Algorithm/Architecture Co-Exploration of Visual Computing on Emergent Platforms: Overview and Future Prospects

Gwo Giun Lee, *Senior Member, IEEE,* Yen-Kuang Chen, *Senior Member, IEEE,* Marco Mattavelli, and Euee S. Jang, *Associate Member, IEEE*

***Abstract*—Concurrently exploring both algorithmic and architectural optimizations is a new design paradigm. This survey paper addresses the latest research and future perspectives on the simultaneous development of video coding, processing, and computing algorithms with emerging platforms that have multiple cores and reconfigurable architecture. As the algorithms in forthcoming visual systems become increasingly complex, many applications must have different profiles with different levels of performance. Hence, with expectations that the visual experience in the future will become continuously better, it is critical that advanced platforms provide higher performance, better flexibility, and lower power consumption. To achieve these goals, algorithm and architecture co-design is significant for characterizing the algorithmic complexity used to optimize targeted architecture. This paper shows that seamless weaving of the development of previously autonomous visual computing algorithms and multicore or reconfigurable architectures will unavoidably become the leading trend in the future of video technology.**

***Index Terms*—Algorithm/architecture co-exploration, complexity analysis or characterization, dataflow, graphs, multicore, reconfigurability, visual computing.**

I. Introduction

NIKLAUS EMIL WIRTH introduced the innovative idea that *Programming* = *Algorithm* + *Data Structure*. Inspired by this paradigm, we advance this to the next level by stating that *Design* = *Algorithm* + *Architecture*.

Traditional design methodologies are usually based on the execution of a series of sequential stages: the theoretical study of a fully specified algorithm, the mapping of the algorithm to a selected architecture, the evaluation of the performance, and the final implementation. However, these straightforward design procedures are no longer adequate to cope with the increasing demands of video design challenges. Conventional sequential design flow yields the independent design and development of the algorithm from the architecture. However, with the ever increasing complexity of both algorithm and system platforms in each successive generation, such sequential steps in traditional designs will inevitably lead to the scenario that designers may either develop highly efficient but highly complex algorithms that cannot be implemented or else may offer platforms that are impractical for real world applications because the processing capabilities cannot be efficiently exploited by the newly developed algorithms. Hence, seamless weaving of the two previously autonomous algorithmic development and architecture development will unavoidably be observed.

As the algorithms in forthcoming visual systems become more complex, many applications in digital video technology must be deployed with different profiles having different levels of performance. Fig. 1 ranks the spectrum of visual computing algorithms based on qualitative complexity analysis.[1]

Extrapolating from a high-level description, as illustrated in Fig. 1, future visual computing algorithms will have better content adaptivity, extended use of temporal information, and further increase the physical size and resolution of the image sequences as they continuously deliver better visual quality. Recent and future video coding standards such as MPEG have been and will continue to focus on video coding tools that are better adapted to the content and that are more refined in motion estimation for more accurate motion compensation models that yield greater complexity. Furthermore, the increase in image-sequence sizes, from standard definition to high definition (HD) and beyond, is also within the already defined roadmaps of video coding standards development.

Similarly, the complexity of video processing algorithms, such as motion adaptive deinterlacers [1], [2], scan-rate converters, and other format converters, is also characterized by studying the three complexity features discussed above. Content-adaptive algorithms for high definition video processing, such as those in scalers, deinterlacers, etc., which are based on the texture contents from surrounding neighbors, have also been documented in the literature [3], [4] and used

Manuscript received February 1, 2009. First version published September 1, 2009; current version published October 30, 2009. This paper was recommended by Associate Editor L.-G. Chen.

G. G. Lee is with the Motion Picture Expert Group, Villar Dora 10040, Italy, and also with the Department of Electrical Engineering, National Cheng Kung University, Tainan 70101, Taiwan (e-mail: clee@mail.ncku.edu.tw).

Y.-K. Chen is with Intel Corporation, Santa Clara, CA 95054 USA (e-mail: y.k.chen@ieee.org).

M. Mattavelli is with the Multimedia Architectures Research Group, École Polytechnique Fédérale de Lausanne, Lausanne 1015, Switzerland (e-mail: marco.mattavelli@epfl.ch).

E. S. Jang is with the Department of Computer Science and Engineering, Hanyang University, Seoul 133-791, Korea (e-mail: esjang@hanyang.ac.kr).

Digital Object Identifier 10.1109/TCSVT.2009.2031376

[1]Within the context of this paper the terms "complexity analysis" and "characterization" both refer to the analysis of complexity metrics on number of operations, potential for parallelism, data transfer, bandwidth, etc.

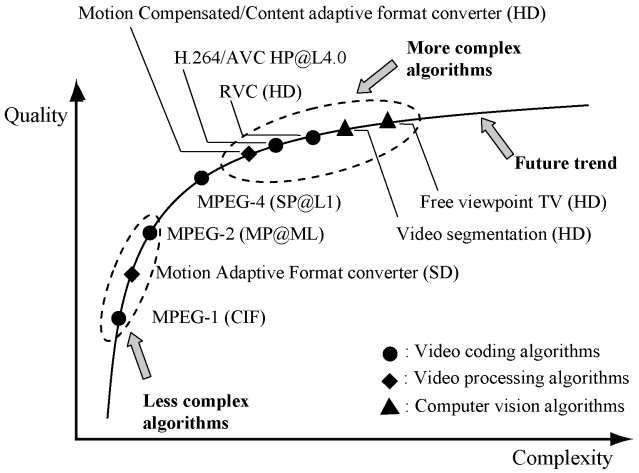

Fig. 1. Complexity spectrum for advanced visual computing algorithms.

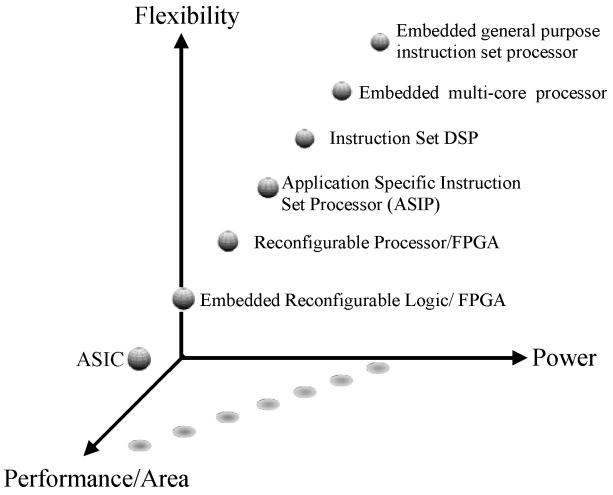

Fig. 2. Spectrum of platforms.

in many applications. Motion-compensated and content-aware algorithms for increasingly higher definition video processing technologies will also be seen in the future.

In addition to being content-adaptive, computer vision algorithms are even more complex as they are content aware as the required cognitive capabilities are added. Computer graphics algorithms are highly computationally intensive: some graphics algorithms, such as real-time rendering, demand a huge amount of processing power, and visualizing graphic content is possible when using many tremendously complex graphics algorithms. The recent evolution of the processing power of graphics processing units (GPUs) is a good indication that the graphics algorithms require ever increasing computing power to provide a better user experience.

Rapid and continuous improvements in semiconductor and architecture design technologies have yielded innovations in system architectures and platforms that offer advanced visual computing algorithms that target different applications. Each application has versatile requirements for trading off the performance per unit of silicon area (performance/silicon area), flexibility of usage, algorithm changes, and power consumption. Conventional implementations of algorithms were usually placed at two architectural extremes: pure hardware or pure software. Although application-specific integrated circuit implementation of algorithms provides the highest speed or best performance, this is however achieved via trading off platform flexibility. Pure software implementations on single-chip processors or CPUs are the most flexible, but require a higher power overhead and yield a slower processing speed. Hence, several other classes of architecture, such as instruction set digital signal processors and application specific instruction set processors, have also been introduced (Fig. 2). We argue and will show that embedded multicore processors and reconfigurable architectures will become the leading trend in architectures that use visual computing algorithms for versatile visual systems.

Because visual algorithms are becoming ever more complex, successfully mapping them onto platforms optimal for versatile applications is a key consideration for forthcoming design methodologies. The aforementioned sequential design flow may provide either excellent visual algorithms that are highly complex and, therefore, cannot be implemented, or that can be used only on system platforms with limited applications because of their poor visual quality. Hence, with the future in mind, we introduce the concept of algorithm/architecture co-exploration, which is now a leading paradigm.

The concurrent exploration of algorithm and architecture optimizations consists of extracting complexity measures of algorithms featuring architectural characteristics that, together with the dataflow model, yield the best mapping onto a platform for targeted applications. It is important to note that the traditional order of growth used in the complexity analysis of computer algorithms is insufficient because it is based on the ideal Random Access Machine, which is merely a single point or platform within the broad spectrum of platforms (Fig. 2). Because contemporary and future platforms are beyond the nanometer scale range, the complexity measures we discuss provide quantitative measures of the intrinsic complexity of algorithms for envisioning even the scope or spectrum of future system architectures and platforms.

