System Modeling and Software Implementation of MPEG-4 Video Encoder

Literature Survey

For

EE382C Embedded Software Systems

Prof. B. L. Evans

by

Chen He and **Shi Zhong**

March 26, 2000

Abstract

MPEG-4 standard provides support for content-based interactivity, high compression, and/or universal accessibility and portability of audio and video content. Due to its content-based representation nature (except the simple profile used for wireless video communication) and flexible configuration structure, any MPEG-4 hardware implementation is likely to be very application specific. Therefore, software-based implementation seems to be a natural and viable option. In addition, a software-based approach allows flexibility and portability, which are extremely desirable features for MPEG-4 based interactive multimedia systems.

In this survey, we will review existing system-level modeling and software-based implementation approaches for real-time MPEG-4 codecs. It is the inherent parallelism and flexible configuration structure with the MPEG-4 encoder that motivate us to model the encoder using Process Networks (PN) - the concurrent computation model, and to implement a scalable software-based encoder under the framework proposed by Allen and Evans. We also intend to adopt the dynamic shape-adaptive data partitioning in each Video Object Plane (VOP) encoder to maximize the parallelism by dividing the tasks and balancing the load on each process node. Therefore, our implementation will utilize two levels of parallelism of the MPEG-4 encoder: control parallelism (VOP level) and data parallelism (Macro-Block level).

1. Introduction

MPEG-4 is an ISO/IEC standard developed by MPEG (Moving Picture Experts Group) and adopted in 1998. The mandate for MPEG-4 was to standardize algorithms for audio-visual coding in multimedia applications so as to support content-based interactivity, high compression, and/or universal accessibility and portability of audio and video contents [6,7]. The visual part of the standard provides profiles for object-based coding of natural, synthetic, and hybrid visual contents, in which bit rates targeted are within 5-64 kbps for mobile applications, 2 Mbps for TV/film applications and 19 Mbps for HDTV applications.

Due to its content-based representation nature and flexible configuration structure, MPEG-4 is considerably more complex than its predecessor standards (i.e., MPEG-1 and MPEG-2). Any hardware implementation is likely to be very application-specific. Therefore, software-based implementation seems to be a natural and viable option. In addition, a software-based approach allows flexibility and portability, which are extremely desirable features for MPEG-4 based interactive multimedia systems. The main obstacle in such an approach is that it requires a large amount of computing power to support real-time encoding and decoding operations. The latest developments in parallel and distributed multi-processor systems, however, promise possible real-time performance for computation extensive signal processing applications at an affordable cost (such as a cluster of workstations). A software-based MPEG-4 encoder implemented by He et al [4] and a real-time beamformer implemented on multiple workstations by Allen and Evans [1] are two successful examples.

In following sections, we first present a brief study of the MPEG-4 video processing standards. Then, we review and analyze the existing modeling and implementation methods for real-time signal processing applications. Finally, we propose our project plan.

