POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Executive Summary of the Thesis

Exploring the Energy/Quality Trade-off of Non Volatile Memory in Intermittent Computing

Laurea Magistrale in Computer Science and Engineering - Ingegneria Informatica

Author: Rei Barjami

Advisor: Prof. Luca Mottola

Co-advisor: Prof. Antonio Rosario Miele

Academic year: 2022-2023

1. Introduction

The rise of the Internet of Things (IoT) has resulted in an increased demand for affordable and compact embedded devices. IoT devices normally rely on batteries as their primary energy source, which brings several challenges. Batteries can be dangerous if not adequately protected, they are expensive and bulky, they require high maintenance costs, and have a detrimental environmental impact. To address these issues, ambient energy harvesting provide a viable alternative. Energy harvested from the environment is, however, unpredictable.

As a result, execution become *intermittent*: periods of active computing alternate with periods where the device is powered off, waiting to harvest enough energy from the environment to resume computing [1]. When an energy failure occurs, data stored in volatile memory is lost. To allow the program to progress against energy failures, systems preserve the program's state in Non Volatile Memory (NVM). When the computation restarts, systems can continue from where they left.

Saving the state onto NVM is an overhead since the energy consumed by this operation does not directly contribute to application progress.

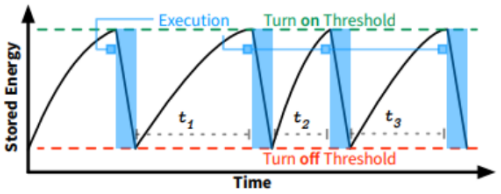


Figure 1: An intermittent execution.

Moreover, NVM is less energy efficient than its volatile counterpart: the energy penalty may heavily impact the system's efficiency. Emerging types of NVM technologies may be used to reduce this overhead. Spin-Transfer Torque Magnetic Random-Access Memory (STT-MRAM) technology allows systems to reduce the energy consumed for write operations at the cost of introducing stochastic errors in the written data. This characteristic, however, can be turned to one's advantage by employing Approximate Computing (AC) techniques [3].

In this thesis, we seize the opportunity and explore how AC can be used to reduce the overhead of NVM writes in intermittent computing. We specifically quantify the amount of energy it allows to save and the degradation in the Quality of Results (QoR) that it produces. AC is a paradigm that sacrifices precise computations

for faster performance or reduced energy consumption. It involves exploiting the resilience of certain applications to errors or noise.

2. Background

Our work intersects intermittent computing and modern NVM technology.

2.1. Intermittent Computing

Figure 1 shows an intermittent execution, where the device executes a computation until the energy stored in the capacitor reaches the turn-off threshold. Upon reaching this threshold, the device powers off. The capacitor begins recharging until it reaches the turn-on threshold, then the computation resumes. The time that separates two consecutive execution periods is not constant, as it depends on energy availability.
If partial results have not been saved to the NVM, results produced until the power-off are lost. Devices used in intermittent computing are extremely resource-constrained and hence unable to run full-fledged operating systems to manage energy failures. Ad-hoc solutions exist to persist the program's state in NVM when running applications on bare hardware. Two classes of such techniques are available: checkpoint-based systems and task-based programming abstractions.
Checkpoint-based systems save program state in NVM at specific points in code, resuming execution from the last checkpoint in case of energy failures. Task-based programming abstractions require programmers to split the program into smaller units called "tasks" that execute with *transactional* semantics.

2.2. STT-MRAM

STT-MRAM is an emerging NVM technology. It is more energy-efficient than flash memory and other NVM technologies like FeRAM and MRAM. Devolder et al. [2] noted that STT-MRAMs suffer from stochastic switching. During a write operation, the applied signal may be unable to switch the value of a cell, leading to a write failure. The probability of the bit to correctly switch depends on how much current we apply for the write.
This means that by tuning the amount of current used for writing in STT-MRAM, we can modify the energy it consumes to perform the write while also altering the Write Error Rate (WER). This property can be exploited to apply AC techniques [3], opening an avenue to explore the trade-off between energy efficiency and QoR of the program running in the system.