Furthermore, many complex algorithms such as those for computer vision that was hard to implement can now be modeled by better dataflow representation discussed in this paper and hence making realization feasible.

II. Advanced Visual Computing Algorithms for Versatile Applications

Digital video technology has substantially transformed our daily life by providing the user with many new ways of enjoying media content. A great deal of video content is now created not only by the traditional content creators (e.g., studios and broadcasting companies), but also by users themselves—who used to be only the end users of the content creation. Newly created objects and services [such as user-created content, personal broadcasting, video telephony, and Internet protocol television (IPTV)] would not exist without *advanced* digital video technology.

At the core of the recent digital media evolution are visual computing technologies that include the acquisition,

representation, manipulation, analysis, synthesis, distribution, and rendition of visual information. Visual computing emerged from various areas in image and video-processing, computer vision, computer graphics, and visualization. Interoperability among the digital video devices was crucial for accelerating the evolution of visual computing technologies. This interoperability was made possible by supporting common formats at every stage of visual-content processing.

This section surveys advanced visual computing algorithms for emerging applications in four areas: video coding, video processing, computer vision, and computer graphics.

A. Video Coding

Video coding standards—produced by international standardization organizations such as the International Standards Organization (ISO) and the International Telecommunication Union—have laid a good foundation for fostering common video formats in different applications. Among the existing video coding standards, the MPEG format MPEG-1 was used in video CD players to replace video cassette players. MPEG-2 has been the most successful video coding standard: it serves many application areas, including digital TV, DVD, satellite/cable broadcasting, and high definition TV. MPEG-4 Advanced Video Coding (AVC)/H.264 is now the most advanced and highest performance video coding standard; MPEG-4 is used in emerging media applications such as digital video broadcasting/digital multimedia broadcasting, and IPTV. Currently, MPEG-4 AVC/H.264 is even challenging its predecessor, MPEG-2, by competing in existing video applications such as HD-DVD.

Recently, MPEG has developed several new coding standards, such as scalable video coding (SVC), multiview video coding (MVC), and reconfigurable video coding (RVC) [5]. The SVC and MVC standards are based on MPEG-4 AVC. Improving adaptation to video content and adding more coding modes has allowed more sophisticated algorithms for these two coding standards. Since the arrival of MPEG-4 AVC, many designers have tried to develop an efficient decoder, one able to reduce the complexity of the design architecture [6]–[8]. RVC is one of the first attempts in video coding standardization in providing a framework for specifying video coding standards by means of a library of modular components and their connections in configuring a codec [5]. It should be noted that the notion of components in video coding is no longer at the codec level, but at the tool level constituting the codec.

We expect that the complexity of video coding tools will further increase as the demand for more compression and higher quality is always present because of compelling market demand, especially from the newly emerging areas such as HDTV and digital cinema aiming for 2 K and 4 K resolutions [9]. The research on more efficient encoding and decoding technologies, such as motion estimation for existing and new codecs, will continue to thrive [10]–[12].

B. Video Processing

Although video coding is certainly one of the major areas in visual computing, newly emerging application fields are arising primarily from video processing areas with very high computation requirements. Various digital video processing technologies, such as video analysis [13]–[15], automatic video annotation [16], object segmentation [17]–[19], and region-based representation [20], are good examples. These pre and postprocessing video technologies are anticipated to be enriched with the more sophisticated algorithms for better visual quality and compression. Such technologies may include combined motion estimation and segmentation [21]–[23].

C. Computer Vision

Traditionally, attempts have been made by computer vision technologies in the reconstruction of the structure of a 3-D scene rendered by video data or several images. Memory and computation-intensive technologies in computer vision are no longer used exclusively only in research labs. Originated from computer vision, MVC and free viewpoint TV (FTV), have been standardized in MPEG for commercial applications [24]–[26].

In MVC, multiple video bitstreams (e.g., 16 video cameras) are encoded and transmitted. These bitstreams are decoded using multiple decoders and subsequently used to produce a synthetic view generated from an arbitrary viewing angle [27]–[29]. Having tens of decoders in a system was something unimaginable merely a decade ago, but has now become the reality. FTV [30] is yet another computational intensive technology. Using hundreds of cameras and based highly on content information, the video data is captured to build a ray space for reconstructing a free viewpoint video at the terminal.

Medical applications are one of the most prominent fields of computer vision. The principle of vision technologies in medicine is the same as in other areas: the extraction of visual information from image data for medical diagnosis. For example, computer tomographic image reconstruction and medical image registration are among the very active topics of computer vision for medical applications [31]–[33]. Along with other tasks in computer vision, applications in medical imaging have been regarded as computation-intensive tasks because of the large size of the test-data sets used (in tens of gigabytes) and of the highly detailed low-level features that must be extracted from medical images acquired from different energy modalities.

D. Computer Graphics

The complexity of providing visual computing services from digital video to 3-D games is ever increasing, and this has been one of the compelling reasons to continuously develop new and more powerful platforms. For most computer graphics algorithms, ordinary single-CPU-based systems are insufficient. However, graphics computing is now possible via discrete GPUs with speeds up to ten times faster than those integrated with the CPU. GPUs have been introduced for computationally intensive processes in the graphics pipeline, and primarily for interactive 3-D graphics such as games. Many computationally intensive operations in the graphics pipeline are now processed by GPUs [34].

Visual processing technologies in computer graphics are classified as: 1) modeling and animation, and 2) rendering. Modeling and animation technologies, despite the huge amount of data they require to be processed, are not as computationally intensive as rendering technologies. This is why there are two different levels of visual quality, such as those in animated movies and in games.

Many modeling and animation technologies have been standardized as industry standards [35]–[37]. Like video, the compression of modeling and animation data received more attention in standardization activities like ISO/MPEG [38]–[40]. Animation framework extension is an MPEG standard defining many modeling and animation technologies [36], [41].

Rendering technologies demand high computational capabilities provided by graphics architectures to support collision detection, physics simulations, scene management, and ray tracing. With further evolution of GPU architecture in graphics, highly sophisticated rendering technologies will be used in commercial devices in the very near future [42]–[44].

III. Visual Computing on Multicore and Reconfigurable Architectures

Before we describe the details of algorithm/architecture co-exploration, we first survey the spectrum of the emerging architectures for visual computing, describe the challenges of mapping and porting algorithms onto those platforms, and explain the need for algorithm/architecture co-exploration.

Because of the increasing complexity of visual computing, and of the expectations of users that they will have the best possible visual experience, it is necessary that advanced platforms provide higher performance, better flexibility, and lower power consumption. To offer numerous functionalities in a single platform with high flexibility and a faster time-to-market, programmable or reconfigurable designs are preferred because these designs will remain on the market longer. Systems with multiple cores, reconfigurable components, or both, open new possibilities for visual system designers to create highly complex visual computing algorithms that are not feasible on single-CPU platforms.

Multicore processors are theoretically capable of providing tremendously high performance. Cell processors, for example, offer 200 Giga floating-point operations per second [45]; NVIDIA's GeForce GTX 280 GPU delivers 933 giga floating point operations per second [46]; and Intel's experimental 80-core floating-point processor exceeded 1 trillion floating point operations per second [47] in 2007. Intel's Larrabee processors [48] have revealed incessant increase in the number of processors. In addition, multiprocessor system-on-chips have emerged as an important class of very-large-scale integration systems since their first debut in 2000 [49]. Systems with multiple cores have become the preferred choice in many vendors' solutions [50], [51], ranging from a simple game console [52], [53] to a high-end workstation [54]. Even in embedded systems, multicore platforms are just as prevalent, e.g., [57], [58]. Hence, it is desirable that highly sophisticated visual computing algorithms are mapped onto multiple cores.

In general, there are four classes of multicore architectures and fine-grain reconfigurable architecture.

The first class comprehends homogenous general-purpose multicore processors that are often seen in general-purpose computers, e.g., laptops, desktops, or workstations [54]. In the past, we used to increase the performance through higher clock speed. However, increasing the clock speed comes with a cost of power consumption. For example from the mid-1980s to the late 1990s, the power consumption of Intel's microprocessors doubled every 2 to 3 years and was approaching a level in which we can no longer provide power and dissipate heat efficiently. Almost all the major computer vendors are shipping their systems with such kinds of multicore processor architectures today. This is because a dual-core processor at a 50% clock frequency can provide the same performance of a single core in theory, but consumes only 25% of the power. It becomes a new trend that the major processor vendors are exploiting multicore architectures. Recently, homogenous multicore processors are also available for embedded applications [58]. In order to provide efficient performance to various general-purpose applications, this class of processors consists of cores that are equipped with a "reasonable" amount of cache, whose coherency is provided by hardware means. These cores implicitly communicate with each other via a memory-sharing architecture.

