MUSTARD: A coupled, stochastic/deterministic, discrete/continuous technique for predicting the impact of Random Telegraph Noise on SRAMs and DRAMs

Karthik .V. Aadithya*‡, Sriramkumar Venogopalan*, Alper Demir†, and Jaijeet Roychowdhury*
*Department of Electrical Engineering and Computer Science, 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, SRAMs and DRAMs have become vulnerable to Random Telegraph Noise (RTN). The bias-dependent, random temporal nature of RTN presents significant challenges to understanding its effects on circuits. In this paper, we propose MUSTARD, a technique and tool for predicting the impact of RTN on SRAMs/DRAMs in the presence of variability. MUSTARD enables accurate, non-stationary, two-way-coupled, discrete stochastic RTN simulation seamlessly integrated with deterministic, continuous circuit simulation. Using MUSTARD, we are able to predict experimentally observed RTN-induced failures in SRAMs, and generate statistical characterisations of bit errors in SRAMs and DRAMs. We also present MUSTARD-generated results showing the effect of RTN on DRAM retention times.

Categories and Subject Descriptors

B.3.1 [Semiconductor Memories] SRAM, DRAM
B.3.3 [Performance Analysis and Design Aids] Simulation

General Terms:

Algorithms, Design, Reliability

Keywords:

Random Telegraph Noise, SRAM/DRAM design, Circuit Simulation

1. INTRODUCTION

Static and Dynamic Random Access Memories (SRAMs and DRAMs) provide high-speed volatile data storage in virtually all electronic systems, including CPUs, GPUs, cameras, smartphones, *etc.* Indeed, they are one of the most important subsystems of modern ICs and SoCs – in multi-core CPUs, for example, on-chip cache (SRAM) can account for almost half the total die area [1].

In recent years, aggressive technology scaling has introduced new challenges for SRAM/DRAM (henceforth "xRAM") design. In particular, a phenomenon known as Random Telegraph Noise (RTN[1]) has become a serious concern.

Fig. 1 depicts the impact of variability and noise on SRAM design safety margins as technologies have progressed from 90nm to 22nm. The stacked bars quantify, in supply voltage (V_{dd}) terms, how static noise [2], threshold voltage shifts due to local/global parameter variations [3], NBTI [4] and RTN [5, 6] reduce SRAM design margin. The dashed line at the top represents V_{dd} scaling. It can be seen that the design margin has been shrinking steadily and that the contribution of RTN to this reduction, though small, has been growing rapidly. At the 22nm node in particular, increased RTN is in a position to eliminate the design safety margin entirely and determine whether an SRAM cell functions correctly or not. Such RTN-induced read/write failures have been reported and are of urgent concern [5, 7]. RTN is also considered responsible for Variable Retention Time (VRT) and bit errors in DRAMs [8, 9].

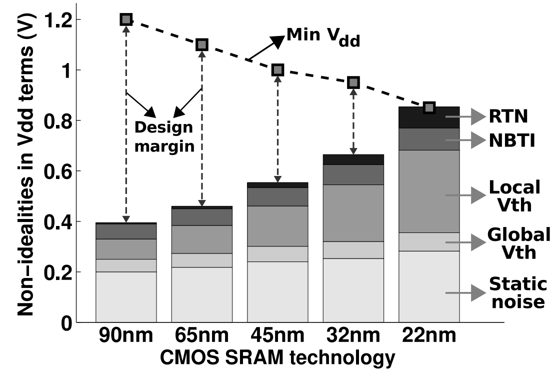

Figure 1: Impact of RTN on SRAM design margins (data courtesy Yasumasa Tsukamoto, Renesas Electronics Corp., Japan).