3. Research Question

Writes in NVM are energy-hungry operations, consuming a relevant amount of the energy stored by the device's capacitor. In intermittent computing, NVM writes might add up to 60% run-time overhead [5]. This study investigates the potential benefits of approximating writes in NVM to reduce this overhead, in exchange of a reduction in the quality of the generated output. This thesis aims to answer the following research questions:

1. What are the benefits of reducing the energy consumption of NVM writes?
2. How does this reduction impact the data processing and QoR?
3. Can we obtain significant reductions in the energy consumptions of NVM writes while maintaining an acceptable QoR?

To examine the advantages and the drawbacks that can be obtained by using AC to reduce the cost of writes in NVM, we identify three potential settings:

- **Setting 1**, the capacitor is fixed, and a checkpoint-based approach is used.
- **Setting 2**, the capacitor is fixed, and a task-based approach is used.
- **Setting 3**, the capacitor is not fixed, but the workload for each energy burst is fixed.

Our goal is to provide a comprehensive understanding of the trade-offs between energy efficiency and results accuracy.

3.1. Benefits

The energy saved by reducing the energy required to perform writes in NVM can be invested to obtain different benefits in an intermittent computing system. For each of the above-defined settings, we obtain different benefits. We analyze them *qualitatively* next. The discussion here provides a stepping stone for the *quantiative* assessment that follows.
Setting 1. The saved energy can be invested to extend the computing phase, as shown in Figure 3. A longer computing phase allows the program to make more progress in a single charge,

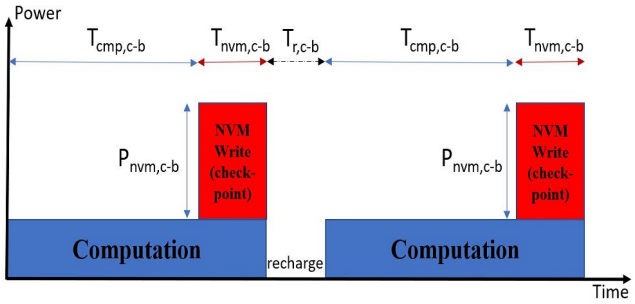


Figure 2: Checkpoint-based intermittent computation.

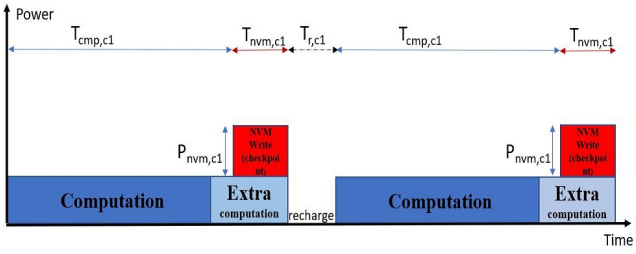


Figure 3: Increasing computing phase by reducing NVM write energy cost.

which reduces the number of energy cycles, and so checkpoints needed to complete the execution of the program. This improves system throughput since energy failures are fewer, and the system's uptime increases, thus improving the system's availability. In Figure 3 we show how, compared to the original case shown in Figure 2, the computing phase length increases.

Setting 2. The programmer can exploit the energy saved to define longer tasks compared to the original task-based scenario. Longer tasks enable the program to be completed in fewer tasks [5], reducing energy failures. Similarly to Setting 1, the system's uptime increases, increasing availability and throughput.

Setting 3. Here, the energy saved allows using a smaller capacitor to compute in an energy burst the same workload as in the original checkpoint-based or task-based scenario. Smaller capacitors provide various advantages. They have less leakage power, recharge faster, and occupy less area, making them well-suited for embedded systems where space is often a constraint.

