# AppropinQuo: a Platform Emulator for Exploring the Approximate Memory Design Space

Giulia Stazi, Antonio Mastrandrea, Mauro Olivieri, Francesco Menichelli
Sapienza University of Rome
Dept. of Information Engineering, Electronics and Telecommunications (DIET)
Rome, Italy
Email: {stazi,menichelli,mastrandrea,olivieri}@diet.uniroma1.it

*Abstract*—**In this work we present AppropinQuo, a flexible and configurable emulator for embedded platforms with approximate memory. The emulator includes models of the effects of approximate memory circuits and architectures, that depend on the internal structure and organization of the cells. The ability to emulate a complete platform, including CPU, peripherals and hardware-software interactions, is particularly important since it allows to execute the application as on the real board, reproducing the effects of errors on output. In fact, output quality is related not only to error rate but it also depends on the application, implementation and its data representation.**

**AppropinQuo allows to run actual applications and operating system as on the physical platform, to analyze the behavior and to expose the effects of specific approximate memory circuits and architectures on output quality. By exploring the design space regarding approximate memories, a complete characterization of the application is possible, as a step toward the determination of the trade-off between saved energy and output quality (energy-quality tradeoff).**

## I. Introduction

In modern digital systems, memory represents a significant contribution to overall system consumption [1]. This is mainly going in parallel with the increasing performance of computing platforms, which put under stress memory bandwidth and capacity. Applications as deep learning, high definition multimedia, 3D graphic, contribute to the demand for such systems, with added constraints on energy consumption.

Techniques for reducing energy consumption in SRAM and DRAM memories have been proposed in many works. A class of very promising approaches, generally called approximate memory techniques, is based on allowing controlled occurrence of errors in memory cells [2]. Although errors may degrade quality of output results, the effects can be tolerated by many applications, known as ETAs (error tolerant applications)[3].

In approximate memories, errors are allowed by design and are non-negligible for the software application. Their presence is the result, in general, of design implementations introduced to significantly reduce energy consumption [4], [5], [6]. Different strategies can be actively used in order to reduce energy consumption introducing approximation: in SRAM, for example, supply voltage scaling can be applied [7], [8]; while in DRAM refresh rate can be reduced or completely disabled [9], [10], [4], since the refresh operation degrades performance and wastes energy (e.g. when the system is in standby mode, it can reach up to 50% of total power consumption [1]).

## II. Related works and contribution

Emulators of embedded system platforms, including hardware-software interactions, are considered fundamental tools to support the design and optimization flow. During the first design phases, they provide the ability to explore different architectures and ideas, allowing to collect data regarding functionality, performance, energy consumption, with reduced development cost and time compared to physical prototypes. This is true for functional and performance emulators [11], [12], energy consumption emulators [13], [14] and fault emulators [15], [16].

Even if emulators including faults in memory units have already been developed, the fault models are not specific to the area of approximate memories and many effects of approximate memory circuits and architectures cannot be emulated using general fault models.

The contribution of the work described in this paper with respect to [17] is: (a) to present a complete implementation for modeling approximate SRAM; (b) to add DRAM models for approximate memories; (c) to add bit dropping models for SRAM and DRAM; (d) to add ECC protected cells models. These models include the ability of emulating the effects of different approximate designs and implementations, which depend on the internal structure and organization of the cells. By exploring the design space and its effects on output, a complete characterization of the application is possible, allowing the determination of the trade-off between saved energy and output quality.

## III. Error injection models for approximate memories

AppropinQuo is an extension of QEmu [11], which is a generic and open source emulator of hardware architectures capable of running different operating systems. In AppropinQuo we developed specific units to model approximate memories inside the architecture; in particular the approximate memory model is implemented as a QEmu *MemoryRegion*, mapped in the I/O memory space, that receives faults according to the error injection models. These error models take into account the techniques proposed and the circuital implementations for approximate DRAM and SRAM. Reducing refresh rate in DRAMs and allowing errors is a strategy for reducing power consumption that has produced many research works in the field of approximate memories [1], [5], [10]. The effects of these techniques at bit level can vary depending on DRAM architectures and will be discussed in Section III-A. As regards SRAMs, the techniques are more varied and applied at circuit level [7], [8], [18]. In general, energy reduction is obtained by scaling supply voltage, but the effects at bit level can be different, depending on circuit design. They are described in Section III-B.

Along with technology dependent models, some techniques specifically proposed for approximate memories work at architectural or logic level. They will be discussed in Section III-C and III-D.

