This paper appears in the 2017 IGSC Workshop on SiMS

POWSER: A Novel User-Experience Based Power Management Metric

Maithili P Gandhe
University of Texas, Austin, USA
maith_us@yahoo.com

Lizy K John
University of Texas, Austin, USA
ljohn@ece.utexas.edu

Andreas Gerstlauer
University of Texas, Austin, USA
gerstl@ece.utexas.edu

ABSTRACT

With the rapid increase in deployment of a variety of applications on mobile devices, and an increasing amount of activity and time spent on the devices, battery life continues to be a major source of concern. At the same time, hardware and software advances in mobile computing have enabled multiple usage functions to be integrated into one device, making energy consumption of the device more critical. Some approximations in computing can be made without noticeable degradation in quality to achieve savings in power. We explain the concept of **user-experience based power management (UEPM)** and propose a novel simple to use **user-perceived-quality-energy metric** called POWSER to build a UEPM-based system with a holistic objective. POWSER can also provide optimal operating points for the applications to perform fast dynamic online optimizations of applications such as video. Quality-energy tradeoffs have been carried out at various levels before. However, existing approaches do not take the final user-experience into account. POWSER is an easy to compute and easy to use weighted metric based on the nature of quality-energy tradeoffs. POWSER models QoS parameters in software that can be propagated down to hardware to implement further power savings across the system stack. Keeping acceptable user experience, POWSER provides an optimal set of operating points for each application by exploiting power savings of up to 34% in image applications.

1. INTRODUCTION

Traditionally, power management at the system level involves dynamic voltage frequency scaling [8] and sleep states [9]. These schemes offer energy savings per the implementation of the scheme for the system. The energy saved could sometimes be inadequate and at other times aggressive depending on the mode of the system and its intended use. We thereby introduce the concept of user-experience based power management that would utilize these schemes in an effective way. The first question we ask is, 'What is User-Experience'? To answer this question correctly, the context in which we quantify user-experience needs to be identified. For e.g. user-experience on a high powered gaming system might depend on the quality of the screen, performance of the system and its audio quality. In our study, we analyze user-experience in relation to portable devices. User satisfaction on mobile devices such as smartphones is envisioned to be closely tied to the application software experience as well as battery life of the device. Software experience depends on the type of the application being run e.g. chasing-type video game vs solitaire. Battery life depends on two factors. The first is the usage mode of the device e.g. playing videos, clicking/viewing pictures or simply texting messages. The second factor is the power consumption of the device.

We explain the concept of user experience via an image interpolation experiment. In this experiment, two images were chosen as shown in Figure 1. First one is that of a screen and the second one is that of a hamburger. Both these images were fed to a MATLAB image interpolation routine and scaled down to one tenth their size. Hence the resolution of the pictures was affected by the same factor.

As seen in the second column of Figure 1, the screen picture degradation appears worse to the eye than the quality loss on the hamburger picture. The gradient of the pixel change or edges directly contribute to how degradation is perceived by human eye. Abrupt pixel changes leading to crisp edges in the picture degrade worse than a picture with smoother edges. In the pictures shown in Figure 1, the scaling factor applied to both images was 0.1x hence the pixel loss rate was the same. However human perceived quality of both degraded images appears to be different. User experience can therefore be closely correlated to human perceived quality for image applications rather than a metric based on pixel loss rate such as peak signal-to-noise ratio (PSNR). Moreover a metric like PSNR requires both pictures to be the same size.