The second class of multicore architectures includes homogenous special-purpose, programmable multicore processors. One of the prominent examples is the GPU as seen in the high-end graphics cards for personal computers [46]. In the past, GPUs were traditionally designed for a specific purpose and thus dedicated pipelines were used initially. However, as the users demand more realistic visual effects, increasingly more programmable functionalities, such as vertex and fragment shaders, are added to the graphics pipeline. Therefore, in the past few years, GPUs begin to develop in the trend homogenous, special-purpose, programmable multicore processors. Furthermore, with more general-purpose programming environment, GPUs are beginning to be applied for many non-graphics purposes. This is often known as the GPGPU phenomenon. Due to the nature of the graphics applications, the cores in this second class of processor architectures are designed with limited amount of cache, but with very large memory bandwidth for supporting heavy streaming workloads.

The third class of multicore architectures is the heterogeneous general-purpose multicore processors that are often seen in the special-purpose systems. Each of the cores may be programmable, but they are designed and optimized for different and specific control-intensive or compute-intensive tasks. For example, Cell Broadband Engine [52], [53], jointly developed by Sony Computer Entertainment, Toshiba, and IBM, has one power processor element and eight synergistic processing elements. The power processor element will work with conventional operating systems and act as the controller for the eight synergistic processing elements, which are designed for highly data parallel execution. Industrial Technology Research Institute's parallel architecture core (PAC) project provides another good example of heterogeneous multicore platform for multimedia applications [59]. The audio/video

codec parallelism is exploited from fine-grain 5-way very long instruction word (VLIW) perspective [60]. At the system level the dynamic voltage and frequency scaling scheme is adopted to further reduce the power consumption in PAC-based system-on-chip (SoC) for multimedia applications [61]. Processors in this class are often designed with local storage and communication, which are both managed explicitly by software.

The fourth class is heterogeneous special-purpose multicore processors, which are often seen in embedded systems [7], [55], [56]. Some of these cores may still be programmable, while some of the others may be dedicated only to a particular function, e.g., motion estimation, entropy decoding, etc. Toshiba's SpursEngine, for example, has a combination of four synergistic processor elements from the cell broadband engine and four dedicated accelerators supporting various video codecs [57]. Processors in this class are also designed with local storage and communication, which are both managed explicitly by software.

In reconfigurable computing, coarse grain dataflow models can be implemented via multicore processors and fine grain dataflow models are realized using reconfigurable architectures. Reconfigurable architecture designs have also become an emerging technology to support multiple algorithms with high similarity. Different video coding standards such as those in MPEG, VC-1, and even AVS, for example, share high percentage of commonalities in algorithmic processing. A proper choice of lower data granularity will reveal the commonalities of the algorithm. If these algorithms, targeted for different purposes or different standards, will not be used or executed at the same time, they can then share the same processing element or datapath in reconfigurable architectures.

Due to the specific architectural characteristics of such different classes of multicore processors discussed, the implementation of visual computing algorithms for different applications may result in preference on one class of processors over the others. For example, each class of processors has its own unique data storage or core-to-core communication mechanism. No specific architecture is a clear winner due to various different characteristics and complexities of the broad spectrum of visual computing algorithms. This is the typical case for which algorithm and architecture co-design become essential.

The objective of algorithm and architecture co-design is to facilitate the mapping of highly sophisticated visual computing algorithms onto the most suitable platform, as depicted in Fig. 2, based on the constraints of the targeted application. In general, as coarse grain dataflow models are preferably mapped onto multicore platforms and dataflow models with finer or lower granularity are ported onto reconfigurable architectures. Important steps to achieving these objectives require the consideration of the following: 1) the characterization of algorithmic intrinsic complexity for the appropriate choice of data granularity in corresponding dataflow models, and 2) the data dependence analysis for the characterization of the potential parallelism in algorithmic complexity analysis.

Characterizing the algorithms with intrinsic complexity measures helps in making the best choice among the four classes of multicore processors and results in their optimal usage. By understanding the complexity and by measuring the degree of parallelism and other criteria discussed below, visual computing algorithms can also be mapped onto reconfigurable architectures, e.g., RVC. Depending on the application in mind, this broad spectrum of algorithms can also be mapped onto other platforms depicted in Fig. 2. In short, it is highly essential that algorithm and architecture co-exploration be applied in obtaining the best design for versatile applications.

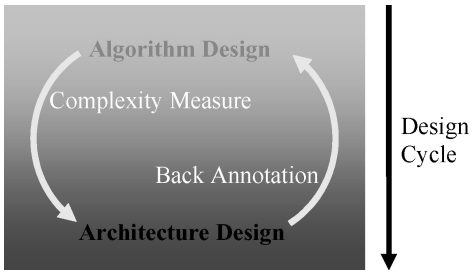

Fig. 3. Algorithm/architecture co-exploration.

IV. Algorithm/Architecture Co-Exploration

Visual system development can be categorized into a number of different design scenarios and can be named architecture, algorithm, and algorithm/architecture design. The first scenario, architecture-oriented design, involves implementing a fully specified algorithm that must be mapped onto a platform specifically designed for the algorithm. The second scenario, algorithm-oriented design, allows the algorithm, or parts of it, to be freely specified, but requires it to be mapped onto an already existing architecture. The third scenario, algorithm/architecture design, allows both algorithm and architecture to be freely developed, but is subject to the usual project and application constraints: performance per unit of silicon area, flexibility, power consumption, and so on. The implementation of video decoders with nonexisting platforms is a typically architecture-oriented design, video encoders onto existing platforms are typically algorithm-oriented design, and other video processing applications, such as de-interlacing, scan-rate up-conversion, and decoder resilience to transmission errors, are an algorithm/architecture design. Concurrent exploration of both algorithmic and architecture optimizations (Fig. 3) is an appropriate approach and is now becoming a new design paradigm.

In this top-down design methodology, it is important to understand the anticipated architecture and evaluate the design space early (Fig. 3) [62]–[64]. The profiling of high-level software codes for complexity analysis has been documented [65], [66]. Because the codes were written to be executed on processors, these methods are high-level and specific for mapping onto processor-oriented platforms. algorithm/architecture co-exploration includes a generic method that enables mapping algorithms onto all the architectures in the spectrum of platforms by extracting complexity measures and quantifying the dataflow models of the algorithms for the mapping and

architecture design. These intrinsic algorithmic complexity measures, together with the dataflow model, will help in the architecture-oriented design. Even for the algorithm-oriented design in the second scenario, complexity characterization or analysis of the dataflow will reveal its optimal utilization.

Furthermore, as illustrated in Fig. 3, in algorithm/architecture co-exploration, it is also possible to back annotation or feedback the architectural features characterized by algorithmic intrinsic complexity metrics for algorithmic development which will result in much more efficient designs.

A. Dataflow Models for the Representation and Co-Design of Algorithms and Architectures

Dataflow representations provide good models for the co-exploration of algorithms and architecture. These models should depict the algorithmic behavior and implicitly reveal and specify the required architecture for the visual system design.

Transaction level modeling (TLM) is a form of dataflow representation that raises the level of abstraction of standard algorithmic descriptions. TLM is a discrete-event model of computation in which modules interchange events with time stamps. Interactions between software and hardware are modeled using shared data channels. Inside this model of computation, modules can be specified at several levels of abstraction, making it possible to specify functionalities and untimed state machine models for the application, as well as to specify an instruction-accurate performance model for the architecture. However, the level of abstraction of this model of computation remains quite low for video algorithm co-design problems. At a higher level of abstraction, we find Kahn process networks (KPN), introduced in 1974 [67]. This model consists of concurrent processes that communicate through unbounded and unidirectional first in, first out. Each process is sequential, but the global execution at the processes level is parallel. The Y-chart Application Programmer's Interface model [68] extends KPNs because it associates a data type with each channel, and because it introduces nondeterminism by allowing dynamic decisions to select the next communication channel. Thus, scheduling shared resources can be modeled.

Graphs also provide good representations of algorithms. In directed acyclic graphs (DAGs), nodes represent atomic operations as well as directed edge data dependences between operations. DAGs depict the dataflow of an application and can be applied at different levels of granularity, ranging from logic operators [69] to tasks [70], [71] or packet processing [72]. These depictions can be extended in DAGs with periods and deadlines, in which computation tasks are annotated with execution deadlines and periods. As in DAGs, in synchronous dataflow (SDF) graphs, nodes represent atomic operations as well as directed edge data dependences between nodes. SDF extends DAG because each SDF node is annotated with the number of data exchanges produced and consumed by the computation at the node. It is static information from which feasible schedules can be determined. This information is also useful for determining the memory requirements for buffering data between the processing entities. Another type of graph is the control data flow graph (CDFG), which can be extracted from the source code description of a program. Directors regulate how actors in the design fire and how tokens are used to communicate between them. This mechanism allows different models of computation (MoC) to be constructed within Ptolemy II. The graph describes the static control flow of the program, showing all the possible paths of computations and dynamic decision points at run-time. It also describes the flow of the data inside the program, showing the concurrency of the application. CDFGs are discussed in [72]. Other graphs focus on the communications between the processing entities like arrival curves [73] (for modeling the workload imposed on an application or system as well as on the output generated by the system) or communication analysis graphs (CAGs) [74]. CAGs represent communication and computation traces extracted from system-level simulations. It is a DAG with communication and computation nodes, which includes timing information. A CAG can thus be seen as an abstract task-level performance model of the application that also includes a schedule.