Predicting the impact of RTN and understanding its effects on xRAM operation presents several challenges. Unlike most other important sources of uncertainty in xRAMs, RTN is *temporally random* and can feature a wide range of time scales (see §2). This makes both measurement[2] and prediction much more involved than for, *e.g.*, local and global uncertainties. Moreover, the magnitude and temporal properties of RTN in a MOS transistor depend strongly on gate bias and current, large and rapid swings in which are the norm during xRAM operation. Because of this, the statistics of generated RTN in xRAMs are strongly *non-stationary* [10–12], making analytical approaches, which rely largely on simple stationarity assumptions, inadequate for analysis and prediction. Additionally, analytical approaches that rely on statistical averaging over large trap[3] populations are inappropriate for aggressive technologies, since trap populations in xRAM MOS devices today number roughly 10 [13, 14]. Furthermore, the continuous-time large-signal operation of xRAMs, and the discrete-event RTN that affects this operation, are *bi-directionally coupled* (Fig. 4); in other words, signal swings in the xRAM affect the generation and statistics of RTN noise, while at the same time, generated RTN can trigger large changes to these very signal swings. Moreover, as indicated in Fig. 1, other sources of variability need to be considered together with RTN to properly account for observed bit errors.

Several physics-based models for predicting the core statistics and current modulation mechanisms involved in RTN are available [14, 15], although which model is best suited for a given device in a given technology remains unclear [12]. Simulation approaches for individual traps operating under stationarity assumptions have been proposed, including a recent at-

[1]also known as RTS (Random Telegraph Signal) noise.

[2]Identical tests carried out on the same SRAM/DRAM chip at different time points can yield completely different success/failure outcomes.

[3]Traps are the source of RTN; please see §2.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
DAC 2011, June 5-10, 2011, San Diego, California, USA.
Copyright 2011 ACM ACM 978-1-4503-0636-2/11/06 ...$10.00.

tempt [16] to apply these approaches to circuits such as SRAMs, but such approaches do not take non-stationarity, bi-directional coupling, statistical trap populations and variability properly into account, while also suffering from inherent computational and accuracy limitations even for single stationary traps. Recently, the SAMURAI technique [11] has addressed the problem of accurately generating non-stationary RTN waveforms for simulation purposes. SAMURAI, however, does not deal with bi-directional coupling or variability, nor does it fully address the impact of statistical trap populations in xRAMs.

In this work, we present a new technique and tool, MUSTARD[4], which takes dynamic generation of non-stationary RTN, bi-directional xRAM-RTN coupling, statistical distributions of trap parameters and variability into account fully and correctly. The key concept behind MUSTARD is to run a deterministic, continuous-time simulation of the SRAM circuit at the SPICE level, which is coupled bi-directionally with a Markov uniformisation based, discrete event, stochastic simulation of non-stationary trap activity for generating RTN. Variability is modelled as part of the SRAM circuit and statistical analysis over trap count, trap locations, energies, *etc.*, is performed. Another important feature of MUSTARD is that it is well-suited to small trap populations and does not rely on approximations or simplifications from large-population averages.

We present experimental results using MUSTARD, detailing how RTN affects SRAM and DRAM cells in the presence (as well as absence) of other variability. We also depict failure probabilities as a function of supply voltage and threshold voltage variability. Following measurement practice [6], we use MUSTARD to plot failure probabilities as the supply voltage varies, predicting quantitatively the extent to which RTN increases the probability of bit errors.

To the best of our knowledge, no existing tool or technique is capable of conducting RTN analysis for xRAMs under such general and realistic conditions. As such, MUSTARD is suitable for use in real xRAM design situations, predicting failure probabilities that can be compared directly against measurements. Moreover, MUSTARD provides "debuggability" and fault-mechanism tracing capabilities, that can help explain and understand measurements, as well as devise and evaluate design/fabrication techniques for mitigating RTN generation and impact. MUSTARD can also be used to evaluate and discriminate between the different core physics-based models for RTN generation that are available [14, 17, 18], since it provides an accurate and convenient mechanism for assessing their impact on measurements of real xRAMs.

In §2, we detail the RTN mechanisms used in MUSTARD, while §3 focusses on MUSTARD's simulation algorithms and architecture. Results on SRAM and DRAM cells are presented in §4 and §5, respectively.

2. THE MUSTARD RTN MODEL: TIME-INHOMOGENEOUS MARKOV CHAINS COUPLED WITH DAES

RTN is produced by the random capture and release of electrons by dangling bonds located in a MOS transistor's oxide layer [15]. As depicted in Fig. 2 (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 [15].

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 [17]. As a result, every 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 current $I_d(t)$.