978-1-5386-3470-7/17/$31.00 ©2017 IEEE

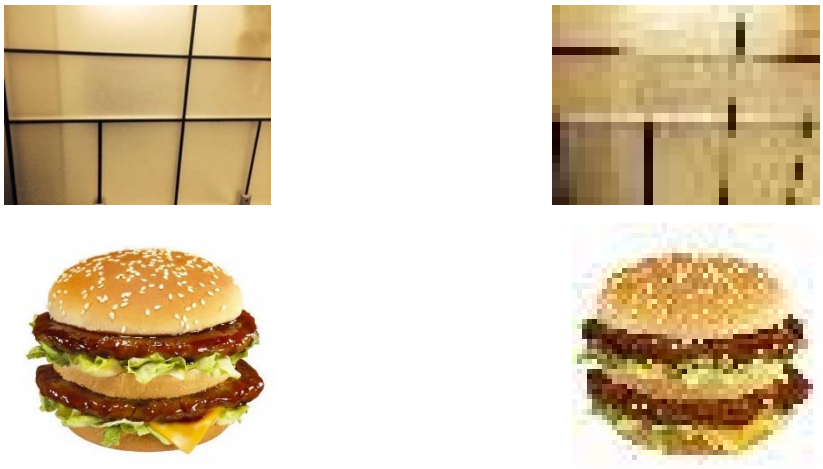

Figure. 1. Two images scaled by the same factor (0.1X) in MATLAB. Original and scaled versions are shown. The scaled hamburger has better user-perceivable quality than the scaled screen.

Recent work has been done in application resilience [12] that characterizes applications for their sensitivity to approximation. The closest work has been to design quality programmable vector processors [13] where in the concept of quality has been introduced at the HW/SW interface via a quality-aware instruction set architecture. The quality of the quality metric for various applications impacts resilience analysis for it in a big way [12]. However they have laid the onus of the quality metric on the user and the metric is provided as a user-defined input for their characterization framework. The metrics used are mathematical metrics and do not necessarily correlate to human perception. We believe that a metric that would affect application resilience analysis hugely needs to incorporate key features per application set that affects quality as perceived by a human. Also the metric should be close-to-flesh since the consumer for these applications is usually a person [3]. Also, in real systems a quality-energy (QE) metric affects design choices rather than a pure quality metric. Hence the need to develop a QE metric that can be easy to use, compute and propagate.

There are two primary contributions in this work. First, we propose a novel human-perceptible QE metric called POWSER. It is an easy to compute and easy to use metric that exploits the trade-offs between quality and energy and provides QoS parameters for downstream use. The metric also incorporates user preferences for QE.

Secondly we demonstrate the use of POWSER across the system stack. POWSER is used to provide a set of optimal operating points per application as well as convey information about the application set for QoS tunable quality knobs across the system stack. The QoS parameters are embedded in the quality metric used for the application set and affected by human perception.

The rest of the paper is organized as follows. Section 2 gives an overview of our user-experience based power management approach. Section 3 describes our results. Section 4 outlines what we envision as applications of POWSER. Section 5 discusses related work and Section 6 offers our concluding remarks.

2. PROPOSED APPROACH

We use approximate computing approach wherein data variables are approximated in the code thereby losing some precision in the data but saving on computation and memory energy. Thus we meet the user satisfaction requirements and enhance the multimedia experience by saving energy thereby extending the battery life of the device. For this paper we target image benchmarks since those can be considered as user-relevant applications.

Figure 2 shows the conceptual structure of a user-experience based power manager. We propose UEPM optimizations across the system stack and hence we are looking for a metric that can be propagated across the stack. A consistent metric available across the system stack will facilitate a UEPM system built on a holistic objective.

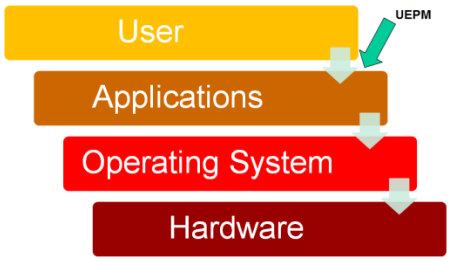

Figure.2 UEPM in a system stack.