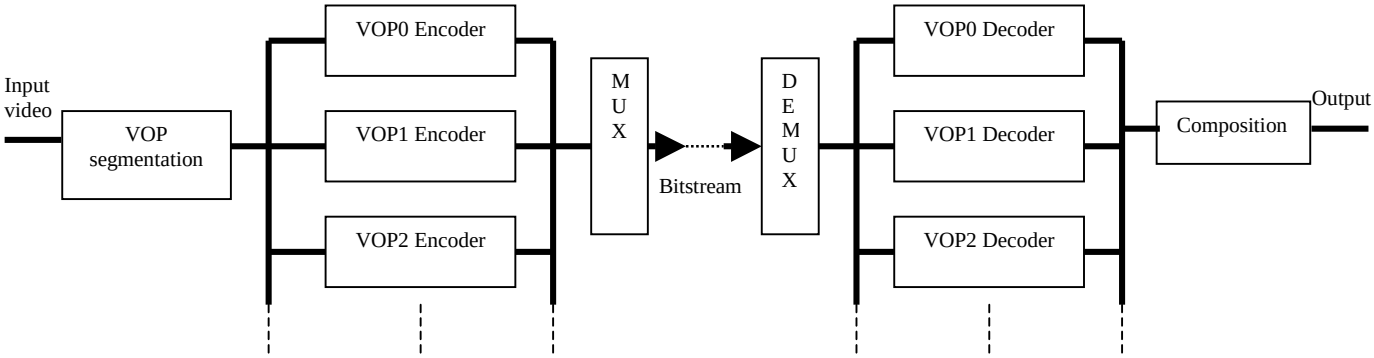

Fig.1. MPEG-4 video codec (encoder and decoder) structure

2. MPEG-4 Video Encoder

To enable the content-based interactivity, MPEG-4 Video Verification Model [6] introduces the concept of VOP (video object plane). Each frame of an input video sequence is segmented into a number of arbitrarily image regions (VOPs), with each of them possibly describing a particular image or video object of interest within scenes. So the video encoder is composed of a number of VOP encoders, which is shown in Fig.1. The inputs to the encoder are video objects (VOs) – a sequence of VOPs of arbitrary shape and position belonging to the same physical object in a scene. The shape, motion and texture information of the VOPs belonging to the same VO are encoded and transmitted or coded into a separate video object layer (VOL) to support separate decoding of VOs. The same coding scheme is applied to each video object separately. On the decoder side, the reconstructed VOs are composed together and presented to users.

The basic coding structure of a VOP encoder consists of shape coding (for arbitrarily shaped VOs), motion estimation/compensation, and DCT-based texture coding [7]. The basic diagram of MPEG-4 video encoder is shown in Fig. 2.

The encoder processes the successive images of a VOP sequence block by block. There are three kinds of macroblock (MB) within a VOP window (the window containing the VOP with the minimum number of MBs): the transparent MB, the contour MB, and the standard MB. The

contour and standard MB include the pixels belonging to the object, and the transparent MB lies completely outside the object (corresponding to the background).

The shape information of a VOP is coded prior to coding motion vectors. The shape coder performs the compression on the *alpha* plane, which contains the shape information of the VOP window. The shape coding is applied to all three kinds of MBs, while in subsequent processing steps, only the motion and texture information contained in the standard and contour MBs are coded.

Block-based motion estimation and compensation techniques are employed in MPEG-4 encoder to efficiently remove temporal redundancies of the video objects. A padding technique is applied on the reference VOP to allow polygon matching, where a polygon defines the part of the contour MBs belonging to the active area inside the VOP. The error measure widely used is the sum-of-absolute-differences (SAD) [4], which is defined as

$$SAD(x, y) = \sum_{i=1}^{N} \sum_{j=1}^{N} | B_{x,y}(i, j) - \hat{B}_{x-u,y-v}(i, j) |$$

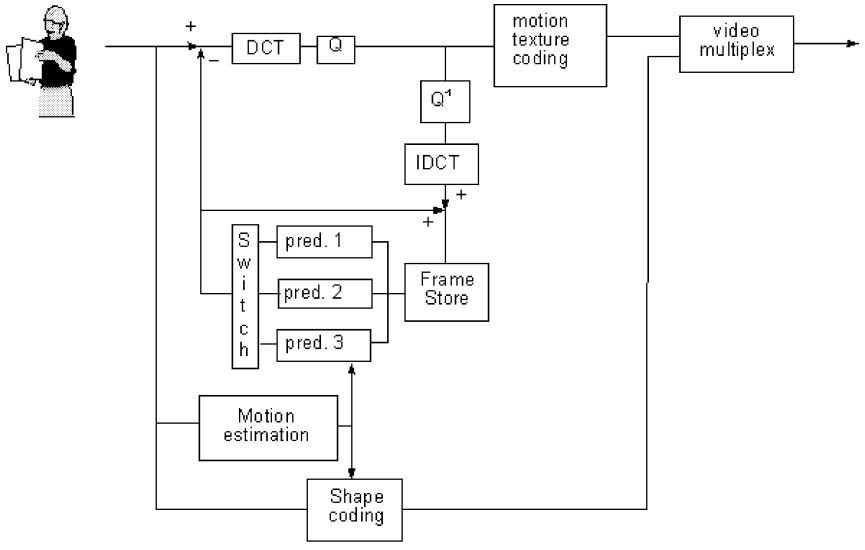

Fig. 2 Basic block diagram of MPEG-4 Video Encoder

where $B_{x,y}(i, j)$ represents the $(i, j)th$ pixel of a MB from the current picture at the spatial location (x, y), $\hat{B}_{x-u,y-v}(i, j)$ represents the $(i, j)th$ pixel of a MB from the reference picture at the spatial location (x, y) displaced by the vector (u, v), and N is either 16 or 8. The SAD calculation is only performed on the pixels inside the object.

