Modeling and Simulation of H.26L Encoder

Literature Survey

For

EE382C Embedded Software Systems

Prof. B.L. Evans

By

Mrudula Yadav and Gayathri Venkat

March 25, 2002

Abstract

The H.26L standard is targeted for low bit rate and low delay applications like video conferencing applications; however many other applications are considered to be within the scope of the design effort and tests indicate that H.26L is fully suitable for a very broad range of applications. The H.26L standard currently in its design stage provides improved coding efficiency and number of new features relative to prior standards. The design of H.26L is strongly intended to lead to a simple and clean solution avoiding any excessive quantity of optional features or profile configurations.

In this survey, we will review H.26L standard, existing system level modeling and software based implementation approaches for real-time video codecs. The inherent parallelism and switching block structure of the H.26L encoder motivate us to model the encoder using Boolean Dataflow model of computation and implement a software based encoder in Ptolemy Classic, a framework for simulating heterogeneous systems.

1. Introduction:

The International Telecommunications Union – Telecommunications Standardization Sector (ITU-T) Video Coding Experts Group (VCEG), officially chartered as ITU-T Q.6/SG16, is now undertaking the design of the next generation of video coding standard in a project known as “H.26L”. The primary focus of the H.26L work has been to achieve a significant improvement in compression efficiency, relative to prior standards. Other focus is to have a simple syntax specification which will lead to a simple and clean solution avoiding any excessive quantity of optional features or profile configurations.

Due to its flexibility and clean slate design, H.26L is considered to be more complex than its predecessor standards (H.261, H.263). Until recently, video encoding and decoding were only possible using Application Specific Integrated Circuits (ASICs) or using multiple DSP platforms. Although ASICs offer best speed performance, they are limited by their hardware structure. Due to the limitations of the ASIC solutions, many of the general-purpose computers have multimedia extensions. These enhancements greatly help in making real time video encoding in software, a reality [3]. The software-based implementation also allows flexibility and scalability. However, software based implementation requires extensive computation to support 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 affordable cost. A software based MPEG4 encoder by He et al [5] is a successful example.

In the following sections, we first present a brief study of H.26L 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

2. H.26L Standards:

2.1 Overview

The main goals of the new ITU -T H.26L standardization effort are, to enhance compression performance and to provide a “network-friendly” packet-based video representation addressing “conversational” (i.e., video telephony) and “non conversational” (i.e., storage, broadcast, or streaming) applications [2].

The underlying coding scheme defined by H.26L is superficially similar to schemes that are successfully employed in prior video coding standards, such as H.263 and MPEG-2. The coding scheme of H.26L includes the use of translational block-based motion compensation [1], DCT-based residual coding, scalar quantization with an adjustable step size for bit rate control, zigzag scanning, and run-length VLC coding of quantized transform coefficients. However, the clean-slate approach and specific optimizations led to some key features that differentiate H.26L from all other standards.

One fundamental concept of H.26L is the separation of the design into two distinct layers: a *video coding layer* that is responsible for efficiently representing video content, and a *network adaptation layer* that is responsible for packaging the coded data in an appropriate manner based on the network on which it is used. In this paper we focus on the video coding layer.

Motion compensation in H.26L is more flexible and efficient than in other standards [1]. Support for the use of multiple previous reference pictures for prediction is included in the core of the standard (this was previously available only in the newest high-capability version of H.263, approved in 2001). A much larger number of different motion compensation block sizes are available for motion compensation (H.263 supported two such block sizes, while H.26L

supports seven). The motion vectors can be specified with higher spatial accuracy than found in earlier standards, with quarter-pixel accuracy as the lower-complexity method and eighth-pixel accuracy sometimes available as a higher-performance method. The use of a deblocking filter within the motion compensation loop is specified in order to reduce visual artifacts and improve prediction (in-loop deblocking only appeared before in a high-capability optional mode of H.263).

H.26L is also unique in that it employs a purely integer spatial transform [1] (an approximation of the DCT) which is primarily 4x4 in shape, as opposed to the usual floating-point 8x8 DCT specified with rounding-error tolerances as used in earlier standards. The small shape helps reduce blocking and ringing artifacts, while the precise integer specification eliminates any mismatch issues between the encoder and decoder in the inverse transform.

The H.26L VCL test model has achieved a significant improvement in rate –distortion efficiency, providing nearly a factor of two in bit rate savings when comparing against the H.263+ test model [2].