In the traditional designs, data-width, precision of operations and storage accuracy is predetermined and fixed results can be expected for a fixed set of inputs. Approximate computing introduces imprecision in dataflow and results are obtained based on criticality of annotations in the code. Criticality of annotations in the code depends on kernels chosen to approximate, usage of approximate data and the amount of time spent in the approximate functions. As the approximation intensity in the code increases, it requires less work to compute results. Hence energy consumption reduces. However quality of the data degrades due to the errors introduced. Based on this inversely proportional relationship of quality and energy similar to that of energy-delay, we define a

generic, easy to use, easy to compute yet effective novel metric called POWSER. POWSER can be defined as a weighted dot product metric as shown below.

$$POWSER = Q_L * E \quad \text{- Equation. 1}$$

Where $Q_L = (1/Q)^{\alpha}$

Q_L = quality degradation/loss

Q = Quality

E = energy consumption

α = user-input normalized quality-energy weightage parameter, default value is 1

Historically, systems have been optimized for quality metrics such as PSNR that do not take into consideration human perception [15]. The choice of the quality metric affects evaluation of approximations and is therefore critical [12]. For image applications, we propose using BLIINDS-II as described in our experimentation strategy in Section 3. BLIINDS-II provides a quality number that correlates better with human visual system than a PSNR metric [6]. Quality 'Q' can be defined differently for each class of applications, for e.g. in image applications, Q can also account for display size distortion if the application under evaluation is expected to be used on varying screen sizes. Therefore 'Q_{Li}' for image applications would be defined as follows:

$$Q_{Li} = \mu * (Q_B)^{\alpha} \quad \text{- Equation. 2}$$

Where,

μ = display size distortion and is defined as image size/display size

Q_B = BLIINDS-II Quality Metric

Since the BLIINDS-II metric reports loss of quality number, it can be used directly to calculate Q_{Li}.

The application class specific parameters can be propagated down to compiler and hardware to operate the system in the appropriate power saving mode and thus exploit further energy savings at the micro-architecture and circuit level.

2.1 Simulation Framework

We use EnerJ that is a Java based approximate computing framework to annotate applications using approximations in the code. The simulator instruments method calls, creation and destruction of objects, arithmetic operators and memory accesses to collect statistics and inject faults [5]. The runtime system is invoked by the instrumentation calls and is implemented as a Java library. EnerJ uses a modified ISA for approximate instructions. Annotations are added for data types as well as for function iterations. The power numbers are obtained from EnerJ's energy model of power dissipation based on instruction profiling. The quality number is obtained by running BLIINDS-II which is a non-reference, natural scene statistics based image quality estimator model that provides a quality number that correlates better with human visual system than a peak signal-to-noise ratio metric [6]. The BLIINDS-II output represents a degree of deviation from the ideal image with a higher number representing higher distortion based on four features extracted from the image.

2.2 Benchmarks

For this paper we use two image benchmarks for our experiments. The first is a simple Raytracer that provides an output image by tracing the path of light through pixels in an image plane and simulating the effects of its encounters with virtual objects. The second is a Canny Edge Detector which is a popular algorithm used in the first step of edge detection in the image processing pipeline. We chose Simple Raytracer since it is representative of a typical graphic imaging algorithm. CannyEdgeDetector was chosen since type and number of edges in a picture is critical to how degradation is perceived by humans as can be seen in Figure 1.

2.2.1 Simple Raytracer

The simple Raytracer uses a checkerboard image as input. We annotate the code for four levels of approximation by varying the parameters for luminance, chrominance calculations in the RGB based model in the code. The most approximate image exhibits a power savings of ~35% with a BLIINDS-II quality degradation of 7.5 as shown by the Quality-Energy (QE) data in Table 1.

Table 1. Simple Raytracer Code Annotations Quality-Energy

Image Quality	Power (Relative %age)	Normalized Mean Pixel Difference (%)	Quality metric (BLIINDS2)
Best (Precise)	100	0	41.5
A1	91.74	0.093	45.5
A2	66.83	0.113	47
A3	66.27	0.116	47.5
Worst (A4)	65.78	0.239	49

Figure 3 shows the output images for varying code annotations for the Simple Raytracer. A white grainy deterioration of the image is observed due to approximation of data. A1, A2, A3 and A4 are approximate versions of the code.

