CECS

CENTER FOR EMBEDDED & CYBER-PHYSICAL SYSTEMS

Simulation Infrastructure and System Modeling for Control-theory based Quality Configurable Memories

Biswadip Maity, Majid Shoushtari, Amir M. Rahmani and Nikil Dutt

Center for Embedded and Cyber-Physical Systems

University of California, Irvine

Irvine, CA 92697-2620, USA

{maityb, anamakis, amirr1, dutt}@uci.edu

CECS Technical Report 19-03

July 01, 2019

Simulation Infrastructure and System Modeling for Control-theory based Quality Configurable Memories

Biswadip Maity, Majid Shoushtari, Amir M. Rahmani and Nikil Dutt

July 2019

1 Introduction

Memory is the primary performance and energy bottleneck in emergent embedded workloads. Memory approximation aims to reduce the bottleneck by trading off quality for performance or energy gains. This technique applies to applications that can tolerate degradation in the quality of service (QoS). However, in order to maintain the required QoS at runtime, additional mechanisms are needed to monitor the QoS and tune the knobs to meet the requirements. In this report, we describe a simulation infrastructure for runtime control of such a system equipped with quality-configurable approximate memory. This technical report serves as a supporting document for our current work under review [3].

2 Related work - Bit Error Rate (BER) models

Previous works in the field of memory approximation have looked into the application of approximation in different memory technologies. Depending on the type of memory technology, researchers have come up with one or more knobs that can tune the degree of approximation in the memory subsystem to save energy by trading-off accuracy. The degree of approximation is represented by the probability of bit flips when accessing the memory and is called bit error rate (BER). Although BER metric is common across technologies, the mapping of BER to a physical technology knob depends on the type of memory used. In this section, we summarize the models of BER for various memory technologies:

1. **Static RAM (SRAM)**: In SRAM cells, the minimum operating voltage (Vmin) is the control knob to trade accuracy for energy gains. Wang and Calhoun developed models of cell failure probability for read, write, and hold operations in [8]. Ansari *et. al.* modeled the failure rate of an SRAM cell based on the Vmin in a 90nm technology [1]. The model is shown in Figure 1.

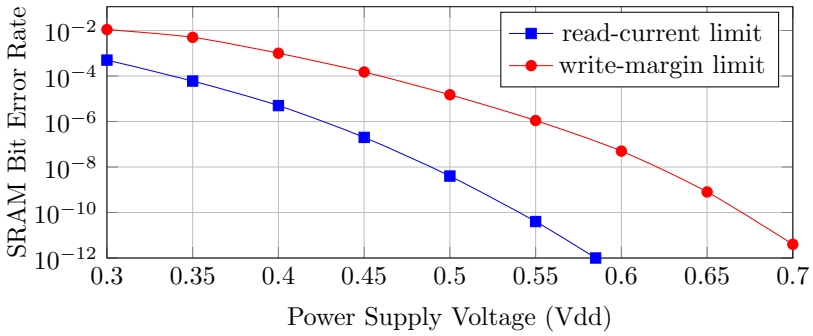


Figure 1: Bit error rate for an SRAM cell with varying Vdd values in 90nm. Figure from [1].

2. **Dynamic RAM (DRAM)**: Previous works have shown that BER in a DRAM can be modelled as a function of temperature and refresh-rate [2, 5]. One such model from [5] which relates the error rates under different refresh cycles (at $40°C$) is presented in Table 1.

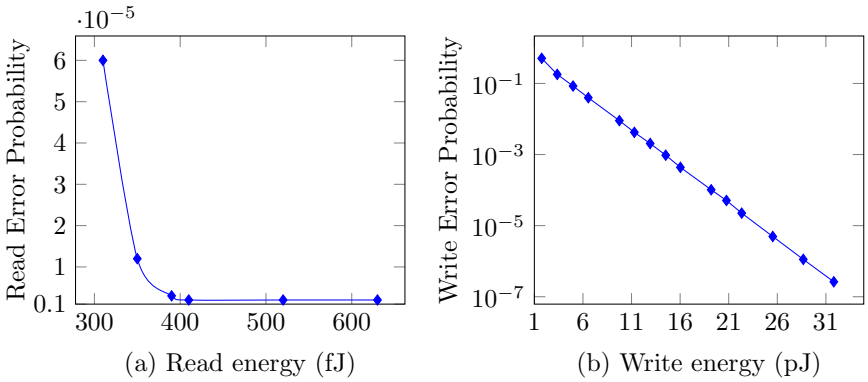
Refresh Cycle [s]	Error Rate per DRAM cell	Bit Flips per Byte
1	4.0×10^{-8}	3.2×10^{-7}
2	2.6×10^{-7}	2.1×10^{-6}
5	3.8×10^{-6}	3.0×10^{-5}
10	2.0×10^{-5}	1.6×10^{-4}
20	1.3×10^{-4}	1.0×10^{-3}

Table 1: DRAM error rate under different refresh cycle (at $40°C$). Table from [5] based on results from [2].

