Quality Evaluation Strategies for Approximate Computing in Embedded Systems

Olaf Neugebauer, Peter Marwedel, Roland Kühn, Michael Engel

Quality Evaluation Strategies for Approximate Computing in Embedded Systems

Olaf Neugebauer[1], Peter Marwedel[1], Roland Kühn[1] and Michael Engel[2]

[1]TU Dortmund University, Dortmund, Germany
firstname.lastname@tu-dortmund.de
[2]Coburg University of Applied Sciences and Arts, Coburg, Germany
engel@multicores.org

Abstract. The quest for increased performance at lower energy consumption rates, especially in embedded systems used in smart systems, has reached physical limits that can no longer be exploited using traditional optimization techniques. One popular way to achieve additional gains is to intentionally perform inaccurate computations. Our framework eases evaluation, analysis and comparison of approximation techniques in terms of energy consumption, run time, quality and user-defined criteria. Applied to a set of benchmarks, we obtain valuable insights into related side effects, including increased file sizes indicating that a careless utilization of approximate computing threatens its usefulness.

Keywords: Approximate Computing, Energy Consumption, Quality Metrics

1 Introduction

Growing demands for processing power, especially in embedded systems bundled in smart systems such as autonomous cars and media signal processing, increase the pressure on hardware and software developers to create well performing systems which are also power and energy efficient. Traditionally, hardware manufacturers solved the challenge by increasing the frequencies of processors and memories, whereas the reduction of power and energy consumption was mostly achieved by shrinking semiconductor structure sizes. More recently, multiple, potentially heterogeneous, cores are being combined onto a single chip to tackle the needs of versatile performance requirements. However, frequency increase is limited by thermal and energy constraints and parallelism is restricted by synchronization overhead.

One increasingly popular method to mitigate the rising impact of these performance and power walls is to intentionally perform inaccurate computations but still satisfy the quality of service (QoS) requirements of a given application. Similar ideas have been exploited especially in the media processing domain to reduce file sizes and throughput requirements by the introduction of *lossy encoding*.

To determine the effectiveness and efficiency of approximate computing approaches, it becomes obvious that it is required to ensure that results stay within an acceptable range. As a consequence, suitable quality metrics are required. The selection of such metrics is complex, error-prone, and highly application dependent. Espe-

cially if metrics lead to contrary or insufficient conclusions, it is necessary to take multiple metrics into account. Our assessment framework gives indications of an application's performance with respect to run time, energy consumption and quality. An extensible implementation enables introduction of user-defined metrics and objectives. We found that the careless utilization of approximate computing threatens its usefulness. We observed increased output file sizes which can render the benefits of approximate computing useless in later processing steps.

2 Contribution to Smart Systems

Today's smart systems expose new requirements to embedded systems. Coupled closely with sensors and actuators, embedded systems play an important role in the whole processing chain of modern smart systems. We think, approximate computing offers new opportunities for the embedded system domain. By either applying approximation techniques to the source code or accepting inaccurate results, applications not designed for this domain can be executed on embedded systems. We showed that pattern recognition can be performed within certain limits on mobile devices. We ported an image-based virus detection application [12] from a dedicated computer to a high performance embedded system by using approximation techniques.

The next step to an aerial virus detection system with dozens of sensors is to move at least the preprocessing to a low-power embedded system closer to the sensor. We believe that new enhanced approximation techniques are key to achieve this goal. In this paper we present our assessment framework which enables us to pursue our overall goal of an aerial virus detection as an example of modern smart systems. The framework is not restricted to our application; a universal approach allows its usage in other domains.

In this paper we start with well-known approximation techniques and applications. One important result of our analyses is that focusing solely on a single metric or category can lead to significantly incorrect conclusions in the evaluation of approximation techniques; in addition, we discovered mutual influences between approximation techniques and lossy compression which can have negative side effects on objectives such as the file size. Since it is not obvious which approximation approach is suitable for a given application, the framework enables a design space exploration (DSE) of approximation techniques to assess the related gains and impacts.

