Learning to Approximate Computing at Run-time

Paulo Garcia, Mehryar Emambakhsh, Andrew Wallace

***Abstract*—Intelligent sensor/signal processing systems are increasingly constrained by tight power budgets, especially when deployed in mobile/remote environments. Approximate computing is the process of adaptively compromising over the accuracy of a systems output in order to obtain higher performance for other metrics, such as power consumption or memory usage, for applications resilient to inaccurate computations. It is, however, usually statically implemented, based on heuristics and testing loops, which prevents switching between different approximations at run-time. This limits approximation versatility and results in under- or over-approximated systems for the specific input data, causing excessive power usage and insufficient accuracy, respectively. To avoid these issues, this paper proposes a new approximate computing approach by introducing a supervisor block embedding prior knowledge about runtime data. The target system (i.e., signal processing pipeline) is implemented with configurable levels and types of approximations [1]. Data processed by the target system is analysed by the supervisor and the approximation is updated dynamically, by using prior knowledge to establish a confidence measure on the accuracy of the computed results. Moreover, by iteratively evaluating the output, the supervisor block can learn and subsequently update tunable parameters, in order to improve the quality of the results. Our approach also envisions switching between multiple approximation and learning engines at run-time. We detail and evaluate this approach for tracking problem in computer vision. Results show our approach yields promising trade-offs between accuracy and power consumption.**

***Index Terms*—field programmable gate array (FPGA), optimisations, power, image processing, dataflow**

I. Introduction

Power/performance trade offs are well established compromises in the design of all embedded systems [2]. In both hardware and software domains, there is a great deal of formal and empirical knowledge which guides system architects towards optimal design time decisions, and myriad runtime operation modes (i.e., power saving modes) controlled locally or remotely [3]. Approximate computing promises unprecedented power savings by introducing a trade off between power and another dimension: accuracy [4]. For applications resiliant to innacurate computations [5], or where there isn't a single golden result [6], approximate computing methods can improve traditional design strategies for power reduction: essential in the dark silicon era [7].

Despite its promise, approximate computing is still an immature technology: a formal model of the impact of approximations on other design metrics does not yet exist [8]. Hence, most approximate computing applications require two premises to be implemented successfully: (a) adequate test data are available, to correctly model the accuracy impact of approximations [9]; and, (b) approximations are performed iteratively at design time, in order to meet the required power/accuracy goals, and remain static throughoput deployment [10].

This is a stark contrast to performance/power trade offs, where well established benchmark suites offer near total coverage of application scenarios [11]: in approximate computing, test data that allows adequate modeling of accuracy is often unavailable. In performance/power trade offs, systems can self-tune their operation based on load and run time parameters to dynamically adjust metrics [12]. In approximate computing, approximations are static: mainly because there is no trusted method to determine if accuracy suffices, without access to ground truth [13]. In this paper, we tackle this problem: adjusting the level of approximations at run time, for signal processing applications. Our hypothesis states that prior knowledge about processed data can guide built-in approximation engines, dynamically modifying the level of approximations whilst ensuring that accuracy suffices for the required task.

Specifically, this paper offers the following contributions:

- We introduce the concept of prior knowledge-guided approximations. This represents a statistical measure of approximation impact, unlike test data-based empirical measures prevalent in the state of the art [14].
- We introduce a model of run time approximations, which use prior knowledge to ensure that accuracy suffices, without access to ground truth, unlike iterative comparisons to ground truth prevalent in the state of the art [15].
- We describe and evaluate a proof of concept of our approach, using an Extended Kalman Filter for motion tracking [16], where we have prior knowledge about the target's motion.

The remainder of this paper is organized as follows: Section II describes a top level view of our methodology, explaining how prior knowledge can guide approximations dynamically. Section III describes a case study of our proposed method, where prior knowledge is used to dynamically adjust the approximations applied to an Extended Kalman Filter for tracking. Section IV describes our experimental setup and the obtained results. Finally, Section V presents our conclusions and future work.

II. Learning to approximate

Our methodology is based on equipping the processing pipeline with variable levels of approximation, i.e., configurable approximations, and an approximation engine within a supervisor block (which may contain additional optimisations such as parameter tuning): a block diagram is depicted in Fig. 1. Depending on the application and the deployment technology (e.g., CPU, ASIC, FPGA) the nature of approximations

P. Garcia, M. Emambakhsh and A. Wallace are with the School of Engineering and Physical Sciences, Heriot-Watt University, Edinburgh EH14 4AS, U.K. E-mail: {p.garcia, a.m.wallace}@hw.ac.uk.

Manuscript received XXX XX, XXXX; revised XXX XX, XXXX.

