importance of RTN, we also plot the stationary power spectral density of thermal noise in the device using the model $S_{thermal}(f) = \frac{8}{3}kTg_m$ (where k is the Boltzmann constant, T is the temperature and g_m is the device transconductance at the applied bias).

The results are presented in Fig. 7 (a to f). In all these plots, τ is measured in seconds, $R(\tau)$ in A^2, all frequencies are in Hz and all spectral densities are in A^2/Hz.

From Fig. 7, it is seen that the RTN traces predicted by SAMURAI closely match analytical expressions in both the time domain (auto-correlation plots (a)-(c)) and the frequency domain (spectral density plots (d)-(f)).

VI. SAMURAI: COMPARISON WITH 2-STAGE RTN GENERATION

In this section, we compare SAMURAI against a recently proposed simulation-based approach [1] for RTN trace generation. This approach, suggested by Ye et. al., is based on the observation that white noise, after being filtered by an RC circuit, exhibits a power spectral density similar to that of stationary RTN. After empirical refinements, Ye. et. al proposed the circuit structure of Fig. 9 to generate approximate, RTN-like traces for a single trap operating under constant gate bias.

As seen from Fig. 9, each trap in the circuit is simulated by a 2-stage "RC filter followed by comparator" structure driven by a

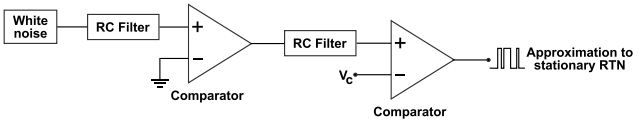

Fig. 9. Schematic proposed by Ye et. al. [1] for producing traces that approximate stationary (constant bias) RTN.

white noise source (the white noise sources of different traps are statistically independent of each other). For a given trap, the output of the 2-stage circuit is a two-level signal that is taken to be the trap occupancy function. The parameters of the circuit, i.e., the R and C values of the filter and the threshold V_c of the second comparator, are chosen depending on the stationary trap activity parameters τ_c (which is $1/\lambda_c$) and τ_e (which is $1/\lambda_e$). Expressions for choosing these parameters are given in [1].

For comparison with SAMURAI, we have implemented the above method, and performed simulations on a variety of trap configurations, as follows:

- As with the validation experiments, we first generate several trap configurations by sampling different values for the 3 parameters V_{gs}, E_{tr}, and y_{tr}. As before, in each experiment, we fix two of these parameters and sweep the third over an appropriate range. For each trap so generated, we compute τ_c and τ_e (the reciprocals of λ_c and λ_e) using equations (1) and (2) of §III.
- [At this point, we start the timer.] For each trap, we compute the parameters of the 2-stage circuit (i.e., R, C and V_c) according to the equations in [1]. We then carry out transient simulation (over the time interval $[0, 1000(\tau_c+\tau_e)]$, in time steps of $0.1(\tau_c+\tau_e)$) of the 2-stage circuit on a SPICE-level circuit simulator; for this, we have used the tool SpiceOpus [24]. [At the end of the simulation, we stop the timer and denote the elapsed time by $t_{2-stage}$.]
- We also simulate each trap under constant bias, for the same time interval $[0, 1000(\tau_c+\tau_e)]$, using SAMURAI. We also time SAMURAI, letting $t_{SAMURAI}$ denote the elapsed time in this case.
- Both the 2-stage method and SAMURAI return trap occupancy functions, which we convert into RTN traces $I_{RTN}(t)$ using Eq. (3). For both methods, we numerically estimate the autocorrelation function $R(\tau) = E[I_{RTN}(t)I_{RTN}(t+\tau)]$, and then translate it into frequency domain by computing the stationary PSD $S(f)$.
- We plot the power spectral densities produced by both methods alongside the analytical expressions known for constant-bias, stationary RTN (the top 3 plots of Fig. 8). For each method, we also plot the average error in the PSD, $|S_{predicted} - S_{analytical}|$, converted to dB scale (the middle 3 plots of Fig. 8). Finally, we plot the wall-clock run-times $t_{2-stage}$ and $t_{SAMURAI}$ for each trap, and calculate the ratio of these as the relative speed-up offered by SAMURAI (the bottom 3 plots of Fig. 8).

While carrying out the above simulations, we observed that for all trap configurations, SAMURAI closely matched the analytical expression. However, this did not hold for the 2-stage method, which often deviated significantly from the analytical spectrum. Therefore, in each of the top 3 plots of Fig. 8, we have presented one value of the swept parameter for which the 2-stage method matches the analytical PSD well, and one value where it does not match so well. (Note: In the top three plots, all power spectral densities are in A^2/Hz.)