## A. DRAM orientation dependent models

DRAM memory cells use a single transistor and a single capacitor to store a bit, represented as charge on the capacitor. Lowering the refresh rate of DRAMs determines that the charge loss induced by leakage current will proceed until discharge. The effects on bit value depend on the DRAM circuit architecture, that we discuss briefly.

In DRAM, single cells are organized in arrays (memory banks) and are connected to an equalizer and a sense amplifier (Fig. 1). Being differential, every sense amplifier is connected to two bitlines in order to determine whether the charge of one of them can be interpreted as logical 0 or 1: when a bitline is activated, the other holds the reference precharge voltage ($V_{DD}/2$). The sense amplifier architecture, which is a specific manufacturer design choice, determines the DRAM cells orientation. In particular the following implementations exist:

- *true-cells*: cells store a logical value of '1' as $V_{DD}$ and a logical value of '0' as $0V$;
- *anti-cells*: cells store a logical value of '0' as $V_{DD}$ and a logical value of '1' as $0V$;
- *mixed-cells*: a combination of both true-cells and anti-cells.

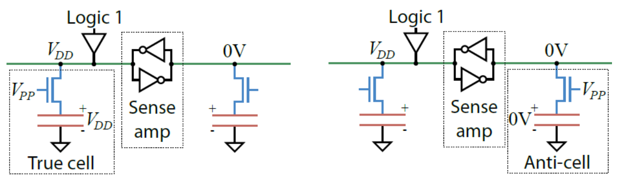

Fig. 1. DRAM true cell and anti cell. Source:[19]

When lowering refresh rate, the corresponding charge loss appears, at logic level, as a bit flip, whose orientation depends on the internal DRAM array structure. In particular, the following errors can emerge and are implemented in our model:

- *true-cell error model*: when a cell loses charge, a '1' to '0' bit flip is observed;
- *anti-cell error model*: when a cell loses charge, a '0' to '1' bit flip is observed;
- *mixed-cell error model*: both '1' to '0' and '0' to '1' bit flip occurs.

Each error model is characterized by an error probability (fault rates $p\_01$, $p\_10$ are defined respectively for the emulation of true-cell and anti-cell error effects). The probability distribution is assumed uniform in the array of cells, as showed in many works (e.g. [20]), and expressed as an error rate (*errors per second per bit*).

## B. SRAM models

SRAM approximate memories are designed with aggressive supply voltage scaling. In SRAM bitcells, read and write errors are caused by low read margin (RM) and write margin (WM) [21]. Since process variations affect RM and WM in opposite directions, the corner defines which is the critical margin (i.e. the slowfast (SF) corner makes the bitcell write critical, the fastslow corner makes it read critical). Under voltage scaling, WM and RM are degraded, increasing read and write bit error rates (BERs). The degradation is in general abrupt (BER increases exponentially at lower voltages), but techniques have been proposed to make such degradation graceful [22].

Given this behavior, our model implements an error on access mechanism, which happens when a cell is activated to perform a read or write operation. Depending on the access we distinguish three kind of errors:

- *Error on write*: introduced during a write operation, the bit stored in the cell is flipped with respect to the bit coming from the data bus;
- *Destructive error on read*: introduced during a read operation, the bit stored in the cell is flipped and passed to the data bus (both cell and data bus contain the corrupted bit);
- *Non-destructive error on read*: introduced during a read operation, the bit stored in the cell is not corrupted during the operation, but it is flipped when passed to the data bus.

For each one of these access errors, a uniform probability distribution in the array of cells is assumed, expressed as *error per access*.

## C. Bit dropping fault model

Bit dropping is a bit-level technique which consists in completely disabling some memory bitlines. The approach showed to be interesting since cells can be completely powered off or even omitted [8]. The dropped bitlines correspond to a certain number of LSBs in each word, starting from the consideration that the impact of errors is exponentially lower for smaller bit weights. The technique is transversal and can be applied in both SRAM and DRAM memory circuits (e.g in [8] for SRAM memory cells, the drop signal disables the precharge circuit during read and write operations; while in DRAMs, for dropped bitlines, the refresh operation is completely disabled).

In our model, bit dropping is implemented as follows:

- in SRAMs, when a word in memory is read or written and the bit dropping is enabled, a given number of LSBs is set always to ‘0’.
- in DRAMs, when a word in memory is read or written and the bit dropping is enabled, a given number of LSBs is set to ‘0’ or ‘1’ depending on memory cell orientation distribution.