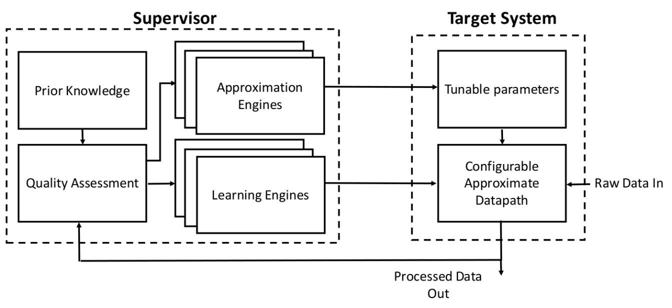

Fig. 1: Block diagram

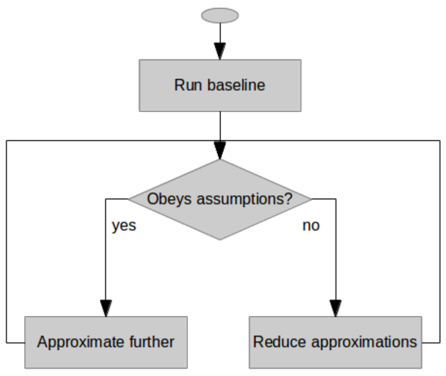

Fig. 2: Approximation engine's runtime behaviour.

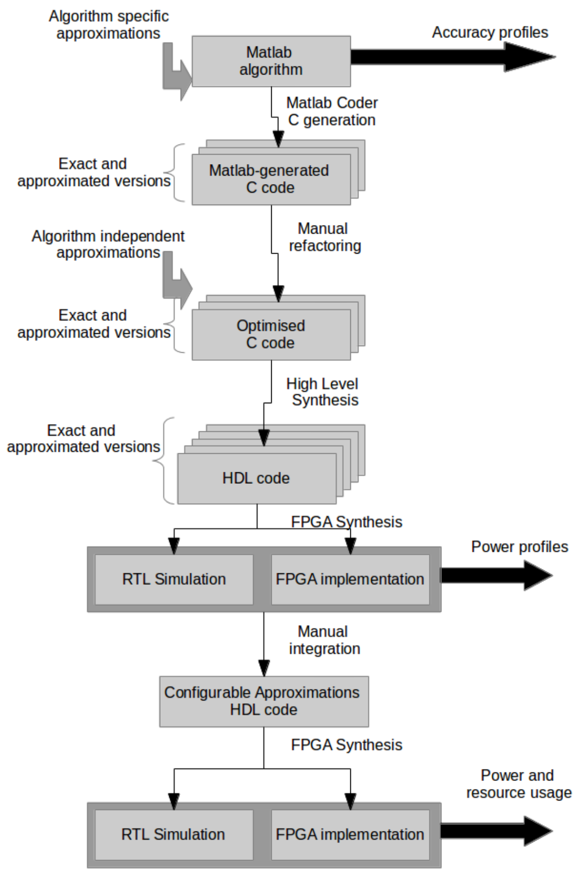

Fig. 3: Experimental design flow.

varies, but our method is applicable across the entire spectrum. Traditional approximation methods include bit width reduction [4], memoisation [17], predictive memory access [18], arithmetic re-writes [10], input-based approximations [19], etc.

Based on prior knowledge about the data, our approximation engine dynamically monitors the processing pipeline's output, and verifies whether or not the calculations still obey the assumptions about the data. If yes, then it is assumed that current levels of accuracy are still within error bounds: hence, the pipeline can be approximated further. If not, then it is assumed that accuracy has exceeded error bounds, and the level of approximation is reduced. This behaviour is described in Fig. 2. Using this method, it is possible to converge on an approximation strategy that optimises power consumption *in situ*, without access to ground truth.

************ Mehryar stuff*************

Our approach is predicated on runtime-configurable levels of approximation. In software solutions running on CPUs and GPUs, this can be achieved through different software versions [20] or through Instruction Set Architecture (ISA) level approximations [21]. In bespoke hardware solutions implemented on ASIC or FPGA, through configurable hardware versions which clock- or power-gate accurate circuitry [22]; this is the approach we use on our experiments, which we detail in Section IV.

III. Case Study: Extended Kalman Filter Tracking

IV. Experimental Results

how we go from matlab doen to FPGA 4.1 power profiles 4.2 results

V. Conclusions and Future Work

Acknowledgment

We acknowledge the support of the Engineering and Physical Research Council, grant references EP/K009931/1 (Programmable embedded platforms for remote and compute intensive image processing applications) and EP/K014277/1 (MOD University Defence Research Collaboration in Signal Processing).

References

[1] V. Vassiliadis, K. Parasyris, C. Chalios, C. D. Antonopoulos, S. Lalis, N. Bellas, H. Vandierendonck, and D. S. Nikolopoulos, "A programming model and runtime system for significance-aware energy-efficient computing," *SIGPLAN Not.*, vol. 50, no. 8, pp. 275–276, Jan. 2015.
[2] S. Das, J. R. Doppa, P. P. Pande, and K. Chakrabarty, "Reliability and performance trade-offs for 3d noc-enabled multicore chips," in *Design, Automation & Test in Europe Conference & Exhibition (DATE), 2016.* IEEE, 2016, pp. 1429–1432.
[3] S. Senni, L. Torres, G. Sassatelli, and A. Gamatie, "Non-volatile processor based on mram for ultra-low-power iot devices," *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, vol. 13, no. 2, p. 17, 2016.