An algorithmic model can also be represented with co-design finite state machines (CFSMs). The communication between CFSM components is asynchronous, whereas, within a finite state machine, the execution is synchronous, based on events. This representation allows the expression of concurrence. Asynchronous data flow models that specify using actor-based program languages have recently also been applied in the video coding field. A network of concurrent actors called *functional units* has been selected as an MoC for the specification of the ISO/International Electrotechnical Commission (IEC) MPEG RVC standard. The CAL actor language, originally proposed by Janneck and Eker [75], has been profiled and standardized by ISO/IEC MPEG [76], [77]. In this framework, CAL is used for specifying the data-flow description of a video decoder and for reference software of the RVC toolbox library [78]. The MPEG RVC data flow specification provides a very attractive starting point for algorithm-architecture co-design. The CAL data flow specifications can be directly synthesized to software, hardware, and mixed software/hardware implementations using appropriate tools that enable efficient exploration of the design space [79]–[81]. Moreover, because it is based on autonomous concurrent actors, the RVC data flow specification can potentially be directly mapped onto multicore architectures [82]. Although efficient mapping may require appropriate partitioning and platform-dependent optimizations, the RVC data flow specifications provide very attractive features for multicore platforms. That was not the case for the classical specifications of video compression, which were based on imperative sequential models for which concurrency and parallelism were not explicitly provided but needed to be extracted using ad hoc reverse engineering techniques.

Architecture models instead focus more on platform-specific features. These models are useful for analyzing and revealing architectural information, such as speed, power, memory, and area consumption. They may consist of an abstract description of the hardware with a set of associated constraints and characteristics, or of an executable description. Abstract models are only symbolic representations of performance, whereas

executable specifications allow for a more concise analysis of the underlying hardware for communication, computation, and memory.

B. Algorithmic Intrinsic Complexity Characterization

One of the essentials of algorithm/architecture co-exploration is that intrinsic complexity metrics or measures of algorithms that provide architectural information, can be feedback or back-annotated in early design stages to facilitate concurrent exploration of both algorithmic and architectural optimizations [83], [84]. To understand the anticipated architectural features and electronic ingredients in the early design stages, it is important that complexity measures intrinsic to the algorithm be estimated and are not biased toward either hardware or software.

1) *Number of Operations:* The number of arithmetic or logical operations is one of the most basic measures (metrics) that quantifies the complexities of the algorithms during the computation, because more operations will require more computational power in either the software on the CPU platform or the hardware on other system platforms. Hence, the number of operations in addition, subtraction, multiplication, and division can be used to measure the complexity of the algorithm. However, these arithmetic operations are not necessarily equally complex as multiplications are constructed from additions (Booth's algorithm [85]). Division is the most complicated operation because it involves addition, subtraction, and more. Because subtraction can be done using addition of two's complement, the two operations are quantified to be equally complex. Logical operations such as *larger than*, *smaller than*, and *equal to* are also thought to have the same complexity as addition and subtraction because they can be done using subtraction. Shifting is the least complex because only the required significant bits are used in the right and left shifting operations.

The precision of these operations also needs to be considered in complexity analysis, because stricter requirements for precision need operators with more bits. In addition, both arithmetic and logical binary operators have different complexities when either a second variable or a constant is added to the variable first operand. This is because of the difference in the degree of complication in accessing a variable or a constant.

2) *Degree of Parallelism:* The degree of parallelism provides yet another intrinsic measure of algorithmic complexities; this measure is independent of design constraints and has been a significant topic of research for many years [86], [87]. This metric characterizes the tendency with which the operations can be processed independently of each other based on their data dependence. Hence, data defined with appropriate granularity can be unfolded [88] and assigned to more or different corresponding processing elements with a greater degree of parallelism should there be less or possibly no dependence in the data. These algorithms are therefore characterized as being less complex. Greater dependence in the data would then result in less parallelism, therefore giving a more complex algorithm.

Prihozhy *et al.* [89] defined a metric that measures the parallelization potential of an algorithm by estimating the ratio between the complexity and the length of the critical path of the algorithm. The complexity is defined as the total number of operations, and the length of the critical path quantifies the maximum number of operations that could be sequentially executed because of the inherent computational dependences intrinsically embedded in the algorithm. This criterion thus quantitatively measures the potential of performance speedup should the algorithm be processed in parallel rather than sequentially. Janneck *et al.* [90] measured the degree of parallelism by the generation of causation races from dataflow models by studying the computation dependences of algorithms with which resource assignment and scheduling were performed in architecture exploration.

The methodology used by Kung *et al.* [91] to design array processors began with the analysis of dependence graphs capable of characterizing data dependences in order to extract the intrinsic parallelism embedded in algorithms. This nonetheless provided systemic guidelines for both hardware and software designs by exploiting the parallelism contained within algorithms depicted in dependence graphs.

The degree of parallelism also provides an insightful criterion for selecting the appropriate CPU architecture required, such as single instruction multiple data, multiple instruction multiple data, VLIW, and even the multicore architectures described above.

3) *Pipeline Depth:* Data required to process several portions of the algorithm reveal the intrinsic pipeline depth of algorithms. Because some data are reused in the dataflow of algorithms, these data must be stored, thereby inherently increasing the number of stages in the pipeline depth. In addition to the local memory required to store these data, the inherent pipeline depth also results in latencies, which, in turn, increase the lower boundary on the clock speed required for real-time applications.

4) *Memory Requirements:* Visual computing algorithms are frequently required to store intermediate data in memory. Because more-complicated algorithms require more memory, insight into the complexity of visual signal processing can thus be gained from the sizes of the memory used by analyzing the lifetime of the required intermediate data. Furthermore, algorithms normally prescribe the scheduling of data being processed. Hence, to perform the proper algorithmic processing, memory must be appropriately configured based on the scheduling of the intermediate data.

Memory configuration is also an intrinsic algorithmic metric in algorithm/architecture co-exploration design methodology, which is not biased toward either software or hardware designs. For hardware applications, data can be saved, based on algorithmic scheduling, in local memory. In software, this metric facilitates the data arrangement schemes for cache or a scratch-pad system within the embedded CPU.

5) *Data Transfer and Bandwidth:* Visual computing algorithms require the transfer of a large amount of data within a specified time. Therefore, the average bandwidth is a measure of the complexity of the algorithm by providing an estimate of the amount of data transferred in one second. The average bandwidth directly estimated from the algorithm is intrinsic to the algorithm itself, whether the algorithm is

implemented in software or hardware. However, the instantaneous or peak bandwidth requirements of algorithms are significantly influenced by design constraints from several architectural details, including the type of data transaction, memory hierarchy, data alignment in memory, and datapath architecture. Consequently, peak bandwidth is not intrinsic to algorithms. This metric, however, provides further insight into clock-speed requirements and bit widths of the interconnect bus in the design of system platforms for reconfigurable and multicore architectures.

C. Dataflow Modeling and Complexity Characterization for Multicore and Reconfigurable Systems Design

In multicore architectures, complexity measurements, especially of the potential for parallelism embedded in the dataflow, will also help in mapping onto homogeneous or multigrain heterogeneous platforms. For software, a quantitative analysis of the complexity metrics will also provide information for designing retargetable compilers for multigrain, heterogeneous architectures. It may even help in porting operating systems onto the platforms, because we are now familiar with the complexity and, therefore, know how to do scheduling, etc.

In addition, by properly choosing lower data granularities for dataflow models, the complexity analysis technique described in this subsection can be used to extract features common to various algorithms and formats adapted for versatile video content, in designing highly efficient reconfigurable visual computing architectures. This will help, of course, in the datapath from the hardware perspective resulting in different reconfigurable architectures in lower granularities. RVC in MPEG is a good example of this category.

Constructing a good dataflow model and doing a thorough quantitative exploration of the complexity that characterizes the algorithms reveals system architecture information and, therefore, provides a systematic top-down methodology for designing and mapping onto the broad spectrum of platforms at different levels of granularity and performance (Fig. 2).

D. Innovative Architectures with Multiple Processors and Reconfigurability for Video Coding and Processing

In this section, we survey how visual computing algorithms are implemented on multicore platforms and how architecture-algorithm co-exploration can be used to map visual computing applications onto multicore platforms.