2.2 H.26 L Codec:

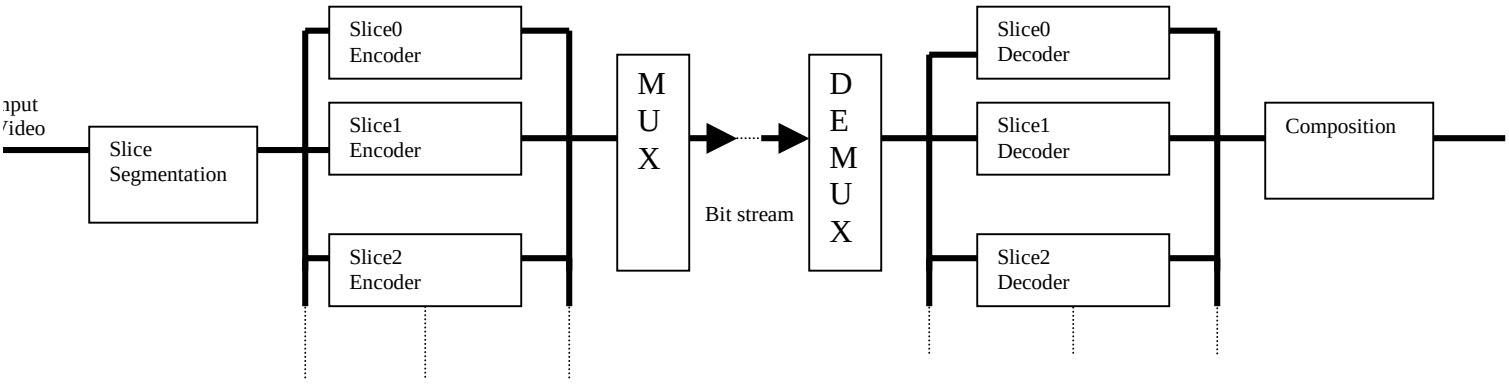

Fig 1. H.26 L video codec (Encoder and Decoder) structure

The input video sequence consists of sequence of pictures. Pictures are divided into macro blocks of 16x16 pixels. A number of consecutive macro blocks in coding order can be organized in slices. Slices represent independent coding units in a way that they can be decoded without referencing other slices of the same frame. Efficient parallel processing of the encoder can be achieved through an effective scheduling algorithm [5].

2.3 H.26L Encoder

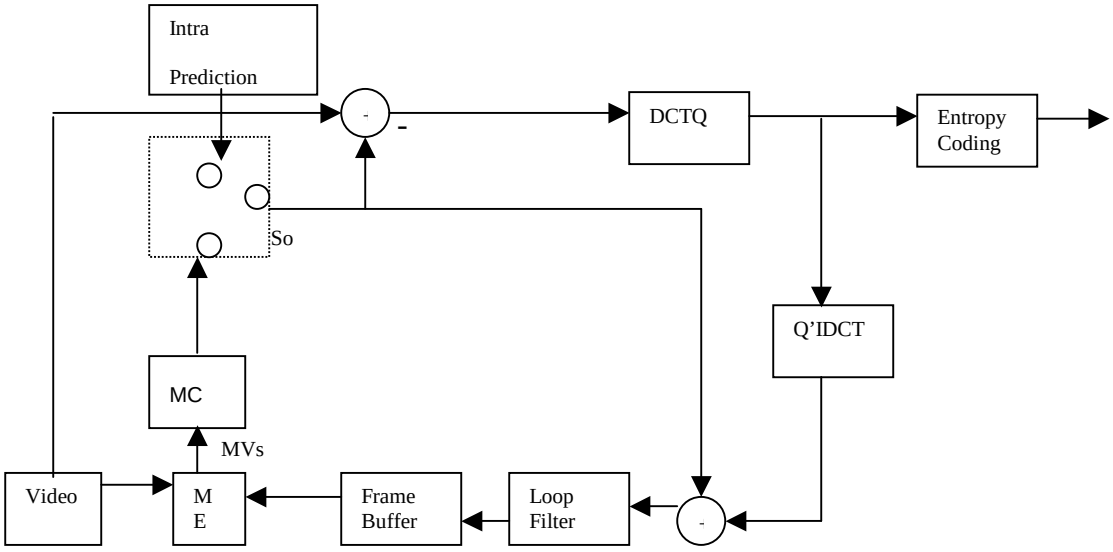

Fig 2. H.26L Encoder

Fig 2 [8] represents the diagram of the H.26L encoder. Every input macro block needs to be predicted in H.26L. The So in Fig 2 is used to select the correct prediction method for inter and intra macro block. The intra predictions are derived from the neighboring pixels in left and top blocks. The unit size of spatial prediction is either 4x4 or 16x16. As H.26L allows more than one previous frames for prediction in inter frame, Inter prediction is calculated from one of these previous frames. Seven block sizes, i.e., 16x16, 16x8, 8x16, 8x8, 8x4, 4x8 and 4x4, are supported in H.26L. The spiral search finds the minimum cost for each block size in the given range [8]. The cost includes signal SAD (Sum of Absolute Difference) and overhead bits for

coding block size information and motion vectors. The optimal block size is decided based on these minimum costs.