3.2. No Free Lunches

By approximating NVM writes, we introduce errors in the data we save. The introduced errors can affect the QoR, making the output unusable. The higher the degree of approximation used, the more errors are introduced, resulting in a greater decrease in the QoR. We show an example of this in Figure 4, where the write errors cause a "salt and pepper" effect in the output image. Therefore, tuning the level of approximation used in NVM writes is crucial.

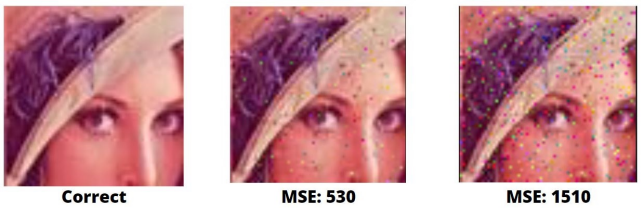


Figure 4: Mean Square Error increase with higher approximation levels.

In intermittent computing, the checkpoint written in NVM or the output of a task may be used as input in the next power cycle. This also means that errors introduced in the early power cycles have a greater impact on the final result than errors introduced in later stages. One approach to reduce this "snowball" effect is to use an increasing level of approximation, starting with a small approximation in the early phases and gradually increasing to higher approximations in later phases.

3.3. Qualitative → Quantitative

In this thesis, we want to quantitatively *explore if the energy savings we can obtain by applying this approximation are significant* or if, in practice, the amount of energy saved is negligible, making the benefits obtained minimal. At the same time, we intend to quantitatively *explore how the QoR decreases* by introducing errors in the data saved in NVM.

Intersecting the two lines of investigation allows us to ultimately conclude whether ad where using AC in intermittent computing is beneficial.

4. Methodology

We now discuss the approximation technique we utilize, the evaluation platform, the evaluation framework, and the benchmarks utilized to generate our experimental results.

4.1. Hardware Platform

Our hardware platform comprises an microcontroller (MCU) that uses STT-MRAM as NVM. As explained before, writes in

STT-MRAM are stochastic, and the probability of a bit not switching value correctly, so the WER, varies with the current used for the write operation. By adjusting this parameter, we can save energy, while increasing the WER.
To structure the analysis, we define five quality levels, each with a different set current and WER [3]. We call Q0 the baseline, with almost 100% probability of the bit to switch. In fact, its WER is 10^{-8}. Quality levels progress from Q1 to Q4, where Q4 provides the best energy efficiency but with a higher WER of 10^{-3}.
We consider MCUs from the TI MSP430 family, widely used in intermittent computing. Specifically, we chose three different MCUs, namely the MSP430L092, MSP430G2x53, and a newly-designed MSP430 core by Singhal et al. [4], called MSP430Singhal. These MCUs have the same RISC16 architecture with differerent energy consumption, amount of RAM and ROM, and operating frequencies. We chose multiple MCUs to explore the impact of AC w.r.t. the overall system's energy consumption and not just the NVM operation. Even if we save energy in writing to the NVM, the overall energy consumption may still be significant if the computing core defeats the savings due to more efficient NVM operations.

