

# Error Tolerant Multimedia Stream Processing: There's Plenty Of Room At The Top (Of The System Stack)

Yiannis Andreopoulos, *Member, IEEE*

*Invited Paper*

**Abstract**—There is a growing realization that the expected fault rates and energy dissipation stemming from increases in CMOS integration will lead to the abandonment of traditional system reliability in favor of approaches that offer reliability to hardware-induced errors across the application, runtime support, architecture, device and integrated-circuit (IC) layers. Commercial stakeholders of multimedia stream processing (MSP) applications, such as information retrieval, stream mining systems, and high-throughput image and video processing systems already feel the strain of inadequate system-level scaling and robustness under the always-increasing user demand. While such applications can tolerate certain imprecision in their results, today's MSP systems do not support a systematic way to exploit this aspect for cross-layer system resilience. However, research is currently emerging that attempts to utilize the error-tolerant nature of MSP applications for this purpose. This is achieved by modifications to all layers of the system stack, from algorithms and software to the architecture and device layer, and even the IC digital logic synthesis itself. Unlike conventional processing that aims for worst-case performance and accuracy guarantees, error-tolerant MSP attempts to provide guarantees for the expected performance and accuracy. In this paper we review recent advances in this field from an MSP and a system (layer-by-layer) perspective, and attempt to foresee some of the components of future cross-layer error-tolerant system design that may influence the multimedia and the general computing landscape within the next ten years.

**Index Terms**—error-tolerant multimedia; stochastic computing; throughput-distortion computation; new computation paradigms; cross-layer system resilience

## I. INTRODUCTION

**A**DVANCES in computer-based information processing hinge on the premise of inexpensive doubling of the processing capability of microprocessors every 18 to 24 months (Moore's law). However, today it is widely acknowledged that this is threatened by fundamental limitations of silicon-based transistor integration that lead to excessive energy dissipation and unacceptable fault rates for future microprocessors [1]–[5]. In a last strive to avoid such limitations, the microprocessor industry has extended conventional single-processor architectures to networks of processors (cores), ranging from

2~16 large cores (multicore) to 4096 small cores on a single integrated circuit die (manycore) [5]. This final attempt to sustain computational information processing advances is, however, not expected to be a panacea, as it significantly increases interconnection and programming complexity, as well as the energy consumption and fabrication costs [5]. It seems that R. P. Feynman's acclaimed "There's plenty of room at the bottom" philosophy<sup>1</sup> is now in jeopardy for silicon CMOS technology. As such, it becomes evident that we must look at applications and their exact precision and resilience requirements in the event that it shall become necessary to trade-off precision for speed, energy consumption and resilience to computational errors. Multimedia stream processing (MSP) is a particularly pertinent class of applications for trading off precision for increased system resilience and decreased utilization of system resources. This is due to the inherent error-tolerant nature of MSP applications, but also due to their significant computational and energy requirements stemming from today's high-volume data streams. Hence, by reversing the existing scaling paradigm, a growing number of researchers investigate whether there is "plenty of room at the top" of the system stack for ultra-efficient, error-tolerant, MSP in multicore, manycore and custom hardware platforms.

### A. New Research Vision for Error-tolerant MSP - An Analogy to Signal Processing for Communications

Unlike the signal processing systems area, signal processing for communications has reached a similar barrier early on: it was discovered in the early 1960's that it is not physically (or economically) viable to amplify the transmission power of a wireless or wired communication system so that the received signal would always be error free; instead, it was acknowledged that transmission errors are not aberrations, but, rather, that they are inherent to communications and must be treated as such. Thus, all communications systems today are designed to withstand certain error rates at all layers of the communication protocol stack by introducing signal compression (to reduce the required bandwidth) and redundancy (to alleviate transmission-channel impairments). The aim is to provide for graceful resilience and performance scaling according to transmission rates and the tolerated

The author is with the Electronic and Electrical Engineering Department, University College London, Roberts Building, Torrington Place, London, WC1E 7JE, Tel. +442076797303, Fax. +442073889325, Email: i.andreopoulos@ucl.ac.uk. Copyright (c) 2010 IEEE. Personal use of this material is permitted. However, permission to use this material for any other purposes must be obtained from the IEEE by sending a request to [permissions@ieee.org](mailto:permissions@ieee.org).

<sup>1</sup>R. P. Feynman, *There's Plenty of Room at the Bottom*; talk given on Dec. 29th, 1959 at the annual meeting of the American Physical Society at Caltech, Pasadena, CA.

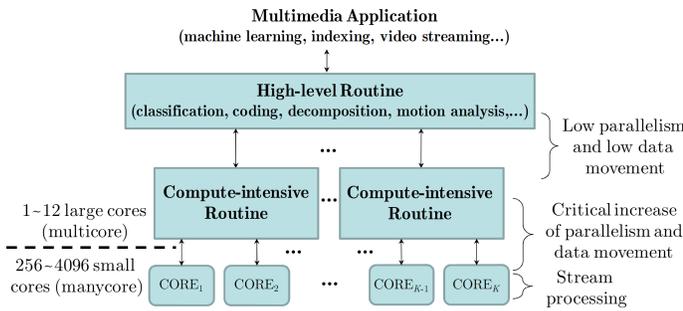


Figure 1. Multimedia stream processing applications viewed from a system perspective. Compute-intensive routines consist of digital signal processing primitives (e.g. convolution, decomposition and template matching), linear algebra routines (e.g. matrix multiplication, system inversion and matrix factorizations), data-dependent algorithms such as entropy coding or probability table updates, etc.

signal distortion. Following this analogy, perhaps it is time to consider computational multimedia stream processing as a computation channel [6], [7] or as a stochastic computing system [8], [9] optimizing for expected performance and not for the worst case. Instead of solely striving for advances in “channel quality” (i.e. increasing fault-free CMOS integration and processor operating frequencies at substantial cost and complexity), we can instead design multimedia processing in software and hardware that can withstand certain error rates at all layers of the system stack [1], [2], [4]. Similar to communications systems, the aim would be to obtain graceful and resilient approximations of the output results with increased processing (i.e. “channel”) resources. Owing to their potential error-tolerance capability, many multimedia stream processing applications currently have enormous scaling potential left unexploited [1], [6]–[8], [10], [11].

### B. Multimedia Stream Processing Systems and Applications

We define an MSP system as any system that organizes an application into streams of data inputs (such as image pixels, sensor measurements, web-page crawler results, etc.) processed by numerical linear algebra and signal processing routines. Stream processing routines (or “kernels”) are performance-critical functions that are restricted from arbitrary data accesses except within their predefined inputs and outputs [12]. A general system overview of stream processing is shown in Figure 1. As indicated in the figure, the level of parallelism in a practical implementation tends to increase for compute-intensive routines used within high-level routines. At the same time, data movement increases substantially, thereby creating a bottleneck in terms of energy consumption and cycles required for data transfer to and from processing cores. Because of this bottleneck and the fact that multimedia processing is, in general, tolerant to imprecision, MSP presents a very important class of applications amenable to resource-precision tradeoffs.

Examples of multimedia stream processing systems fitting the schematic of Figure 1 are: document retrieval engines [13], multimedia low-level or high-level analysis [14], coding and visualization [15], computer graphics, games and computer

vision algorithms [16], etc. Pictorial illustrations of how such applications fit the system overview of Figure 1 are given in Figure 2. From a computational perspective it is almost easier to list the multimedia applications that do not lend themselves to stream processing, since so many are compatible with this paradigm. Thus, it can be said that MSP systems are the backbone of our digital society: from world-wide-web indexing to voice or action recognition and mobile media, MSP systems perform the bulk of the operations required for these increasingly-complex services in real time.

### C. Error Tolerance in Multimedia Stream Processing Systems

All MSP applications aim for average error or mean squared error guarantees against ground-truth or “oracle” systems rather than worst case error. For example, all face recognition, machine learning and webpage ranking algorithms optimize for the expected recall percentage (or percentage of mis-detection/misclassification) against ground-truth results and not for the worst-case. Within commercial services (e.g. Google page ranking or image search) individual users will not notice the occasional degradation of the recall accuracy by a few percentile points, but they will notice service interruptions due to system failure or inadequate server capacity. In addition, most inputs in MSP applications stem from imperfect sampling processes (audio/visual sensors, web-crawler data, etc) and lossy signal compression performed for bandwidth reduction introduces further artifacts. Consequently, MSP systems today waste precious resources doing overly-precise calculations. This means that, in return for throughput increase (in samples, queries, or measurements per second), many, if not most, MSP applications can accept graceful degradation in their results’ accuracy under stochastic performance guarantees.

Have precision vs. computation aspects been adequately exploited so far in systems? Current high-performance multicore and manycore systems (aka “CPUs” and “GPUs”) do provide a few data types that can be used to adjust data precision (i.e. 8/16/32/64-bit integers and 16/32/64-bit floating-point representations). Moreover, within reconfigurable computing architectures based on field-programmable gate arrays (FPGAs), the wordlength can be adjusted according to the algorithm specification [17], [18]. However, the problem of allocating optimal wordlength per input data component, memory, and interconnect unit is known to be NP hard [17], [18] and one can only provide for a static configuration (and precision) during runtime [17]. Thus, while the computing architecture can be configured for a specific algorithm [18], a completely new configuration will be required for a different algorithm (or, in most cases, even for a different precision requirement). Importantly, such a configuration cannot be calculated and applied dynamically, at runtime, without significant overhead. Even in software designs, where one could perform type conversions at runtime (e.g. 32-bit float to 16-bit signed int), such conversions are prohibitively costly in SIMD (single-instruction-multiple-data) CPU or GPU processing. Thus, all general-purpose hardware and linear algebra and signal processing routines today are optimized for 32/64-bit floating-point.