[4]**M**arkov **U**niformisation based **S**imulation of **T**rap **A**ctivity for **R**TN-aware **D**esign. We expect to release the MUSTARD tool under an open source license in Summer 2011.

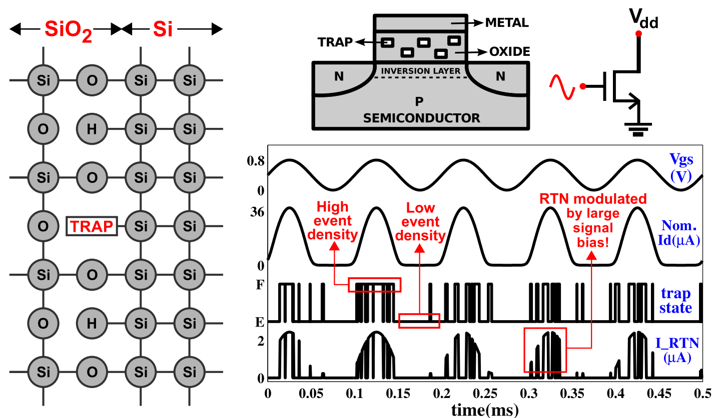

Figure 2: Left: Dangling bonds at the Si/SiO$_2$ interface. Right: Trap activity leading to large signal modulated RTN current under sinusoidal gate bias.

Furthermore, the propensity of a trap to capture/release an electron is not constant, but depends on the instantaneous gate bias $V_{gs}(t)$. Mathematically, the probability that an *empty* (*filled*) trap *tr* changes state to become *filled* (*empty*), within a short time interval dt, is given by $\lambda_{c,tr}(t)dt$ ($\lambda_{e,tr}(t)dt$), where the capture (emission) propensity $\lambda_{c,tr}(t)$ ($\lambda_{e,tr}(t)$) is a non-trivial function (whose exact form is specified in [14]) of $V_{gs}(t)$. This bias-dependence makes the electron capture/release process non-stationary [12].

Finally, the formula relating trap states to $I_{RTN}(t)$ exhibits a complicated dependence on the biasses $I_d(t)$ and $V_{gs}(t)$. For instance, one model proposed for NMOS RTN is the following equation [19]:

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

where N_{filled} denotes the number of *filled* traps, q is the electronic charge ($\sim 1.6 \times 10^{-19}$ Coulomb), W and L are the transistor dimensions and C_{ox} is the oxide capacitance per unit area. More elaborate models have also been suggested [18].

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. 2 (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 *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 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.

MUSTARD captures all the above concepts using the mathematical abstraction of a time-inhomogeneous Markov chain.

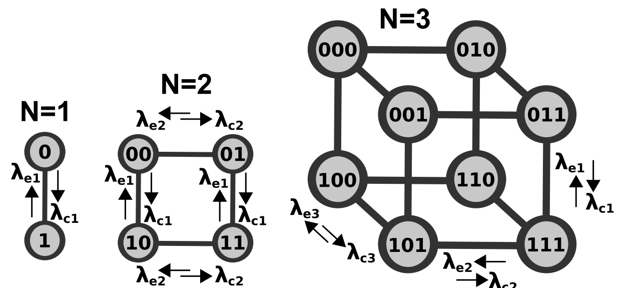

Figure 3: Hypercube Markov state transition graphs that model the RTN produced by 1, 2, and 3 trap systems. For the last case, although the figure does not explicitly show it, all parallel edges are assigned identical propensities.

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 tran-

sistor). 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 kth trap changes state, the kth 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. 3 for $N = 1$, 2 and 3).

Each dimension in the above 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. 3), which results in a time-inhomogeneous Markov chain.

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 is summarised by Fig. 4.

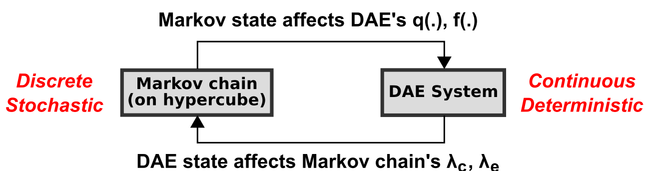

Figure 4: The MUSTARD model for RTN: A non-stationary Markov chain coupled with a DAE.