The middle 3 plots of Fig. 8 compare the average error in the PSDs produced by both methods. It is seen that SAMURAI clearly surpasses the 2-stage method from an accuracy perspective. Also, the 2-stage method can sometimes mis-predict even stationary RTN by a wide margin (almost 7dB), whereas the error in SAMURAI is always contained within about 1dB (the theory of uniformisation guarantees that SAMURAI is *stochastically exact*; however, a small error is expected to arise because (a) numerical truncations/approximations are used both in the simulation and in the autocorrelation/PSD estimation, and (b) the time interval of simulation/integration is not infinite, but of finite length $1000(\tau_c+\tau_e)$).

The bottom 3 plots of Fig. 8 highlight the speed-ups that SAMURAI offers over the 2-stage method. For every trap configuration that we tested, SAMURAI was many times faster than the 2-stage method; indeed, a logarithmic scale is needed for convenient visual depiction. In the plots, the speed-ups offered by SAMURAI are indicated alongside the log(run-time) bars for each trap. As the plots show, SAMURAI can sometimes be 4 orders of magnitude faster than 2-stage RTN generation.

Algorithm 2: Circuit simulation with non-stationary RTN in MUSTARD

Input: Circuit DAE D, initial circuit and trap states at time t_0, final time t_f
Output: Circuit simulation trace with realistic, non-stationary RTN in time $[t_0, t_f]$

```
// uniformise the RTN Markov chain to a high-rate λ*
λ* = 0;
foreach trap tr in the circuit do λ* += λ_c,tr(t0) + λ_e,tr(t0);
t_curr = t0; x_curr = x0; tr_curr = tr0;
while t_curr < tf do
    // generate a candidate time for the next RTN event
    t_next_RTN = t_curr + exprand(1/λ*);
    // simulate the circuit's DAE D until t_next_RTN
    while t_curr < min(tf, t_next_RTN) do
        record(t_curr, x_curr, tr_curr);
        t_next = min(t_curr + t_step, t_next_RTN, tf);
        x_next = LMSSolve(D, t_curr, x_curr, t_next);
        // LMSSolve can be any standard DAE solution method
        // e.g., Forward Euler, Backward Euler, Trapezoidal
        t_curr = t_next; x_curr = x_next;
    end
    // time to make a probabilistic decision
    // to either keep or discard the candidate RTN event
    if t_curr < tf then
        u = rand(); // u is uniformly distributed in [0,1]
        sum = 0;
        foreach trap tr in the circuit do
            // compute propensity of trap tr to change state
            // detailed stochastic models available for this
            // by default, MUSTARD uses [3]
            λ[tr] = tr_curr[tr] ? λ_e,tr(x_curr) : λ_c,tr(x_curr);
            if u ≥ sum/λ* AND u < (sum+λ[tr])/λ* then
                // Keep the RTN event: trap tr changes state!
                tr_curr[tr] = !(tr_curr[tr]);
                // update circuit's DAE to reflect RTN event
                // many literature models available for this
                // by default, MUSTARD uses [18]
                x_curr = new_ckt_state_after_RTN_event_at_tr;
                D = new_ckt_DAE_after_RTN_event_at_tr;
                break;
            end
            sum += λ[tr];
        end
    end
end
```

VII. MUSTARD: A CAD TOOL FOR BI-DIRECTIONALLY COUPLED RTN SIMULATION

So far, we have presented and validated an algorithm for uni-directionally coupled RTN simulation. We now extend this algorithm for bi-directionally coupled RTN trace generation. The extended algorithm forms the core of MUSTARD.

A. Simulation of bi-directionally coupled Markov/DAE systems by uniformisation

Algorithm 2 describes MUSTARD's strategy for generating genuinely non-stationary RTN at the circuit level. Similar to SAMURAI, the algorithm works by first uniformising the time-inhomogeneous RTN hypercube Markov chain into a high-rate but time-homogeneous Markov chain, whose events are generated using the well-known Gillespie's algorithm [25]. Again like SAMURAI, later in the simulation, a probabilistic decision is made to discard some of the generated RTN events, which restores the non-stationary statistics expected of the original RTN hypercube. Reasoning in a way similar to SAMURAI, we conclude that no statistical test will be able to distinguish between the RTN traces produced by Algorithm 2 and the RTN traces measured on a fabricated circuit such as an SRAM or DRAM.

The crucial difference between MUSTARD and SAMURAI is that: MUSTARD ensures that the effects of individual RTN events on the circuit's DAE are incorporated in a coupled manner, i.e., the DAE is updated as soon as the RTN events occur (corresponding to lines 19 and 20 in Algorithm 2). This enables MUSTARD to overcome the main limitation of SAMURAI. Moreover, MUSTARD has the same run-time as SAMURAI, so the improved accuracy of MUSTARD over SAMURAI, is achieved at no extra computational cost.