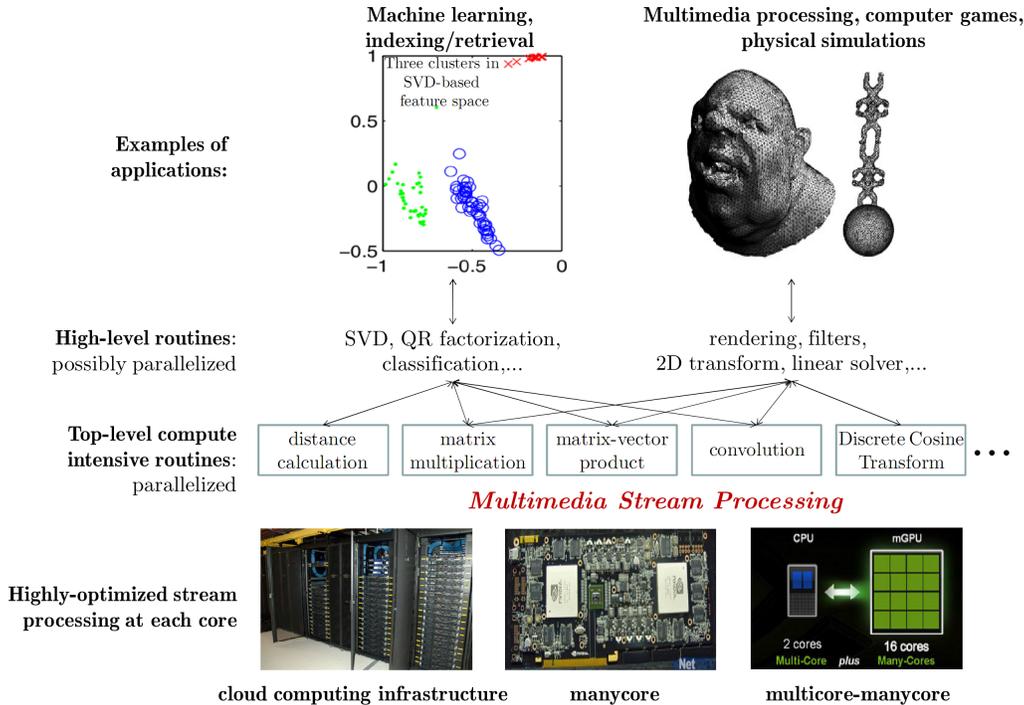


Figure 2. Multimedia stream processing applications viewed from a system perspective. Compute-intensive routines consist of digital signal processing primitives (e.g. convolution, decomposition and template matching), linear algebra routines (e.g. matrix multiplication, system inversion and matrix factorizations), data-dependent algorithms such as entropy coding or probability table updates, etc.

This lack of runtime adaptability to the algorithmic precision is now beginning to limit the scaling potential of such systems quite severely, especially in view of the “power wall” problem of CMOS technology [5], [11], [19]. Planet-level SP services like Google, Facebook, Twitter & Amazon EC2 already feel the strain of energy consumption and throughput limitation created by millions of tasks per second. Faced with a 100-fold predicted increase of processing volume within the next few years due to new applications and new users, their only avenue for handling such demand would be to utilize systematic approaches for precision/fault-tolerance/performance. This is also reflected in recent exascale computing studies, which state that, to achieve the leap from petaFLOP to exaFLOP computing (from  $10^{15}$  to  $10^{18}$  operations per second) by 2020: “The [numerical] libraries must be able to find optimal mapping of the required precision in terms of speed, precision, and energy usage.” and “Ideally, the user could specify the required precision for the result, and the algorithm would choose the best combination of precision on the local hardware in order to achieve it. The actual mechanics would be hidden from the user.” [ [19], pp. 27,31]. This critical aspect of dynamic precision adaptation within MSP is prominently highlighted in this paper, and algorithm-oriented and systems-oriented possible solutions are reviewed in Sections II and III.

#### D. Paper Organization

In this paper we survey previous and on-going efforts for error tolerance in multimedia systems with a view towards enabling future high-performance multimedia systems to circumvent the limitations of technology scaling. Section II takes an

application-oriented approach and presents a review of existing work on complexity-precision scaling and error tolerance in multimedia stream processing systems. In a complementary manner, Section III takes a system-oriented view and presents some recent and emerging research efforts<sup>2</sup> in all layers of the system stack, from high-performance software libraries to the hardware components layer. Section IV attempts to consolidate these two viewpoints by highlighting possibilities for advanced multimedia computing systems that incorporate error tolerance both for specific applications and for specific system layers in an integrated manner. Finally, Section V provides some concluding remarks.

## II. ERROR-TOLERANT MULTIMEDIA SYSTEMS FOR RESOURCE-DISTORTION ADAPTATION

In this section, we review multimedia applications and their potential for tolerance to errors by the production of approximate results, either by component adaptation (e.g. changes in the transform decomposition of a video coding system) or by system-level resource-precision optimization. We separate the discussion in three categories: (a) stream representation, analysis and coding; (b) information indexing and multimedia retrieval systems; (c) learning and recognition applications. These three classes of applications encompass a wide range of MSP systems used in practice.

<sup>2</sup>This section also highlights related papers published in the same issue as this overview paper.

### A. Stream Representation, Analysis and Coding

These are applications that make heavy use of matrix multiplication, matrix-vector products, short-length convolution, multidimensional transform decompositions, and data-dependent memory-intensive processing as their computationally-intensive kernels (see Figure 1 and Figure 2). Examples are: transform analysis and synthesis, motion estimation and compensation, entropy encoding or decoding, graphics rendering and animation [15], [16], super-resolution and construction of large 3D scenes from multiple views [20], etc.

1) *General Theory*: Pearl examined the notion of complexity for inexact computations [21] and established bounds on the minimum complexity of assigning a number of distinct calculations to a number of machines (or “computation units”) under predetermined average distortion in the result. It is shown that providing exact complexity estimates is hard as both complexity and distortion are problem-dependent. For example, in an approximate sorting algorithm that performs swaps of two elements at each step, complexity can be defined as the number of steps performed; at the same time, distortion is “*the damage incurred [to the application] by receiving the partially sorted list [...] instead of the fully sorted one*” [21]. On the other hand, when generating specified binary sequences of finite length with sequential circuits (e.g. in the case of channel decoding with the Viterbi algorithm), complexity can be defined as the number of states the machine must employ, while distortion can be defined by the Hamming distance [21]. Pearl shows that complexity bounds of such problems depend on the maximum number of possible assignments of tasks to computation units, which is lower bounded via Shannon’s rate-distortion function when computed on the input data. This indicates that, asymptotically, both the complexity of a computer program and its execution time vary according to the rate-distortion function of the input stream to be processed. This was indeed proven much later by Sow and Eleftheriadis [22] under the assumption of an elementary computing system (Turing machine). For systems aiming at signal representations, denoising and coding, the investigation of links between the rate-distortion and the complexity-distortion functions remains an active area of research [23], [24].

Given that complexity bounds for Turing machine automata under approximate results may not map well to real-world execution time complexity of a modern-day computing machine, several authors proposed theoretical analysis for specific cases of transforms decompositions in coding systems. Goyal and Vetterli [25] investigated the relationship between scalar quantization and transform decompositions in coding of Gaussian sources. Assuming signal-dependent optimal decompositions (such as the Karhunen-Loeve transform) they show that, unlike the conventional design (Figure 3), scalar quantization can precede the transform, a bitplane quantizer can be employed and, under a discrete approximation of the transform design (i.e. integer-to-integer mapping), very similar rate-distortion performance can be achieved even under the widely-used case of scalar entropy coding. Moreover, the use of an integer-to-integer transform allows for complexity reduction because the

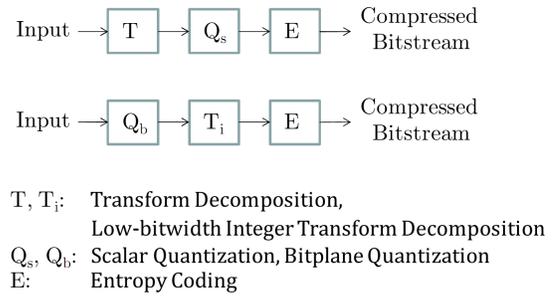


Figure 3. Top: Conventional transform coding system; Bottom: Reversal of quantization and transform leading to the utilization of a low-bitwidth integer transform and universal entropy coding.

same entropy code can be used for all transform coefficients and the wordlength used for the implementation can be reduced significantly [26].

Advanced aspects on reversing the order of quantization and transform in analysis or coding systems were also investigated independently by Nawab, Chandrakasan, *et al* [27], [28], Andreopoulos *et al* [29]–[31], and Lee *et al* [32] with respect to incremental (progressive) computation and fixed-quantization for Fourier transforms and discrete wavelet transforms computations. Discretizing the input to the transform by processing input images in a bitplane-by-bitplane manner has the advantage that it can be computed using look-up tables or small-bitwidth computations. This was proposed for the discrete cosine transforms (DCT) [33] and the discrete wavelet transform [29]. Such approaches also have the advantage that each input layer can be processed independently (thus introducing bit-level parallelism) and computational faults introduced by the hardware can be addressed in an inherently layered manner by scheduling the most significant bitplanes to the most reliable computational cores or units [33]. In addition, the required quantization and transform precision can be optimally adapted according to the energy consumption or complexity (computations) budget and the application’s precision profile [29], [32].

2) *Rate-Distortion-Complexity Modeling of Multimedia Analysis and Coding Systems*: Beyond the interplay between the quantization and the transform decomposition and its impact in the rate-distortion-complexity (R-D-C) space, other works investigated new representations that can enable scalable R-D-C models of transform systems. For Fourier analysis, Chen and Sundaram proposed the preprocessing of signals by fixed polynomial basis functions followed by weighted Fourier transforms of the pre-computer basis functions [34]. This enabled operational complexity-distortion points not achievable by conventional fast Fourier transform (FFT) computations. For R-D-C scalability of larger systems (e.g. an entire image or video encoder or decoder), it was identified early on that appropriate complexity metrics are required [35]–[38]. Rather than focusing on execution time [39], task-specific complexity metrics can be defined, such as the number of entropy coding operations, the number of motion compensated prediction operations and the number of symbol encoding or decoding operations [35]–[38] [40]–[42]. These algorithm-specific met-

rics have been termed as generic complexity metrics (GCMs) and their impact in real systems was defined as real complexity metrics (RCMs) [35]. A particular algorithm and its realization in software or hardware can be modeled with respect to its GCMs and their mapping to RCMs. GCM decomposition frameworks for the analysis of the operations performed for a particular video encoding or decoding were proposed in recent work [36]. For multi-view image coding and decoding systems, GCM-alike complexity models and their mapping to real complexity (in terms of time and memory usage) and the corresponding rate-distortion (R-D) characteristics were proposed [40], [41]. The link between complexity metrics and data flow models and directed acyclic graphs has been discussed in the overview paper of Lee *et al* [43]. Importantly, similar to previous works, the authors highlight that R-D optimization is only a subcase of the general R-D-C optimization problem of a multimedia encoder or decoder.