Many visual computing algorithms can be mapped onto multicore platforms and reconfigurable architectures. Examples include video encoding/decoding, analysis, understanding, search, retrieval, and medical imaging. Video coding has been studied more extensively than other visual computing applications, from submodules like motion estimation and deblocking filters to complete codecs have been implemented on platforms with multiple cores. Examples of all four previously discussed classes of multicore architectures for visual computing are available: 1) software implementation of video codecs on homogenous general-purpose multicore processors [27], [28], [113], [92]; 2) implementation on homogenous special-purpose processors, e.g., GPU [93]–[95]; 3) implementation of video codec on heterogeneous [60], [97], [98]; and 4) implementation on heterogeneous and dedicated multiple-IP to accelerate video codecs is shown in [7], [55], and [56]. The next class of visual computing applications on multicore is video analysis, understanding, search, and retrieval. Almost no software implementation of video analysis, understanding, search, and retrieval existed a few years ago, given the huge amount of computation required. In the last couple of years, examples [99]–[101] have begun to emerge. Video analysis, understanding, search, and retrieval have various algorithms (unlike video codecs which have a standard set of algorithms), most of which are implemented on homogenous general-purpose processors. Recently, some researchers have studied on how to implement computer vision applications on GPUs [103]. Yet another class of visual computing applications that we mentioned earlier is medical imaging, e.g., volume reconstruction and volume rendering. They have become very popular real-time applications on GPUs [106]–[107] because the algorithms are similar to graphics algorithms. For special-purpose GPUs, programmable processors offer a large number of computations. Currently from the perspective of dataflow modeling, it can be seen that traditional signal processing algorithms such as video coding have a more regular dataflow model whereas the algorithms which have been implemented more recently, such as video analysis, search, and retrieving are modeled with dataflow models that are more irregular and complex.

While mapping the algorithms to the multicore architecture, we learned that algorithm and architecture co-design is unavoidable. For example, for future systems with hundreds of cores, we must consider whether there is enough parallelism in the applications, cf., Section IV-A. Meenderinck *et al.* [107] perform a comprehensive parallelism analysis of the H.264 decoder, which was considered hard to conventionally parallelize. They found that if the decoder is parallelized using a dynamic 3-D wave-front approach, even mobile video will have enough parallelism. For another instance, to increase parallelism, it is important to break data dependence. To achieve the best compression quality, H.264/AVC has incorporated many algorithms with heavy data dependence. Wang *et al.* [108] carefully reviewed the deblocking filter algorithm and observed that the results of each deblocking filtering step affect only a limited region of pixels. Therefore, a novel algorithm is proposed to take advantage of the parallelism. For yet another example, as mentioned earlier, one important factor in efficient implementation on a multicore platform is performance analysis. Li *et al.* [109] showed that properly parallelizing the media-mining workloads is key to effectively using existing small-scale multicore processors and future large-scale multicore platforms, but requires extra effort. One important factor in parallelization is performance analysis, and memory-subsystem performance in particular. After performance bottlenecks are identified, various parallelization techniques can be used. In addition to parallelizing emerging media-mining workloads, the method presented in this paper is also applicable to other applications.

Algorithm/architecture co-exploration is also significant in the design of finer grain reconfigurable architectures.

Examples can be found in the emerging RVC, multipurpose DCT, and reconfigurable de-interlacers [110], [111]. Lee *et al*. [83], [110], [111] have also shown that back annotation or feeding back of the architectural features characterized by algorithmic intrinsic complexity metrics for algorithmic development and reference result in a much more efficient design. With application to 3-D spatio-temporal motion estimation for video coding [83], [112], [113], Lee *et al*. have demonstrated significant reduction in design cost while the algorithmic performance still surpasses recent published works and even full search under many circumstances.

V. Conclusion

To cope with the increasing complexity of future visual computer algorithms and with expectations in rendering the best visual experience, we have shown the importance of concurrently optimizing both algorithm and architecture so that architecture information is back annotated to the algorithm and is thus being considered as early as the algorithm design phase. We have also shown that advanced multicore platforms provide the best tradeoff for higher performance per unit area (gate count), better flexibility, and lower power consumption. Reconfigurable architectures with lower granularity are also favored in applications for higher performance and lower power, but at the cost of less flexibility. AAC for multicore platforms and reconfigurable architectures will inevitably become a trend for implementing the coming generation of visual computing algorithms.

References

[1] G. G. Lee, M.-J. Wang, H.-T. Li, and H.-Y. Lin, "A motion-adaptive deinterlacer via hybrid motion detection and edge-pattern recognition," *EURASIP J. Image Video Process.*, vol. 2008, Mar. 2008.

[2] G. G. Lee, H.-T. Li, M.-J. Wang, and H.-Y. Lin, "Motion adaptive deinterlacing via edge pattern recognition," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS 2007)*, New Orleans, LA, May 2007, pp. 2662–2665.

[3] G. G. Lee, H.-Y. Lin, M.-J. Wang, R.-L. Lai, C. W. Jhuo, and B.-H. Chen, "Spatial-temporal content-adaptive deinterlacing algorithm," *Institut. Eng. Technol. Image Process.*, vol. 2, no. 6, pp. 323–336, Dec. 2008.

[4] G. G. Lee, H.-Y. Lin, M.-J. Wang, R.-L. Lai, and C.-W. Jhuo, "A high-quality spatial-temporal content-adaptive deinterlacing algorithm," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, Seattle, WA, May 2008, pp. 2594–2597.

[5] *The MPEG Homepage* [Online]. Available: http://www.chiariglione.org/mpeg

[6] M. Horowitz, A. Joch, F. Kossentini, and A. Hallapuro, "H.264/AVC baseline profile decoder complexity analysis," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, no. 7, pp. 704–716, Jul. 2003.

[7] T.-C. Chen, Y.-W. Huang, and L.-G. Chen. "Analysis and design of macroblock pipelining for H.264/AVC very-large-scale integration architecture," in *Proc. 2004 Int. Symp. Circuits Syst. (ISCAS)*, vol. 2. May 2004, pp. 273–276.

[8] S.-H. Wang, T. Lin, and Z.-H. Lin. "Macroblock-level decoding and deblocking method and its pipeline implementation in H.264 decoder SOC design," *J. Zhejiang Univ. Sci. A*, vol. 8, no. 1, pp. 36–41, Jan. 2007.

[9] *Digital Cinema System Specification V5.1*, document draft aO331.doc, Digital Cinema Initiatives, LLC, Apr. 2005 [Online]. Available: http://www.dcimovies.com.

[10] C. Zhu, X. Lin, L. Chau, and L. Po, "Hexagonal search for fast block motion estimation," *IEEE. Trans. Circuits Syst. Video Technol.*, vol. 14, no. 10, pp. 1210–1214, Oct. 2004.

[11] K. R. Namuduri, "Motion estimation using spatio-temporal contextual information," *IEEE. Trans. Circuits Syst. Video Technol.*, vol. 14, no. 8, pp. 1111–1115, Aug. 2004.

[12] G. G. Lee, H.-Y. Lin, and M.-J. Wang, "Rate control algorithm based on intra-picture complexity for H.264/AVC," *IET Image Process.*, vol. 3, no. 1, pp. 26–39, Feb. 2009.

[13] G. G. Lee, H.-Y. Lin, D. W.-C. Su, and M.-J. Wang, "Multiresolution-based texture adaptive algorithm for high-quality deinterlacing," *Instit. Electron. Inform. Commun. Eng. Trans. Inf. Syst.*, vol. E90-D, no. 11, pp. 1821–1830, Nov. 2007.

[14] G. G. Lee, D. W.-C. Su, H.-Y. Lin, and M.-J. Wang, "Multiresolution-based texture adaptive M detection for de-interlacing," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, Kos, Greece, May 2006, pp. 4317–4320.

[15] L. Liu, S. Kesavarapu, J. Connell, A. Jagmohotionan, L. Leem, B. Paulovicks, V. Sheinin, L. Tang, and H. Yeo, "Video analysis and compression on the STI cell broadband engine processor," in *Proc. IEEE Int. Conf. Multimedia Expo*, Jul. 2006, pp. 29–32.

[16] A. Vailaya, M. A. T. Figueiredo, A. K. Jain, and H.-J. Zhang, "Image classification for content-based indexing," *IEEE Trans. Image Process.*, vol. 10, no. 1, pp. 117–130, Jan. 2001.

[17] S.-Y. Chien, S.-Y. Ma, and L.-G. Chen, "Predictive watershed: A fast watershed algorithm for video segmentation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, no. 5, pp. 453–461, Jul. 2003.

[18] A.-R. Mansouri and J. Konard, "Multiple motion segmentation with level sets," *IEEE Trans. Image Process.*, vol. 12, no. 2, pp. 201–220, Feb. 2003.

[19] Y. Wang, K.-F. Loe, T. Tan, and J.-K. Wu, "Spatiotemporal video segmentation based on graphical models," *IEEE Trans. Image Process.*, vol. 14, no. 7, pp. 937–947, Jul. 2005.