3. **Spin-Transfer-Torque Magnetic RAM (STT-MRAM)**: Oboril *et. al.* proposed relaxing thermal stability factor of STT-MRAM to enable fast and energy-efficient cache memories[6]. Ranjan *et. al.* developed a model for error probabilities as the read/write energy is varied for Spintronic Memories [7] and is shown in Figure 2.

In the experiments concerned with the simulation of quality-configurable memory, the underlying technology is abstracted by choosing BER as the control knob. Since there are models for each technology which maps the technology-dependent control knob to BER, the models can be incorporated in the infrastructure by adding another lookup-table or an equation which translates between BER and the technology-dependent control knob. As an example, the equation for STT-MRAM writes, as obtained from the model in Figure 2 (b), is:

$$WriteEnergy(pJ) = \min\{-4.836 \times log_{10}(BER) - 1.022, 36\} \quad (1)$$



(a) Read energy (fJ) (b) Write energy (pJ)

Figure 2: Energy vs. error probability trade-off for an STT-MRAM bit-cell from [7].

The minimum function is used to limit the energy to the value when write operation is performed without any approximation (i.e., BER = 0).

3 Simulation infrastructure

To simulate the behavior of a system with approximate memory, we developed a Sniper-based [4] memory fault injector (FI). The simulation infrastructure is described in this section.

3.1 Host system

To carry out the experiments a host system with an eight-core Intel Xeon(R) CPU E3-1230 V2 processor with 16 GB of RAM running *Ubuntu 16.04* Operating System is used. Sniper 6.1 is compiled with *gcc-4.7.4* and uses *PIN* front end.

3.2 Simulated system

The simulated system in Sniper has a single-core processor with *x86* instruction set architecture (ISA) based Gainestown micro-architecture and two levels of on-chip cache (L1 and L2). A fault injector (FI) is added which can inject faults into read / write operations of the memory hierarchy (e.g., cache, TLB, DDR) [1]. We make the following changes to implement the FI:

1. The application source code is first analyzed to detect the non-critical parts of the data. Although future work could explore automatically detecting the non-critical data elements, currently programmer expertise is

[1] Code repository at https://github.com/duttresearchgroup /memapprox-control.

needed to detect these addresses. In our target applications, we used a combination of both static analysis and Valgrind to select the non-critical data elements. Typically, large data structures which hold signal buffers (e.g., images, video) are good candidates for non-critical data.

2. To inject faults only into the non-critical data objects of the program, the source code of the program is annotated with `add_approx()` and `remove_approx()` methods to declare the address of the non-critical data objects in the program. The annotations use SimAPI commands in Sniper which are magic-instructions to communicate from the user application to the simulator at runtime. The annotated user-program is then compiled with Sniper API libraries.

   When developing a real platform, we expect to have a quality configurable memory subsystem with separate instances of: (1) Regions which do not have any quality configuration, and do not allow any errors; (2) Regions with configurable knobs (e.g., voltage in SRAM, refresh rate in DRAM, read/write current amplitude in NVM). The critical data elements can only be mapped to the non-configurable regions, whereas the non-critical can be mapped to either region.

3. Whenever `add_approx()` or `remove_approx()` methods are invoked, the methods are captured by the FI. FI records these addresses into a table (dedicated memory) in the simulator. At runtime, all the memory accesses are instrumented. Sniper is used with Pin as frontend and runs as a Pintool which is used for dynamic instrumentation. For each memory access, if the virtual address of the access falls into the any of the given address boundaries in the table, FI attempts to inject a fault into the part of data referenced by that memory access. The probability of a bit-flip is determined by the value of BER in the simulation framework.

4. A separate controller, which is implemented in the middleware, is capable of receiving the results from quality monitors at runtime from user-applications. The controller continually monitors the application's output quality. The controller can communicate with the simulator and dynamically change the degree of approximation through SimAPI commands. The controller changes the degree of approximation by setting the read/write BER knobs in the simulation framework.

4 System Model

The goal of the System Identification process is to identify a model of the system using black-box techniques. We use the System Identification toolbox in Matlab to model the system and understand the system dynamics of the generated model. The modeling and analysis are performed for each memory component for reads and writes. This section outlines the steps in detail.