3) *Links to System-level Complexity Adaptation and Complexity-Distortion Scalability*: The R-D-C characteristics of representation, analysis and coding systems for multimedia streams have also been studied from a practical standpoint. Several authors proposed complexity profile measurement of the video encoders and decoders embedded in real systems with the emphasis on creating a complexity verifier for stream admission control or complexity-driven adaptation [39], [44]–[46]. These are based on systematic measurements of execution time with the usage of different parameters for transform and analysis (motion estimation/compensation, deblocking) as well as different entropy coding modes. The key idea behind these frameworks is adaptation, i.e. the flexible and dynamic (runtime) reconfiguration of a multimedia analysis or coding system. This is a very active area of research with work on dynamic voltage scaling (DVS) for multimedia systems [47]–[49], algorithmic modifications in computer graphics for complexity scalability with approximate results [15], [16], predictive energy-precision scalability [48], [50], [51] and game-theoretic parameter R-D-C adaptation of video coding systems [52]. Most research efforts in this area consider a few tasks (e.g. decoding and inverse prediction of a few video frames) and attempt to provide DVS for energy saving under worst-case based or average based cycle prediction per task. Given that multiple tasks can be performed at different precision and complexity (cycles) [47], [48], [51], R-D-C adaptation can be achieved in this manner. Through the use of linear programming for the optimal scheduling of multiple multimedia processing and decoding tasks under delay deadlines, a bound has been derived recently on the optimal scheduling under zero miss rate (i.e. all tasks finishing prior to their deadline) and an on-line algorithm achieving performance very close to the bound has been demonstrated for video coding based on multi-level motion-compensated prediction [51].

### B. Information Indexing and Multimedia Retrieval

The predominant computational kernels of this class of multimedia stream processing algorithms include matrix and vector products, singular value decomposition calculations,

linear solvers, template (or string) matching and distance metric calculation, etc. Application examples utilizing such kernels are: document clustering [53], multimedia retrieval engines (such as images/video/music/forensic-indices/metadata-based retrieval [54]), webpage ranking systems [13], etc. Given the prevalence of such systems, significant emphasis has been placed on their efficient parallelization of their computationally-intensive elements (kernels) in computer clusters or GPUs [55].

By changing the overall design perspective of the indexing and retrieval problem, distributed approaches for “approximately correct” indexing and retrieval with no single point of failure have emerged<sup>3</sup> [56]. The architecture of these approaches is inherently robust to computational errors or even system crashes: a number of machines contain partial indexing of the database of documents and they are queried randomly or pseudorandomly. This provides approximate results with the accuracy depending on the configuration of the nodes to be queried and the number of results required per query. Thus, such systems can be designed to have no single point of failure [56].

Finally, concerning multimedia retrieval systems in particular, the survey of Datta *et al* [57] points to various high-level analysis and retrieval systems that are robust to noise in the input data or in the calculated low-level feature points used for matching and retrieval processes (e.g. corner and edge points in images). Well known studies have already analyzed the resilience of low-level feature extraction to noise [58] and recent work [59]–[61] has indicated significant complexity-precision tradeoffs in feature extraction algorithms by incremental or approximate computation of their computationally-intensive kernels (transforms, distance metric calculations, matrix-vector products) in space or frequency domain. Hence, one can envisage that, subject to complexity and error tolerance constraints, systems that provide best-effort multimedia retrieval will begin to emerge in the future. Concerning audio retrieval, state-of-the-art approximate computation software designs for convolution were proposed recently [7], and their coupling with a music matching and an MPEG-7 descriptor system indicates that near three-fold improvement of processing throughput can be achieved with no effect in the precision of the analysis and retrieval process.

### C. Computationally-intensive Learning and Recognition Tasks

The dominant computation kernels of this category are the same as the ones of the previous subsection. Learning algorithms for large data sets have traditionally been known to be robust to noise in the input or processed data [62]. Thus, the scalability and robustness options provided by such systems are inherently suitable for error-tolerant designs that adapt system resources according to the desired precision. For example, game-theoretic optimization of distributed classifier chains under precision-complexity constraints was shown to achieve significant resource-precision scalability for speaker recognition from voice recordings [63]. Asanovic *et al* [64]

<sup>3</sup>See also <http://yacy.net/> for an example open-source instantiation.

and Anastasia and Andreopoulos [6] demonstrated that back-propagation learning algorithms with matrix operations are robust to significant noise levels in the performed computations. Specifically, it has been found that back-propagation learning can be implemented reliably, even when the update process is converted from floating-point to 16-bit fixed-point representation [64], or to noisy computation by companding and packing with (approximately) 24dB signal-to-noise ratio (against the single-precision floating-point results [6]). Since the latter approach is an emerging approach proposed recently, it is reviewed in more detail in the next section. Face recognition by independent component analysis [65] or 2D principal component analysis [66] was also shown to be robust to amplitude distortions [6] as long as the input image pixel's phase remains undistorted. Similarly, state-of-the-art object recognition in scale-space representations [67] was shown to be very robust even under amplitude changes and random noise insertion in the input image data that reduces the input accuracy to less than 5 bits/pixel. From an implementation perspective, this indicates that occasional memory read/write errors are not expected to affect the performance of an object or face recognition algorithm. However, errors in loop indexes that will distort the order (i.e. phase) of the input data, as well as faults in loop indexing variables, can have a severe effect in the recognition performance. This means that for learning and recognition tasks, protecting the data-memory space using error control coding (ECC) circuits is not important, as learning and recognition algorithms will scale their recognition rate gracefully in the presence of data errors. However, protection of data loop indexing and instruction memory variables is important as errors in the algorithm flow could cause severe distortions.

### III. SYSTEM-ORIENTED OVERVIEW OF ERROR-TOLERANT MULTIMEDIA STREAM PROCESSING

Given the possibilities for resource/distortion scaling presented for the broad classes of multimedia stream processing applications of the previous section, we elaborate on the different possibilities of practical system adjustment for resource scaling and resilience under approximations of outputs or the potential existence of processing errors stemming from the software, scheduling, architecture, or hardware layers. Unlike the previous section that focused on particular classes of multimedia stream processing algorithms, here we summarize research advances in system layers, from the software layer to the architecture, device and IC layers.

#### A. Advances in Approximate (Precision-aware) Multimedia Stream Processing Software Components

In the software layer, the de-facto standard libraries for high-performance MSP today are the Basic Linear Algebra Subprograms (BLAS), the Linear Algebra Package (LAPACK) and digital signal processing or computer vision libraries. Several optimized designs exist for these, focusing on single-instruction-multiple-data architectures (e.g. NAG, Goto [68], ATLAS, Intel IPP, OpenCV, Matlab, and AMD ACML), manycore platforms (e.g. Matlab Parallel Computing Toolbox,

PLASMA/MAGMA [69]), and embedded processors (e.g. Eigen for ARM Neon).

All high-performance realizations of MSP algorithms try to make maximum usage of computational kernels of such libraries to ensure their realization remains as modular and highly-optimized as possible. This modularization also allows for automated design flows that can optimally select hardware resources from a multi-component architecture for data-dominated MSP algorithm flows [17], [18] [70]. Finally, the importance of this modularization of algorithmic components into high-performance computational kernels of software libraries is demonstrated by the recent definition of the reconfigurable video coding specification [71] within MPEG video coding standards.

We first review conventional acceleration and fault-tolerance approaches and then present some emerging ideas on throughput-distortion scaling of computation for linear algebra operations that form the compute- and memory-intensive software kernels (primitives) of the error-tolerant MSP applications of the previous section.

1) *Acceleration Techniques and Conventional Fault Tolerance* : Acceleration techniques for these libraries generally follow two different approaches: tailoring and customizing the computational routines to particular systems (e.g. to exploit problem-specific sparsity for sparse linear solvers) [72], or exploiting mixed precision for accelerated processing in certain computations, e.g. for iterative linear solvers [73]. In both cases, notable accelerations have been reported via the use of manycore platforms, hybrid multicore/manycore systems, or reconfigurable systems [18]. State-of-the-art designs today use Streaming SIMD Extensions (SSE) for multicore processors and automated parallelization within the compilation and scheduling, e.g. via the CUDA and OpenCL frameworks.

In terms of tolerance to hardware-induced errors, conventional fault tolerance techniques follow ECC approaches in software [74], [75] or hardware [55], [76]. These approaches create redundant computations composed of mixtures of the existing input data in a linear operation (e.g. matrix product or convolution) and form checkpoints from the redundant computations in order to check the useful results for correctness using parity bits or Hamming codes. They generally incur 70~150% performance penalty in terms of the achieved giga floating-point operations per second (GFLOPS) [75], [76] and they can only detect and correct a limited number of errors per computation, typically one or two errors per input substream of data. Complementary to such general-purpose fault-tolerant approaches, error tolerance in software or hardware for MSP was proposed by Breuer, Ortega *et al* [77]–[79] by exploiting the fact that multimedia operations such as motion estimation and filtering tend to create localized errors that can be masked from the user if detected or treated properly. As an example, Chung and Ortega [78] analyzed the motion estimation of an MPEG encoder based on an RTL implementation of the algorithm and concluded that most hardware-induced faults either did not create an error in the output or they only led to bandwidth increases due to erroneous motion vector estimation or redundant frame transmissions.

While these approaches for acceleration and error tolerance

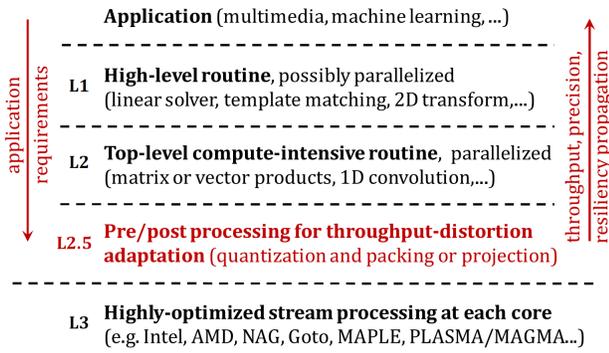


Figure 4. General framework of throughput-distortion adaptation in MSP applications utilizing high-performance linear algebra and digital signal processing software libraries [6], [7].

are certainly encouraging, they still fail to capture the massive parallelism offered by the hardware platforms themselves or, for mixed-precision methods, the speed-vs.-accuracy trade-off between single and double-precision floating point. In fact, it is becoming all the more evident that massively parallel architectures today are significantly underutilized [80]. To quantify this, one can assess the efficiency of a particular computation via the peak performance figure of merit. Peak performance can be defined as the ratio between the practically-obtained processing throughput over the maximum-possible processing throughput for a particular SP algorithm on a given platform. While well-known linear algebra libraries today achieve over 0.9 (or 90%) peak performance in single cores [68], this figure is significantly lower for multicore and manycore processors. For example, the latest performance figures for manycore processors from NVIDIA indicate peak performance of 0.52 (52%) or less for basic matrix multiplication routines [80]. This means that (at least) 48% of the available processing cycles' budget is wasted in data I/O between cores and memory, cache misses and stall cycles. This substantial drop of peak performance illustrates the difficulty in exploiting multicore and manycore processors to their limit.