Therefore, in order to truly generate non-stationary RTN at the circuit level, one needs to develop the capability to stochastically simulate the above two-way-coupled Markov/DAE system. In the next section, we describe an exact algorithm for carrying out such a simulation.

3. MUSTARD: CORE SIMULATION ALGORITHM AND ARCHITECTURE

3.1 Simulation of two-way-coupled Markov/DAE systems by uniformisation

Algorithm 1 describes MUSTARD's strategy for generating genuinely non-stationary RTN at the circuit level. Briefly, 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 [21]. 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. This technique can be shown to be stochastically exactly equivalent to the original non-stationary Markov system (as proved in [22–24]); in other words, the theory of Markov uniformisation guarantees that no statistical test will be able to distinguish between the RTN traces produced by Algorithm 1 and the RTN traces measured on a fabricated circuit such as an SRAM or DRAM.

Moreover, by ensuring that the effects of RTN on the circuit's DAE are incorporated in a coupled manner (*i.e.*, as soon as the RTN events occur), Algorithm 1 overcomes the main limitation of SAMURAI [11]. Best of all, Algorithm 1 has the same run time as SAMURAI. Thus, the improved accuracy of MUSTARD over SAMURAI, is in fact achieved at no extra computational cost!

Algorithm 1: 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 < t_f 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(t_f, t_next_RTN) do
        record(t_curr, x_curr, tr_curr);
        t_next = min(t_curr + t_step, t_next_RTN, t_f);
        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 etc.
        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 < t_f 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 [14]
            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 [17]
                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
```

3.2 MUSTARD's software architecture

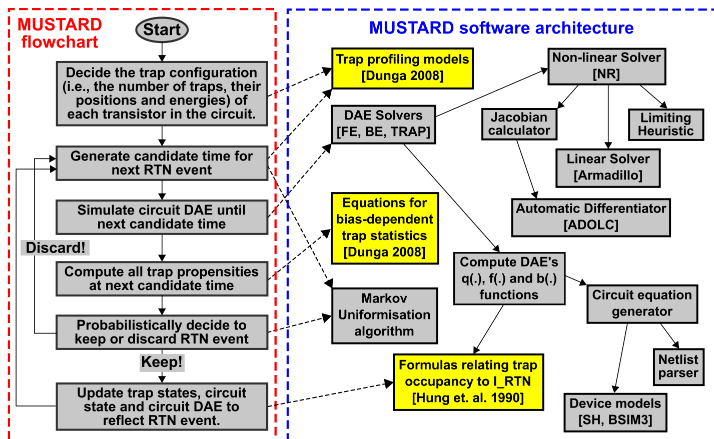

Figure 5: The flowchart (left) describes the functionality required for Algorithm 1, while the dependency graph (right) illustrates how the functionality is implemented in MUSTARD. An arrow from u to v indicates that the module for u depends on the module for v. Except ADOLC [25] and Armadillo [26], all other modules have been programmed by us in C++.

Fig. 5 illustrates the software architecture underlying MUSTARD. As the figure shows, we have implemented much of the circuit simulation functionality from scratch, in order to integrate RTN simulation more efficiently with circuit simulation. Also, our design for MUSTARD is highly 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. 5 also provides references

for the default RTN models currently used by MUSTARD.

4. SRAM RESULTS