The intra VOPs as well as the residual errors after motion-compensated prediction are coded using DCT coding on 8 × 8 blocks. Scanning of the DCT coefficients followed by quantization and run-length coding is performed using the techniques and VLC (variable length code) tables defined in the MPEG-1/2 and H.263 standards, as well as the provision for quantization matrices [2]. Alternative techniques such as shape adaptive DCT and wavelet transform may be applied for texture coding.

Other main functionalities supported by MPEG-4 video encoder are spatial and temporal scalability and error robustness at VOL and VOP level. Scalability is an important feature when the same video objects are to be made available through channels of different bandwidth or receivers of different processing capability, or to respond to different user requests.

3. System-level Modeling and Software Implementation of MPEG-4 Encoder

3.1 Model of Computations

Formal System-level modeling provides portability and scalability over heterogeneous software environments and guarantees determinacy and correctness. There exist a number of system-level computation models:

In the *Synchronous Dataflow (SDF)* formal model, each actor consumes and produces the same fixed number of tokens on each firing. The flow of data through the graph does not depend on the values of data. For the SDF model, static scheduling is possible and determination and boundedness can be determined in finite time. *Boolean Dataflow (BDF)* generalizes SDF by

adding *if-then-else* and *for-loops* constructs to support data-dependent control flow. *Dynamic Dataflow (DDF)* generalizes BDF by adding run-time recursion. Although static schedules can be constructed for some BDF and DDF graphs, these graphs are usually scheduled dynamically.

Kahn Process Network (PN) is a concurrent model of computation that is a superset of data flow models. As a directed graph, each arc represents a FIFO queue for communication and each node represents an independent, concurrent process. It is demonstrated by Allen and Evans that PN model implemented with POSIX threads can be used to realize scalable software-based real-time signal processing systems on workstations.

Petri net is a graphical and mathematical modeling tool for describing and studying systems with concurrent, asynchronous, distributed, nondeterministic and parallel characteristics [4]. It consists of *places*, *transitions*, and *arcs* that connect them. A Petri net is executed by the firing rules that transmit the marks or tokens from one place to another, and such firing is enabled only when each input place has a token inside. Thus, by using firing transition and a token distribution state, Petri nets can describe the information flow or system activities in a straightforward way.

3.2 High-level Modeling and Concurrency Analysis of MPEG-4 Encoder

Kim and Evans [5] described a generic dataflow of video codec system modeled using homogeneous SDF (HSDF), in which each functional block in Fig. 2 is implemented as a star in Ptolemy environment. Primary contributions of their work include: a) system modeling using HSDF at a high level of abstraction (thus independent of heterogeneous software platforms); and b) HSDF models are implemented in Ptolemy environment which provides system-level simulation and synthesis tools. However, no simulation results are reported.

Hamosfakidis and Paker [3] discussed the concurrency feature in an MPEG-4 video encoding task. Concurrency emerges as a basic feature in the MPEG-4 video encoding due to the nature of the standard that deals with object-oriented coding. A real-time MPEG-4 scheme

should take into account characteristics such as VOP's deadlines and precedence constraints (which can be modeled by dataflow models or Process Networks).

3.3 Implementing Real-time Signal Processing Systems on Workstations

Allen and Evans [1] proposes a framework for implementing real-time signal processing systems on workstations. It is originally designed for a real-time sonar beamformer implementation on UNIX workstations using POSIX threads and process network model. Basic ideas include:

1) Conventional implementation of UNIX operating system is not capable of deterministic real-time performance while POSIX extensions provide support for real-time applications on UNIX workstations. In this case, the workstation is both the development platform and target architecture.
2) Computational Process Network serves as the reliable formal design methodology for organizing and developing real-time multiprocessor software. It provides necessary scalability and guarantees determinate and complete execution, and bounded scheduling.
3) By carefully designing the processing node and dividing the process tasks, the parallelism is exploited and run-time overhead reduced. Real-time sonar beamforming at 4GFlops is achieved after scaled up to 12 333MHz UltraSPARC-II processors.