### D. Looseness level and models

Bit level approximate techniques have been introduced in order to exploit the exponential weight that bits assume in data words. In [8] selective voltage scaling is proposed in order to modulate error rate, at the cost of an increase in circuit complexity; in [23] DRAM banks are reorganized and refresh rate modulated in order to obtain a similar effect. From the results, the technique appears effective, but the effectiveness is dependent on the microprocessor ISA and its data representation and organization in memory.

In order to support the emulation of bit level techniques, we have inserted the concept of *looseness level* and *looseness mask* in AppropinQuo. The looseness level allows to define a 32-bit configurable mask (constant for the whole memory array) that is applied to every 32-bit word in memory. Its scope is the selective protection of bits from faults (i.e. the MSBs). Bits within a word are not affected by faults when the corresponding bit in the looseness mask is set to zero (i.e. with a looseness mask set to 0x0FFFFFFF, the 4 MSBs are exact, while the 28 LSBs are affected by faults). The structure of the looseness mask allows to effectively tune approximation at bit level for 32-bit, 16-bit and 8-bit data. The technique introduces a new level of freedom in the design space, that can be explored in search of a better energy-quality trade-off.

### E. ECC protected cells models

Error correcting code (ECC) is a technique widely used for designing robust memories. ECC corrects both read and write errors and has been proposed in approximate memories as: (1) a uniform ECC that equally protects all bits [24]; (2) a selective ECC (SECC), that mitigates failures only on MSBs, that have a stronger impact on quality [25]. SEEC techniques in union with bit dropping have been demonstrated to be particularly interesting, since check bits can be taken from unused (dropped) LSBs, thus saving area and energy [8].

In order to explore ECC and SECC techniques for approximate memories, we implemented the class of Single Error Correction (SEC) codes of *(n, k)* Hamming codes, where *k* is the number of the information bits (i.e., protected) and *n* is the code length (information bits plus check bits). Other kind of codes, as BCH and ReedSolomon, suffer from larger complexity, which makes them impractical for error-tolerant applications [26].

## IV. Results

As a case study, we show the impact of approximation techniques on a signal processing application (digital FIR filtering). Due to space constraints, we report results derived from a limited number of approximate memory architectures and configurations. The digital FIR filter (100 taps) has been implemented using 32-bit integer arithmetic. The input buffer, output buffer and internal tap registers have been allocated in approximate memory. As for the underlying hardware platform, an x86 based system was configured as target architecture.

Fig. 2 shows the output SNR, with respect of the exact case, for an approximate DRAM memory. In the hypothesis of a circuit consisting of true-cells, different error rates and looseness levels are explored. For example, an error rate of about $10^{-3}$ is obtained in case of a 60x increase in refresh period [5]. The figure also shows that the looseness level can be used in order to rise SNR orthogonally to error rate, at the cost of keeping some bits exact. As expected in this application, each exact bit as an impact of $+6dB$ on SNR.

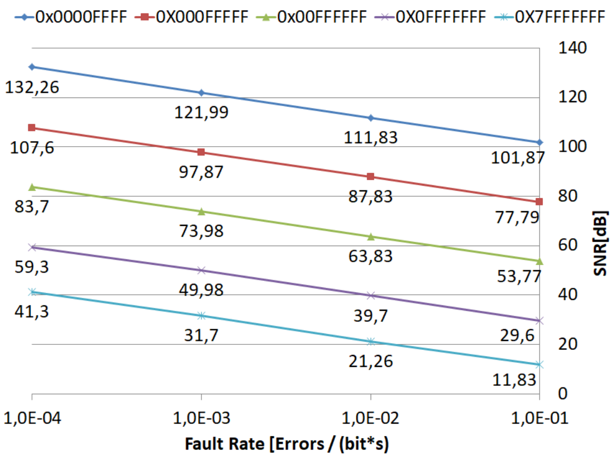

Fig. 2. Output SNR for approximate DRAM

Table I reports the output SNR in case of an approximate SRAM memory. The SRAM are explored in two corners, one containing only error on read (EOR) and the other only error on write (EOW). In this case, due to the access pattern of the application, it is possible to note that the EOW case produces higher SNR than the EOR case. Again, by modulating looseness level, higher SNR values can be obtained.

Table II reports the output SNR in case of bit dropping. The results show that the LSB dropping is an energy-wise approach in case of high BER (as can happen in case of $V_{DD}$ scaling at voltages below the minimum operating voltage), since it completely eliminates the energy associated with dropped bitlines.