We have applied MUSTARD to conduct coupled, non-stationary analysis of RTN in 22nm SRAMs. Our simulations 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 SRAM cells with varying trap configurations, threshold voltages and supply voltages. We now present these findings.

4.1 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 [27]) and studied its non-stationary RTN patterns using MUSTARD.

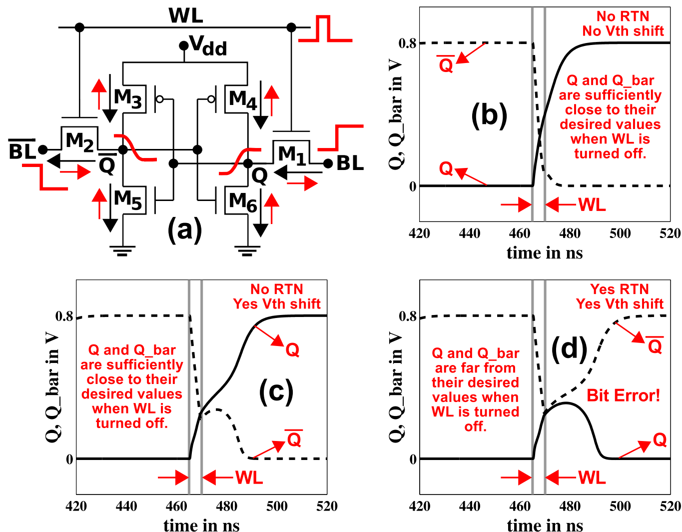

Figure 6: Part (a): Writing the bit 1 to a 6T SRAM cell. Black (Red) arrows indicate the direction of I_d (I_{RTN}) in each transistor. Parts (b)-(d): RTN, coming on top of a 100mV V_{th} shift due to parameter variations, can produce an SRAM write error.

Fig. 6 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 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 "debuggability" 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. 7). From this, it is possible to precisely pinpoint which RTN events triggered the failure (Fig. 7(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. 7 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.

4.2 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.

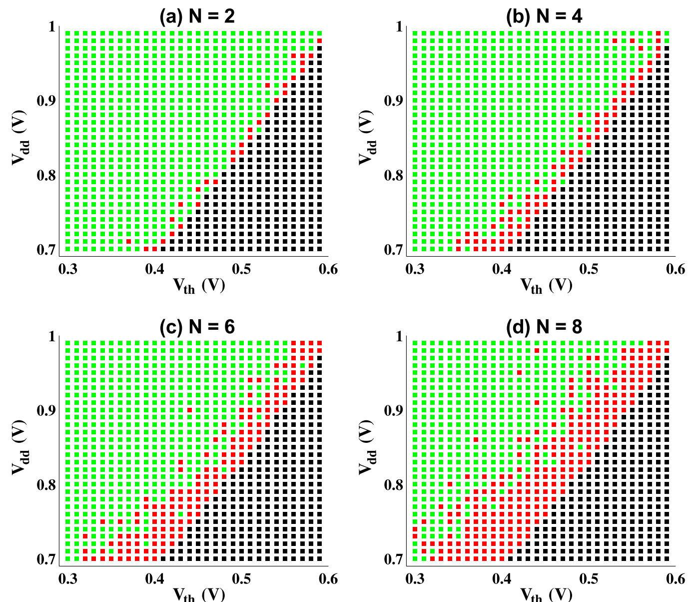

Figure 8: MUSTARD simulations of RTN in 22nm SRAM cells, where V_{th}, V_{dd} and N are swept over appropriate ranges. For each (V_{th}, V_{dd}, N) triple, several random N-traps-per-transistor configurations are sampled and MUSTARD-simulated over a random bit pattern. Black squares indicate a bit error 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).

Fig. 8 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 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. 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. 9 quantifies the bit-error impact of RTN on SRAM arrays. Using a normal distribution for V_{th} and the trap profiling model proposed in [14], 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}.

5. DRAM RESULTS

We have also applied MUSTARD to investigate the impact of RTN on DRAM refresh time (a.k.a. retention time). Fig. 10 presents our findings.

Fig. 10 (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 dif-

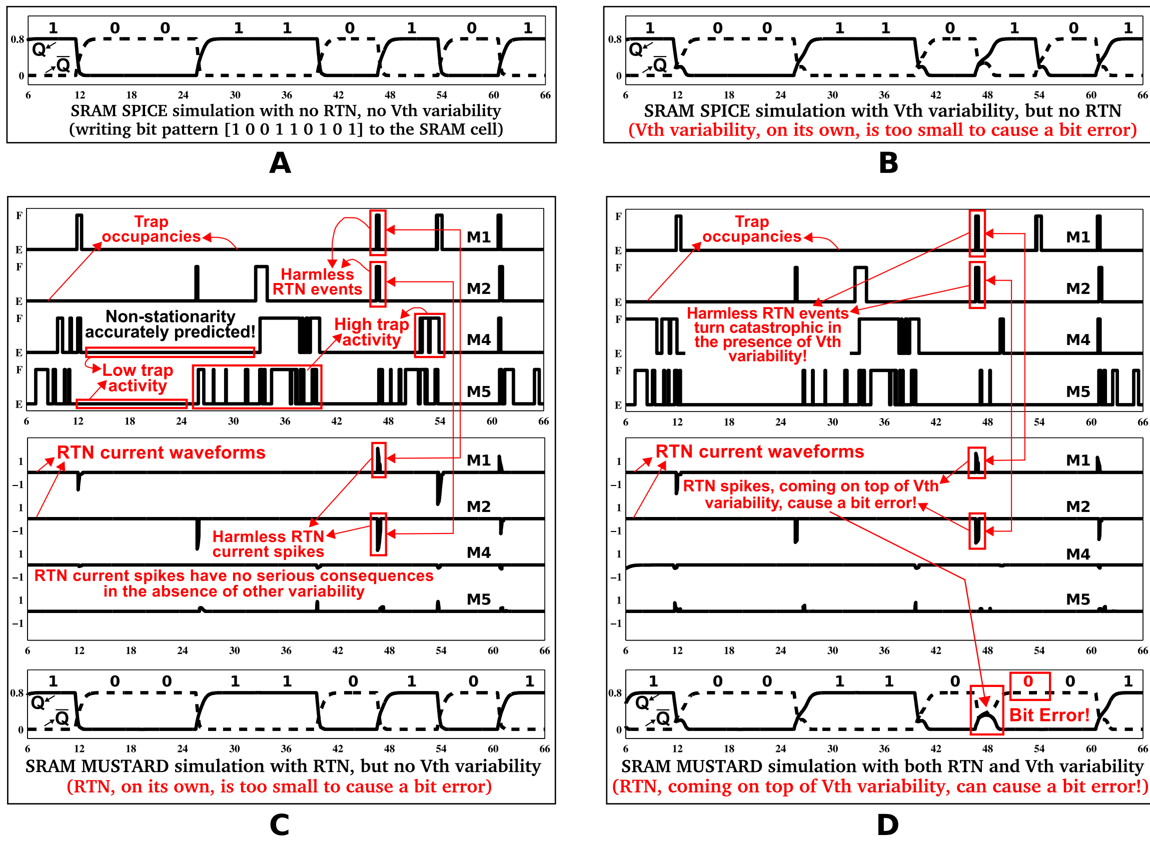

Figure 7: Examination of MUSTARD's simulation trace (including trap occupancies and RTN currents of each transistor) during the clock cycles prior to the write failure of Fig. 6. Part (D) pinpoints the RTN events responsible for the bit error. Part (C) shows that similar events are harmless in the absence 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.

Value stored in DRAM cell as a function of time (while writing a 1)

MUSTARD-simulated DRAM trap occupancies (while writing a 1)

MUSTARD-simulated DRAM RTN currents (while writing a 1)

RTN currents start at 0, attain a peak value and then taper off.

Minimum DRAM refresh time vs RTN strength for various Vth