[20] P. Salembier and F. Marqués, "Region-based representations of image and video: Segmentation tools for multimedia services," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 9, no. 8, pp. 1147–1169, Dec. 1999.

[21] M.-J. Wang, G. G. Lee, and H.-Y. Lin, "Extraction of perceptual hue feature set for color image/video segmentation," in *Proc. 2008 Pacific-Rim Conf. Multimedia (PCM)*, Tainan, Taiwan, Dec. 2008, pp. 875–878.

[22] M.-F. Fu, O. Au, and W.-C. Chan, "Fast global motion estimation based on local motion segmentation," in *Proc. Int. Conf. Image Process.*, vol. 2. 2003, pp. 367–370.

[23] S.-C. Cheng, "Visual pattern matching in motion estimation for object-based very low bit-rate coding using moment preserving edge detection," *IEEE Trans. Multimedia*, vol. 7, no. 2, pp. 189–200, Apr. 2005.

[24] *Description of Exploration Experiments in 3-D Video Coding*, document N10173.doc, ISO/IEC JTC1/SC29/WG11, Oct. 2008.

[25] *Preliminary FTV Model and Requirements*, document N9168.doc, ISO/IEC JTC1/SC29/WG11, Jul. 2007.

[26] *View Synthesis Tools for 3-D Video*, document M15851.doc, ISO/IEC JTC1/SC29/WG11, Oct. 2008.

[27] K. Ugur, H. Liu, J. Lainema, M. Gabbouj, and H. Li, "Parallel encoding-decoding operation for multiview video coding with high coding efficiency," in *Proc. IEEE Integr. 3-D Television Capture Transmission Display Conf.*, May 2007, pp. 1–4.

[28] Y. Yang, G. Jiang, M. Yu, and D. Zhu, "Parallel process of hyper-space-based multiview video compression," in *Proc. IEEE Int. Conf. Image Process.*, Oct. 2006, pp. 521–524.

[29] R. I. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*. Cambridge, U.K.: Cambridge University Press, 2000.

[30] M. Tanimoto, "Overview of free viewpoint television," *Signal Process. Image Commun.*, vol. 21, no. 6, pp. 454–461, Jul. 2006.

[31] F. Xu and K. Mueller, "Real-time 3-D computed tomographic reconstruction using commodity graphics hardware," *Phys. Med. Biol.*, vol. 52, no. 12, pp. 3405–3419, Jun. 2007.

[32] W. Plishker, O. Dandekar, S. S. Bhattacharyya, and R. Shekhar, "Toward a heterogeneous medical image registration acceleration platform," in *Proc. IEEE Biomed. Circuits Syst. Conf.*, Montreal, Canada, 2007, pp. 231–234.

[33] W. Plishker, O. Dandekar, S. S. Bhattacharyya, and R. Shekhar, "Toward systematic exploration of tradeoffs for medical image registration on heterogeneous platforms," in *Proc. IEEE Biomed. Circuits Syst. Conf.*, Nov. 2008, pp. 53–56.

[34] W. Mark, "Future graphics architecture," *ACM Queue*, vol. 6, no. 2, pp. 54–64, Mar.-Apr. 2008.

[35] *The Virtual Reality Modeling Language*, VRML97: ISO/IEC 14772-1, 1997.

[36] *MPEG-4 Animation Framework Extension*, ISO/IEC 14996-16, 2006.

[37] *Extensible 3-D*, X-3D: ISO/IEC 19775-1, 2004.

[38] M. Deering, "Geometry compression," in *Proc. ACM Special Interest Group Comput. Graphics Interactive Tech.*, 1995, pp. 13–20.

[39] M. Isenburg and J. Snoeyink, "Coding with ASCII: Compact, yet textbased 3-D content," in *Proc. 3-D Data Process. Vis. Transmission*, Padova, Italy, 2002, pp. 609–616.

[40] E. S. Jang, J. D. K. Kim, S. Y. Jung, M.-J. Han, S. O. Woo, and S.-J. Lee, "Interpolator data compression for MPEG-4 animation," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 14, no. 7, pp. 989–1008, Jul. 2004.

[41] M. Bourges-Sevenier and E. S. Jang, "An introduction to the MPEG-4 animation framework eXtension," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 14, no. 7, pp. 928–936, Jul. 2004.

[42] C. Loop and J. Blinn, "Real-time GPU rendering of piecewise algebraic surfaces ACM transactions on graphics (TOG) archive," in *Proc. Assoc. Comput. Machinery's Special Interest Group Graphics Interactive Tech.*, vol. 25, no. 3. 2006, pp. 664–670.

[43] X. Hao and A. Varshney, "Real-time rendering of translucent meshes," *ACM Trans. Graph.*, vol. 23, no. 2, pp. 120–142, Apr. 2004.

[44] N. A. Carr, J. D. Hall, and J. C. Hart, "GPU algorithms for radiosity and subsurface scattering," in *Proc. Assoc. Comput. Machinery's Special Interest Group Graph. Interactive Tech.*, 2003, pp. 51–59.

[45] S. Williams, J. Shalf, L. Oliker, S. Kamil, P. Husbands, and K. Yelick, "Scientific computing kernels on the cell processor," *Int. J. Parallel Program.*, vol. 35, no. 3, pp. 262–298, Jun. 2007.

[46] M. Garland, S. L. Grand, J. Nickolls, J. Anderson, J. Hardwick, S. Morton, E. Phillips, Y. Zhang, and V. Volkov, "Parallel computing experiences with CUDA," *IEEE Micro*, vol. 28, no. 4, pp. 13–27, Jul.–Aug. 2008.

[47] S. Vangal, J. Howard, G. Ruhl, S. Dighe, H. Wilson, J. Tschanz, D. Finan, P. Iyer, A. Singh, T. Jacob, S. Jain, S. Venkataraman, Y. Hoskote, and N. Borkar, "An 80-Tile 1.28TFLOPS network-on-chip in 65 nm CMOS," in *Proc. IEEE Int. Solid-State Circuits Conf. Dig. Tech. Papers*, Feb. 2007, pp. 98–589.

[48] L. Seiler, D. Carmean, E. Sprangle, T. Forsyth, M. Abrash, P. Dubey, S. Junkins, A. Lake, J. Sugerman, R. Cavin, R. Espasa, E. Grochowski, T. Juan, and P. Hanrahan, "Larrabee: A many-core x86 architecture for visual computing," *IEEE Micro*, vol. 29, no. 1, pp. 10–21, Jan. 2009.

[49] W. Wolf, A. A. Jerraya, and G. Martin, "Multiprocessor system-on-chip (MPSoC) technology," *IEEE Trans. Comput. Aided Design Integr. Circuits Syst.*, vol. 27, no. 10, pp. 1701–1713, Oct. 2008.

[50] G. Blake, R. G. Dreslinski, and T. Mudge, "A survey of multicore architectures," *IEEE Signal Process. Mag.*, vol. 26, no. 6, pp. 26–37, Nov. 2009.

[51] L. J. Karam, I. AlKamal, A. Gatherer, G. A. Frantz, D. V. Anderson, and B. L. Evans, "Trends in multicore DSP platforms," *IEEE Signal Process. Mag.*, vol. 26, no. 6, pp. 38–49, Nov. 2009.

[52] P. Hofstee, "Power efficient processor architecture and the cell processor," in *Proc. 11th Int. Symp. High-Performance Comput. Architecture*, 2005, pp. 258–262.

[53] T. Chen, R. Raghavan, J. N. Dale, and E. Iwata, "Cell broadband engine architecture and its first implementation: A performance view," *IBM J. Res. Dev.*, vol. 51, no. 5, pp. 559–572, Sep. 2007.

[54] P. Kongetira, K. Aingaran, and K. Olukotun, "Niagara: A 32-way multithreaded sparc processor," *IEEE Micro*, vol. 25, no. 2, pp. 21–29, Mar.–Apr. 2005.

[55] Y. Qiu, W. Badawy, and R. Turney, "An architecture for programmable multicore IP accelerated platform with an advanced application of H.264 codec implementation," *J. Signal Process. Syst.*, vol. 57, no. 2, pp. 123–137, Nov. 2009.

[56] M. Berekovic, H. J. Stolberg, and P. Pirsch, "Multicore system: Onchip architecture for MPEG-4 streaming video," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 12, no. 8, pp. 688–699, Aug. 2002.

[57] M. Baron, "MPF 2008: Mixed architectures dominate consumer sockets," *Microprocessor Rep.*, Aug. 11, 2008.

[58] Y. Hara, "Renesas rolls multicore embedded processor," *EE Times*, Apr. 20, 2007.

[59] T.-J. Lin, C.-N. Liu, S.-Y. Tseng, Y.-H. Chu, and A.-Y. Wu, "Overview of ITRI PAC project: From VLIW DSP processor to multicore computing platform," in *Proc. IEEE Int. Symp. Very-Large-Scale Integr. Design Automation Test*, Apr. 2008, pp. 188–191.