The residue after prediction is transformed with 4x4 integer DCT. Two different entropy-coding techniques Context-Based Adaptive Binary Arithmetic Coding (CABAC) and Universal Variable Length Coding (UVLC) are used in H.26L to compress quantized coefficients. UVLC provides a simple and robust method to code all mode information and DCT coefficients. But the performance at moderate or high bit rates is not good. Therefore, CABAC is proposed as another option in H.26L. CABAC has three distinct advantages [8]: (1) context model provides estimation of conditional probabilities of the coding symbols; (2) arithmetic code permits non-integer number of bits to be assigned to each symbol; (3) adaptive arithmetic code permits the entropy coder to adapt itself to non–stationary symbol statistics. For each intra macro block, the difference between the original image and low quality prediction is encoded.

3. System-level Modeling of H.26L Encoder

3.1 Models of Computations

Formal System-level modeling provides portability and scalability over heterogeneous software environments and guarantees determinacy and correctness [9]. 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.

3.2 Modeling of H.26L Encoder:

Kim and Evans [10] 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. However, no simulation results are reported.

The switching features [8] like deciding between the prediction modes of the standard, deciding between the best block sizes, best reference frame etc motivates us to model the encoder using BDF rather than using SDF. SDF may not be an efficient way as switching cannot be implemented using SDF. The Fig3 represents the high level BDF modeling of the macro block processing in the H.26L Encoder. The parallelization methods [3] can then be applied to improve the performance of the encoder.

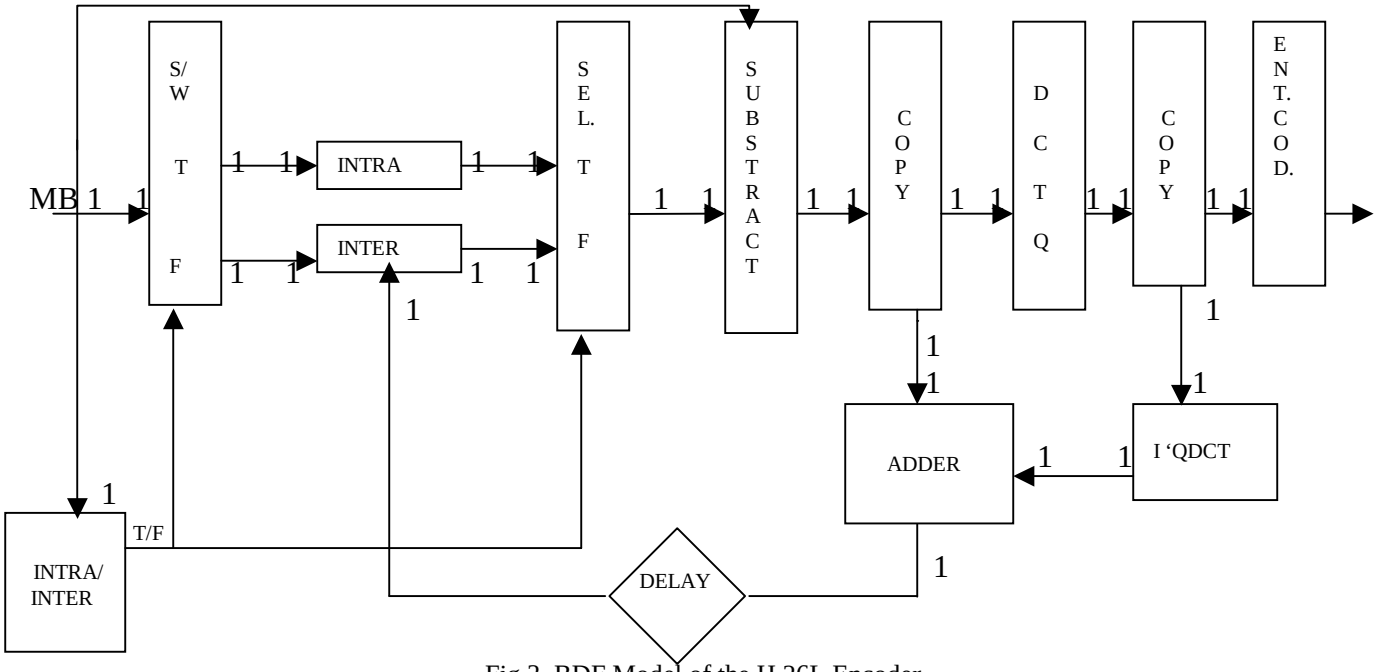

Fig 3. BDF Model of the H.26L Encoder

The firing rules for the BDF are the same as that of SDF and each actor is enabled only when it has all the inputs. The arcs in the BDF model represent the flow of data tokens from one actor to another. The token type is block based (16 *16). The block based data type is used to facilitate the computation of inverse transform/quantization. Some macro block information is also needed in the InterPrediction block and is embedded in the block-based type.

