

# ALP: Energy-Efficient Support for All Levels of Parallelism for Complex Media Applications\*

Ruchira Sasanka    Manlap Li    Sarita V. Adve    Yen-Kuang Chen    Eric Debes  
University of Illinois at Urbana-Champaign    Intel Corporation

Real-time complex media applications are becoming increasingly common on general-purpose systems such as desktop, laptop, and handheld computers. General-purpose systems are becoming more popular for these applications because of the growing realization that programmability is important for this application domain as well, due to a wide range of multimedia standards and proprietary solutions. However, such applications demand high performance and energy efficiency from general purpose processors, especially in the mobile systems where media applications have become popular. This work seeks to develop a general-purpose architecture that can meet the performance demands of future media applications in an energy-efficient way, while also continuing to work well on other common workloads for desktop, laptop, and handheld systems.

Fortunately, most media applications have a lot of parallelism that can be exploited for energy-efficient high-performance designs. The conventional wisdom has been that this parallelism is in the form of large amounts of data-level parallelism (DLP). Much of the recent effort for media applications therefore has been on architectures that target such DLP in various ways; e.g., Imagine, VIRAM, SCALE, RAW. Most quantitative evaluations of these architectures, however, are largely based on small kernels, which are easily computed in real-time on today's general-purpose processors (e.g., speech codecs such as adpcm, color conversion such as rgb2cmyk, filters such as fir, and autocorrelation.).

Our work examines significantly more complex applications – high quality DVD resolution MPEG encode and decode, speech and face recognition, and ray tracing. (All but MPEG decode are not possible to compute in real-time today on general-purpose processors.) We find that although there is a lot of DLP in these complex applications, this DLP is often interspersed with a significant amount of control. This control-based complexity arises both from the inherently complex nature of the tasks performed and through increasingly intelligent decisions for improving performance.

This work makes two broad contributions.

## (1) A study of parallelism in complex media applications.

First, we make the case that contemporary media applications require energy-efficient support for multiple types of parallelism including ILP, TLP, and multiple forms of DLP such as sub-word SIMD (*SIMD*), vectors, streams, and vectors and streams of short vectors.

Specifically, the forms of DLP seen in our applications vary in (i) the *amount* of parallelism available and (ii) in the *similarity* of the work that needs to be done on each data element. The applications we analyzed require many forms of data-level parallelism like sub-words, vectors, or streams. Furthermore, for significant parts of several applications, ILP and TLP were the only practical forms of exploiting parallelism (e.g., Huffman coding in MPEG encode

and ray tracing).

## (2) ALP: Energy-efficient exploitation of ILP, TLP, and DLP with an evolutionary programming model and hardware.

Our second broad contribution is a complete architecture, called ALP, that effectively supports *all levels* of parallelism described above in an energy-efficient way, using an *evolutionary programming model and hardware*. Based on results from our prior work and commercial processors like Power5, ALP uses a conventional CMP/SMT substrate (CMP with superscalar SMT cores) to support TLP and ILP. ALP also supports conventional SIMD for small amounts of similar DLP. The key novelty in ALP is in its support for larger amounts of DLP through a technique we call *indexed vectors and streams*.

The programming model for indexed vectors and streams lies between SIMD and conventional vectors/streams. Indexed vectors exploit the regular data access patterns that are the hallmark of DLP by providing support for conventional vector *memory* instructions. They differ from a conventional vector implementation in that computation on indexed vector data is performed by conventional SIMD instructions. Each architectural vector register is associated with an internal hardware register that indicates the “current” element of the vector. A SIMD instruction specifying a vector register as an operand accesses and auto-increments the current element of that register. Thus, a loop containing a SIMD instruction accessing (indexed) vector register V0 marches through V0, much like a vector instruction. Indexed streams are similar to indexed vectors – the main difference is that they may have unbounded length.

Our choice of supporting vector/stream *data* without supporting vector/stream *computation* exploits a significant part of the benefits of vectors and streams on modern systems, but without need for dedicated vector/stream compute units. Specifically, ALP largely exploits existing storage and data paths in conventional superscalar systems and does not need any new special-purpose structures. ALP reconfigures part of the L1 data cache to provide an indexed vector (or stream) register file when needed.

Relative to a single-thread superscalar without SIMD, for our application suite, ALP achieves aggregate speedups from 5X to 58X, energy reduction from 1.5X to 15X, and energy-delay product (EDP) reduction of 7.3X to 873X. These results include benefits from a 4-way CMP, 2-way SMT, SIMD, and indexed vectors/streams. Our detailed results show significant benefits from each of these mechanisms. Specifically, for applications with DLP, adding indexed vector/stream support to a system with all the other enhancements in ALP achieves speedups of 1.15X to 3.24X, energy savings of 1.1X to 2.0X, and an EDP improvement of 1.24X to 6.5X (harmonic mean of 2X).

\*Submitted for publication.