## V. Conclusion

In this paper we presented an emulator for embedded platforms with approximate memory, that includes error injections models derived from approximate memory circuits proposed in literature for DRAMs and SRAMs. Given an application, it is possible to explore the impact of memory errors and approximate memory techniques on its output, discovering the relation between memory errors and output quality. The fault

TABLE I
SNR $[dB]$ FOR SRAM

| | **Looseness Level** | **Fault rate** $[errors/(bit \times s)]$ | | | |
|---|---|---|---|---|---|
| | | $10^{-4}$ | $10^{-3}$ | $10^{-2}$ | $10^{-1}$ |
| EOW | 0x0000FFFF | 127.3 | 117.6 | 107.3 | 94.6 |
| EOR | | 125.0 | 114.9 | 104.2 | 90.6 |
| EOW | 0x000FFFFF | 104.2 | 93.5 | 83.4 | 70.5 |
| EOR | | 101.4 | 90.9 | 80.3 | 66.4 |
| EOW | 0x00FFFFFF | 80.3 | 69.3 | 59.6 | 46.5 |
| EOR | | 77.5 | 66.8 | 56.3 | 42.6 |
| EOW | 0x0FFFFFFF | 56.4 | 45.5 | 35.3 | 22.6 |
| EOR | | 53.4 | 42.8 | 32.9 | 18.9 |
| EOW | 0x7FFFFFFF | 38.2 | 27.6 | 17.2 | 4.6 |
| EOR | | 35.5 | 24.9 | 14.8 | 1.0 |

TABLE II
SNR $[dB]$ FOR BIT DROPPING

| **# of dropped LSBs** | | | |
|---|---|---|---|
| **16 bits** | **20 bits** | **24 bits** | **28 bits** |
| 82.2 | 52.9 | 28.2 | 4.0 |

models have been introduced in a modular way in the emulator, in order to allow extensions as new techniques and circuits for approximate memories will be proposed.

We showed, as case study, the the impact of approximation techniques on a signal processing application, producing a figure of output degradation dependent on memory technology, error rates, looseness level. The exploration allows to select the point of work of approximate techniques, dependent on accepted output quality. As future work, the introduction of an energy consumption model for approximate memories in the emulator will allow to explore and directly determine the energy-quality tradeoff of a given combination of application and hardware platform.

## REFERENCES

[1] J. Liu, B. Jaiyen, R. Veras, and O. Mutlu, "Raidr: Retention-aware intelligent dram refresh," in *ACM SIGARCH Computer Architecture News*, vol. 40, no. 3. IEEE Computer Society, 2012, pp. 1–12.

[2] C. Weis, M. Jung, É. F. Zulian, C. Sudarshan, D. M. Mathew, and N. Wehn, "The role of memories in transprecision computing," in *Circuits and Systems (ISCAS), 2018 IEEE International Symposium on*. IEEE, 2018, pp. 1–5.

[3] G. Stazi, L. Adani, A. Mastrandrea, M. Olivieri, and F. Menichelli, "Impact of approximate memory data allocation on a h.264 software video encoder," in *Approximate and Transprecision Computing on Emerging Technologies ATCET2018, Workshop on*, 2018.

[4] S. Liu, K. Pattabiraman, T. Moscibroda, and B. G. Zorn, "Flikker: saving dram refresh-power through critical data partitioning," *ACM SIGPLAN Notices*, vol. 47, no. 4, pp. 213–224, 2012.

[5] A. Raha, S. Sutar, H. Jayakumar, and V. Raghunathan, "Quality configurable approximate dram," *IEEE Transactions on Computers*, vol. 66, no. 7, pp. 1172–1187, 2017.

[6] G. Stazi, F. Menichelli, A. Mastrandrea, and M. Olivieri, "Introducing approximate memory support in linux kernel," in *Ph. D. Research in Microelectronics and Electronics (PRIME), 2017 13th Conference on*. IEEE, 2017, pp. 97–100.

[7] F. Frustaci, M. Khayatzadeh, D. Blaauw, D. Sylvester, and M. Alioto, "Sram for error-tolerant applications with dynamic energy-quality management in 28 nm cmos," *IEEE Journal of Solid-State Circuits*, vol. 50, no. 5, pp. 1310–1323, 2015.

[8] F. Frustaci, D. Blaauw, D. Sylvester, and M. Alioto, "Approximate srams with dynamic energy-quality management," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 24, no. 6, pp. 2128–2141, 2016.

[9] D. T. Nguyen, H. Kim, H.-J. Lee, and I.-J. Chang, "An approximate memory architecture for a reduction of refresh power consumption in deep learning applications," in *Circuits and Systems (ISCAS), 2018 IEEE International Symposium on*. IEEE, 2018, pp. 1–5.