In summary, the key contributions of this paper are:

- An analysis of the impact of exemplary approximation techniques on representative applications showing the difficulty of using single-metrics QoS analyses,
- An insight into conflicts between unfocused use of approximation and application-inherent lossy compression techniques,
- Enabling DSE for approximation techniques through an automatic evaluation of multiple metrics,
- Enabling runtime and energy assessment on real embedded hardware,
- Easy integration of user-defined metrics and objectives, and
- Determining the impact of approximation techniques in early design stages.

3 On Approximate Computation and Metric Selection

Approximate computing promises to be a worthwhile approach to overcome or at least shift the influence of performance and power walls. In recent years, related techniques have seen significant adoption [11]. On the hardware side, approximate computing can be applied to almost all components. For example, approximate adders [7], logic complexity reduction at the transistor level [4], and reduced refresh rates of memories [9]. Software approaches are usually more flexible than hardware ones since they can be adapted to specific scenarios or dynamically adjusted at run-time. Application knowledge can be introduced, exploring the specific requirements and capabilities of a given piece of software. Software approximation techniques can be executed on commodity hardware. Also, these techniques avoid the problem of obtaining unexpected results from inaccurate hardware computations and the problem of testing inaccurate hardware. Omitting iterations is the basic idea behind loop perforation [10, 15]. Using approximate data types and operations are the key features of EnerJ [14]. The Green system [2] enables programmers to express approximate functions and loops. Parallel applications can suffer from large synchronization overhead. Relaxing the synchronization by keeping the output quality in an acceptable range can improve the performance [13].

The impact of approximations on accuracy or QoS of applications can be quantified with metrics. Numerous metrics for different domains have been developed in the last decades. The presented approaches either rely on user-defined or heavily application-dependent quality metrics. In most cases, only one metric is considered. Akturk et al. [1] presented hints on metric selection for different domains. However, choosing the appropriate metric during the development process is a complicated task as we show in the following. First, we present some of the most important metrics used in today's approximate computing research. This paper focuses on applications from the image processing domain. Thus, metrics concerning this domain are discussed.

Metrics can be at least classified into two categories. The first category focuses on the signal itself where mathematical properties are used to quantify the quality. The second category takes the context information and recipient of the result into account. For image processing applications with humans as recipients, this category is called *Perceptual Visual Quality Metrics* (PVQM) and according to Lin et. al [8] the first category is called *Signal Fidelity Metrics* (SFM).

SFM are widespread and often used due to their simplicity. Prominent examples are the Mean-Squared Error (MSE), Root-Mean-Squared Error (RMSE), Mean-Absolute Error (MAE) and Peak-Signal-to-Noise Ratio (PSNR). In contrast to typical SFM, PQVM try to involve human visual system characteristics in the evaluation. Here, we present two metrics from this domain, the Universal Image Quality Index (UIQI) [16] and its successor the Structural Similarity Index (SSIM) [17]. Since the UIQI tends to be unstable in some cases, the SSIM uses some improvements to obtain stable results.

Original MSE = 460; SSIM = 0.95 MSE = 459; SSIM = 0.50

Fig. 1. Quality comparison according to MSE and SSIM

During the development and analysis of applications using approximate computing, a selection of proper quality metrics is of high importance. Figure 1 shows three versions of the same picture with Figure 1a being the reference picture. In Figure 1b the image has been brightened, and in Figure 1c is distorted with Gaussian blur. Applying a signal fidelity metric like MSE results in almost identical values. Thus, judging which picture is *better* is complicated. In contrast to SFM, PVQM consider the human as receiver of the results. The resulting SSIM value indicates that Figure 1b is much *better* than Figure 1c. Not only for humans, also for pattern recognition applications these results are relevant. For example, an edge detection algorithm would usually perform better on images with sharp edges such as the image from Figure 1b.

This example highlights the importance of quality metrics and their selection for approximate computing. Selecting proper metrics is complex, error-prone and usually requires deep application knowledge.

4 Framework

The *Quality Comparison for Approximate Programs on Embedded Systems* (QCAPES) framework supports developers to introduce approximate computing into their applications taking the special requirements of embedded systems into account. As described in the previous sections, this paper focuses on applications from the multimedia domain. However, as this section shows, the framework can be applied to nearly all domains. QCAPES is available under an open source license[1].