B. MUSTARD's software architecture

We intend to soon release MUSTARD as an open-source CAD tool for RTN analysis. To encourage early adoption, we have tried to implement a modular, easily extensible, easily maintainable architecture for MUSTARD.

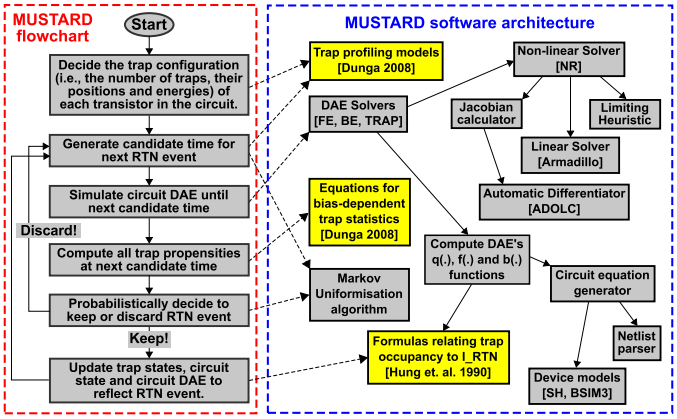

Fig. 10. The flowchart (left) describes the functionality required for Algorithm 2, while the dependency graph (right) illustrates how the functionality is implemented in MUSTARD. An arrow from u to v indicates that the module for u depends on the module for v. Except ADOLC [26] and Armadillo [27], all other modules have been programmed by us in C++.

Fig. 10 illustrates the software architecture underlying MUSTARD. As the figure shows, we have implemented much of the circuit simulation functionality from the bottom up, in order to integrate RTN simulation more efficiently with circuit simulation. Also, our design for MUSTARD is modular, with the RTN-related modules (highlighted in yellow) maintained separately from the rest of the simulator. This makes it easy to experiment with RTN under various trap configurations, statistical parameters, equations for modulating trap occupancies, etc. Fig. 10 also provides references for the default RTN models currently used by MUSTARD.

VIII. APPLICATIONS TO SRAM DESIGN

We have applied the techniques discussed above to conduct coupled, non-stationary analysis of RTN in 22nm SRAMs. Our simulations are able to accurately reproduce experimentally observed effects, such as RTN-induced SRAM write failures. Furthermore, in order to characterise the impact of RTN on entire SRAM arrays in the presence of local and global parameter variations, we have simulated a large number of SRAM cells with varying trap configurations, threshold voltages and supply voltages. We now present these results.

A. Prediction of RTN-induced SRAM write failures

RTN-induced write failures have been experimentally observed in deep sub-micron SRAMs [5]. To reproduce these findings, we designed a 22nm 6T SRAM cell (using the BSIM3 model, with parameters obtained from [28]) and studied its non-stationary RTN patterns using Markov uniformisation.

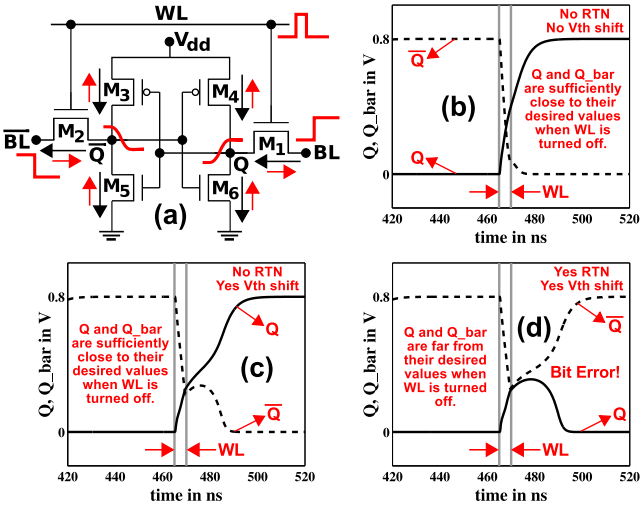

Fig. 11. Part (a): Writing the bit 1 to a 6T SRAM cell. Black (Red) arrows indicate the direction of I_d (I_{RTN}) in each transistor. Parts (b)-(d): RTN, coming on top of a 100mV V_{th} shift due to parameter variations, can produce an SRAM write error.

Fig. 11 illustrates an RTN-induced write failure scenario predicted by MUSTARD, for the above SRAM cell. Part (a) shows the biasses under which this failure occurs. The expected directions of I_d (black arrows) and I_{RTN} (red arrows) are also indicated next to each transistor.

If there is no RTN and no V_{th} shift, we see from part (b) that the node Q properly settles to logical 1 (or V_{dd}).

We now introduce a 100mV shift (to model local and global parameter variations) to the V_{th} of pass transistors M1 and M2. Even with this shift, the SRAM cell, in the absence of RTN, is able to latch on to the correct value of Q by the end of the clock cycle (part (c)).