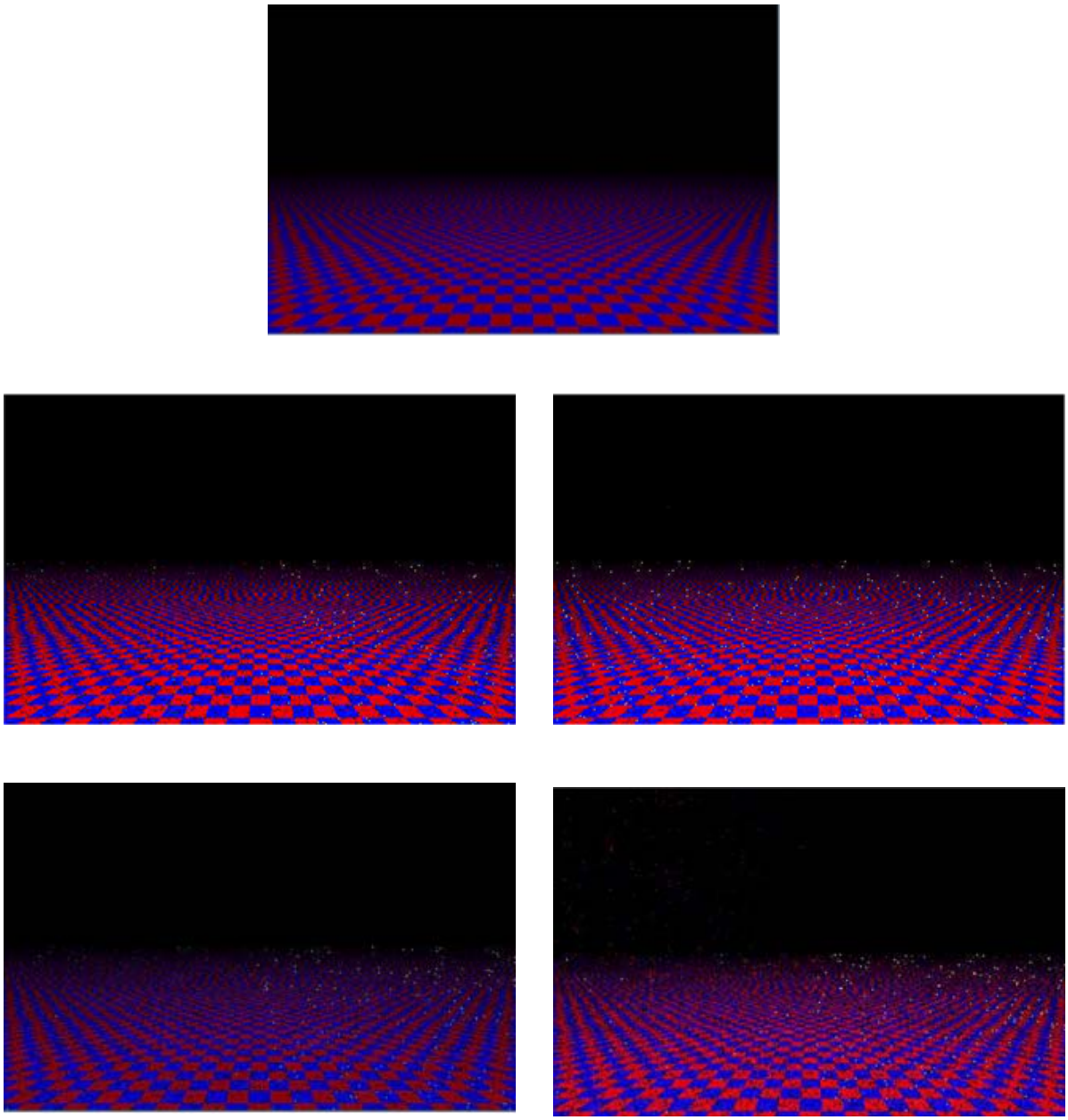

Figure 3. Simple Raytracer Code Annotated Images (Top Left to Bottom Right– Precise, A1, A2, A3, A4)