2) *Throughput-Distortion Computation via Adaptive Companding and Packing* : Ideas have begun to emerge recently on precision-adaptive MSP with respect to throughput-distortion scaling of linear algebra and signal processing operations [6], [7], [81]–[85]. Figure 4 illustrates how such approaches fit within the system overview of MSP applications of Figure 1. The key principle of such frameworks comprises two steps:

- Compaction of inputs by quantization and packing or projection, prior to stream processing at each core. In this way the core operation remains uninterrupted and any high-performance software or hardware library can be used for the realization of the core processing.
- Extraction of the computed results to derive meaningful approximations of the outputs.

Using these approaches, the precision can be controlled according to the desired acceleration. Since the process can be performed in software and does not break the regular data access patterns or the memory-access locality of each operation, all high-performance primitives of multicore/manycore

processors (e.g. SSE instructions and automated parallelization and scheduling in CUDA and OpenCL) can be used. Thus, significant acceleration or energy scaling can be obtained over state-of-the-art MSP realizations with processing throughput that can significantly surpass 100% of the peak performance of a given platform with graceful degradation of precision. In the remainder of this subsection we summarize recent advances on quantized-and-packed linear processing [6], [7], [81]–[84]. The reader is also referred to the work of Borland and Constantinides in this issue that focuses on scalable precision analysis and control [85] for custom hardware realizations.

Packed linear image processing hinges on the idea that the dynamic range of a 32-bit or 64-bit numerical representation can be used for the concurrent calculation of multiple small-dynamic-range integer operations if the operands are positioned (or “packed”) in such numerical representation with appropriate spacing from each other [81], [82]. This has been proposed for a variety of image processing operations such as bound estimation, image cross-correlation and orientation correlation [81], [82], incremental image convolution and motion estimation [30], integer block-transform decomposition [83] and integer wavelet transforms [84]. If the linear operations are not mapping integers to integers, quantization can be applied prior to packing [6], [7] which creates an inherent throughput-distortion tradeoff in the performed linear operations, as explained in the following.

Consider a linear operation  $\text{op}$  that can be applied to  $M$  input data (e.g. image, video, audio, etc.) blocks  $\mathbf{B}_m$  concurrently<sup>4</sup> (with  $m \in \{0, \dots, M-1\}$ ,  $M \geq 2$ ), using operator matrix  $\mathbf{K}$ :

$$\mathbf{U}_m = \mathbf{B}_m \text{op} \mathbf{K}. \quad (1)$$

This can be a block transform decomposition/reconstruction, or a convolution/cross-correlation operation using processing kernel  $\mathbf{K}$  [81], [82]. In the general case of quantized (approximate) processing via packing [6], [7], the first step is to perform companding, e.g.

$$\tilde{\mathbf{B}}_m = \llbracket c_{\mathbf{B}} \mathbf{B}_m \rrbracket, \quad \tilde{\mathbf{K}} = \llbracket c_{\mathbf{K}} \mathbf{K} \rrbracket, \quad (2)$$

where  $c_{\mathbf{B}}$ ,  $c_{\mathbf{K}}$  are the companding coefficients (determined based on the precision requirements of the application in question [6], [7]) and  $\llbracket a \rrbracket$  rounds  $a$  to the nearest integer. Operational<sup>5</sup> packing then forms a single block  $\mathbf{D}$  by:

$$\mathbf{D} = \sum_{m=0}^{M-1} \tilde{\mathbf{B}}_m \varepsilon^m, \quad (3)$$

with  $\varepsilon > 0$  an appropriate packing coefficient. Concurrent processing of multiple inputs then takes place by

$$\mathbf{R} = (\mathbf{D} \text{op} \tilde{\mathbf{K}}). \quad (4)$$

Considering the use of an operational real-number representation, such as single or double-precision floating point, the

<sup>4</sup>The  $M$  input blocks can be parts of different images (or other media) that are processed concurrently, or parts of the same image.

<sup>5</sup>The term *operational* refers to an algorithm or representation realizable by a computer.

results can be unpacked sequentially [81], [82]. First, all packed results are shifted to the non-negative region of zero by:

$$\mathbf{R}^+ = \mathbf{R} - L_{\min} \cdot \mathbf{J} \quad (5)$$

with  $L_{\min} = A_{\min} \sum_{m=0}^{M-1} \varepsilon^m$ ,  $A_{\min}$  the minimum possible value of the results<sup>6</sup> of (1) and  $\mathbf{J}$  the unit matrix (matrix of ones). Each result is subsequently unpacked iteratively from  $\mathbf{R}^+$  by the following:

$$m = 0: \mathbf{R}_{\{0\}}^+ = \mathbf{R}^+, \mathbf{U}_{\{0\}}^+ = \lfloor \mathbf{R}_{\{0\}}^+ \rfloor, \quad (6)$$

$$\forall m \in \{1, \dots, M-1\}: \begin{cases} \mathbf{R}_{\{m\}}^+ = \frac{1}{\varepsilon} (\mathbf{R}_{\{m-1\}}^+ - \mathbf{U}_{\{m-1\}}^+), \\ \mathbf{U}_{\{m\}}^+ = \lfloor \mathbf{R}_{\{m\}}^+ \rfloor \end{cases} \quad (7)$$

where  $\mathbf{R}_{\{m\}}^+$  indicates the contents of  $\mathbf{R}^+$  during the  $m$ th unpacking and  $\lfloor a \rfloor$  the largest integer smaller or equal to  $a$ . Finally, the results are derived from  $\mathbf{U}_{\{0\}}^+, \dots, \mathbf{U}_{\{M-1\}}^+$  by offsetting to their original range and performing inverse companding by (dequantization):

$$\forall m \in \{1, \dots, M-1\}: \tilde{\mathbf{U}}_{\{m\}} = c_{\mathbf{BK}, \text{op}} (\mathbf{U}_{\{m\}}^+ + A_{\min}) \quad (8)$$

with  $c_{\mathbf{BK}, \text{op}}$  the inverse companding operator determined based on the specifics of the linear operation performed. For example, for the case of matrix multiplication [6] or convolution [7], i.e.  $\mathbf{U}_m = (\mathbf{B}_m \mathbf{K})$  or  $\mathbf{U}_m = (\mathbf{B}_m * \mathbf{K})$  respectively, we have (for both):  $c_{\mathbf{BK}, \text{op}} = (c_{\mathbf{B}} c_{\mathbf{K}})^{-1}$ . The execution time reduction for increasing distortion<sup>7</sup> stems from increased values of  $M$  (which in turn is controlled by the companding coefficients  $c_{\mathbf{B}}$  and  $c_{\mathbf{K}}$ ), as more results are calculated concurrently – albeit at lower precision [6], [81]. Finally, while the packing approach illustrated by (3)-(7) packs only the input blocks  $\tilde{\mathbf{B}}_m$ , this has been extended recently to companding and packing of both operands  $\tilde{\mathbf{B}}_m$ , and  $\tilde{\mathbf{K}}$  [6], [7].

A conceptual illustration of how the floating-point representation noise affects the quantized-and-packed results of (4) is given in Figure 5. This noise is significantly amplified in packed representations as the “lower” side result (multiplied by  $\varepsilon = 0.0001$ ) is in the decimal part of the number. This representation noise creates the notion of computational capacity in this approach [6]: for given quantization distortion, there is a limit on the throughput increase achieved via increased packing (i.e. increased values for  $M$ , surpassing  $M = 2$  shown in Figure 5), beyond which the distortion stemming from the floating-point

<sup>6</sup>The minimum and maximum possible values of the output can be calculated *a-priori* for given op and  $\mathbf{K}$ , under the known dynamic range of the input.

<sup>7</sup>where distortion is defined in the root sum-squared-error sense  $\|\mathbf{U}_m - \mathbf{U}_m\|_2$

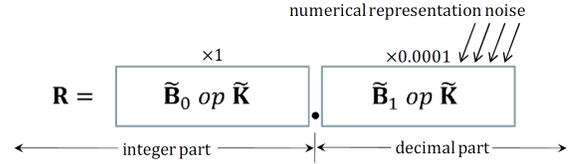


Figure 5. Conceptual example of the result of (4), with  $M = 2$  packings and  $\varepsilon = 0.0001$ .

computation<sup>8</sup> surpasses the companding-induced distortion<sup>8</sup>. The interdependency between throughput and distortion and the notion of computational capacity make this approach a computation channel for linear signal processing operations.

*Summary of results:* Recent experiments [6] demonstrated practical accelerations of up to 175% for approximate linear convolution operations following the approach summarized here, even against state-of-the-art realizations such as the Intel IPP convolution library. Within the MSP frameworks of a music matching system [54] and an MPEG-7 metadata calculation from audio streams, it was demonstrated that this leads to virtually no effect on the applications’ precision [6]. Similarly, other experiments [7] show that the peak performance of the generic matrix multiplication (GEMM) routine of BLAS achieved on a multicore processor can be increased by up to 92% in comparison to the state-of-the-art double-precision GEMM routine of the Goto library [68]. This leads to peak performance of almost 180% for double-precision matrix multiplication under approximate results [6]. When tested within a face recognition [66] and a metadata-based analysis system for music recordings, this approach demonstrated that the accuracy of the system remains virtually unaffected [6]. Such substantial gains in throughput can be exchanged for fault tolerance using the well-known methods outlined in Subsection III.A.1. In terms of theoretical results, it was shown that an optimal coupling of the quantization-induced and representation-induced (floating point) noise exists for matrix multiplication operations of independent identically distributed input sources [6]; this results in the maximum processing throughput under a predefined distortion for the output results (in SNR, versus the full-precision results). It is expected that such a result could be extended to broader classes of linear algebra operations.

### B. Multicore and Manycore Scheduling and Runtime Support for Multimedia Stream Processing

From the operating system’s and runtime support’s perspective, it is generally acknowledged [1]–[5], [86]–[88] that the critical issues for the execution environment of Figure 1 are:

- estimating the MSP time per core (of thread) for efficient task allocation in a multicore/manycore environment;

<sup>8</sup>Or, equivalently: decreasing the companding distortion (by increasing the companders) leads to increased noise stemming from the floating-point computations, as more space is needed to pack the quantized inputs and results (i.e.  $\varepsilon$  in Figure 5 becomes smaller); thus, for each packing (i.e. acceleration)  $M$ , there is a limit on the quantization accuracy, beyond which the distortion stemming from floating-point computation surpasses the quantization distortion.

- increasing job concurrency for MSP, and
- allowing for error tolerance and graceful degradation under transient errors or core failures.