Now we bring in a small amount of RTN, by injecting one trap each into transistors M1, M2, M4 and M5. With the introduction of RTN, we see that the SRAM cell is no longer able to respond by the end of the clock cycle (part (d)). This is because RTN currents in transistors M1 and M4 oppose the nominal currents driving Q to 1, while RTN currents in M2 and M5 oppose the nominal currents driving $\overline{Q}$ to 0. As a result, the signals Q and $\overline{Q}$ are delayed sufficiently to cause a bit error (part (d)).

Thus, MUSTARD can reproduce results previously obtainable only by measurement. In addition, MUSTARD offers significant "debuggabil-

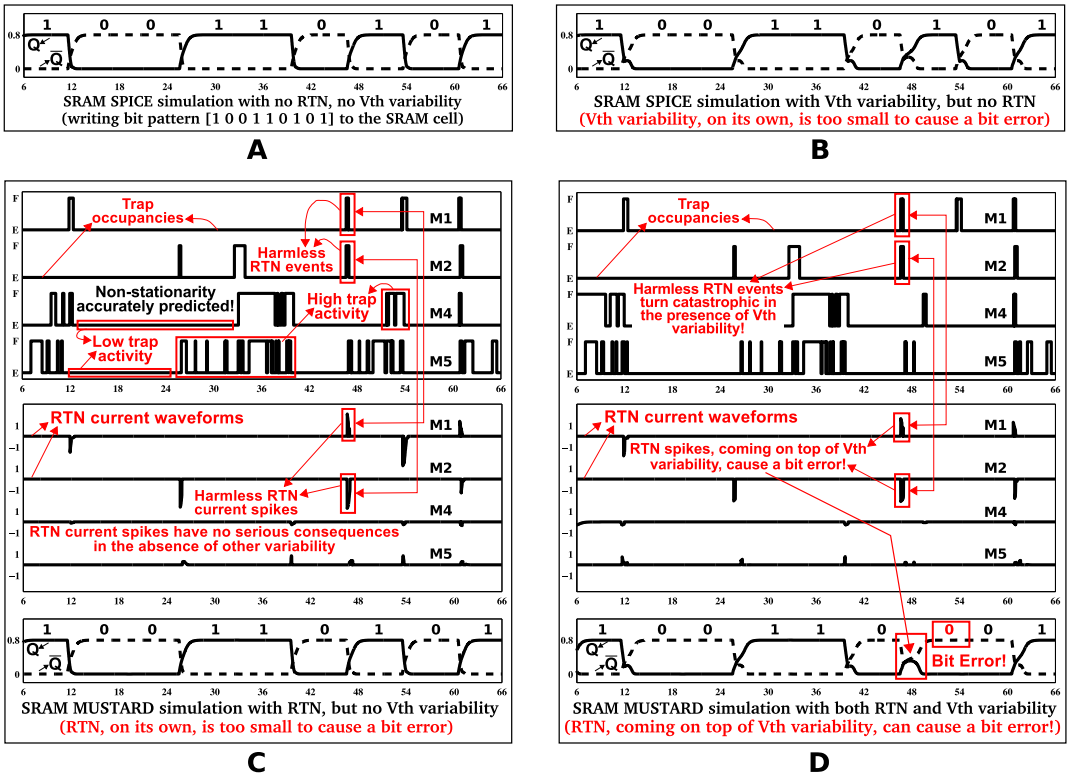

Fig. 12. Examination of MUSTARD's simulation trace (including trap occupancies and RTN currents of each transistor) during the clock cycles prior to the write failure of Fig. 11. Part (D) pinpoints the RTN events responsible for the bit error. Part (C) shows that similar events are harmless in the absence of V_{th} shifts. In all plots, the x-axis denotes time in ns. For the $Q/\overline{Q}$ plots, the y-axis denotes voltage (in volts). For trap occupancy plots, the y-axis is discrete, with E meaning *empty* and F meaning *filled*. In the I_{RTN} plots, the y-axis denotes current in μA.

ity" advantages over pure measurement. For example, now that a bit error has been discovered, the entire simulation trace of RTN events leading up to the bit error can be examined (Fig. 12). From this, it is possible to precisely pinpoint which RTN events triggered the failure (Fig. 12(D)). By contrast, pure measurement can only indicate that the SRAM cell has been compromised due to RTN; it cannot provide further insight into which RTN events were responsible for the failure, how likely such events are under normal operating conditions, etc.

Parts (A) and (B) of Fig. 12 show reference simulations (without RTN) of the SRAM cell with and without V_{th} shifts. Part (C) shows that RTN current spikes that occur in the absence of V_{th} shifts are unable to cause bit errors. However, *in the presence of V_{th} shifts*, similar spikes do produce bit errors (Part (D)). Indeed, it is apparent that the RTN events of M1 and M2 (pointed out in the figure), which produce RTN spikes (also pointed out in the figure) just as the SRAM cell is switching from 0 to 1, must have been responsible for this particular bit error.