4.2. Framework

We build an evaluation framework that takes as input the characterization of STT-MRAM cells, the source code of a benchmark, the MCUs characterization, and the benchmark-specific quality metric. It returns the energy consumption of the benchmark in different levels of approximation and how the quality of the output of the benchmark degrades with different levels of approximation. This framework is composed of widely used simulators NVSim and MSPSim.
NVSim is a simulator that allows to estimate the characteristics of various NVM technologies, such as flash memory, phase-change memory, and STT-MRAM. The simulation is performed under different conditions, including varying temperature, voltage, and current for write/read operations. NVSim outputs the latency, endurance, area, leakage power, and energy consumption of the modeled NVM technology,
We utilize NVSim to emulate STT-MRAM. To fully understand the behavior of the memory array, it is necessary to accurately characterize the electrical properties of each cell, such as the current required for set and reset operations and the voltage needed for reading the cell value. By emulating STT-MRAM cells with different write currents, NVSim allows us to compute the energy cost of writes in STT-MRAMs with the quality levels defined above.
MSPSim is an emulator for the MSP430 MCU that accurately simulates the behavior of the computing core, RAM, and peripherals, allowing us to execute and debug programs in a simulated environment. One of the most relevant features of MSPSim is the time-accurate emulation of the computing core, including its instruction set, registers, and memory operations. MSPSim allows us to run our code and accurately measure performance as we would on the physical hardware.
MSPSim does not emulate any NVM on its own. We modify MSPSim to allow us to consider a part of the address space as it is STT-MRAM. This portion of the address space is divided into sections; each assigned an approximation level. This determines the probability of errors occurring when values are written to that section. Higher approximation levels result in more errors. This way, we can simulate the approximation technique for STT-MRAMs described above. When running a program using MSPSim, we can choose which data needs to be saved in the simulated STT-MRAM portion of the address space and in which section with what approximation level the data should be saved.

4.3. Benchmarks

There is no standard benchmark selection for evaluating intermittent computing systems, which have strict constraints on energy, memory, and computation resources. Other works in literature [5] use benchmarks from the MiBench2 suite for evaluating intermittent systems.
For our evaluation, we also select benchmarks from the MiBench2 suite. We chose a heterogeneous set of benchmarks to cover various aspects relevant to the evaluation, while keeping into account resource constraints of the target platform that may prevent certain benchmarks to run in the first place, for example, because of memory limitations. We consider the following aspects:

- Whether the benchmark is amenable to ap-

Benchmark	Pipeline	Quality metric
FFT	No	Avg. Relative Error (ARE)
PicoJpeg	No	RMSE
Susan Edge Detection	Yes	Precision and recall
Only Writes	No	

Table 1: The benchmark we use for our experimental evaluation and their properties.

Quality	WER	Curr. (μA)	E. x bit (pJ)
Q0	10^{-8}	1153	166
Q1	10^{-6}	865	94
Q2	10^{-5}	769	74
Q3	10^{-4}	673	57
Q4	10^{-3}	577	43

Table 2: Quality-energy map for a 32nm STT-MRAM.

proximation or it requires 100% accurate computing.

- Whether the benchmark may be considered as a pipeline of subtasks or as a single task.

For each benchmark, we choose a quality metric to assess the QoR of the approximate execution in comparison to the correct execution. The choice of the quality metric is specific to the benchmark. The benchmarks we select are summarized in Table 1.

5. Experimental Evaluation

We now show the experimental results obtained using the framework described previously.

5.1. STT-MRAM Characterization

This section presents experimental results on the energy consumed when writing a bit in STT-MRAM and how much this value can be reduced with our approximation technique.
For each quality level, we show the energy consumption of writing a bit cell obtained with NVSim. Table 2 summarizes the energy consumption results, showing that even with the first level of approximation, the energy saved is high, being 56% of the energy consumed in the baseline. The energy consumption decreases further with stronger approximation, with Q4 consuming only 25% of the energy of the correct case. It is important to point out that the decrease in energy consumption does not follow a linear pattern, with the reduction being more significant in the initial levels of approximation. However, this reduction becomes less pronounced with the strongest quality levels.