It is natural to apply the framework to those real-time signal processing applications that need both the formal system design and extensive computation power. Obviously, MPEG-4 encoder is one of them. The key to achieving real-time capability is to deliberately design the processing node in the model.

3.4 Software-based MPEG-4 Encoder Design

Both formal model designs and real-time software implementations for object-based MPEG-4 encoders have been sparse so far. He *et al* [4] described a software implementation on a cluster

of UNIX workstations using parallel processing. With 20 workstations, their encoder yields an encoding rate higher than real time. Linear speed-up with number of workstations has been observed. Three major steps in their design are:

1) *System modeling*: a Petri-Net-based modeling scheme is used to capture the spatio-temporal relationships between MPEG-4 video components at various levels (video session, video object, or video object plane level). This is used to guarantee the deterministic system behavior and the precedence relationships among model components.
2) *Scheduling*: an effective scheduling algorithm (a variant of *Earliest Deadline First* algorithm) is employed to allocate objects to different processors, to ensure synchronization among various objects, meet the presentation deadline requirements with guarantee of QoS (quality of service), and maximize the speed-up in terms of compression time. This is equivalent to exploiting control parallelism.
3) *Dynamic data partitioning*: by further dividing computation of one object among multiple processors, the computation load on each processor is further balanced. This is equivalent to exploiting data parallelism.

In He's implementation, real-time capability depends mainly on scheduling and partitioning, but not on the modeling part. In Allen and Evans' framework, load balancing and partitioning is integrated into the process network model and scheduling among multiple processors is done automatically, which provides a unified approach and better scalability.

4. Proposed work

The inherent parallelism and flexible configuration structure of the encoder and existing work described above motivate us to model the encoder using a formal system-level computation models, e.g. Computational Process Networks, and implement a scalable real-time software-

based encoder on workstations using the framework constructed by Allen and Evans [1]. To attain the above objectives, our project plan is:

1) Use computational process networks to model the encoder system, thus to provide determinacy, correctness, and complete and bounded execution.
2) Maximize the parallelism by dividing the tasks and balancing the load in each processing node, i.e. controlling the granularity of the processing nodes which will significantly affect the real-time capability and scalability of the implementation, and for which we intend to adopt the dynamic shape-adaptive data partitioning [4] in each VOP encoder.
3) Implement (part of) the encoder based on Pthread library and Allen's C++ implementation of the process networks [1] and analyze the simulation results.

References

1. G. E. Allen and B. L. Evans, "Real-Time Sonar Beamforming on Workstations Using Process Networks and POSIX Threads", IEEE Transactions on Signal Processing, pp. 921-926, Mar. 2000
2. G. Cote, B. Erol, M. Gallant, and F. Kossentini, "H.263+: Video Coding at Low Bit Rates," *IEEE Trans. Circuits and Systems for Video Technology*, vol. 8, no. 7, pp. 849-866, Nov. 1998.
3. A. Hamosfakidis and Y. Paker, "Concurrency analysis for real time MPEG-4 video encoding," *IEEE Int. Conf. on Multimedia Computing and Systems*, Jun. 1999, vol. 2, pp. 862-866.
4. Y. He, I. Ahmad and M. L. Liou, "A software-based MPEG-4 video encoder using parallel processing," *IEEE Trans. Circuits and Systems for Video Tech*., vol. 8, no. 7, pp. 909-920, Nov. 1998.
5. J.-I. Kim and B. L. Evans, "System modeling and implementation of a generic video codec," *Proc. IEEE Second Workshop on Multimedia Signal Processing*, Los Angeles, CA, Dec. 1998, pp. 311-316.
6. T. Sikora, "The MPEG-4 Video Standard Verification Model," *IEEE Trans. Circuits and Systems for Video Technology*, vol. 7, no.1, pp. 19-31, Feb. 1997.
7. ISO/IEC JTC1/SC29/WG11 N2995, *Overview of the MPEG-4 Standard*, http://drogo.cselt.stet.it/mpeg/standards/mpeg-4/mpeg-4.htm, Oct. 1999.