The user provides an executable without modifications which is considered as the reference and one or several inaccurate versions of the application to the framework. By passing a configuration file, the user selects metrics, objectives and data sets which are considered for the evaluation. Several metrics and some testing data from the multimedia domain are already included in the framework. Due to the open structure of QCAPES, new metrics and objectives can be added easily. Evaluation results are stored in a database and visualizations are generated.

Considering multiple metrics is crucial to evaluate the results generated by approximate versions of a given program in terms of output quality. This framework includes all previously discussed signal fidelity and perceptual-based metrics (cf. Section 3). For UIQI and SSIM, overlapping pixel by pixel or non-overlapping

[1] http://sfb876.tu-dortmund.de/auto?self=Software

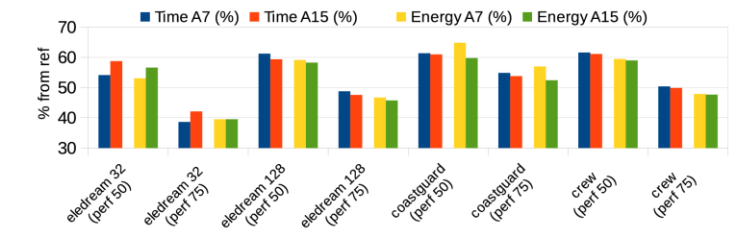


Fig. 2. Run time and energy consumption for the x264-encoder

Blockwise sliding window implementations are provided. In addition, windows of block size 4×4, 8×8 or 16×16 pixels are used for the local quality evaluation. For the SSIM a Gaussian filter can be used to reduce the influence of pixels at the edges. To summarize, a total of 13 PVQM settings are available in the QCAPES framework. Additional metrics included, e.g. numerical applications, are not covered in this paper. The user of the framework selects which metrics and objectives should be used. In this paper the Odroid-XU3, containing a modern heterogeneous MPSoC, is used as a target platform. It provides facilities to measure the energy consumption of key components.

5 Evaluation

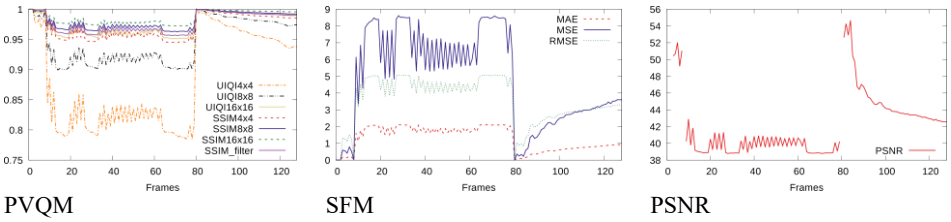
To highlight the importance of considering multiple quality metrics and objectives simultaneously, this paper presents two use-case studies from the multimedia and image processing domain typical for embedded systems. In these experiments, state of the art approximation techniques were applied.

In the first use-case essential loops of the x264[2] video encoder were perforated following the work from Sidiroglou et al. [10, 15]. The perforated applications were evaluated with two videos (640×360 pixel, 32 and 128 frames) from PARSEC [3] and two videos (352×288 pixel, both 300 frames) from Xiph.Org [18]. We used *perf*[3] to validate Sidiroglou's selection of run time critical loops to perforate since an older version of the x264-encoder was used. We identified loops to perforate in two functions (`x264_pixel_sad_x3_8x8` and `x264_pixel_sad_x3_16x16`). We choose to perforate the loops with a rate of 50% and 75% resulting in an execution of every second or forth iteration, respectively.

For each video, energy consumption and run time are measured on the Cortex-A15 and Cortex-A7 (cf. Figure 2). As expected, run time and energy consumption decrease with higher perforation rates. The results from run time reduction matches the conclusion of the original work. However, energy consumption measured on a real embedded system extends the existing work.

QCAPES enables analysis with multiple metrics. Figure 3 shows Y-component of the YCbCr encoded with perforation rate of 50%. The blockwise calculation led to

[2] x264-snapshot-20160203-2245
[3] Linux profiling with performance counters



PVQM SFM PSNR

Fig. 3. Frame-by-frame analysis of the Y channel for x264 encoding of eledream 128. Higher values in 3a and 3c are better, and in 3b lower values are better.

almost identical results as pixel-based, while the run time for blockwise calculation was significantly shorter than pixel-based calculation. This indicates a fast alternative to pixel-based evaluation. Compared to the encoding time, applying blockwise PVQM seems feasible during run time, especially when idle cores are available. In general, the resulting videos were *good* with both perforation rates.

