Modeling and Simulation of H.26L Encoder

Final Report

For

EE382C Embedded Software Systems

Prof. B.L. Evans

By

Mrudula Yadav and Gayathri Venkat

May 8, 2002

Abstract

The H.26L standard is targeted for low bit rate and low delay video applications such as 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. Since H.26L has a wide range of applications, any hardware implementation is likely to be application specific. Therefore, software-based implementation that follows flexibility and portability seems to be a natural and viable option.

The Synchronous Data Flow (SDF) model of computation (data driven and statically schedulable) is the most suitable model for the video processing applications since the number of tokens produced and consumed by any computation block (actor) at its input/output port remains constant throughout the execution. In this report, we will present our SDF model of the H.26L encoder and the results of our simulation under multi-processor environment. Our model yields a better performance in terms of speed of execution in comparison with the original sequential implementation since the benefits from concurrent execution outweigh the overheads created by interprocessor communication.

1.Introduction

UB Video Inc., a leading provider of software solutions for video communications, announced on March 11, 2002 the availability of UBLive-26L-C64, the world's first H.26L based video processing solution on Texas Instruments TMS320C64X digital media platform, for real-time video applications, allowing users of such applications to enjoy excellent video quality on the existing low bandwidth network infrastructure [11].

The H.26L standard is being developed through the International Telecommunications Union – Telecommunications Standardization Sector (ITU-T) Video Coding Experts Group (VCEG) and the ISO/IE MPEG standardization committee. H.26L offers substantially improved coding efficiency, offering the same quality as MPEG-4 or H.263 for as little as 50% of the bandwidth required by the latter two standards [2]. The main objective behind the H.26L project is to develop a high-performance video coding standard by adopting a "back to basics" approach where simple and straightforward design using well-known building blocks is used.

The incorporation of multimedia technology in many of the general purpose computers have greatly helped in making real time video encoding in software a reality [3]. The software-based implementation as against hardware implementation allows for greater flexibility and scalability. A software-based implementation of a simple profile MPEG-4 encoder by He *et al* [9] is an example.

In this paper, we will first present the background of H.26L video processing standard, followed by a short description of H.26L encoder, which is our main focus in this project. Then we will model the H.26L encoder in SDF model and simulate the model in Ptolemy environment. We will finally test the model in the multiprocessor environment for its effectiveness and present our conclusions.

2. Overview of H.26L Video Encoder

A fundamental concept in 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 [4,8]. In this paper, we focus on the video coding layer.

Pictures are divided into macroblocks of 16*16 pixels. The H.26L standard requires every input macroblock needs to be predicted. The intra predictions, which reduce spatial redundancy, are derived from the neighboring pixels in left and top macroblocks [7]. Inter prediction (motion estimation and compensation), which reduce temporal redundancy, can be done from more than one previous frame. The standard also allows for seven different block sizes from which an optimum block size with minimum cost is found and used for prediction. The minimum cost is calculated based on the distance measure between two candidate macroblocks and overhead bits for coding block size information and motion vectors. The prediction capability of the motion compensation algorithm in H.26L is further improved by allowing motion vectors to be determined with higher levels of spatial accuracy of the order of quarter-pixel and eighth-pixel which are not found in previous standards.

The coding scheme of H.26L also includes the Discrete Cosine Transform (DCT)-based residual coding [1], scalar quantization with an adjustable step size for bit rate control and zigzag scanning.

Two different entropy-coding techniques, Context-Based Binary Adaptive Arithmetic Coding (CABAC) and Universal Variable Length Coding (UVLC) are employed in H.26L to further compress the quantized coefficients.

3. System Level Modeling of the H.26L Encoder

3.1 Synchronous Dataflow Graph

Formal system-level modeling provides portability and scalability over heterogeneous software environments and guarantees determinacy and correctness [9]. Synchronous Dataflow (SDF) model is a data driven model in which the flow of data through the graph does not depend on the values of data. Each block in the model consumes and produces the same fixed number of tokens on each input/output port on every firing. Static scheduling is possible and boundedness can be determined in finite time. As compilation can be done statically, generation of efficient code is possible. All these factors make SDF a suitable model of computation for modeling signal processing applications.

3.2 Modeling of the H.26L Video Encoder

Kim and Evans [10] described a generic dataflow of video codec system modeled using homogeneous SDF (HSDF), in which each functional block of an encoder is implemented as a star in the Ptolemy environment. Hai [11] described a SDF model of the H.263 Encoder and simulation results indicate that the model provides efficiency in terms of code size and the memory size. The successful results from the above approach and core strengths of SDF as stated above motivates us to model the H.26L encoder using the SDF model of computation.