All three aspects can be assisted by a multicore/manycore scheduling approach that encompasses the error-tolerant nature of MSP applications and their software components. Conventional synchronous data flow (SDF) models [89] approach the scheduling problem of multiple subtasks in multiple cores as seen in Figure 6(a). Each subtask of the three MSP tasks  $a_1$ ,  $a_2$ ,  $a_3$ , requiring an anticipated number of cycles  $c_{i,j}$  [with  $1 \leq i \leq K$  and  $1 \leq j \leq 3$  in the example of Figure 6(a)], is statically scheduled in one of the  $K$  available cores by, for example, following round-robin scheduling, or allocation to the least-recently-used core. Each subtask's result is returned by each of the  $K$  cores in the output data streams at time instants  $t_{i,j}$ . Synchronization and combination of the subtasks takes place before returning the final three results to the higher-level function. For example, the input subtasks could be texture, motion-vector, and audio decoding for a video decoding application. In this case, the output results will need to be synchronized and combined together to produce the three decoded audio and video frames  $a_1$ ,  $a_2$ ,  $a_3$  before being flushed to the video player thread for display.

Conventional SDF scheduling for multi-stream processing can be extended to decrease inter-thread dependencies (e.g. by duplicating some data structures accessed by multiple threads), to replicate tasks of a different core if the current core finished its execution flow and remains idle (for fault tolerance purposes), to increase cache efficiency by increasing data locality (e.g. stripe-based processing in video frames), etc. The work of Li *et al* [90] comprises a good academic overview of such techniques within the context of video analysis and data mining applications. Moreover, the MapReduce system proposed by Google [88] provides the most prevalent practical system exemplifying such distributed, fault-tolerant scheduling. MapReduce performs massively parallel stream processing tasks over computing clusters by splitting each input stream into multiple pieces, assigning multiple map and reduce tasks to individual workers (that are copies of the algorithm to be performed on each piece) and producing the results of each task into distinct output files [88]. The map functionality maintains the key/value pairs that need to be maintained in memory in order for the compilation of the final results to be successful. The crucial aspects that make MapReduce interesting are the massive scaling potential and the fact that it hides the details of parallelization, fault tolerance, locality and load optimization from the user, thus making it easy to use [88]. The inherently redundant and distributed nature of MapReduce allows for robustness to unequal load balancing and network interruptions, as multiple copies and multiple key/value pairs are kept in different parts of the computing cluster. This also makes the overall computation immune to sporadic core failures.

Whether for a single computing cluster or for a set of clusters, error-tolerant scheduling can encompass a significantly-higher number of possible cycles' budgets, characterized by vectors  $\mathbf{c}_{i,j}$ , which are coupled with the corresponding expected distortion impact (indicated by vectors  $\mathbf{d}_{i,j}$ ) caused in

the particular MSP element when the core processing utilizes a different cycle count. The variability in cycles-distortion operational points can occur due to: (i) hardware-induced errors; (ii) the error-tolerance capabilities of the application itself; (iii) the throughput-distortion scaling capabilities allowed by the signal processing or linear algebra computational kernel as illustrated in the previous subsection. This error-tolerant scheduling scenario is shown in Figure 6(b). Depending on the particular core configuration and the utilized cycles, the resulting output data streams are produced at times  $t'_{i,j}$  (that will be different from the original times  $t_{i,j}$ ) and will be synchronized and combined together to return the final three results  $\hat{a}_1$ ,  $\hat{a}_2$ ,  $\hat{a}_3$  (which are potentially approximate, i.e.  $\hat{a}_j \neq a_j$ ) to the higher-level MSP routine.

Such error-tolerant scheduling creates a significantly-larger exploration space where resilience-distortion-complexity tradeoffs can be formed based on the adopted scheduling approach. We summarize here some of the recent work in this area. Liu *et al* investigated fault-generating processors under scheduling with multiple possible voltage (frequency) levels for each core [91]. It is shown that, despite being an NP-complete problem, a near optimal scheduling solution (in terms of minimum energy efficiency and maximum resiliency to transient hardware errors) can be accomplished in polynomial time. In this issue, Mastronarde *et al* [87] propose a dynamic scheduling solution based on the formulation of the scheduling problem as a Markov decision process. Validation is performed by simulations with a multi-processor architecture that allows for dynamic power scaling. Leem *et al* [92] propose the error-resilient system architecture (ERSA) framework for high error resiliency to high-order bit errors in inputs of probabilistic clustering aimed at recognition and mining systems. Under robust scheduling in a multicore environment, it is shown that ERSA retains high accuracy even under very high error rates for the input streams. Subramanyan *et al* [93] propose error-tolerant scheduling by extending the conventional concept of redundant thread execution to include feedback from the leading thread to the trailing thread. This means that the high parallelization possible in modern multicore and manycore processor environments is used more efficiently since the overall execution of the two threads remains robust to transient errors while the execution time is accelerated due to the collaborative exchange of messages between the two threads executing the same MSP algorithm. Liu *et al* [89] propose to extend the error-tolerant SDF model of Figure 6(b) to include reconfiguration at the core level itself, based on FPGA realizations. For correlated multi-stream processing, significant improvement in throughput per unit-area of hardware is demonstrated against the static SDF scheduling of Figure 6(a). Finally, Anastasia and Andreopoulos [31] demonstrate energy-distortion tradeoffs by combining incremental computation with time-driven execution via a control thread that terminates the execution of the multimedia processing thread once the allocated time (or number of cycles) has been surpassed. Due to the inherent robustness of incremental computation, graceful degradation is achieved in the output results even under aggressive scheduler-driven

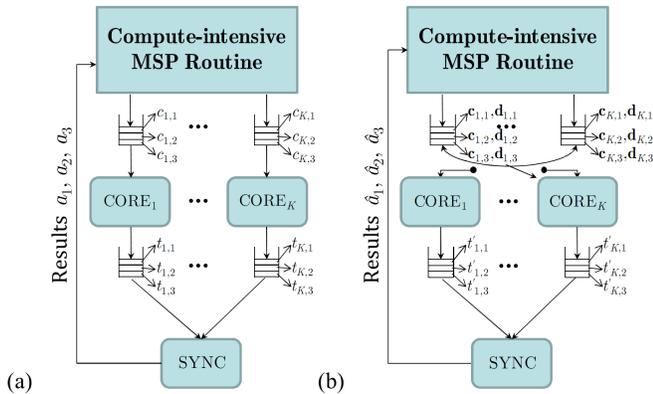


Figure 6. Task concurrency and synchronization for the production of three output results from three groups of  $K$  streamed inputs (subtasks), following the hierarchy of MSP system components of Figure 1; (a) Conventional SDF model with fixed job cycles; (b) Error-tolerant adaptive scheduling with multiple cycles-distortion pairs.

termination of the execution with no performance penalty against the conventional (non-incremental) computation of the MSP algorithm.

### C. Error-tolerant Architecture, Hardware Components and Cross-layer Reliability

Within system-level design, all conventional CMOS-based technologies ensure that processor chips remain fault-free during their lifetime. This has shaped the commonly accepted notion of “digital” in applications as being “lossless”, despite the fact that: (i) all practical multimedia inputs (such as image/video samples or other sensor-derived data) are approximations of the physical reality; (ii) perfect reliability (and bit-level reproducibility of linear algebra routines) now comes at considerable cost (chip designers today have highly-complex fault detection mechanisms in their design and fabrication flows [55]); and (iii) the fundamental scaling limitations of CMOS technology will lead to significantly-increased fault rates for SRAM and latch circuits below 22nm [86]. For these reasons, an emerging field of cross-layer system reliability research [1] is now gaining significant traction. This concept distinguishes between error-tolerant and error-intolerant applications (e.g. safety-critical applications). Errors can be controlled at different layers of the system stack [94], [95]. For example, error detection can be entirely be implemented in hardware while error recovery can done within a combined hardware/architecture/application approach [95]. This has the advantage of leveraging the energy and cycles’ overhead for the detection and correction of hardware-induced errors between the application, architecture and hardware component layers. In such cases, chip-level or component-level error analysis and detection techniques, such as the important work of Nicolaidis on soft error mitigation [96], [97], can be complemented with recovery at the architecture and software level. Since it may be prohibitively expensive to correct all errors, or many errors may even have a benign effect on the multimedia application [78], [95] or scheduling [93] (as indicated by the previous sections), it becomes imperative

to create an intermediate control plane where error reporting and mitigation decisions can take place. Under this scenario, the error tolerance capability of a system extends beyond a simple yes/no answer to whether a system can completely recover from any combination of  $F$  errors within a set of interconnected hardware components [95]. Rather, designers are interested to know “how often” and “under what scenarios” a system is not protected from errors and, in the case of MSP systems, what is the impact of the erroneous computations on the output multimedia streams.

Such a view of cross-layer resilience for system-on-chip (SoC) design for throughput-energy-distortion (T-E-D) trade-offs in MSP applications is shown in Figure 7 (based on Carter *et al* [95]). The key differences between this stack and the conventional system stack are: (i) the communication of expected application performance (in terms of T-E-D) from the application to the operating system or scheduler; (ii) the assumption of a cross-layer interface (CLI), positioned at the SoC architecture layer, that propagates information on the detected hardware errors (and possible diagnostics and mitigation techniques) from the lower layers (circuit and device layer) to the higher layers (operating system and the application itself); (iii) the ability to reconfigure hardware components and to mitigate hardware failures in order to meet the expected T-E-D performance. While still at a very early stage of development, such approaches have shown initial promise for error-tolerant systems [1]–[4], [86], [94], [95].

A critical component of any CLI-based framework is the error mitigation at the IP block and circuit and device layers. To this end, previous work has focused on different techniques, such as Markov random fields [98], stochastic processors [99], stochastic logic [100], Razor [101], and space-time redundancy techniques at the integrated circuit level [97]. Within this issue, we refer the reader to two new contributions on error control and error mitigation techniques [102], [103], ranging from cross-layer techniques for protection of multimedia applications from soft errors, to reliable information processing under unreliable hardware via detection and estimation techniques. Most of these techniques aim for deterministic detection of “erroneous” outputs under the expectation of certain rates of hardware-induced errors. This can be considered wasteful for multimedia applications where the inputs are inherently noisy and the targeted result of the application is a stochastic metric, e.g. maximization of the expected signal-to-noise-ratio for a coding system or the expected recall rate for a multimedia retrieval application. A few notable exceptions that have emerged in the last few years are: (i) the notion of stochastic processors [8], [9], [99], (ii) variable-precision hardware [104] or inexact design [105] and (iii) stochastic logic [100]. In the first case, the hardware is deliberately under-designed or used beyond its safety margin via techniques such as voltage over-scaling in order to produce occasional errors that are software-correctable. In the second case, arithmetic hardware (typically multiply-accumulate units) is designed to have multiple precision levels [104]; alternatively, pruning techniques (using heuristics) are derived in order to remove computational units (blocks) according to their expected impact in the precision of the results [105]. In this way, errors affect the multimedia

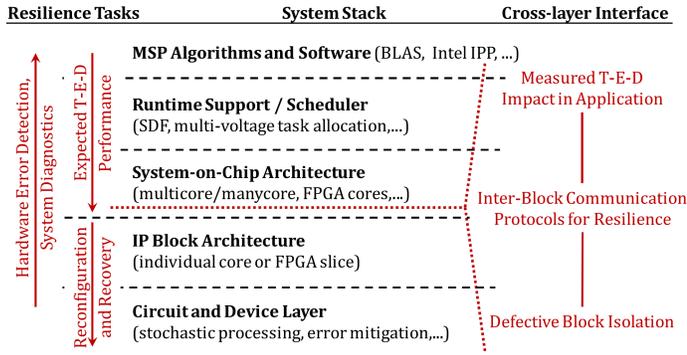


Figure 7. A view of cross-layer resilience for System-on-Chip design focusing on throughput-energy-distortion (T-E-D) tradeoffs of MSP applications with a cross-layer interface positioned at the SoC layer [94].

application in a controllable manner. In the last case, the input is essentially represented by its probability encoding (probability of appearance of a certain bit sequence) and the resulting data-path synthesis for numerical computations is altered to compute the probabilistic encoding of the output under a given processing flow [100]. Such approaches form a significant departure from conventional digital logic synthesis and are particularly suitable for error-tolerant applications like linear or polynomial computations (e.g. convolution) that are common in multimedia signal processing.