For memory limited or distributed systems, the file size is an additional important aspect. By usage of a user-defined objective, the file size was also measured. The results indicate that dropping iterations leads to an increased file size. At a perforation rate of 50%, the file size never deviated more than 5% from the reference, while a perforation rate of 75% could lead to an increase of 20%. A detailed analysis of the frame types for the eledream 128 video shows that increasing the perforation rate increases the amount of P- and decreases B-frames.

At this point it should be emphasized again, that QCAPES compares the output of a reference implementation with the output of one or more approximate implementations. Due to this fact, the quality between the different implementations is examined. For example, in an approximate version of a video encoder, a larger fraction of P-frames could be generated, which would result in better output quality of the approximate version than the reference implementation. We have chosen this method, since it is not always possible to obtain access to the source material or it is not even desirable to get the closest output compared to the source (e.g. filter operations).

This result indicates that improvements achieved by approximation could be canceled out if a later memory or bandwidth constrained process like wireless transmission follows. Therefore, considering multiple metrics and objectives is crucial especially in the resource restricted embedded systems domain.

For the second experiment we used cjpeg, a JPEG encoder from the jpeg-9b package [6]. Here we studied two approximation techniques, namely reduced data type precision and algorithmic choice. In this case, cjpeg provides three different DCT algorithms and an adjustable compression quality. Three standard test pictures (512×512 pixel) [5], and a large picture (4000×4500 pixel) [18] were analyzed.

Run time and energy consumption in relation to the different algorithms are very similar. For the small images, only minor differences are measured. On the Odroid-XU3, the float implementation is the fastest and most energy efficient choice. The file size differs too, but not as much as for the videos.

Table 1. Impact of different quality settings for lena_color

Benchmark	MSE	RMSE	MAE	PSNR	Benchmark	MSE	RMSE	MAE	PSNR
lena_int_100	0.12	0.34	0.12	57.48	lena_fast_100	0.72	0.85	0.56	49.55
lena_int_80	0.24	0.49	0.15	54.40	lena_fast_80	1.05	1.02	0.55	47.92
lena_int_50	0.27	0.52	0.08	53.76	lena_fast_50	1.09	1.04	0.40	47.75
lena_int_30	0.22	0.47	0.06	54.69	lena_fast_30	1.21	1.10	0.33	47.32
lena_int_10	0.28	0.53	0.02	53.67	lena_fast_10	1.48	1.22	0.18	46.41

To highlight some insights, SFM results for lena_color are listed in Table 1. For this image we have evaluated five different quality settings to assess if the quality between the different implementations change. In the comparison of the *fast* and the *float* implementation we could observe very interesting results. As shown in Table 1 the MAE decreases with lower quality settings, while the MSE and RMSE increase. A possible explanation could be that small errors vanish, while larger errors increase. By using the square in MSE to avoid averaging effects, larger errors can have a greater effect on the MSE and all other metrics based upon it, even if the absolute error is getting smaller. This assumption is supported by the cjpeg documentation; the fast implementation tends to produce worse results on higher quality settings than on lower settings. This shows that it is always advisable and recommended not to rely on just one specific metric or metric family, but to involve multiple metrics in the evaluation.

6 Conclusion and Future Work

Evaluating a small set of complex, real-world benchmarks using QCAPES has already demonstrated the usefulness of an automatic, multi-metric assessment framework to discover previously unexpected effects of the naïve application of approximation techniques. In cases where approximation techniques interfere with application-specific inherent optimizations, e.g., the application of psychoacoustic or visual models for lossy data compression, the consideration of additional metrics, such as file size (and, consequently, energy consumption for transfer or storage) is necessary to efficiently apply approximation techniques without introducing undesirable side effects. We expect similar effects to show up especially in applications which are supposed to benefit most from approximation, thus multi-criterial assessment will be required for holistically building of efficient approximate systems. In addition, we propose to involve application domain experts in the QoS assessment process. Our JPEG example shows that using multiple metrics can be used to identify the structure of errors. In addition, contradicting metrics evaluating the same approximation approach demonstrate that a careful selection and combination of metrics is required for real-world use cases. Here, QCAPES provides an easy approach to integrate the simultaneous evaluation of different metrics. This is becoming especially relevant since we expect a large amount of the output of the approximate applications to be consumed by subsequent pieces of software, e.g., in autonomously driving cars, instead of a human end user who can detect and complain about insufficient quality. A dynamic QoS adaptation at runtime is a worthwhile goal, e.g. to adapt a system's output to changing requirements or to cope with previously unknown input data. Our evaluation has shown that the overhead of QoS assessment is small compared to the compu-