[10] M. Jung, D. M. Mathew, C. Weis, and N. Wehn, "Efficient reliability management in socs-an approximate dram perspective," in *Design Automation Conference (ASP-DAC), 2016 21st Asia and South Pacific*. IEEE, 2016, pp. 390–394.

[11] F. Bellard, "Qemu, a fast and portable dynamic translator." in *USENIX Annual Technical Conference, FREENIX Track*, 2005, pp. 41–46.

[12] N. Binkert, B. Beckmann, G. Black, S. K. Reinhardt, A. Saidi, A. Basu, J. Hestness, D. R. Hower, T. Krishna, S. Sardashti *et al.*, "The gem5 simulator," *ACM SIGARCH Computer Architecture News*, vol. 39, no. 2, pp. 1–7, 2011.

[13] S. K. Rethinagiri, O. Palomar, R. Ben Atitallah, S. Niar, O. Unsal, and A. C. Kestelman, "System-level power estimation tool for embedded processor based platforms," in *Proceedings of the 6th Workshop on Rapid Simulation and Performance Evaluation: Methods and Tools*. ACM, 2014, p. 5.

[14] K. Chandrasekar, C. Weis, Y. Li, S. Goossens, M. Jung, O. Naji, B. Akesson, N. Wehn, and K. Goossens, "Drampower: Open-source dram power & energy estimation tool," *URL: http://www. drampower. info*, vol. 22, 2012.

[15] K. Parasyris, G. Tziantzoulis, C. D. Antonopoulos, and N. Bellas, "Gemfi: A fault injection tool for studying the behavior of applications on unreliable substrates," in *Dependable Systems and Networks (DSN), 2014 44th Annual IEEE/IFIP International Conference on*. IEEE, 2014, pp. 622–629.

[16] A. Höller, A. Krieg, T. Rauter, J. Iber, and C. Kreiner, "Qemu-based fault injection for a system-level analysis of software countermeasures against fault attacks," in *Digital System Design (DSD), 2015 Euromicro Conference on*. IEEE, 2015, pp. 530–533.

[17] F. Menichelli, G. Stazi, A. Mastrandrea, and M. Olivieri, "An emulator for approximate memory platforms based on qemu," in *International Conference on Applications in Electronics Pervading Industry, Environment and Society*. Springer, 2016, pp. 153–159.

[18] J. Kwon, I. J. Chang, I. Lee, H. Park, and J. Park, "Heterogeneous sram cell sizing for low-power h. 264 applications," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 59, no. 10, pp. 2275–2284, 2012.

[19] J. Liu, B. Jaiyen, Y. Kim, C. Wilkerson, and O. Mutlu, "An experimental study of data retention behavior in modern dram devices: Implications for retention time profiling mechanisms," in *ACM SIGARCH Computer Architecture News*, vol. 41, no. 3. ACM, 2013, pp. 60–71.

[20] D. M. Mathew, M. Schultheis, C. C. Rheinländer, C. Sudarshan, C. Weis, N. Wehn, and M. Jung, "An analysis on retention error behavior and power consumption of recent ddr4 drams," in *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2018*. IEEE, 2018.

[21] K. Itoh and M. Horiguchi, "Low-voltage scaling limitations for nanoscale cmos lsis," *Solid-State Electronics*, vol. 53, no. 4, pp. 402–410, 2009.

[22] F. Frustaci, M. Khayatzadeh, D. Blaauw, D. Sylvester, and M. Alioto, "13.8 a 32kb sram for error-free and error-tolerant applications with dynamic energy-quality management in 28nm cmos," in *Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2014 IEEE International*. IEEE, 2014, pp. 244–245.

[23] J. Lucas, M. Alvarez-Mesa, M. Andersch, and B. Juurlink, "Sparkk: Quality-scalable approximate storage in dram," in *The memory forum*, 2014, pp. 1–9.

[24] M. Spica and T. Mak, "Do we need anything more than single bit error correction (ecc)?" in *Memory Technology, Design and Testing, 2004. Records of the 2004 International Workshop on*. IEEE, 2004, pp. 111–116.

[25] I. Lee, J. Kwon, J. Park, and J. Park, "Priority based error correction code (ecc) for the embedded sram memories in h. 264 system," *Journal of Signal Processing Systems*, vol. 73, no. 2, pp. 123–136, 2013.

[26] C. W. Slayman, "Cache and memory error detection, correction, and reduction techniques for terrestrial servers and workstations," *IEEE Transactions on Device and Materials Reliability*, vol. 5, no. 3, pp. 397–404, 2005.