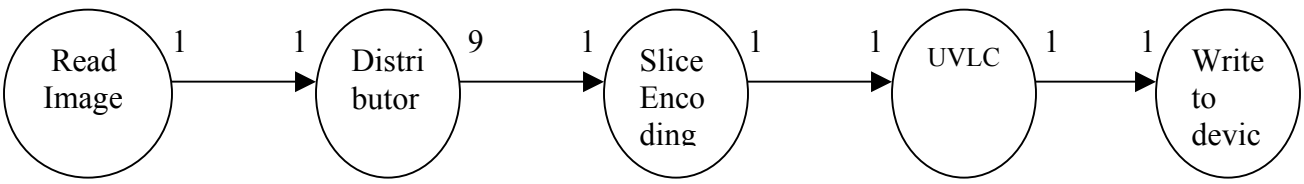


Fig. 1 SDF Model of H.26L Encoder

Zheng, *et al* [13] have exploited the GroupOfBlocks (GOB) level parallelism in the H.263 Encoder on multiprocessor workstations and the results indicate a linear speedup with a number of processors. This motivates us to exploit the inherent slice level parallelism in the H.26L Encoder since each image is made up of slices and each slice represents an independent coding unit.

Fig.1 represents the SDF modeling of the H.26L Encoder. The arcs in the SDF model represent the flow of data tokens from one block to another. Each image is of size (176*144) and is divided into slices. Each slice consists of number of macroblocks (9 in our case) of size (16*16 block of pixels). The tokens are both frame based (176*144) as well as slice based (9*(16*16)). Once the image is read from the device, the distributor segments the image into different slices and outputs to the slice encoding block. The slice encoding block processes the encoding of the macro block in a loop and the result is then entropy coded using either of the entropy coding methods – UVLC or CABAC. Fig 2 represents the homogenous SDF model of the macro block encoding. Here each token is block based of size (16*16). Each macro block is processed by the Intra/Inter block (IP), which decides whether the macro block has to be Intra predicted or Inter predicted.

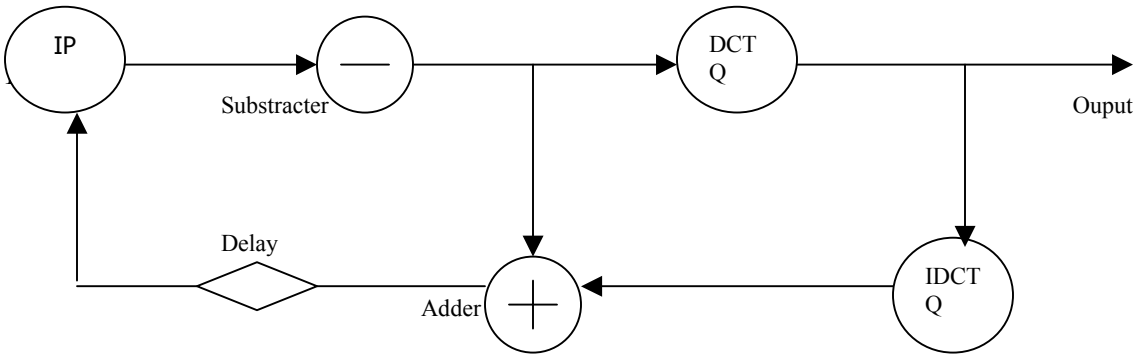


Fig. 2. SDF Model of Macro Block Encoder

Intra prediction block (IP) does the prediction of the pixels in the spatial domain. Inter prediction block (IP) computes the motion vectors based on the previous reference frames.

The subtract block subtracts the predicted input from the original input. The transform/ quantization (DCTQ) block computes the transform based on integer spatial transform that is an approximation of Discrete Cosine Transform (DCT) and also quantizes of the coefficients based on the quantization parameter. The inverse transform/quantization (IDCTQ) does the inverse of the transform and quantization and is fed as a feedback loop to the inter prediction block. There is also an initial delay token to avoid deadlock.

4. Software Implementation

Ptolemy Classic and PeaCE (Ptolemy extension as *Codesign* Environment) are the two environments to simulate the SDF model. Ptolemy Classic 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. It supports various models of computations including SDF. SDF is one of the mature domains in Ptolemy having a large library of stars and demonstrations. A model in SDF domain can be easily migrated to other domains in Ptolemy. PeaCE is a codesign environment that is built on top of Ptolemy Classic. It provides the following extra features in addition to features provided by Ptolemy in the SDF domain [11]:

1. Computation modules are specified with an extended SDF model that supports controlled global state.

2. Fractional Rate Dataflow (FRDF) Model is introduced where several optimization can be performed to minimize the memory size.

We chose Ptolemy classic [6] as against its extension PeaCE for simulation because our model did not require sharing of global space among different blocks and our scope was limited to minimize the timing required for encoding. However our model can be easily converted into a FRDF model and simulated in the PeaCE environment to optimize for buffer size management.