#### IV. A UNIFIED VIEW OF ERROR-RESILIENT MULTIMEDIA STREAM PROCESSING

Figure 8 presents a speculation of how the different components identified for error tolerance and cross-layer mitigation of hardware-induced errors could be consolidated in future MSP system designs. A mixture of algorithmic, hardware and integration is expected to be required in order to create seamless throughput-energy-distortion scalability in future error-tolerant multimedia stream processing that may operate beyond the limitations of fault-free CMOS designs. Starting from algorithms for multimedia processing, several approaches such as the ones outlined in Section II and Section III.A could offer several advances (resource-distortion adaptation, approximate MSP and result recovery under faults). Resilient scheduling approaches and mitigation of core failures can incorporate some of the new approaches outlined in Section III.B. Within the architecture and the device layer, resilient memory designs with graceful degradation under hardware-induced errors (via layers of ECC protection) can offer prioritization for the components that need high levels of resilience to soft errors while allowing for voltage overscaling techniques to cause soft errors on memory components that do not store critically important data to the MSP application. Stochastic logic at the IC layer or stochastic computation can become a viable alternative for logic design flow in error-tolerant applications, especially aiming for MSP applications with average error guarantees, such as multimedia analysis, indexing and retrieval systems. Finally, all such components can function within a unified control plane comprising a cross-layer interface for T-E-D scalability and hardware error mitigation.

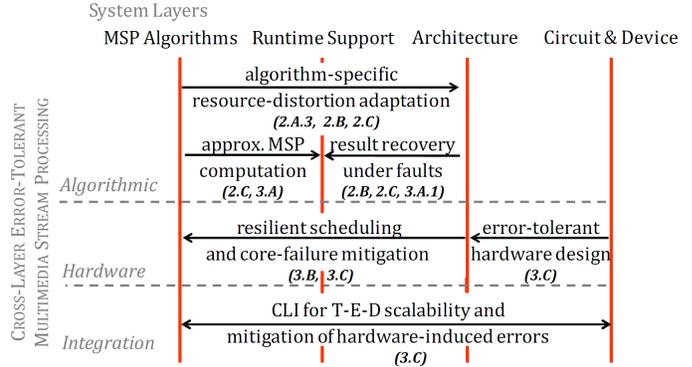


Figure 8. Possible integration of various components at different layers of the system stack for error-tolerant multimedia stream processing beyond the limitations of CMOS technology scaling. The numbers in parentheses refer to the section of the paper where related research is discussed.

Unlike the application-specific or hardware-specific techniques proposed today for error tolerant computing, the integration of components illustrated in Figure 8 comprises an approach oriented towards system layers. Within each layer, standard components can be designed in a scalable and resilient manner, similar to the OSI protocol stack in communications. Allowing for seamless parameter exchange and reconfiguration in this manner may provide very significant technology scaling to levels unfathomable under the rigid, worst-case oriented, design flow of existing systems.

#### V. CONCLUSION

Similar to MMX/SSE/AVX instructions for single-input-multiple-data (SIMD) processing (1996-present) and Graphics Processing Units (GPUs, 1999-present), it is possible that the next revolution in stream processing will also be inspired by multimedia computing systems: it may center on adaptively exploiting precision and noise in the input data streams for ultra-high performance throughput and energy scaling under approximate results. We have reviewed several approaches for error-tolerance and application-oriented rate-distortion-complexity scalability and then presented a summary of different approaches for error-resilient components across the different layers of the system stack. These approaches indicate that there seems to be substantial capability for error tolerance within several classes of multimedia processing algorithms. This capability can bring important gains in throughput and resilience of practical implementations in future generations of CMOS-based integrated processors that may incur errors from the architecture, device and IC layers of the system stack. This throughput/distortion scaling potential, stemming from the error-tolerant nature of multimedia stream processing algorithms, is labeled in this paper as “plenty of room at the top of the system stack”. The system-layer oriented summary of different approaches presented in Figure 8 indicates a possible stack of error-resilient system components that could be used in future multimedia stream processing applications offering throughput-energy-distortion scaling beyond what is possible today. Addressing each system layer and providing components that offer error tolerance with controls that can

be tuned for resource-distortion tradeoffs within a (potentially) noisy computing environment appears to be a very significant research challenge for the multimedia systems community in the next 10 years.