B. Statistical inferences about RTN-induced SRAM write failures

SRAM arrays typically contain thousands of cells, spanning a wide range of V_{th} values and trap populations. To ascertain the impact of RTN on such circuits, we have conducted a large number of MUSTARD simulations.

Fig. 13 shows sample plots generated by sweeping the pass transistor's V_{th} and the supply voltage V_{dd} over appropriate ranges, for various RTN strengths (i.e., with 2, 4, 6 and 8 traps per transistor). As seen from the figure, the (V_{th}, V_{dd}) space is roughly banded into 3 regions: (1) a region that contains bit errors even without RTN (black), (2) a region that contains bit errors with RTN, but would

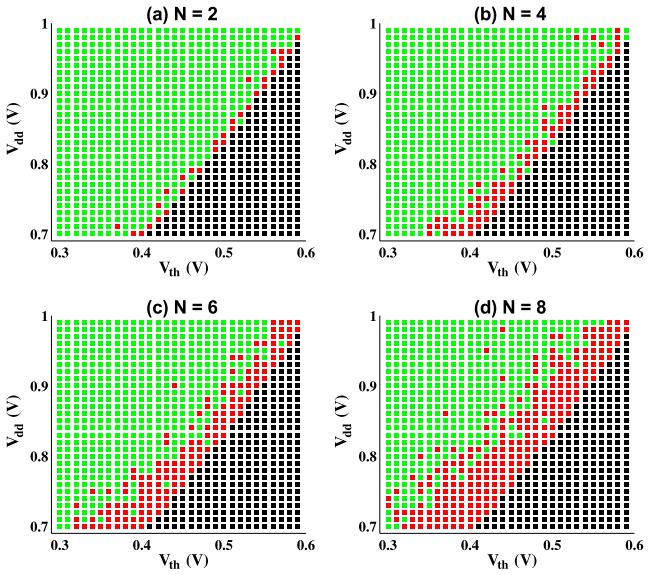

Fig. 13. MUSTARD simulations of RTN in 22nm SRAM cells, where V_{th}, V_{dd} and N are swept over appropriate ranges. For each (V_{th}, V_{dd}, N) triple, several random N-traps-per-transistor configurations are sampled and MUSTARD-simulated over a random bit pattern. Black squares indicate a bit error even in the absence of RTN. Red squares indicate a bit error in the presence of RTN, which would not have occurred if RTN had been absent. Green squares indicate robust configurations (i.e., no bit errors even in the presence of RTN).

not contain bit errors had there been no RTN (red), and (3) a region that does not contain bit errors even in the presence of RTN (green). The red region, therefore, measures the incremental contribution of RTN towards lowering the SRAM design margin, similar to the RTN component of each stacked bar in Fig. 1. As expected, this region becomes bigger as the number of traps increases. On average, the effect of RTN seems equivalent to a V_{th} shift of about 0.02V to 0.06V, which tallies well with measured data [5].

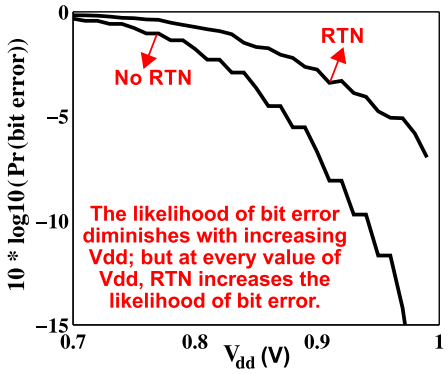

Fig. 14. Bit error probabilities as a function of V_{dd}. For each V_{dd}, several SRAM cells - with randomly distributed trap configurations and V_{th} values - are sampled and MUSTARD-simulated. V_{th} is sampled from a normal distribution while trap configurations are sampled using the trap profile model proposed in [3].

Fig. 14 quantifies the bit-error impact of RTN on SRAM arrays. Using a normal distribution for V_{th} and the trap profiling model proposed in [3], we have computed the probability of an SRAM write failure as V_{dd} varies between 0.7V and 1.0V. As expected, the bit error probability decreases with increasing V_{dd}, whether or not RTN is present. However, *in the presence of RTN*, the bit error probability diminishes with V_{dd} *at a reduced rate*, leading to a higher bit error probability at *every* value of V_{dd}.

We note that all the simulations described above (and those described in the next section) could be just as easily carried out using SAMURAI instead of MUSTARD. Indeed, we have previously applied SAMURAI [15] to predict RTN-induced SRAM failures. Here however, we have chosen to use MUSTARD, in order to demonstrate the ability to account for bi-directionally coupled RTN.