Our main accomplishment is the implementation of the H.26L encoder, exploiting the slice level parallelism in the algorithm using the models in Fig. 1 and Fig. 2. Software implementation involved the following steps:

- *Segmentation into SDF blocks*. The original sequential code in C is partitioned into self contained blocks. Each block is encapsulated into a C++ class and coded into the .pl files of Ptolemy.
- *Migration from SDF to CGC (Code Generation in C)*. CGC is a Code Generation domain in Ptolemy and it generates C code rather than run simulations. The conversion from SDF into the CGC domain required adding of methods like addCode(), codeBlock(), addInclude() [6] and some inclusion of macros. The go() method is modified to invoke the addCode() method.
- *Single and Mutli Processor Target*. Target architecture is one of the key features of code generation domains. Every application has a user-specified target architecture, selected from a set of targets supported by the user-selected domain which supports scheduling, compiling, assembling, and downloading code. CGC domain supports single processor target *(default-CG*) and also multi processor targets (unix_multi_C and NOWam (Networks Of Workstations Active Messages)). The generated code in the CGC domain is run under single processor target and also under multi processor targets. The first step in multiprocessor scheduling, or parallel scheduling, is to translate SDF graph to an

acyclic precedence expanded graph (APEG). The APEG describes the dependency between invocations of blocks in the SDF graph during execution of one iteration. Hence, a block (slice encoding block in our case) in a multirate SDF graph may correspond to several APEG nodes.[6]. Parallel processing is accomplished by the scheduler which maps the slice encoding block in Fig.1 onto several APEG nodes and then schedules the APEG nodes onto processors.

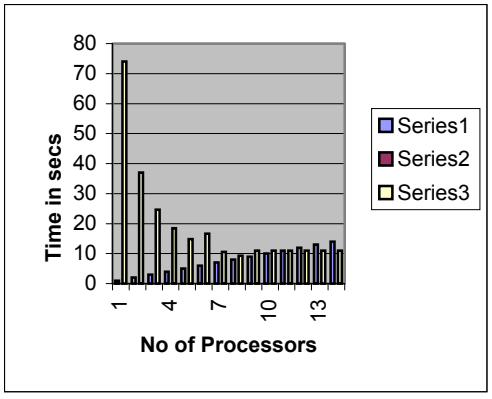
- *Generation of Gantt Chart.* Gantt Chart gives the display of the schedule of the tasks executed in different periods of time. This is automatically displayed under multi processor target.

5. Simulation Results

We tested our simulation of the SDF model in Ptolemy Classic running under Sun Solaris 2.5 on a single processor. We recorded the time required for running and analyzed the performance difference between our implementation and the original sequential implementation of the encoder. We then tested the simulation in multiprocessor CGC domain for different number of processors. Fig. 3.a shows the timing results of the tests performed. The CGC domain

Existing codec	1.41 sec
SDF	1.33 sec
CGC	0.74 sec

(a)



(b)

Image – 176 *144, Macro Blocks (MB)= 99, Slice = 9 MB

Fig 3 (a).Timing Results (b) Processor verses Time Graph

produces optimized C code which has better performance than the SDF domain in Ptolemy which is built on C++ code. Hence time required to run the simulation in CGC is less than the time required in SDF. The graph plotted with the time versus number of processors shows linear decrease in time with increase in number of processors. We think this is because the computations are distributed on a number of processors by keeping the inter processor communication minimum. However, interdependencies between macro blocks in a slice make it less beneficial to increase the number of processors more than the number of slices in a image. Here the optimum number is 9 as image consists of 9 slices.

6. Limitations

The scope of parallelism exploited in our implementation is limited to I-Frames (Intra frame). Slice Level Parallelism is ineffective for the encoding of P (backward predicted frame) and B (bi-directional predicted frame) frames because H.26L encoder allows for multiple reference frames and this might account for huge interprocessor communication overhead.

7. Conclusions and Future Work

Our model guarantees speedup in the execution time to that of the existing codec, when run in multiprocessor CGC domain. The speedup depends on the total number of slices in the input image and maximum speedup is achieved when the number of processors used is equal to the number of slices. Inter processor communication is kept low.

In addition to testing the model under other CG domains, the future work could include

- Exploiting of block-level parallelism in the computations of transforms, motion vectors and entropy coding, specifically Universal Variable Length Coding (UVLC).
- Migration of SDF model to Fractional Rate Data Flow(FRDF) model and simulation under PeaCE environment to optimize for buffer size management.

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 Proc.,*vol.3, pp.573-576, Oct. 2001.

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

[4] B. Erol, F. Kossentini and H. Alnuweiri, "Efficient Coding and Mapping Algorithms for Software-Only Real-Time Video coding at Low Bit Rates," *IEEE Trans. Circuits and Systems for Video Tech*., 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 Tech*., vol. 8, no.7, pp. 909-920, Nov. 1998.

[6] "The Ptolemy Project," http://www.ptolemy.eecs.berkeley.edu/ (Current May 2002).

[7] "The PictureTel Standards Page," *http://standard.pictel.com*, Aug, 2001.

[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 May 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 Int. Workshop on Multimedia Signal Processing*, Los Angeles, CA, Dec. 1998, pp 311-316.

[11] **"**Ubvideo**,"** *http://www.ubvideo.com/* (Current May 2002).

[12]S.Hai, "PeaCe CoDesign Environment," *http://peace.snu.ac.kr/research/peace* (Current May 2000).

[13] L.Zheng, J.Cosmas and T.Itagaki, "Real Time H.263 Video Encoder Using Mercury Multiprocessor Workstation," *http://www.cms.livjm.ac.uk/pgnet2001/papers/LZheng.pdf*, (Current May 2002).