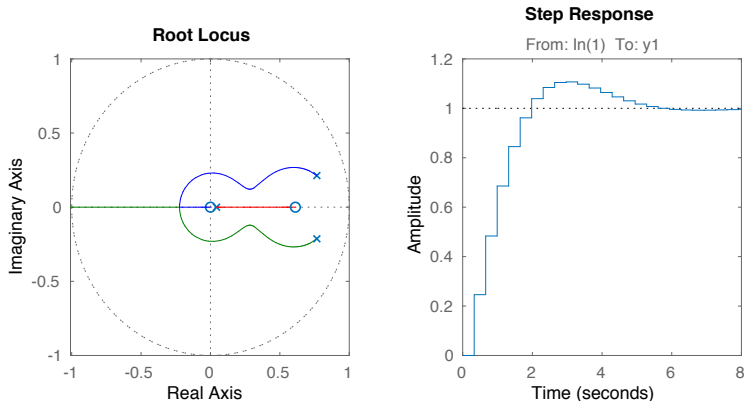


Figure 3: Root locus and step response for system identification of L1 data Writes.

4.1 Generating the data

In the simulation framework described in Section 3, the target application is executed with different bit error rates (BER) to observe the loss in accuracy. The execution starts with BER set to 0, and then it is varied in steps of $2.5e^{-7}$ after every 10 frames are processed. This creates a waveform with step-like input function. The data collection is terminated after 120 steps ($BER = 3e^{-5}$). This captures the degradation in output quality with variation of BER knobs. Depending on the application's tolerance to errors, the number of steps can be changed to capture most of expected quality values during runtime.

4.2 Discrete-time identified transfer function

A system can be modelled using Z-transforms which describe systems using transfer functions. This *transfer function* of a system describes how an input $U(z)$ is transformed into the output $Y(z)$ and is defined as $G(z) = \frac{Y(z)}{U(z)}$. We use the System Identification toolbox in Matlab to model the system using data generated in previous step.

4.3 Stability

A closed-loop system is considered stable, if all the closed-loop poles have a magnitude of less than 1. To determine the magnitude of the closed-loop poles, once the transfer function is obtained, the root locus of the system is plotted in Matlab. The root locus is the locations of all possible roots of the transfer function. The root locus and the step response obtained after modeling L1 write errors are presented in Fig 3. It can be seen that all the poles are within the unit circle, hence the system can be considered stable.

4.4 Maximum overshoot

The *maximum overshoot* is the absolute value of the largest difference between the output signal and its steady-state value, divided by steady state value and is denoted by: $M_P = \frac{|y_{max} - y_{ss}|}{|y_{ss}|}$.

4.5 Summary of system dynamics

We present the summary of the system dynamics for memory writes in the table below:

Component	Transfer function	Stable	Max-overshoot	Mode
L1D	–	-	137%	Static BER=$1e^{-3}$
L1D	$\frac{379.8z^{-1}}{1-0.823z^{-1}+0.0276z^{-2}}$	Yes	21.6%	PI
L1D	–	-	25%	Manual
L2D	$\frac{39.89z^{-1}}{1-1.817z^{-1}+0.8378z^{-2}}$	Yes	26%	PI
L2D	-	Yes	7.5%	Manual
DRAM	$\frac{1.796z^{-1}}{1-0.3254z^{-1}-0.1103z^{-2}}$	Yes	38.3%	PI

5 Runtime control algorithms

After the system identification process, the next step is to configure the memory subsystem knobs correctly. If the BER is too low, then the relaxed-accuracy is under exploited, whereas if the BER is too high, then the QoS requirement is not met. This calls for a runtime management of the BER knobs to meet the dynamic QoS requirement. A good algorithm should be able to adapt to changes in QoS quickly and maintain the target without overshoots and undershoots.

5.1 Manual control

A manual algorithm to set the knob by observing the deviation from the QoS target is presented in Algorithm 1. The minimum change of the BER knob defines the granularity of the control. The algorithm proceeds in steps towards the target QoS with a scaling factor to respond quickly to large changes. The results of tracking using the manual algorithm are presented in [3]. Although the algorithm can reach the target quality, the settling time is very large. Moreover,

Algorithm 2 Pseudo-code for manual recalibration algorithm used in tuning of memory approximation knob.