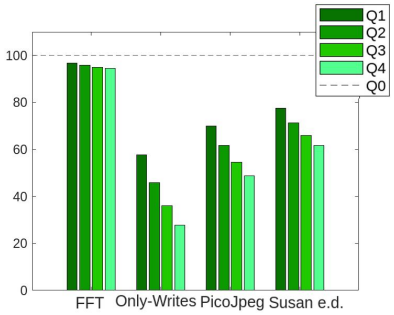
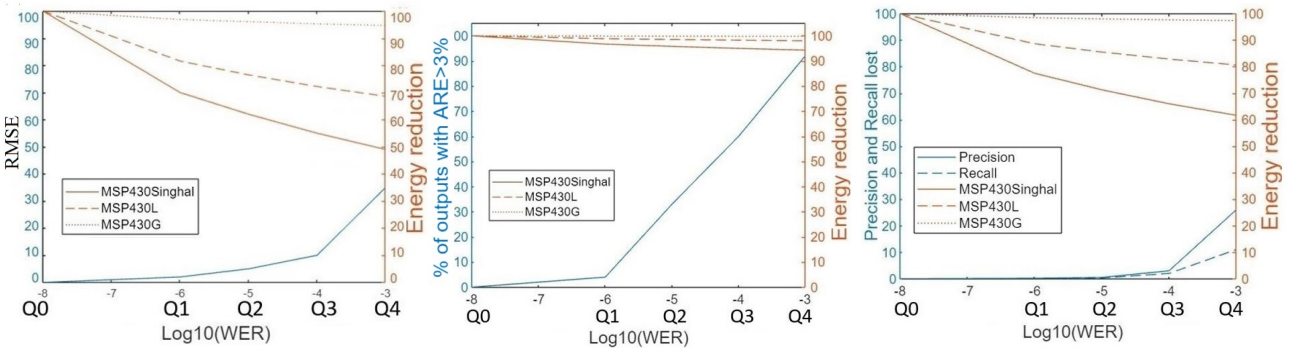


Figure 5: Energy consumption for different benchmarks with various quality levels, normalized to energy consumed with Q0.

5.2. Total Energy Consumption

Figure 5 shows an excerpt of our results, including the total energy consumed for each benchmark normalized to the same quantity consumed by the benchmark executed with Q0 with MSP430Singhal. We show that the reduction in the energy consumed varies dramatically between benchmarks. For the "Only Writes" microbenchmark, which does nothing but write an array of values in NVM and represents the base case, the energy reduction with the maximum approximation level Q4 is significant, being it only 30% of the energy consumed with Q0.
We show that the PicoJpeg yields the highest energy savings. With an approximation level of Q4, the energy consumed is only half of the energy consumed when running the benchmark with Q0. However, in FFT, the energy savings are almost negligible, regardless of the quality level. In Susan Edge Detection, we observe significant energy savings, with the energy consumed with the Q4 quality level being only 0.62 times the energy consumed with Q0.
We note that energy consumption does not decrease linearly while increasing the approximation level. So, energy consumption decreases more significantly from Q1 to Q2 than from Q3 to Q4. For this reason, pushing the approximation level too far is less effective.



(a) PicoJeg trade-off. (b) FFT trade-off. (c) Susan trade-off. $\delta = 1$

Figure 6: Trade-off energy reduction/QoR degradation.

5.3. Quality/Energy Trade-off

The price to pay for energy savings is a decrease of QoR, whose results we summarize here.

PicoJpeg. This benchmark can be seen as a single task, where the approximation happens only at the end when writing the result in NVM. Some pixels at random positions will be incorrect when a bit fails to switch. The only difference when increasing the approximation level is that we introduce more wrong pixels in the image. There is no correlation in the position of failed pixels, but they are spread randomly.

Figure 6a demonstrates this benchmark's energy/quality trade-off, showing how the RMSE compared to the result of a correct execution increases as the approximation level changes. The RMSE almost linearly increases from Q1 to Q3 and experiences a significant increase when moving to Q4, with the RMSE increasing from 9.8 at Q3 to 35 at Q4. On the other hand, the decrease in energy consumption, shown on the right, is more prominent for the first levels of approximation and mitigates in the last ones.

In summary, we observe that for Q3, the RMSE is only 9.8. With that approximation level, we can achieve a *system-wide energy reduction* of 45% and 27.8% with an MSP430Singhal and MSP430L MCU, respectively. At the cost of a small degradation in quality, we can achieve significant energy savings.