2.2.2 CannyEdgeDetector

We use CannyEdgeDetector as our second example. The CannyEdgeDetector is used as the first step for edge detection in image processing. It implements Gaussian Convolution to filter out noise in the original image. This is followed by the step to find edge strength by taking gradient of the image, determining the edge direction and relating the edge direction to a direction that can be traced in an image. Next step is to apply non maximum suppression to suppress any pixel values that are not considered an edge. This gives a thin line in the output image. Finally hysteresis is applied based on upper and lower threshold to eliminate breaking up of edges also called streaking in the output image.

The CannyEdgeDetector algorithm implementation uses parameters GaussianKernelRadius, GaussianKernelWidth, xConv, yConv for Gaussian Convolution, Highthreshold and LowThreshold for hysteresis and xGradient and yGradient are used for gradient calculations to get the thin edge line in the output image. We approximate various combinations of these parameters to analyze quality-energy trade-offs and the five annotated code versions are labeled A1, A2 and A3. The Canny Edge Detector Quality-Energy Numbers are shown in Table 2

below. The input image used is a 320x240 speed limit sign. The quality and power variance in CannyEdgeDetector is small compared to that in Simple Raytracer. The output images for various annotations from precise to most approximate are seen in Figure 4 with the less to more visible spurious edges in the picture.

Table 2. Canny Edge Detector Code Annotations Quality-Energy

Output Image Name	Power (Relative %)	Quality metric (BLIINDS-II)
Precise	100	93.5
A1	97.09	90.5
A2	93.40	81.5
A3	93.00	87.5

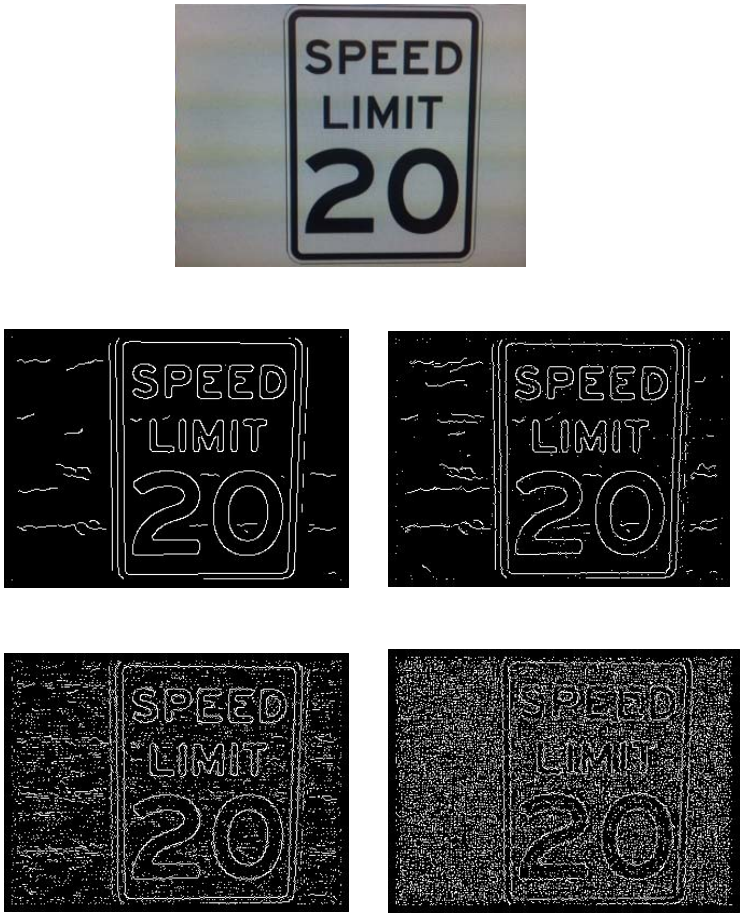

Figure 4. Canny Edge Detector Code Annotated Images (Top Left to Bottom Right – Input, Precise, A1, A2, A3)

3. RESULTS

From Equation 1, POWSER for the five approximations in Simple Raytracer is as shown below in Figure 5.