## REFERENCES

- [1] A. DeHon, N. P. Carter and H. Quinn (Eds), "CCC cross-layer reliability visioning study," Computing Community Consortium, Tech. Rep. [Online]. Available: <http://www.relxlayer.org>
- [2] A. DeHon, H. Quinn, and N. Carter, "Vision for cross-layer optimization to address the dual challenges of energy and reliability," in *Proc. Design, Automat. & Test in Europe Conf. & Expo., DATE'10*, 2010, pp. 1017–1022.
- [3] H. Esmaeilzadeh, E. Blem, R. St Amant, K. Sankaralingam, and D. Burger, "Dark silicon and the end of multicore scaling," in *Proc. 38th Annual Int. Symp. on Comput. Arch., ISCA'11*, 2011, pp. 365–376.
- [4] S. Mitra, K. Brelsford, and P. Sanda, "Cross-layer resilience challenges: Metrics and optimization," in *Proc. Design, Automat. & Test in Europe Conf. & Expo., DATE'10*, 2010, pp. 1029–1034.
- [5] D. Patterson, "The trouble with multi-core," *IEEE Spectrum*, vol. 47, no. 7, pp. 28–32, Jul. 2010.
- [6] D. Anastasia and Y. Andreopoulos, "Throughput-distortion computation of generic matrix multiplication: Toward a computation channel for digital signal processing systems," *IEEE Trans. on Signal Process.*, vol. 60, no. 4, pp. 2024–2037, Apr. 2011.
- [7] M. Anam and Y. Andreopoulos, "Throughput scaling of convolution for error-tolerant multimedia applications," *IEEE Trans. on Multimedia*, vol. 14, no. 2, pp. 797–804, Jun. 2012.
- [8] N. Shanbhag, R. Abdallah, R. Kumar, and D. Jones, "Stochastic computation," in *Proc. Design, Automat. & Test in Europe Conf. & Expo., DATE'10*, 2010, pp. 859–864.
- [9] S. Narayanan, J. Sartori, R. Kumar, and D. Jones, "Scalable stochastic processors," in *Proc. Design, Automat. & Test in Europe Conf. & Expo., DATE'10*, 2010, pp. 335–338.
- [10] K. Asanovic, R. Bodik, B. Catanzaro, J. Gebis, P. Husbands, K. Keutzer, D. Patterson, W. Plishker, J. Shalf, S. Williams *et al.*, "The landscape of parallel computing research: A view from Berkeley," UCB/EECS-2006-183, EECS Department, University of California, Berkeley, Tech. Rep., 2006.
- [11] D. Lammers, "The era of error-tolerant computing," *IEEE Spectrum*, vol. 47, no. 11, pp. 15–15, Nov. 2010.
- [12] B. Khailany, T. Williams, J. Lin, E. Long, M. Rygh, D. Tovey, and W. Dally, "A programmable 512 gops stream processor for signal, image, and video processing," *IEEE J. Solid-State Circ.*, vol. 43, no. 1, pp. 202–213, Jan. 2008.
- [13] Z. Zhu, I. Cox, and M. Levene, "Ranked-listed or categorized results in ir: 2 is better than 1," *Proc. Internat. Conf. Nat. Lang. and Inf. Systems, Nldb'08*, pp. 111–123, 2008.
- [14] K. Petridis, D. Anastopoulos, C. Saathoff, N. Timmermann, Y. Kompatsiaris, and S. Staab, "M-ontomat-annotizer: Image annotation linking ontologies and multimedia low-level features," in *Proc. Knowledge-Based Intell. Inf. and Eng. Syst. (LNCS-4253)*, 2006, pp. 633–640.
- [15] T. Yeh, G. Reinman, S. Patel, and P. Faloutsos, "Fool me twice: Exploring and exploiting error tolerance in physics-based animation," *ACM Trans. on Graphics*, vol. 29, no. 1, article 5, Jan. 2009.
- [16] T. Kim and D. James, "Skipping steps in deformable simulation with online model reduction," *ACM Trans. on Graphics*, vol. 28, no. 5, article 123, 2009.
- [17] T. Todman, G. Constantinides, S. Wilton, O. Mencer, W. Luk, and P. Cheung, "Reconfigurable computing: architectures and design methods," vol. 152, no. 2, pp. 193–207, Mar. 2005.
- [18] A. Roldao-Lopes, A. Shahzad, G. Constantinides, and E. Kerrigan, "More flops or more precision? accuracy parameterizable linear equation solvers for model predictive control," in *Proc. 17th Symp. Field Program. Cust. Comput. Mach.*, 2009, pp. 209–216.
- [19] J. Dongarra, P. Beckman, T. Moore, P. Aerts, G. Aloisio, J. Andre, D. Barkai, J. Berthou, T. Boku, B. Braunschweig *et al.*, "The international exascale project roadmap," *Int. J. of High Perf. Comput. Appl.*, vol. 25, no. 1, pp. 3–60, Jan. 2011.
- [20] R. Newcombe, S. Izadi, O. Hilliges, D. Molyneaux, D. Kim, A. Davison, P. Kohli, J. Shotton, S. Hodges, and A. Fitzgibbon, "KinectFusion: Real-time dense surface mapping and tracking," in *Proc. 10th Symp. Mixed and Augm. Real., ISMAR'11*, 2011, pp. 127–136.
- [21] J. Pearl, "Theoretical bounds on the complexity of inexact computations," *IEEE Trans. Inf. Theory*, vol. 22, no. 5, pp. 580–586, May 1976.
- [22] D. Sow and A. Eleftheriadis, "Complexity distortion theory," *IEEE Trans. Inf. Theory*, vol. 49, no. 3, pp. 604–608, Mar. 2003.
- [23] N. Vereshchagin and P. Vitányi, "Rate distortion and denoising of individual data using kolmogorov complexity," *IEEE Trans. Inf. Theory*, vol. 56, no. 7, pp. 3438–3454, Jul. 2010.
- [24] K. Vereshchagin and P. Vitanyi, "Algorithmic rate-distortion function," in *Proc. IEEE Int. Symp. Inf. Theory*, 2006, pp. 798–802.
- [25] V. Goyal and M. Vetterli, "Computation-distortion characteristics of block transform coding," in *Proc. IEEE Int. Conf. on Acoust., Speech, and Signal Process., ICASSP-97*, vol. 4. IEEE, 1997, pp. 2729–2732.
- [26] J. Ostermann, J. Bormans, P. List, D. Marpe, M. Narroschke, F. Pereira, T. Stockhammer, and T. Wedi, "Video coding with h. 264/avc: tools, performance, and complexity," *IEEE Circ. and Syst. Mag.*, vol. 4, no. 1, pp. 7–28, Jan. 2004.
- [27] J. Ludwig, S. Nawab, and A. Chandrakasan, "Low-power digital filtering using approximate processing," *IEEE J. Solid-State Circ.*, vol. 31, no. 3, pp. 395–400, Mar. 1996.
- [28] A. Chandrakasan, V. Gutnik, and T. Xanthopoulos, "Data driven signal processing: an approach for energy efficient computing," in *Proc. Int. Symp. on Low power Electr. and Design, ISLPED'96*, 1996, pp. 347–352.
- [29] Y. Andreopoulos and M. van der Schaar, "Incremental refinement of computation for the discrete wavelet transform," *IEEE Trans. Signal Process.*, vol. 56, no. 1, pp. 140–157, Jan. 2008.
- [30] D. Anastasia and Y. Andreopoulos, "Software designs of image processing tasks with incremental refinement of computation," *IEEE Trans. Image Process.*, vol. 19, no. 8, pp. 2099–2114, Aug. 2010.
- [31] —, "Scheduling and energy-distortion tradeoffs with operational refinement of image processing," in *Proc. Design, Automat. & Test in Europe Conf. & Expo., DATE'10*, 2010, pp. 1719–1724.
- [32] D. Lee, L. Kim, and J. Villasenor, "Precision-aware self-quantizing hardware architectures for the discrete wavelet transform," *IEEE Trans. on Image Process.*, vol. 21, no. 2, pp. 768–777, Feb. 2012.
- [33] T. Xanthopoulos and A. Chandrakasan, "A low-power DCT core using adaptive bitwidth and arithmetic activity exploiting signal correlations and quantization," *IEEE J. Solid-State Circuits*, vol. 35, no. 5, pp. 740–750, May 2000.
- [34] Y. Chen and H. Sundaram, "Basis projection for linear transform approximation in real-time applications," in *Proc. IEEE Int. Conf. Acoust., Speech and Signal Process., ICASSP'06*, vol. 2, 2006, pp. II–II.
- [35] M. van der Schaar and Y. Andreopoulos, "Rate-distortion-complexity modeling for network and receiver aware adaptation," *IEEE Trans. on Multimedia*, vol. 7, no. 3, pp. 471–479, Jun. 2005.
- [36] Y. Andreopoulos and M. van der Schaar, "Complexity-constrained video bitstream shaping," *IEEE Trans. on Signal Process.*, vol. 55, no. 5, pp. 1967–1974, May 2007.
- [37] B. Foo, Y. Andreopoulos, and M. van der Schaar, "Analytical rate-distortion-complexity modeling of wavelet-based video coders," *IEEE Trans. on Signal Process.*, vol. 56, no. 2, pp. 797–815, Feb. 2008.
- [38] Y. Andreopoulos and M. van der Schaar, "Adaptive linear prediction for resource estimation of video decoding," *IEEE Trans. on Circ. and Syst. for Video Technol.*, vol. 17, no. 6, pp. 751–764, Jun. 2007.
- [39] H. Stolberg, M. Bereković, and P. Pirsch, "A platform-independent methodology for performance estimation of multimedia signal processing applications," *J. of VLSI Signal Process.*, vol. 41, no. 2, pp. 139–151, Feb. 2005.
- [40] I. Bauermann and E. Steinbach, "RDTC optimized compression of image-based scene representations (part i): Modeling and theoretical analysis," *IEEE Trans. on Image Process.*, vol. 17, no. 5, pp. 709–723, May 2008.
- [41] —, "RDTC optimized compression of image-based scene representations (part ii): Practical coding," *IEEE Trans. on Image Processing*, vol. 17, no. 5, pp. 724–736, Jun. 2008.
- [42] D. Turaga, M. van der Schaar, and B. Pesquet-Popescu, "Complexity scalable motion compensated wavelet video encoding," *IEEE Trans. Circ. and Syst. for Video Technol.*, vol. 15, no. 8, pp. 982–993, Aug. 2005.
- [43] G. Lee, Y. Chen, M. Mattavelli, and E. Jang, "Algorithm/architecture co-exploration of visual computing on emergent platforms: overview and future prospects," *IEEE Trans. Circ. and Syst. for Video Technol.*, vol. 19, no. 11, pp. 1576–1587, Nov. 2009.
- [44] J. Valentim, P. Nunes, and F. Pereira, "Evaluating mpeg-4 video decoding complexity for an alternative video complexity verifier model," *IEEE Trans. Circ. and Syst. for Video Technol.*, vol. 12, no. 11, pp. 1034–1044, Nov. 2002.