[60] J.-M. Chen, C.-L. Chen, J.-L. Luo, P.-W. Cheng, C.-H. Yu, S.-Y. Tseng, and W.-K. Shih, "Realization and optimization of H.264 decoder for dual-core SoC," in *Proc. Int. Conf. Signal Process. Multimedia Appl. (SIGMAP)*, Barcelona, Spain, Jul. 2007, pp. 309–316.

[61] S.-Y. Tseng and M.-W. Chang, "DVFS aware techniques on parallel architecture core (PAC) platform," in *Proc. Int. Conf. Embedded Softw. Syst.*, Chengdu, Sichuan, China, Jul. 29–31, 2008, pp. 79–84.

[62] A. Kalavade and E. A. Lee, "A hardware-software codesign methodology for DSP applications," *IEEE Design Test Comput.*, vol. 10, no. 3, pp. 16–28, Sep. 1993.

[63] W. Wolf, *Computers as Components: Principles of Embedded Computing Systems Design*. San Mateo, CA: Morgan Kaufmann, 2000.

[64] W. Wolf, "A decade of hardware/software codesign," *Computer*, vol. 36, no. 4, pp. 38–43, Apr. 2003.

[65] H.-J. Stolberg, M. Berekovic, and P. Pirsch, "A platform-independent methodology for performance estimation of multimedia signal processing applications," *J. Very-Large-Scale Integr. Signal Process. Syst.*, vol. 41, no. 2, pp. 129–151, Sep. 2005.

[66] I. Barbieri, M. Bariani, A. Cabitto, and M. Raggio, "A simulation and exploration technology for multimedia-application-driven architectures," *J. Very-Large-Scale Integr. Signal Process. Syst.*, vol. 41, no. 2, pp. 153–168, Sep. 2005.

[67] G. Kahn, "The semantics of simple language for parallel programming," in *Proc. Int. Federation Inf. Process. Congr.*, vol. 74. 1974, pp. 471–475.

[68] E. A. de Kock, W. J. M. Smits, P. van der Wolf, J.-Y. Brunel, W. M. Kruijtzer, P. Lieverse, K. A. Vissers, and G. Essink, "YAPI: Application modeling for signal processing systems," in *Proc. Design Automation Conf.*, Los Angeles, CA, 2000, pp. 402–405.

[69] S. A. Blythe and R. A. Walker, "Efficient optimal design space characterization methodologies," *ACM Trans. Des. Autom. Electron. Syst.*, vol. 5, no. 3, pp. 322–336, Jul. 2000.

[70] V. Mathur and V. Prasanna, "A hierarchical simulation framework for application development on system-on-chip architectures," in *Proc. 14th Annu. IEEE Int. Appl. Specific Integr. Circuit/System Chip Conf.*, 2001, pp. 428–434.

[71] L. Benini, D. Bertozzi, D. Bruni, N. Drago, F. Fummi, and M. Poncino, "SystemC cosimulation and emulation of multiprocessor SoC designs," *Computer*, vol. 36, no. 4, pp. 53–59, Apr. 2003.

[72] L. Thiele, S. Chakraborty, M. Gries, and S. Künzli, "A framework for evaluating design tradeoffs in packet processing architectures," in *Proc. Annu. ACM IEEE Design Autom. Conf.*, New Orleans, LA, 2002, pp. 880–885.

[73] M. Gries, C. Kulkarni, C. Sauer, and K. Keutzer, "Comparing analytical modeling with simulation for network processors: A case study," in *Proc. Design Autom. Test Eur. Conf. Exhibit.*, 2003, pp. 256–261.

[74] K. Lahiri, A. Raghunathan, and S. Dey, "System-level performance analysis for designing on-chip communication architectures," *IEEE Trans. Comput.-Aided Design Integr. Circuits Syst.*, vol. 20, no. 6, pp. 768–783, Jun. 2001.

[75] J. Eker and J. W. Janneck, "CAL language report: Specification of the CAL actor language," Univ. California, Berkeley, CA, Tech. Memo UCB/ERL M03/48, Dec. 1, 2003.

[76] *Information Technology—Part 4: Codec Configuration*, document ISO/IEC 23001-4.doc, MPEG Systems Technologies, 2009.

[77] *Information Technology—Part 4: Video Tool Library*, document ISO/IEC 23002-4.doc, MPEG video technologies, 2009.

[78] S. S. Bhattacharyya, J. Eker, J. W. Janneck, C. Lucarz, M. Mattavelli, and M. Raulet, "Overview of the MPEG reconfigurable video coding framework," *J. Signal Process. Syst.*, Jul. 2009.

[79] J. W. Janneck, I. Miller, D. Parlour, M. Mattavelli, C. Lucarz, M. Wipliez, M. Raulet, and G. Roquier, "Translating dataflow programs to efficient hardware: An MPEG-4 simple profile decoder case study," in *Proc. Design Automat. Test Eur. Workshop New Wave High Level Synthesis*, Munich, Germany, Mar. 2008.

[80] J. Boutellier, C. Lucarz, S. Lafond, V. Martin Gomez, M. Mattavelli, "Quasi-static scheduling of CAL actor networks for reconfigurable video coding," *J. Signal Process. Syst.*, 2009.

[81] G. Roquier, C. Lucarz, M. Mattavelli, M. Wipliez, M. Raulet, J. W. Janneck, I. D. Miller, and D. B. Parlour: "An integrated environment for HW/SW co-design based on a CAL specification and HW/SW code generators," in *Proc. Int. Symp. Circuits Syst.*, Taipei, May 2009, p. 799.

[82] I. Amer, C. Lucarz, M. Mattavelli, G. Roquier, M. Raulet, O. Deforges, and J.-F. Nezan, "Reconfigurable video coding: The video coding standard for multicore platforms," *IEEE Signal Process. Mag.*, vol. 26, no. 6, pp. 113–123, Nov. 2009.

[83] G. G. Lee, M.-J. Wang, H.-Y. Lin, D. W.-C. Su, and B.-Y. Lin, "Algorithm/architecture co-design of 3-D spatio-temporal motion estimation for video coding," *IEEE Trans. Multimedia*, vol. 9, no. 3, pp. 455–465, Apr. 2007.

[84] M. Ravasi and M. Mattavelli, "High-abstraction level complexity analysis and memory architecture simulations of multimedia algorithms,"

IEEE Trans. Circuits Syst. Video Technol., vol. 15, no. 5, pp. 673–684, May 2005.

[85] A. D. Booth, "A signed binary multiplication technique," *Quart. J. Mech. Appl. Math.*, vol. 4, no. 2, pp. 236–240, 1951.

[86] P. C. Treleaven, "Parallel architecture overview," *Parallel Computing*, vol. 8, nos. 1–3, pp. 59–70, Oct. 1988.

[87] K. Shen, G. W. Cook, L. H. Jamieson, and E. J. Delp, "Overview of parallel processing approaches to image and video compression," in *Proc. Soc. Photo-Optical Instrum. Eng.*, vol. 2186, 1994, pp. 197–208.

[88] K. K. Parhi, *VLSI Digital Signal Processing Systems: Design and Implementation*. New York: Wiley, Jan. 1999.

[89] A. Prihozhy, M. Mattavelli, and D. Mlynek, "Evaluation of the parallelization potential for efficient multimedia implementations: Dynamic evaluation of algorithm critical path," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 15, no. 5, pp. 593–608, May 2005.

[90] J. W. Janneck, D. Miller, and D. B. Parlour, "Profiling dataflow programs," in *Proc. IEEE Int. Conf. Multimedia Expo*, 2008, pp. 1065–1068.

[91] S. Y. Kung, *VLSI Array Processor*. Englewood Cliffs, NJ: Prentice-Hall, 1988.

[92] E. van der Tol, E. Jaspers, and R. Gelderblom, "Mapping of H.264 decoding on a multiprocessor architecture," in *Proc. Soc. Photo-Optical Instrum. Eng.*, vol. 5022. 2003, pp. 707–709.

[93] Y.-K. Chen, E. Q. Li, X. Zhou, and S. L. Ge, "Implementation of H.264 encoder and decoder on personal computers," *J. Visual Commun. Image Representations*, vol. 17, no. 2, pp. 509–532, Apr. 2006.

[94] W.-N. Chen and H.-M. Hang, "H.264/AVC motion estimation implementation on compute unified device architecture (CUDA)," in *Proc. IEEE Int. Conf. Multimedia Expo*, Apr. 2008, pp. 697–700.

[95] Y.-C. Lin, P.-L. Li, C.-H. Chang, C.-L. Wu, Y.-M. Tsao, and S.-Y. Chien, "Multipass algorithm of motion estimation in video encoding for generic GPU," in *Proc. Int. Symp. Circuits Syst.*, May 2006, pp. 4451–4454.