IX. Applications to DRAM Design

We have also applied MUSTARD to investigate the impact of RTN on DRAM refresh time (a.k.a. retention time). Fig. 15 presents our findings.

Fig. 15 (A) shows how the stored value Q of a 22nm DRAM cell evolves with time as the bit 1 is written to it. The plot illustrates this for two different V_{th} values (0.5V and 0.6V), and for three different RTN strengths (0 traps, 5 traps and 10 traps). From the figure, it is seen that RTN affects DRAMs in a manner markedly different from SRAMs. Indeed, whether or not a V_{th} shift is present, RTN *always* has an impact on the stored value Q of a DRAM cell. By contrast, in an SRAM cell, RTN can affect the stored value only in the presence of V_{th} shifts. For the same simulation, Fig. 15 (B) shows the number of filled traps as a function of time (for the 5 and 10 trap scenarios), while Fig. 15 (C) shows the RTN currents $I_{RTN}(t)$ (whose directions are along the blue arrow of Fig. 15 (A)). In all 4 cases, it is seen that $I_{RTN}(t)$ starts at 0, attains a peak value and tapers off towards the end of the write. This can be explained as follows: in the beginning, the transistor is off, so there is no RTN and the traps are likely to be empty [4] (because their emission propensities would be much higher than their capture propensities). As the gate voltage is increased, the traps start demonstrating activity (because their capture and emission propensities are now comparable), which leads to increased RTN current. As Q rises further, the propensities are still comparable, but the nominal current I_d becomes smaller and smaller. Therefore, even though the trap activity continues (as seen from Fig. 15 (B)), the waveform that modulates the trap activity becomes quite small, thereby causing RTN to taper off.

Fig. 15 (D)-(F) show the impact of RTN on DRAM refresh time – an important parameter that characterises how long a DRAM cell can retain a stored value (before leakage currents eventually corrupt it). Fig. 15 (D) shows that a DRAM cell with higher V_{th} needs to be refreshed more often. For two DRAM cells with the same V_{th}, the one with higher trap count needs to be refreshed more often (because, on average, the increased RTN would have weakened its stored value to a greater extent). Fig. 15 (E) shows that the required refresh frequency decreases as V_{dd} increases; however, as the number of traps increases, more frequent refreshes are needed. The final plot (F) shows how the probability of a DRAM bit error depends on refresh frequency; this plot was generated by MUSTARD-simulating hundreds of DRAM cells on random bit patterns.

X. Conclusions and Future Work

In this paper, we have presented two techniques, SAMURAI and MUSTARD, for accurate, non-stationary, coupled RTN analysis at the circuit level. Both these tools are capable of generating realistic, stochastically exact RTN traces for every transistor in a circuit, which can be used to predict RTN-induced failure rates for the entire circuit (such as an SRAM/DRAM) in the presence of variability. While SAMURAI offers the advantage of easy interoperability with any SPICE-like circuit simulator (with absolutely no modification required to the simulator), MUSTARD sacrifices inter-operability for the capability to analyse bi-directionally coupled RTN. Both tools are highly *generic*, i.e., they are designed to work with (a) any circuit design (e.g., 6T/9T SRAMs, 1T/3T DRAMs), (b) any device model (e.g., BSIM3, BSIM4, PSP), (c) any RTN model (e.g., number fluctuation models, mobility fluctuation models), and (d) any variability model (e.g., corner analysis, Gaussian PDFs, non-Gaussian PDFs). In this work, we have used MUSTARD to duplicate experimentally observed RTN-induced bit errors in SRAMs, and variable retention times in DRAMs. We are also able to generate statistical characterisations of RTN-induced failures in SRAMs and DRAMs, in the presence of variability; specifically, given an SRAM/DRAM design and a model for parameter variability, we are able to accurately predict the probability that RTN will cause the circuit to fail. To our knowledge, no CAD tool for RTN analysis offers all the above features. We have also implemented the method proposed by Ye et. al. [1], with which we have compared the accuracy and efficiency of our technique.

In future, we plan to release open-source implementations of both SAMURAI and MUSTARD.

References

[1] Y. Ye, C. C. Wang, and Y. Cao. Simulation of Random Telegraph Noise with 2-stage equivalent circuit. In *Proceedings of the IEEE/ACM International Conference on Computer Aided Design*, 2010.

[2] Y. Tsukamoto, S. O. Toh, C. Shin, A. Mairena, T. J. K. Liu, and B. Nikolic. Analysis of the relationship between random telegraph signal and negative bias temperature instability. In *Proceedings of the IEEE International Reliability Physics Symposium*, pages 1117–1121, 2010.