The code annotations for Raytracer done in A2 and A3 provide a set of optimal operating points for quality-energy efficiency based on POWSER and provide power savings of 34%. The annotated code can then be fed to the compiler and hardware beneath and used symbiotically with hardware power saving methods to get a UEPM optimal system. Approximate components for SRAM, FP and DRAM can be provided in hardware and software manager can route approximate computations to approximate units and precise computations to precise units.

Figure 6 suggests that code annotations for CannyEdgeDetector done in A2 is clearly an optimal operating point for quality-energy efficiency based on POWSER. Maximum power savings obtained in CannyEdgeDetector are 7.1% which is significantly lower than 35% power savings obtained in Simple Raytracer.

4. APPLICATIONS OF POWSER

As seen in SimpleRaytracer and CannyEdgeDetector experiments above, POWSER can be represented as a convex curve that can be used to establish one optimal operating point. In addition, POWSER can also provide a library of operating points for each application that can be used for online optimizations at runtime. Energy and quality can be weighted unequally per requirement of the user to get multiple sets of optimal operating points.

Consider an example of online video streaming. We captured two consecutive frames at time interval t' in a video stream. The data to be displayed between these two frames is very similar. For UEPM optimization at the application layer in the stack, we consider a H.264 video encoder using motion estimation engine [16] that stores differential data between the two consecutive frames.

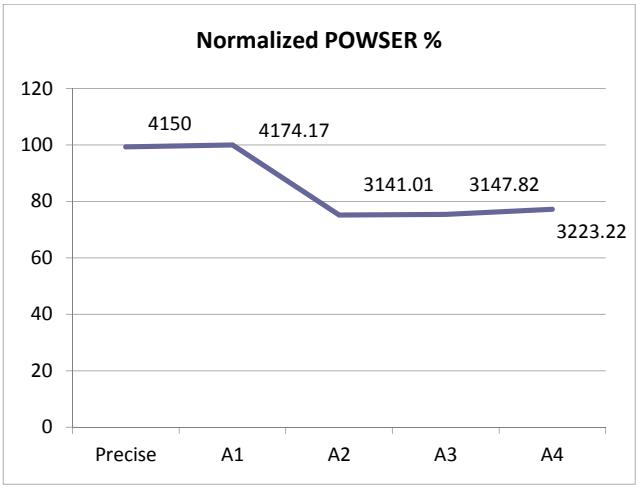

Figure 5. POWSER Q-E characteristics for Simple Raytracer

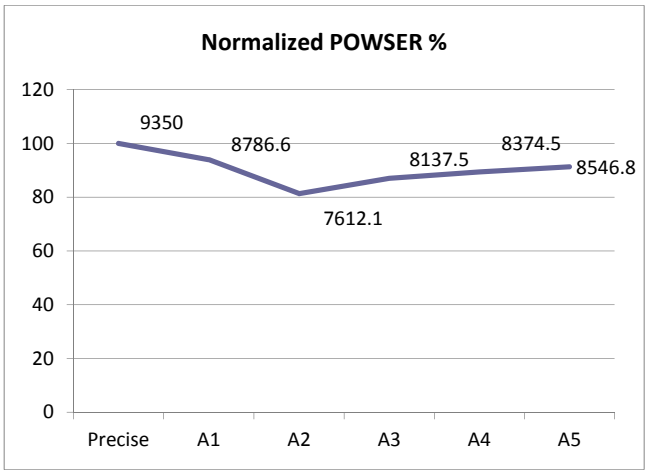

Figure 6. POWSER Q-E characteristics for CannyEdgeDetector

We suggest an algorithm as follows:

Algorithm 1:

```
//set defaults
int P = 0; //P defaults to zero to detect motion in frames
print('Enter user defined threshold value, defaults to 0')
int threshold_pixels = P; //user_defined over-ride for threshold
char display_mode = UEPM_Precise;
int[] curr_frame_pixels = UEPM_1(curr_frame);
//Fetch next frame data
int next_frame_pixels_diff  = data_from_buffer; // for differential data based ME engine
if (next_frame_pixels_diff) < P
{
        display_mode = UEPM_Approx;
        curr_frame = next_frame;
        curr_frame_pixels = UEPM_2(curr_frame);
}
//Functions
display_pixels_type  func UEPM_1 {
//Precise code
return (precise_image);
}
display_pixels_type  func UEPM_2 {
//Annotated code
return (annotated_image); // code is picked based on optimal operating point per UEPM metric
}
```

As shown in the Algorithm 1 above, the next frame to be displayed is approximated if the data differential between the current frame and next frame is less than a certain threshold value specified by user. From a user-experience perspective, it would mean that since the consecutive frames display related scenes, the next frame inherently holds better human perceivable quality [14] and we can approximate the next frame for energy savings. We propose an easy to use QE based UEPM metric called POWSER that is easy to compute yet effective. The approximate code for the next frame corresponds to the lowest operating point on the convex QE curve. Such an operating point would not be available in the individual quality or energy optimizations since the individual Q and E metrics do not follow a convex curve trend.

POWSER contains parameters in computation that can be propagated down as QoS parameters to the OS level. The QoS parameters can be used during compilation to approximate instructions based on instruction profiling for the application set. For e.g. video streaming requires blocks of data to be fetched into the block cache. Such fetches could result in load instructions in an estimated address range and can be marked as approximate. At the hardware level we consider a typical mobile SoC that would contain multiple cores, caches, memory, graphics accelerator, video encoder/decoder engine and south bridge units such as USB, UART, LCD Display, SATA, etc. POWSER can then be propagated to hardware to identify one or more functional units being approximated. Since the functional units in a hardware design are known, the QoS parameters at the hardware-software boundary can be pre-determined and listed into a routing table maintained in hardware. With a fast lookup table, POWSER can be used efficiently to implement design changes for energy savings automatically.

As an example, for image applications the units under consideration are LCD Display and Graphics Accelerator. The parameter 'μ' represents display size distortion and is propagated to the lower levels in the stack as a character. The QoS parameter is then used to index into a lookup table that marks LCD and Graphics Accelerator units for energy savings. Each of these blocks can be optimized separately in hardware for power using DVFS schemes, clock-gating, power-gating and providing alternate approximate functional units.

Thus POWSER can be used across the system stack to enable an optimal QE design point operation. Additionally since the same metric can be used across the stack, additional verification overhead for appropriate approximations at each level of the stack is avoided. Thus POWSER enables implementation of a UEPM based system with a unified objective. Also POWSER provides the ability to develop a pre-annotated library for each application. This library can be used for dynamic online optimizations of applications such as online video. We propose use of a human perceptible quality metric for calculation of quality enabling POWSER to correlate better with user experience as opposed to commonly used metrics such as PSNR or pixel errors. PSNR and pixel errors represent the absolute error present in the system but not all of the error is captured by the user as demonstrated in Figure 1. In order to trade off quality for energy, we identify the data flow and control flow in the application. We then choose variables in the data flow that can be approximated and propagated. The data widths of these variables are truncated thus introducing error. Annotation is done incrementally across all the chosen parameters and quality and energy numbers are obtained. The result is approximate data that gets propagated through the dataflow causing error to propagate as well. Control flow variables are kept as precise as possible in order to maintain correctness of the system.

POWSER thus provides a consistent, easy–to-use, easy-to-compute yet effective UEPM metric for a vertically optimized UEPM based system.