Increasing Vth

Higher trap counts necessitate more frequent refreshes!

Minimum DRAM refresh time vs RTN strength for various Vdd

Increasing Vdd

Higher trap counts necessitate more frequent refreshes!

Pr(bit error) vs DRAM refresh time across Vth, trap configurations

The likelihood of bit error increases with increasing refresh time and decreases with increasing Vdd

Increasing Vdd

Figure 10: 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.

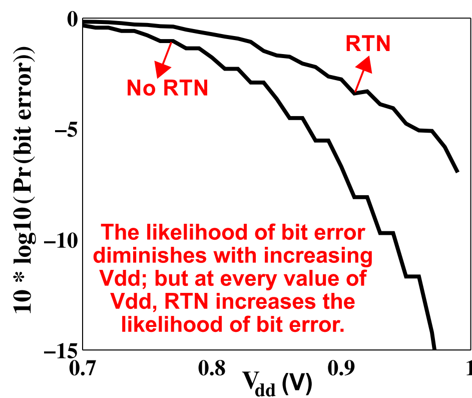

Figure 9: Bit error probabilities as a function of V_{dd}. For each V_{dd}, several SRAM cells - with randomly distributed trap configurations and V_{th} values - are sampled and MUSTARD-simulated. V_{th} is sampled from a normal distribution while trap configurations are sampled using the trap profile model proposed in [14].

ferent 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. 10 (B) shows the number of filled traps as a function of time (for the 5 and 10 trap scenarios), while Fig. 10 (C) shows the RTN currents $I_{RTN}(t)$ (whose directions are along the blue arrow of Fig. 10 (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 [12]. As the gate voltage is increased, the traps start demonstrating activity, 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. 10 (B)), the waveform that modulates the trap activity becomes quite small, thereby causing RTN to taper off.

Fig. 10 (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. 10 (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. 10 (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.

6. SUMMARY AND CONCLUSIONS

In this paper, we have presented MUSTARD, a powerful methodology and CAD tool for RTN analysis of SRAMs and DRAMs. The MUSTARD model for RTN is a time-inhomogeneous Markov chain coupled with a DAE – a generic abstraction that captures all the salient aspects of RTN generation at the trap-level. Consequently, MUSTARD is capable of generating genuinely non-stationary RTN at the circuit level (using Markov uniformisation). In addition, we have developed a complete software system, centred around MUSTARD, that enables accurate, non-stationary, two-way-coupled, stochastic, discrete RTN simulation seamlessly integrated with deterministic, continuous circuit simulation. Using this system, we have been able to duplicate experimentally observed RTN-induced failures in SRAMs. We have also been able to draw statistical inferences about the impact of RTN on entire SRAM arrays, in the presence of local and global parameter variations. We have also investigated the effects of RTN on DRAM retention times.

Acknowledgements: We would like to thank Yasumasa Tsukamoto of Renesas Electronics Corp. for Fig. 1 (and many useful discussions, besides). Support from the Semiconductor Research Corporation (SRC, task 1836.021) is also gratefully acknowledged. We would like to thank David Yeh of SRC, in particular, for his encouragement of this work.