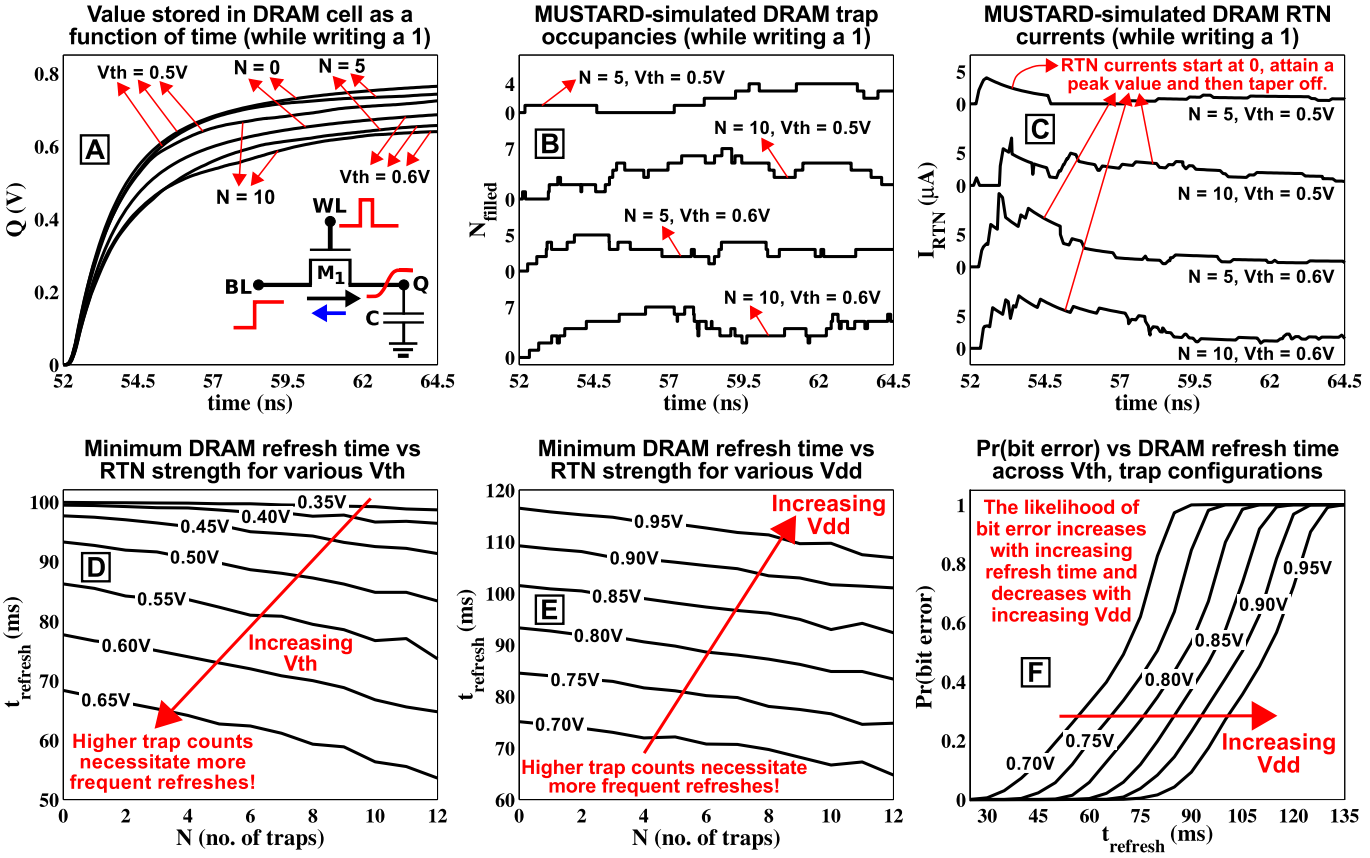

Fig. 15. RTN analysis of DRAMs using MUSTARD. The three plots on the top show that MUSTARD is capable of simulating non-stationary, coupled RTN within a DRAM cell (circuit shown in plot A). The three plots on the bottom are statistical results obtained by MUSTARD-simulating hundreds of DRAM cells with different threshold voltages and trap configurations. Details are explained in the text.

[3] M. V. Dunga. *Nanoscale CMOS modeling*. PhD thesis, The University of California, Berkeley, 2008. http://www.eecs.berkeley.edu/Pubs/TechRpts/2008/EECS-2008-20.html.

[4] H. Tian and A. El Gamal. Analysis of 1/f noise in switched MOSFET circuits. *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing*, 48(2):151–157, 2001.

[5] S. O. Toh, Y. Tsukamoto, Z. Guo, L. Jones, T. J. K. Liu, and B. Nikolic. Impact of random telegraph signals on Vmin in 45nm SRAM. In *Proceedings of the IEEE International Electron Devices Meeting*, pages 767–770, 2009.