FFT. We now evaluate the trade-off for the Fast FFT benchmark. We observed that the approximated outputs follow the same curve as the correct execution, but various spikes are introduced, and the more we increase the approximation level, the more spikes are introduced in the result. Since this benchmark can be seen as a single task, an error in the result saved in NVM does not affect the rest of the execution.

Figure 6b shows how energy consumption decreases with different approximation levels. In the graph, we evaluate the percentage of outputs with an Average Relative Error (ARE) bigger than 3%, which we consider unacceptable. We found that for Q1, only 1% of the obtained results have an ARE bigger than 3%, but this percentage increases to 35% already at Q2. Thus, we argue that the maximum approximation level we can apply is Q1. Furthermore, we observe that even at the strongest approximation level, the energy savings for this benchmark are limited, up to 4.5% with the MSP430Singhal. This reinforces our conclusion that no benefits can be obtained by applying our AC technique for systems running this kind of program.

Susan Edge Detection. The Susan algorithm consists of a pipeline of tasks, and errors introduced in the initial stages of the pipeline may impact the subsequent computations, leading to more complex errors compared to FFT or PicoJpeg. The Susan edge detection benchmark is divided into various well-defined stages. Every stage ends with a write in NVM. This value is then used as input for the next stage. The errors introduced in the approximated execution of this benchmark is that some edges are not recognized, while some pixel that would not be part of an edge are recognized as part of one, so we have false positives and false negatives.

We observe from Figure 6c that also for this benchmark, the decrease in precision and recall is almost linear from Q0 to Q3, and it jumps for Q4. Q3 seems the most promising approximation level. We can note that for Q3 we have a decrease in precision of only 4% and a decrease in recall of 2%, while reducing the energy consump-

tion, with the MSP430Singhal, of about 34% the original energy consumption.

6. Conclusion

In this thesis, we explored the possibility of using a hardware approximation technique involving the stochastic switching of STT-MRAM to reduce the cost of NVM writes in an intermittent computing system. We show that in particular cases, the energy saved is significant, halving the energy consumed to run the program while maintaining a low degradation in the QoR. In other cases, even a minimal approximation that reduces slightly the energy savings causes the QoR to decrease drastically.
Future work could explore dynamically tuning this technique in multi-state pipelined programs given a final application requirements to meet.

7. Acknowledgements

I want to thank my advisor, Prof. Luca Mottola, for his advice and all the support he gave me throughout my research. Working with him was inspiring, and I am genuinely grateful for the opportunity to work under his supervision.
I would also like to thank my co-advisor, Prof. Antonio Miele, for his invaluable help and contribution to my research. His insightful feedback has been critical in helping me face the challenges of my research.
Then, I want to thank my lab colleagues: Andrea M., Matteo, Andrea P., and Mattia. They made my time in the lab such a wonderful experience.
Finally, I would like to thank my family for their encouragement and support throughout my studies. Their faith in me has been a constant source of inspiration and motivation.

References

[1] N. A. Bhatti and L. Mottola. HarvOS: Efficient code instrumentation for transiently-powered embedded sensing. In *IPSN*, pages 209–219, 2017.

[2] T. Devolder et al. Single-shot time-resolved measurements of nanosecond-scale spin-transfer induced switching: Stochastic versus deterministic aspects. *Physical review letters*, 100(5):057206, 2008.

[3] A. Monazzah et al. CAST: content-aware STT-MRAM cache write management for different levels of approximation. *TCAD*, 39(12):4385–4398, 2020.

[4] V. K. Singhal et al. 8.3 A 10.5A/MHz at 16MHz single-cycle non-volatile memory access microcontroller with full state retention at 108nA in a 90nm process. In *ISSCC*, pages 1–3, 2015.

[5] Joel Van Der Woude et al. Intermittent Computation without Hardware Support or Programmer Intervention. In *OSDI*, pages 17–32, 2016.