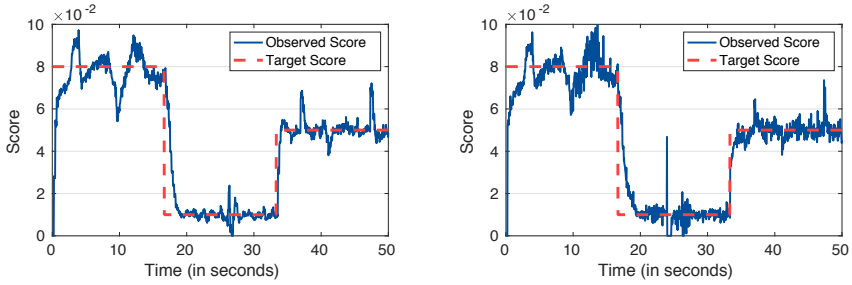
```
1: upper_bound ← 1.1                                   ▷ Upper bound for settling
2: lower_bound ← 0.9                                   ▷ Lower bound for settling
3: resolution ← 3E − 6                                 ▷ Minimum change of manual knob
4: procedure MANUAL_CALIBRATIONS(current_knob, current_error, target_error)
   ▷ The manual recalibration scheme which returns the next knob.
5:     next_knob ← knob
6:     if current_error > upper_bound × target_error then
7:         if target_error > 0 then
8:             multiplier ← log(current_error/target_error) + 1
9:         else
10:            multiplier ← 1
11:        step ← resolution × multiplier
12:        next_knob ← next_knob − step
13:    else if current_error < upper_bound × target_error then
14:        if current_error > 0 then
15:            multiplier ← log(target_error/current_error) + 1
16:        else
17:            multiplier ← 1
18:        step ← resolution × multiplier
19:        next_knob ← next_knob + step
20:    return next_knob                                ▷ The new value is next_knob
```

the manual algorithm does not take into account the history of errors which occurred before.

5.2 PI and PID control

Manual control cannot look into the history of errors and does not have any understanding of system dynamics. Moreover, it lacks any formal analysis to guarantees of stability. We have developed a formal control theory based approach in [3]. As a related work, we discuss the performance of a PI and PID controller here. Figure 4 (a) shows the quality tracking using a PI controller and 4 (b) shows the quality tracking for a PID controller.

In order to assess the reaction to stochastic inputs, we first calculate a trailing moving average of the score at each frame based on the last six frames. Then, we evaluate the Error Sum of Squares (SSE), which is the sum of the squared differences between each observation and the moving mean corresponding to the frame. This metric gives us insight into the reaction of the controller to stochastic variations in the system's input. A low value of this metric would imply that the controller is stable, and is better capable of tracking the score. For the experiment shown in Figure 4, with a PI controller this value is 0.0141, and with a PID controller this value is 0.0187. Thus, we can see that PI is more



(a) PI Controller (b) PID Controller

Figure 4: Quality tracking of a streaming video using different controllers. The L1 data cache writes are exposed to errors based on the value of the BER knob.

effective in reacting to the stochastic nature of the changing inputs. Moreover, from the figure, we can see that PID tracking has a lot more ups and downs compared to the PI tracking, which affirms our analysis.

6 Conclusion

In this report, we presented a simulation infrastructure of quality configurable memory based on Sniper simulator. We used the infrastructure for a black-box modeling technique called *System Identification* and then presented a manual re-calibration algorithm to automatically configure the memory knobs at runtime to maintain a target QoS. The simulation infrastructure currently supports running a single application as workload. Future work will look into infrastructures for running multiple workloads at the same time and developing runtime controllers to better manage the approximation knobs.

References

[1] Amin Ansari, Shuguang Feng, Shantanu Gupta, and Scott A. Mahlke. Enabling ultra low voltage system operation by tolerating on-chip cache failures. In *Proceedings of the 2009 International Symposium on Low Power Electronics and Design, 2009, San Fancisco, CA, USA, August 19-21, 2009*, pages 307–310, 2009.

[2] Vimal Bhalodia, Institute of Technology. Dept. of Electrical Engineering, and Massachusetts Computer Science. Scale dram subsystem power analysis. 03 2007.

[3] Amir M. Rahmani Biswadip Maity, Majid Shoushtari and Nikil Dutt. Self-adaptive memory approximation: A formal control theory approach. under review, .

[4] Trevor E. Carlson, Wim Heirman, Stijn Eyerman, Ibrahim Hur, and Lieven Eeckhout. An evaluation of high-level mechanistic core models. *ACM Transactions on Architecture and Code Optimization (TACO)*, 2014.

[5] Song Liu, Karthik Pattabiraman, Thomas Moscibroda, and Benjamin G. Zorn. Flikker: Saving DRAM Refresh-power Through Critical Data Partitioning. In *Proc. of ASPLOS*, 2011.

[6] Fabian Oboril, Azadeh Shirvanian, and Mehdi Baradaran Tahoori. Fault tolerant approximate computing using emerging non-volatile spintronic memories. In *34th IEEE VLSI Test Symposium, VTS 2016, Las Vegas, NV, USA, April 25-27, 2016*, page 1, 2016.

[7] Ashish Ranjan, Swagath Venkataramani, Xuanyao Fong, Kaushik Roy, and Anand Raghunathan. Approximate storage for energy efficient spintronic memories. In *Proceedings of the 52Nd Annual Design Automation Conference*, DAC '15, pages 195:1–195:6, New York, NY, USA, 2015. ACM.

[8] Jiajing Wang and Benton H. Calhoun. Minimum supply voltage and yield estimation for large srams under parametric variations. *IEEE Trans. VLSI Syst. 2011*, 19(11), 2011.