[6] J. S. Kolhatkar. *Steady-state and cyclo-stationary RTS noise in MOSFETS*. PhD thesis, The University of Twente, The Netherlands, 2005. http://doc.utwente.nl/48261/1/thesis-Kolhatkar.pdf.

[7] P. J. Restle, J. W. Park, and B. F. Lloyd. DRAM variable retention time. In *Proceedings of the IEEE International Electron Devices Meeting*, pages 807–810, 1992.

[8] T. Umeda, K. Okonogi, K. Ohyu, S. Tsukada, K. Hamada, S. Fujieda, and Y. Mochizuki. Single silicon vacancy-oxygen complex defect and variable retention time phenomenon in DRAMs. *Applied Physics Letters*, 88(25):253504(1–3), 2006.

[9] N. Tega, H. Miki, Z. Ren, C. P. D'Emic, Y. Zhu, D. J. Frank, J. Cai, M. A. Guillorn, D. G. Park, W. Haensch, and K. Torii. Reduction of Random Telegraph Noise in high-K metal-gate stacks for 22nm generation FETs. In *Proceedings of the IEEE International Electron Devices Meeting*, pages 771–774, 2009.

[10] M. J. Kirton and M. J. Uren. Noise in solid-state microstructures: A new perspective on individual defects, interface states and low-frequency 1/f noise. *Advances in Physics*, 38(4):367–468, 1989.

[11] A. van der Ziel. *Noise in solid state devices and circuits*. Wiley Interscience, 1976.

[12] S. Lee, H. J. Cho, Y. Son, D. S. Lee, and H. Shin. Characterisation of oxide traps leading to RTN in high-K and metal gate MOSFETS. In *Proceedings of the IEEE International Electron Devices Meeting*, pages 763–766, 2009.

[13] G. I. Wirth, J. Koh, R. da Silva, R. Thewes, and R. Brederlow. Modelling of statistical low frequency noise of deep sub-micron MOSFETS. *IEEE Transactions on Electron Devices*, 52(7):1576–1588, 2005.

[14] A. S. Roy and C. C. Enz. Analytical modeling of large-signal cyclo-stationary low-frequency noise with arbitrary periodic input. *IEEE Transactions on Electron Devices*, 54(9):2537–2545, 2007.

[15] Aadithya, K. V., and Demir, A. and Venugopalan, S. and Roychowdhury, J. SAMURAI: An accurate method for modelling and simulating non-stationary Random Telegraph Noise in SRAMs. In *Proceedings of the Design, Automation and Test Conference in Europe (to appear)*, 2011.

[16] Aadithya, K. V., and Demir, A. and Venugopalan, S. and Roychowdhury, J. MUSTARD: A coupled, stochastic/deterministic, discrete/continuous technique for predicting the impact of Random Telegraph Noise on SRAMs and DRAMs. In *Proceedings of the Design Automation Conference (to appear)*, 2011.

[17] A. van der Ziel. Unified presentation of 1/f noise in electron devices: fundamental 1/f noise sources. *Proceedings of the IEEE*, 76(3):233–258, 1988.

[18] K. K. Hung, P. K. Ko, C. Hu, and Y. C. Cheng. Random Telegraph Noise of deep-submicrometer MOSFETs. *Electron Device Letters, IEEE*, 11(2):90–92, 1990.

[19] K. K. Hung, P. K. Ko, C. Hu, and Y. C. Cheng. A physics-based MOSFET noise model for circuit simulators. *IEEE Transactions on Electron Devices*, 37(5):1323–1333, 1990.

[20] J. Roychowdhury. Numerical simulation and modelling of electronic and biochemical systems. *Foundations and Trends in Electronic Design Automation*, 3(2-3):97–303, 2009.

[21] P. Heidelberger and D. M. Nicol. Conservative parallel simulation of continuous time Markov chains using uniformisation. *IEEE Transactions on Parallel and Distributed Systems*, 4(8):906–921, 1993.

[22] A. P. A. van Moorsel and K. Wolter. Numerical solution of non-homogeneous Markov processes through uniformisation. In *Proceedings of the Twelfth European Multiconference on Simulation*, pages 710–717, 1998.

[23] J. G. Shanthikumar. Uniformisation and hybrid simulation/analytic

models of renewal processes. *Operations Research*, 34(4):573–580, 1986.
[24] http://www.spiceopus.si/.
[25] D. T. Gillespie. A general method for numerically simulating the stochastic time evolution of coupled chemical reactions. *Journal of Computational Physics*, 22(4):403–434, 1976.
[26] https://projects.coin-or.org/ADOL-C.
[27] http://arma.sourceforge.net/.
[28] http://ptm.asu.edu/modelcard/HP/22nm_HP.pm.