[96] M. Kung, O. Au, P. Wong, and C. H. Liu, "Block based parallel motion estimation using programmable graphics hardware," in *Proc. Int. Conf. Audio Language Image Process.*, Jul. 2008, pp. 599–603.

[97] H. Baik, K.-H. Sihn, Y.-I. Kim, S. Bae, N. Han, and H. J. Song, "Analysis and parallelization of H.264 decoder on cell broadband engine architecture," in *Proc. IEEE Int. Symp. Signal Process. Inf. Technol.*, Dec. 2007, pp. 791–795.

[98] X. He, Y. Zhang, X. He, H. Wu, and Y. Zou, "An efficient block motion estimation method on CELL BE," in *Proc. Int. Conf. Audio Language Image Process.*, Jul. 2008, pp. 1672–1676.

[99] A. Gulati and L. Bonetto, "Harnessing parallelism from video-processing DSPs," Part 1, http://dspdesignline.com/showArticle.jhtml? articleID=191502939 and Part 2, http://www.embedded.com/columns/showArticle.jhtml? articleID=191601552

[100] Y.-K. Chen W. Li, J. Li, and T. Wang, "Novel parallel Hough transform on multicore processors," in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP)*, Mar. 2008, pp. 1457–1460.

[101] Y.-K. Chen, W. Li, and X. Tong, "Parallelization of AdaBoost algorithm on multicore processors," in *Proc. IEEE Workshop Signal Process. Syst.*, Oct. 2008, pp. 275–280.

[102] T. P. Chen, D. Budnikov, C. J. Hughes, and Y.-K. Chen. "Computer vision on multicore processors: Articulated body tracking," in *Proc. IEEE Int. Conf. Multimedia Expo*, 2007, pp. 1862–1865.

[103] O. M. Lozano and K. Otsuka, "Real-time visual tracker by stream processing," *J. Signal Process. Syst.*, vol. 57, no. 2, pp. 285–295, Nov. 2009.

[104] K. Mueller and F. Xu, "Practical considerations for GPU-accelerated CT," in *Proc. 3rd IEEE Int. Symp. Biomed. Imaging Nano Macro*, Apr. 6–9, 2006, pp. 1184–1187.

[105] G. C. Sharp, N. Kandasamy, H. Singh, and M. Folkert, "GPU-based streaming architectures for fast cone-beam CT image reconstruction and demons deformable registration," *Phys. Med. Biol.*, vol. 52, no. 19, pp. 5771–5783, Sep. 2007.

[106] M. Ohara, H. Yeo, F. Savino, G. Iyengar, L. Gong, H. Inoue, H. Komatsu, V. Sheinin, S. Daijavaa, and B. Erickson, "Real-time mutual-information-based linear registration on the cell broadband engine processor," in *Proc. 4th IEEE Int. Symp. Biomed. Imag.*, Arlington, VA, Apr. 2007, pp. 33–36.

[107] C. Meenderinck, A. Azevedo, B. Juurlink, M. Alvarez Mesa, and A. Ramirez, "Parallel scalability of video decoders," *J. Signal Process. Syst.*, vol. 57, no. 2, pp. 173–194, Nov. 2009.

[108] S.-W. Wang, S.-S. Yang, H.-M. Chen, C.-L. Yang, and J.-L. Wu, "A multicore architecture based parallel framework for H.264/AVC deblocking filters," *J. Signal Process. Syst.*, vol. 57, no. 2, pp. 195–211, Nov. 2009.

[109] W. Li, X. Tong, T. Wang, Y. Zhang, and Y.-K. Chen, "Parallelization strategies and performance analysis of media mining applications on multicore processors," *J. Signal Process. Syst.*, vol. 57, no. 2, pp. 213–228, Nov. 2009.

[110] G. G. Lee, M. J. Wang, H. Y. Lin, and R.L. Lai, "On the efficient algorithm/architecture co-exploration for complex video processing," in *Proc. IEEE Int. Conf. Multimedia Expo (ICME)*, Hannover, Germany, Jun. 2008, pp. 1057–1060.

[111] G. G. Lee, H.-Y. Lin, M.-J. Wang, B.-H. Chen, and Y.-L. Cheng, "On the verification of multistandard SOCs for reconfigurable video coding based on algorithm/architecture co-exploration," in *Proc. IEEE Workshop Signal Process. Syst. (SiPS)*, Washington D.C., Oct. 2008, pp. 170–175.

[112] G. G. Lee, M.-J. Wang, H.-Y. Lin, D. W.-C. Su, and B.-Y. Lin, "A 3-D spatio-temporal motion estimation algorithm for video coding," in *Proc. IEEE Int. Conf. Multimedia Expo (ICME)*, Toronto, Canada, Jul. 2006, pp. 741–744.

[113] G. G. Lee, K. A. Vissers, and B.-D. Liu, "On a 3-D recursive motion estimation algorithm and architecture for digital video," in *Proc. IEEE Midwest Symp. Circuits Syst. (MWCAS)*, Hiroshima, Japan, Jul. 2004, pp. 449–451.

Gwo Giun Lee (S'91–M'97–SM'07) received the B.S. degree in electrical engineering from National Taiwan University, Taipei, Taiwan, and the M.S. and Ph.D. degrees in electrical engineering from the University of Massachusetts.

He has held several technical and managerial positions in the industry; he has held the position of System Architect at Philips Semiconductors, Eindhoven, The Netherlands, and the position of DSP Architect at Micrel Semiconductors, San Jose, CA. He was a Visiting Professor in the École Polytechnique Fédérale de Lausanne, Lausanne, Switzerland, in 2007, and he also worked as the Director of the Quanta Research Center, Taiwan, before accepting his current post at the faculty of the Department of Electrical Engineering, National Cheng Kung University, Tainan, Taiwan, where he established the Media SoC Laboratory. He is also currently a Delegate with the MPEG International Standards Organization/International Electrotechnical Commission. He is the author of more than 60 technical papers. His research interests include algorithm, architecture, and the system-on-chip design for digital visual and intelligent signal processing.

Dr. Lee serves as an Associate Editor for IEEE TRANSACTIONS OF CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY. He is the main Editor for the Reconfigurable Video Coding Ad Hoc Group.

Yen-Kuang Chen (S'92–M'98–SM'02) received the B.S. degree from National Taiwan University, Taipei, Taiwan, and the Ph.D. degree from Princeton University, Princeton, NJ.

He is currently a Principal Research Scientist at Intel Corporation, Santa Clara, CA. He is one of the key contributors to Supplemental Streaming SIMD Extension 3 in the Intel Core 2 processor family. He has served as a Program Committee Member of over 35 international conferences and workshops. He holds more than 15 patents, 25 pending patent applications, and he has authored 85 technical publications. His research interests include developing innovative multimedia applications, studying the performance bottleneck in current architectures, and designing next-generation processors/platforms with many cores.

Dr. Chen serves as an Associate Editor for IEEE TRANSACTIONS ON CIRCUIT AND SYSTEM FOR VIDEO TECHNOLOGY, IEEE TRANSACTIONS ON MULTIMEDIA, and the *Journal of Signal Processing Systems*. He also served as an Associate Editor for IEEE TRANSACTIONS ON CIRCUIT AND SYSTEM I, from 2004 to 2005.

Marco Mattavelli received the Ph.D. degree from École Polytechnique Fédérale de Lausanne (EPFL), Lausanne, Switzerland, in 1996. He started his research activity at the Philips Research Laboratories, Eindhoven, The Netherlands in 1988 on channel and source coding for optical recording, electronic photography, and signal processing of high-definition television.

In 1991, he joined the "Swiss Federal Institute of Technology" (EPFL) where he got his Ph.D. in 1996. He has been a chairman of a group of MPEG ISO/IEC standardization committee. He is currently leading the Multimedia Architectures Research Group at EPFL. He is the author of more than 100 publications. His current major research interests include methodologies for specification and modeling of complex systems, architectures for video coding, high speed image acquisition and video processing systems, and applications of combinatorial optimization to signal processing.

Dr. Mattavelli received the International Standards Organization/International Electrotechnical Commission Award for his work in 1997 and 2003. He has served as an Invited Editor for several conferences and scientific journals.

Euee S. Jang (A'95) received the B.S. degree from Jeonbuk National University, Jeonju, Korea, and the M.S. and the Ph.D. degrees in electrical and computer engineering from the State University of New York, Buffalo, in 1994 and 1996, respectively.

He is currently an Associate Professor at the Department of Computer Science Education, Hanyang University, Seoul, Korea. He is the author of more than 150 MPEG contribution papers, over 30 journal and conference papers, and two book chapters. He holds 35 patents. His research interests include image and video coding, reconfigurable video coding, and computer graphics objects.

Dr. Jang has received three International Standards Organization/International Electrotechnical Commission Certificates of Appreciation for his contribution to MPEG-4 development. He also received the Presidential Award for the contribution in MPEG standardization from the Korean Government.