Each Macro Block from the input picture frame is taken as an input and is processed by the Intra/Inter actor which decides whether the macro block has to be predicted using Inter prediction mode or using Intra prediction mode. The intra coding actor predicts all pixels in each block in the spatial domain. There are six modes for prediction of 4x4 luminance blocks, including DC prediction (mode 0) and five directional modes, (Vertical, Diagonal, Vertical/Diagonal, Horizontal, Horizontal/Diagonal). In addition, there are four modes of prediction (Vertical, Horizontal, DC prediction, plane prediction) available for coding macro blocks of size 16x16. The inter coding actor predicts the motion vector using median prediction or direct segmentation prediction to the accuracy of ¼ pel in the low complexity mode and 1/8 pel in the higher complexity model.

The subtract actor subtracts the predicted input from the original input. The transform/ quantization actor computes the transform based on integer spatial transform that is an approximation of DCT and also does the quantization of the coefficients based on the quantization parameter. The inverse transform/quantization does the inverse of the transform and quantization and is fed as a feedback loop to the interprediction actor. There is also an initial delay token to avoid deadlock. The entropy-coding actor does either of the two types of encoding –UVLC and CABAC to further compress the quantized coefficients.

4. Ptolemy Classic:

Ptolemy Classic [6] is an environment for simulation and prototyping of heterogeneous systems. It uses modern object-oriented software technology (C++) to model each subsystem in a natural and efficient manner and to integrate these subsystems into a whole.

5. Proposed Work:

By making use of the formal model of computation [6], one can get inherent parallelism. Converting the C code to C++ and importing it into SDF actors introduces additional processing overhead, because the domain must provide a means for communication between actors. By applying a formal model of computation to the video codec and carefully modeling the actors in such a way that the overhead in synchronizing the actors is less, speed and memory efficiency could be gained which can be compared to the existing codec publicly available [7].

In order to achieve the above objectives, our project plan is to:

1. Further model the interprediction actor using the formal model of computation eg. SDF as intensive computation is involved in the interprediction actor.
2. Simulate the model using Ptolemy Classic [6] and analyze the simulation results.

References

[1] F. Kossentini, A. Joch, G. Sullivan and P. Topiwala, "Overview and performance evaluation of the ITU-T draft H.26L video coding standard," *Proc. Society of Photo-Optical Instrumentation Engineers,* Dec. 2001, vol. 4472, no. 290, pp. 290-306.

[2] G. J. Sullivan, T. Wiegand, and T. Stockhammer, "Using the Draft H.26L Video Coding Standard for Mobile Applications, " in *Proc. IEEE International Conference on Image Processing*, Thessaloniki, Greece, Sep. 2001, invited paper.

[3] K. K. Leung, H. C. Yung and Paul Y. S. Cheung, "Parallelization Methodology for Video Coding: An Implementation on the TMS320C80," *IEEE Transactions on Circuits and Systems on Video Technology*, vol.10, no.8, pp. 1431-1425, Dec. 2000.

[4] B. Erol, F. Kossentini, H. Alnuweiri, "Efficient Coding and Mapping Algorithms for Software-Only Real-Time Video coding at Low Bit Rates," *IEEE Transactions on Circuits and Systems on Video Technology*, vol.10, no.6, pp. 843-856, Sept. 2000.

[5] Y. He, I. Ahmed and M. L. Liou, "A software-based MPEG-4 video encoder using parallel processing," IEEE *Trans. Circuits and Systems for Video Techology*, vol. 8, no.7, pp. 909-920, Nov. 1998.

[6] Buck, Ha, Lee, and Messerschmitt, "Ptolemy: A mixed-paradigm simulation/prototype platform in C++," *Proceeedings of the C++ At Work Conference*, Santa Clara, CA, Nov. 1991.

[7] "The PictureTel Standards Page," http://standard.pictel.com (Current March 2002).

[8] Y. He, F. Wu, S. Li, Y. Zhong and S. Yang, "H.26L-based fine granularity scalable video coding," *http://research.microsoft.com/~fengwu/papers/h26l_iscas_02.pdf* (Current March 2002).

[9] C. He and S. Zhong, "System Modeling and Software Based Implementation of MPEG-4 Video Encoder," Proc. *IEEE Asilomar Conference on Signals, Systems and Computers*, Pacific Grove, CA, Oct. 29-Nov. 1, 2000, vol. 2, pp. 1058 –1062.

[10] 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.