- [45] M. Horowitz, A. Joch, F. Kossentini, and A. Hallapuro, "H.264/AVC baseline profile decoder complexity analysis," *IEEE Trans. Circ. and Syst. for Video Technol.*, vol. 13, no. 7, pp. 704–716, Jul. 2003.
- [46] S. Regunathan, P. Chou, and J. Ribas-Corbera, "A generalized video complexity verifier for flexible decoding," in *Proc. IEEE Int. Conf. Image Processing, ICIP'03*, vol. 3, 2003, pp. 289–292.
- [47] W. Yuan and K. Nahrstedt, "Practical voltage scaling for mobile multimedia devices," in *Proc. 12th ACM Int. Conf. Multimedia*, 2004, pp. 924–931.
- [48] Z. He, Y. Liang, L. Chen, I. Ahmad, and D. Wu, "Power-rate-distortion analysis for wireless video communication under energy constraints," *IEEE Trans. Circ. and Syst. for Video Technol.*, vol. 15, no. 5, pp. 645–658, May 2005.
- [49] W. Yuan, K. Nahrstedt, S. Adve, D. Jones, and R. Kravets, "Grace-1: Cross-layer adaptation for multimedia quality and battery energy," *IEEE Trans. Mobile Comput.*, vol. 5, no. 7, pp. 799–815, Jul. 2006.
- [50] J. Srinivasan and S. Adve, "Predictive dynamic thermal management for multimedia applications," in *Proc. 17th Int. Conf. on Supercomput., ICS'03*, 2003, pp. 109–120.
- [51] Z. Cao, B. Foo, L. He, and M. van der Schaar, "Optimality and improvement of dynamic voltage scaling algorithms for multimedia applications," *IEEE Trans. Circ. and Syst.*, vol. 57, no. 3, pp. 681–690, Mar. 2010.
- [52] N. Mastronarde and M. van der Schaar, "A bargaining theoretic approach to quality-fair system resource allocation for multiple decoding tasks," *IEEE Trans. on Circ. Syst. for Video Technol.*, vol. 18, no. 4, pp. 453–466, Apr. 2008.
- [53] W. Xu, X. Liu, and Y. Gong, "Document clustering based on non-negative matrix factorization," in *Proc. 26th Int. ACM Conf. on Res. and Dev. in Inf. Retr.*, 2003, pp. 267–273.
- [54] D. Ellis, C. Cotton, and M. Mandel, "Cross-correlation of beat-synchronous representations for music similarity," in *Proc. IEEE Int. Conf. on Acoust., Speech and Signal Process., ICASSP'08*, 2008, pp. 57–60.
- [55] B. Dally, "The future of GPU computing," in *ACM/IEEE Int. Conf. on Supercomp., SC'09 (presentation)*, 2009.
- [56] H. Asthana, R. Fu, and I. Cox, "On the feasibility of unstructured peer-to-peer information retrieval," *Adv. in Inf. Retr. Theory*, pp. 125–138, 2011.
- [57] R. Datta, D. Joshi, J. Li, and J. Wang, "Image retrieval: Ideas, influences, and trends of the new age," *ACM Computing Surveys*, vol. 40, no. 2, p. 5, Apr. 2008.
- [58] C. Schmid, R. Mohr, and C. Bauckhage, "Evaluation of interest point detectors," *Int. J. of Comp. Vis.*, vol. 37, no. 2, pp. 151–172, Feb. 2000.
- [59] Y. Andreopoulos and I. Patras, "Incremental refinement of image salient-point detection," *IEEE Trans. on Image Process.*, vol. 17, no. 9, pp. 1685–1699, Sept. 2008.
- [60] D. Jun and D. Jones, "An energy-aware framework for cascaded detection algorithms," in *IEEE Worksh. on Signal Process. Syst., SIPS'10*, 2010, pp. 1–6.
- [61] P. Mainali, Q. Yang, G. Lafruit, L. Gool, and R. Lauwereins, "Robust low complexity corner detector," *IEEE Trans. on Circ. and Syst. for Video Technol.*, vol. 21, no. 4, pp. 435–445, Apr. 2011.
- [62] K. Fukunaga, *Introduction to statistical pattern recognition*. Elsevier Acad. Press, ISBN 0122698517, 1990.
- [63] B. Foo and M. van der Schaar, "A distributed approach for optimizing cascaded classifier topologies in real-time stream mining systems," *IEEE Trans. on Image Process.*, vol. 19, no. 11, pp. 3035–3048, Nov. 2010.
- [64] K. Asanovic and N. Morgan, "Experimental determination of precision requirements for back-propagation training of artificial neural networks," TR-91-036, Univ. of California Berkeley, International Computer Science Institute, Tech. Rep., 1991.
- [65] M. Bartlett, J. Movellan, and T. Sejnowski, "Face recognition by independent component analysis," *IEEE Trans. on Neural Net.*, vol. 13, no. 6, pp. 1450–1464, Jun. 2002.
- [66] J. Yang, D. Zhang, A. Frangi, and J. Yang, "Two-dimensional PCA: a new approach to appearance-based face representation and recognition," *IEEE Trans. on Pat. Anal. and Mach. Intel.*, vol. 26, no. 1, pp. 131–137, Jan. 2004.
- [67] D. Lowe, "Object recognition from local scale-invariant features," in *Proc. 7th IEEE Conf. on Comp. Vis., ICCV'99*, vol. 2, 1999, pp. 1150–1157.
- [68] K. Goto and R. Van De Geijn, "High-performance implementation of the level-3 blas," *ACM Trans. on Math. Soft.*, vol. 35, no. 1, article 4, Jan. 2008.
- [69] E. Agullo, J. Demmel, J. Dongarra, B. Hadri, J. Kurzak, J. Langou, H. Ltaief, P. Luszczek, and S. Tomov, "Numerical linear algebra on emerging architectures: The PLASMA and MAGMA projects," in *J. of Phys.: Conference Series*, vol. 180, no. 012037, 2009.
- [70] A. Portero, G. Talavera, M. Moreno, J. Carrabina, and F. Catthoor, "Methodology for energy-flexibility space exploration and mapping of multimedia applications to single-processor platform styles," *IEEE Trans. on Circ. and Syst. for Video Technol.*, vol. 21, no. 8, pp. 1027–1039, Aug. 2011.
- [71] S. Bhattacharyya, J. Eker, J. Janneck, C. Lucarz, M. Mattavelli, and M. Raulet, "Overview of the mpeg reconfigurable video coding framework," *J. of Sig. Process. Syst.*, vol. 63, no. 2, pp. 251–263, Apr. 2011.
- [72] L. Buatois, G. Caumon, and B. Levy, "Concurrent number cruncher: a gpu implementation of a general sparse linear solver," *Int. J. of Par., Emerg. and Distr. Systems*, vol. 24, no. 3, pp. 205–223, Jun. 2009.
- [73] J. Langou, P. Luszczek, J. Kurzak, A. Buttari, and J. Dongarra, "Exploiting the performance of 32 bit floating point arithmetic in obtaining 64 bit accuracy (revisiting iterative refinement for linear systems)," in *Proc. ACM/IEEE Conf. on Supercomput., SC'06*, 2006, pp. 50–50.
- [74] D. Murray and S. Hand, "Spread-spectrum computation," in *Proc. 4th USENIX Conf. on Hot Top. in Syst. Depend.*, 2008, pp. 5–8.
- [75] N. Maruyama, A. Nukada, and S. Matsuoka, "Software-based ecc for gpus," in *Proc. Symp. on Appl. Accel. in High Perf. Comput., SAAHPC'09*, 2009.
- [76] J. Shaffer, D. Luebke, and K. Skadron, "A hardware redundancy and recovery mechanism for reliable scientific computation on graphics processors," in *Proc. 22nd ACM/SIGGRAPH/EUROGRAPHICS Symp. on Graph. Hardw.*, 2007, pp. 55–64.
- [77] M. Breuer, "Hardware that produces bounded rather than exact results," in *Proc. 47th Des. Autom. Conf.*, 2010, pp. 871–876.
- [78] H. Chung and A. Ortega, "Analysis and testing for error tolerant motion estimation," in *Proc. 20th IEEE Int. Symp. on Defect and Fault Tol. in VLSI Syst., DFT'05*, 2005, pp. 514–522.
- [79] K. Lengwehasatit and A. Ortega, "Scalable variable complexity approximate forward dct," *IEEE Circ. and Syst. for Video Technol.*, vol. 14, no. 11, pp. 1236–1248, Nov. 2004.
- [80] E. Phillips, "CUDA accelerated linpack on clusters," in *ACM/IEEE Int. Conf. on Supercomp., SC'09 (presentation)*, 2009.
- [81] D. Anastasia and Y. Andreopoulos, "Linear image processing operations with operational tight packing," *IEEE Signal Process. Lett.*, vol. 17, no. 4, pp. 375–378, Apr. 2010.
- [82] A. Kadyrov and M. Petrou, "The Invaders' algorithm: range of values modulation for accelerated correlation," *IEEE Trans. on Pat. Anal. and Mach. Intel.*, vol. 28, no. 11, pp. 1882–1886, Nov. 2006.
- [83] J. Allen, "An approach to fast transform coding in software," *Elsevier Signal Process.: Image Comm.*, vol. 8, no. 1, pp. 3–11, Jan. 1996.
- [84] C. Lin, B. Zhang, and Y. Zheng, "Packed integer wavelet transform constructed by lifting scheme," *IEEE Trans. Circ. and Syst. for Video Technol.*, vol. 10, no. 8, pp. 1496–1501, Aug. 2000.
- [85] D. Borland and G. Constantinides, "A scalable precision analysis framework," *IEEE Trans. on Multimedia*, this issue.
- [86] S. Nassif, N. Mehta, and Y. Cao, "A resilience roadmap," in *Proc. Des., Autom. & Test in Eur. Conf. & Expo., DATE'10*, 2010, pp. 1011–1016.
- [87] N. Mastronarde, K. Kanoun, D. Aienza, P. Frossard, and M. van der Schaar, "Markov decision process based energy-efficient on-line scheduling for slice-parallel video decoders on multicore systems," *IEEE Trans. on Multimedia*, this issue.
- [88] J. Dean and S. Ghemawat, "MapReduce: Simplified data processing on large clusters," *Comm. of the ACM*, vol. 51, no. 1, pp. 107–113, Jan. 2008.
- [89] M. Liu, Z. Lu, W. Kuehn, and A. Jantsch, "Fpga-based adaptive computing for correlated multi-stream processing," in *Proc. Des., Autom. & Test in Europe Conf. & Expo., DATE'10*, 2010, pp. 973–976.
- [90] W. Li, X. Tong, T. Wang, Y. Zhang, and Y. Chen, "Parallelization strategies and performance analysis of media mining applications on multi-core processors," *J. of Signal Process. Syst.*, vol. 57, no. 2, pp. 213–228, Mar. 2009.
- [91] L. Leem, H. Cho, J. Bau, Q. Jacobson, and S. Mitra, "ERSA: Error resilient system architecture for probabilistic applications," in *Proc. Des., Autom. & Test in Europe Conf. & Expo., DATE'10*, 2010, pp. 1560–1565.
- [92] P. Subramanyan, V. Singh, K. Saluja, and E. Larsson, "Multiplexed redundant execution: A technique for efficient fault tolerance in chip

- multiprocessors,” in *Proc. Des., Autom. & Test in Europe Conf. & Expo., DATE'10*, 2010, pp. 1572–1577.
- [93] P. Korkmaz, B. Akgul, and K. Palem, “Energy, performance, and probability tradeoffs for energy-efficient probabilistic cmos circuits,” *IEEE Trans. Circ. and Syst. I: Reg. Papers*, vol. 55, no. 8, pp. 2249–2262, Aug. 2008.
- [94] N. Carter, H. Naeimi, and D. Gardner, “Design techniques for cross-layer resilience,” in *Proc. Des., Automat. & Test in Europe Conf. & Expo., DATE'10*. European Design and Automation Association, 2010, pp. 1023–1028.
- [95] M. Nicolaidis, “Time redundancy based soft-error tolerance to rescue nanometer technologies,” in *Proc. 17th IEEE. Int. VLSI Test Symp., VTS'99*, 1999, pp. 86–94.
- [96] —, “Design for soft error mitigation,” *IEEE Trans. Dev. and Materials Reliab.*, vol. 5, no. 3, pp. 405–418, Mar. 2005.
- [97] K. Nepal, R. Bahar, J. Mundy, W. Patterson, and A. Zaslavsky, “Designing nanoscale logic circuits based on markov random fields,” *J. of Electron. Test.*, vol. 23, no. 2, pp. 255–266, Feb. 2007.
- [98] —, “Designing nanoscale logic circuits based on markov random fields,” *J. Electron. Test.*, vol. 23, no. 2, pp. 255–266, Feb. 2007.
- [99] G. Varatkar, S. Narayanan, N. Shanbhag, and D. Jones, “Stochastic networked computation,” *IEEE Trans. Very Large Scale Integr.*, vol. 18, no. 10, pp. 1421–1432, Oct. 2010.
- [100] W. Qian, X. Li, M. Riedel, K. Bazargan, and D. Lilja, “An architecture for fault-tolerant computation with stochastic logic,” *IEEE Trans. on Comput.*, vol. 60, no. 1, pp. 93–105, Jan. 2011.
- [101] T. Austin, D. Blaauw, T. Mudge, and K. Flautner, “Making typical silicon matter with razor,” *IEEE Comput.*, vol. 37, no. 3, pp. 57–65, Mar. 2004.
- [102] J. Sartori and R. Kumar, “Branch and data herding: Reducing control and memory divergence for error-tolerant gpu applications,” *IEEE Trans. on Multimedia*, this issue.
- [103] R. A. Abdallah and N. R. Shanbhag, “Robust and energy efficient multimedia systems via likelihood processing,” *IEEE Trans. on Multimedia*, this issue.
- [104] H. Kaul, M. Anders, S. Mathew, S. Hsu, A. Agarwal, F. Sheikh, R. Krishnamurthy, and S. Borkar, “A 1.45 GHz 52-to-162GFLOPS/W variable-precision floating-point fused multiply-add unit with certainty tracking in 32nm CMOS,” in *Proc. IEEE Int. Solid-State Circ. Conf. Dig of Tech. Papers, ISSCC'12*, 2012, pp. 182–184.
- [105] A. Lingamneni, K. Muntimadugu, C. Enz, R. Karp, K. Palem, and C. Piguet, “Algorithmic methodologies for ultra-efficient inexact architectures for sustaining technology scaling,” in *Proc. 9th ACM Conf. Comput. Front.*, 2012, pp. 3–12.



**Yiannis Andreopoulos** (M'00) is Senior Lecturer at University College London (UK). His research interests are in wireless sensor networks, error-tolerant computing and multimedia systems. He received the 2007 “Most-Cited Paper” award from the ELSEVIER EURASIP SIGNAL PROCESSING: IMAGE COMMUNICATION journal and a best-paper award from the 2009 IEEE WORKSHOP ON SIGNAL PROCESSING SYSTEMS. Dr. Andreopoulos was Special Sessions Co-chair of the 10TH INTERNATIONAL WORKSHOP ON IMAGE ANALYSIS FOR MULTIMEDIA INTERAC-

TIVE SERVICES (WIAMIS 2009) and Programme Co-chair of the 18TH INTERNATIONAL CONFERENCE ON MULTIMEDIA MODELING (MMM 2012). He is an Associate editor of the IEEE TRANSACTIONS ON MULTIMEDIA, the IEEE SIGNAL PROCESSING LETTERS and the ELSEVIER IMAGE AND VISION COMPUTING journal.