[4] S. Mittal, "A survey of techniques for approximate computing," *ACM Computing Surveys (CSUR)*, vol. 48, no. 4, p. 62, 2016.

[5] Q. Xu, T. Mytkowicz, and N. S. Kim, "Approximate computing: A survey," *IEEE Design & Test*, vol. 33, no. 1, pp. 8–22, 2016.

[6] S. Venkataramani, K. Roy, and A. Raghunathan, "Approximate computing," in *VLSI Design and 2016 15th International Conference on Embedded Systems (VLSID), 2016 29th International Conference on*. IEEE, 2016, pp. 3–4.

[7] T. Mitra, T. S. Muthukaruppan, A. Pathania, M. Pricopi, V. Venkataramani, and S. Vishin, "Power management of asymmetric multi-cores in the dark silicon era," in *The Dark Side of Silicon*. Springer, 2017, pp. 159–189.

[8] S. Venkataramani, S. T. Chakradhar, K. Roy, and A. Raghunathan, "Approximate computing and the quest for computing efficiency," in *Proceedings of the 52nd Annual Design Automation Conference*. ACM, 2015, p. 120.

[9] A. Yazdanbakhsh, D. Mahajan, H. Esmaeilzadeh, and P. Lotfi-Kamran, "Axbench: A multiplatform benchmark suite for approximate computing," *IEEE Design & Test*, vol. 34, no. 2, pp. 60–68, 2017.

[10] K. Nepal, S. Hashemi, H. Tann, R. I. Bahar, and S. Reda, "Automated high-level generation of low-power approximate computing circuits," *IEEE Transactions on Emerging Topics in Computing*, 2016.

[11] J. L. Henning, "Spec cpu2000: Measuring cpu performance in the new millennium," *Computer*, vol. 33, no. 7, pp. 28–35, 2000.

[12] C. Isci and M. Martonosi, "Runtime power monitoring in high-end processors: Methodology and empirical data," in *Proceedings of the 36th annual IEEE/ACM International Symposium on Microarchitecture*. IEEE Computer Society, 2003, p. 93.

[13] V. K. Chippa, S. T. Chakradhar, K. Roy, and A. Raghunathan, "Analysis and characterization of inherent application resilience for approximate computing," in *Proceedings of the 50th Annual Design Automation Conference*. ACM, 2013, p. 113.

[14] Q. Zhang, F. Yuan, R. Ye, and Q. Xu, "Approxit: An approximate computing framework for iterative methods," in *Proceedings of the 51st Annual Design Automation Conference*. ACM, 2014, pp. 1–6.

[15] J. Han and M. Orshansky, "Approximate computing: An emerging paradigm for energy-efficient design," in *Test Symposium (ETS), 2013 18th IEEE European*. IEEE, 2013, pp. 1–6.

[16] G. Y. Kulikov and M. V. Kulikova, "The accurate continuous-discrete extended kalman filter for radar tracking," *IEEE Transactions on Signal Processing*, vol. 64, no. 4, pp. 948–958, 2016.

[17] S. Sinha and W. Zhang, "Low-power fpga design using memoization-based approximate computing," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 24, no. 8, pp. 2665–2678, 2016.

[18] A. Yazdanbakhsh, B. Thwaites, H. Esmaeilzadeh, G. Pekhimenko, O. Mutlu, and T. C. Mowry, "Mitigating the memory bottleneck with approximate load value prediction," *IEEE Design & Test*, vol. 33, no. 1, pp. 32–42, 2016.

[19] A. Raha, H. Jayakumar, and V. Raghunathan, "Input-based dynamic reconfiguration of approximate arithmetic units for video encoding," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 24, no. 3, pp. 846–857, 2016.

[20] V. Vassiliadis, K. Parasyris, C. Chalios, C. D. Antonopoulos, S. Lalis, N. Bellas, H. Vandierendonck, and D. S. Nikolopoulos, "A programming model and runtime system for significance-aware energy-efficient computing," in *ACM SIGPLAN Notices*, vol. 50, no. 8. ACM, 2015, pp. 275–276.

[21] S. Venkataramani, V. K. Chippa, S. T. Chakradhar, K. Roy, and A. Raghunathan, "Quality programmable vector processors for approximate computing," in *Proceedings of the 46th Annual IEEE/ACM International Symposium on Microarchitecture*. ACM, 2013, pp. 1–12.

[22] W.-T. J. Chan, A. B. Kahng, S. Kang, R. Kumar, and J. Sartori, "Statistical analysis and modeling for error composition in approximate computation circuits," in *Computer Design (ICCD), 2013 IEEE 31st International Conference on*. IEEE, 2013, pp. 47–53.