5. RELATED WORK

In the past work has been done to model and synthesize approximate adders for optimal quality-energy [7]. Quality-Energy tradeoffs have been analyzed using a PSNR metric. Our approach differs from prior work in that it incorporates user experience incorporating BLIINDS-II instead of a raw PSNR metric. This is combined with approximation at the application level for energy savings to propose a novel, easy to use yet effective consistent UEPM metric called POWSER. POWSER can be used across the system stack to build UEPM systems that have a holistic objective.

6. CONCLUSION

This paper proposes a novel, easy-to-use, easy-to-compute yet effective consistent UEPM metric called POWSER that can be used across the system stack to develop UEPM based systems with a unified objective. POWSER can also be used for fast dynamic online optimizations for applications such as video. Since our metric can be used across all layers of the stack it provides inherent checking capability and thus provides a UEPM based system with a unified objective. POWSER can also provide an optimal set of operating points for applications to enable fast dynamic online optimizations for applications such as video.

7. REFERENCES

[1] M. Pedram, Power estimation and optimization at the logic level. Int. J. High Speed Electron. Syst. 5, 2 (June), 179 –202, June 1994.

[2] J. S. Seng and D. M. Tullsen. Architecture-Level Power Optimization - What Are the Limits? Journal of Instruction-Level Parallelism 7, 7(3):1–20, January 2005.

[3] Mallik, A., J. Cosgrove, R.P. Dick, G. Memik, and P. Dinda. PICSEL: Measuring User-Perceived Performance to Control Dynamic Frequency Scaling. in the International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS-2008) 2008

[4] M. Saad and A. C. Bovik, “DCT statistics model-based blind image quality assessment,” in IEEE Int. Conf. Image Processing, Brussels, Belgium, Sep. 2011.

[5] A. Sampson, W. Dietl, E. Fortuna, D. Gnanapragasam, L. Ceze, and D. Grossman. EnerJ: Approximate data types for safe and general low-power computation. In PLDI, 2011.

[6] Saad, Michele A., Alan C. Bovik, and Christophe Charrier. "A DCT statistics-based blind image quality index." Signal Processing Letters, IEEE 17.6 (2010): 583-586

[7] J. Miao, K. He, A. Gerstlauer, M. Orshansky. "Modeling and synthesis of quality-energy optimal approximate adders." Proceedings of the International Conference on Computer-Aided Design. ACM, 2012.

[8] S. Herbert, D. Marculescu. "Analysis of dynamic voltage/frequency scaling in chip-multiprocessors." In Proc. IEEE Intl. Symp. on Low Power Electronics and Design, pp.38-43, 2007.

[9] A. Sinha, A.P.Chandrakasan. "Dynamic Power Management in Wireless Sensor Networks." IEEE Design and Test of Computers Magazine, Vol. 18, No. 2, Mar-Apr. 2001, pp.62-74.

[10] J. Cong, K. Gururaj. "Assuring Application-Level Correctness Against Soft Errors. " ICCAD, pages 150-157, 2011.

[11] Vaibhav Gupta et.al. IMPACT: Imprecise Adders for Low-Power Approximate Computing. In Proc. ISLPED, pages 409-414, 2011.

[12] V.K. Chippa et. al. Analysis and characterization of inherent application resilience for approximate computing. In Proc. DAC '13, pages 1-9, 2013.

[13] S. Venkataramani, V. K. Chippa, S. T. Chakradhar, K. Roy, A. Raghunathan, "Quality programmable vector processors for approximate computing. " In Proc. of the 46th Annual International Symposium on Microarchitecture, pages 1-12, 2013.

[14] M. Saad and A.C. Bovik, " Blind prediction of natural video quality" IEEE Transactions on Image Processing, December 2013

[15] S Winkler, P Mohandas, "The Evolution of Video Quality Measurement: From PSNR to Hybrid Metrics", IEEE Trans on Broadcasting, Vol. 54, No. 3, September 2008.

[16] L. M. Po and W.C. Ma, "A novel four-step algorithm for fast block motion estimation” IEEE Trans. Circuits Syst. Video Technol., vol. 6, pp. 313-317, June 1996.