7. REFERENCES

[1] M. Golden, S. Arekapudi, G. Dabney, M. Haertel, S. Hale, L. Herlinger, Y. Kim, K. McGrath, V. Palisetti, and M. Singh. A 2.6GHz Dual-Core 64bx86 microprocessor with DDR2 memory support. In *Proceedings of the IEEE International Solid-State Circuits Conference*, pages 325–332, 2006.

[2] E. Seevinck, F. J. List, and J. Lohstroh. Static-noise margin analysis of MOS SRAM cells. *IEEE Journal of Solid-State Circuits*, 22(5):748–754, 1987.

[3] R. Venkatraman, R. Castagnetti, and S. Ramesh. The statistics of device variations and its impact on SRAM bitcell performance, leakage and stability. In *Proceedings of the Seventh International Symposium on Quality Electronic Design*, pages 190–195, 2006.

[4] S. V. Kumar, K. H. Kim, and S. S. Sapatnekar. Impact of NBTI on SRAM read stability and design for reliability. In *Proceedings of the Seventh International Symposium on Quality Electronic Design*, pages 213–218, 2006.

[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] 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.

[7] K. Takeuchi, T. Nagumo, K. Takeda, S. Asayama, S. Yokogawa, K. Imai, and Y. Hayashi. Direct observation of RTN-induced SRAM failure by accelerated testing and its application to product reliability assessment. In *Proceedings of the IEEE International Symposium on VLSI Technology*, pages 189–190, 2010.

[8] 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.

[9] 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.

[10] A. Papoulis, S. U. Pillai, and S. Unnikrishna. *Probability, random variables, and stochastic processes*. McGraw-Hill, NY, 2002.

[11] K. V. Aadithya, A. Demir, S. Venugopalan, and J. Roychowdhury. 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*, 2011.

[12] 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.

[13] 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.

[14] 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.

[15] 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.

[16] 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.

[17] 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.

[18] 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.

[19] 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.

[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] 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.

[22] 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.

[23] 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.

[24] J. G. Shanthikumar. Uniformisation and hybrid simulation/analytic models of renewal processes. *Operations Research*, 34(4):573–580, 1986.

[25] https://projects.coin-or.org/ADOL-C.

[26] http://arma.sourceforge.net/.

[27] http://ptm.asu.edu/modelcard/HP/22nm_HP.pm.