tational requirements for the application itself. This motivates to investigate the evolution of QCAPES into an online QoS assessment and control tool, enabling the easy establishment of *software control circuits*. The discussed metrics rely on comparing the "lossy" QoS-reduced application with a reference version. Integrating reduced reference metrics into the framework would increase the usage and flexibility.

Acknowledgements The authors like to thank the German Research Foundation (DFG) for supporting part of this work within the Collaborative Research Center SFB 876 "Providing Information by Resource-Constrained Data Analysis", projects A3 and B2, http://sfb876.tu-dortmund.de/.

References

1. Akturk, I., Khatamifard, K., Karpuzcu, U.R.: On quantification of accuracy loss in approximate computing. In: Proc. of WDDD (2015)
2. Baek, W., Chilimbi, T.M.: Green: A Framework for Supporting Energy-Conscious Programming using Controlled Approximation. In: Proc. of PLDI (2010)
3. Bienia, C., Kumar, S., Singh, J.P., Li, K.: The PARSEC Benchmark Suite: Characterization and Architectural Implications. In: Proc. of PACT (2008)
4. Gupta, V., Mohapatra, D., Park, S.P., Raghunathan, A., Roy, K.: IMPACT: IMPrecise adders for low-Power Approximate Computing. In: Proc. of ISLPED (2011)
5. Image Processing Place: http://www.imageprocessingplace.com (Dec 2016)
6. Independent JPEG Group: http://www.ijp.org (Dec 2016)
7. Kedem, Z., Mooney, V.J., Muntimadugu, K.K., Palem, K.V., Devarasetty, A., Parasuramuni, P.D.: Optimizing energy to minimize errors in dataflow graphs using approximate adders. In: Proc. of CASES (2010)
8. Lin, W., Jay Kuo, C.C.: Perceptual visual quality metrics: A survey. J. Vis. Commun. Image Represent. 22(4) (May 2011)
9. Liu, J., Jaiyen, B., Veras, R., Mutlu, O.: RAIDR: Retention-Aware Intelligent DRAM Refresh. In: Proc. of ISCA. vol. 40 (2012)
10. Misailovic, S., Sidiroglou, S., Hoffmann, H., Rinard, M.: Quality of service profiling. In: Proc. of ICSE. vol. 1 (2010)
11. Mittal, S.: A survey of techniques for approximate computing. ACM Comput.. 48(4) (2016)
12. Neugebauer, O., Libuschewski, P., Engel,M., Müller, H., Marwedel, P.: Plasmon-based Virus Detection on Heterogeneous Embedded Systems. In: Proc. of SCOPES (2015)
13. Renganarayana, L., Srinivasan, V., Nair, R., Prener, D.: Programming with relaxed synchronization. In: Proc. of RACES (2012)
14. Sampson, A., Dietl, W., Fortuna, E., Gnanapragasam, D., Ceze, L.: EnerJ: Approximate Data Types for Safe and General Low-power Computation. In: Proc. of PLDI (2011)
15. Sidiroglou, S., Misailovic, S., Hoffmann, H., Rinard, M.: Managing performance vs. accuracy trade-offs with loop perforation. In: Proc. of ESCE/FSE (2011)
16. Wang, Z., Bovik, A.C.: A universal image quality index. Signal Processing Letters, IEEE 9(3) (2002)
17. Wang, Z., Bovik, A.C., Sheikh, H.R., Simoncelli, E.P.: Image quality assessment: From error visibility to structural similarity. IEEE Trans. on Image Process. 13(4) (2004)
18. Xiph.Org Foundation: http://www.xiph.org (Dec 2016)