MPEG-4 video encoder and decoder implementation on RMI Alchemy Au1200 processor for video phone applications

László Felföldi
Department of Informatics, University of Szeged
H-6720 Szeged, Árpád tér 2., Hungary
lfelfold@inf.u-szeged.hu

Key words: MPEG4, Secure Video phone, Video encoding and decoding, Optimization, MIPS32, Alchemy Au1200

Abstract: MPEG-4 Simple Profile video is widely used as video compression in video phone applications. Because of its high complexity, the design of fully standard-compliant MPEG-4 video encoder with real time speed on low-power mobile processors like ARM or RMI Alchemy family, embedded applications require optimizations at all level. This paper describes the details of the efficient implementation of MPEG-4 Simple Profile video encoder and decoder on Alchemy Au1200 core, making available to run simple secure video phone applications using QCIF resolution video at 30 frames per second

1 Introduction

With the advance of the multimedia communication technologies, video phone applications are getting polular in the past years. The increasing available network bandwidth and computational resources make the video transmission available even for mobile devices in better quality than the VCR systems could produce. MPEG-4 coding [1, 2] is widely used in video streaming applications, and serves as a the base of emerging, more sophistical standards as H.264 (MPEG-4 AVC). MPEG-4 Simple Profile coding, that requires only the implementation of a simplified subset of features, is similar to the coding used in the MPEG-1, MPEG-2, H.261 or H.263, but the improved techniques make higher compression rate or video quality available.
The Simplified version of a video phone consists of the following parts (Fig. 1):

- Camera or Video input device that imports the phisical image into digital data stream
- Video encoder device that compresses the video data into its more compact representation
- In the case of secured application, an encrypter device transcripts the encoded stream into secured stream version.
- The generated stream is forwarded to the other endpoint or endpoints using a selected network protocol, like RTP
- The stream received from the the network needs to be decrypted by the decrypter module
- The video encoder uncompress its input into the format required by the dispay unit
- The display unit or video output device makes the received video visible.

The module that reqires the most computational resources is the video encoding process. Furthermore, the quality and performance of a video phone application mostly depend on the decoder quality and performance, thus its effective implementation has great importance. To acomplish this task, algorithmic and architecture dependent optimization should be applied [3].

This paper aimes to specify an effective implementation of Simple Profile MPEG-4 video encoding and decoding on Alchemy Au1200 processor. Section 2 lists briefly the capabilities and features of the Au1200 architecture. Section 3 deals with the MPEG-4 Simple profile video decoding and its hardware implementation, while Section 4 describes the video encoding process and the possible software optimization techniques.

2 Alchemy Au1200 architecture

The Alchemy Au1200 processor is a high-performance, low-power, high integration System-on-

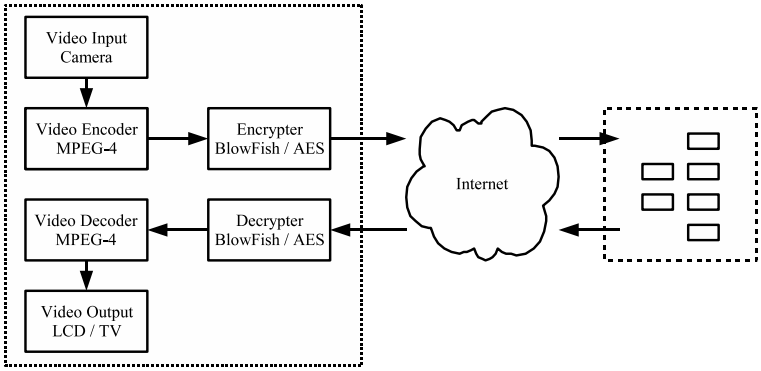

Figure 1: An example for a secure video phone application

chip targeted at personal media players, automotive information and entertainment, multimedia clients and devices where efficient digital media processing and low power are valued. For enhancing support of the special needs of these application, the Au1200 processor, besides the MIPS32 CPU core, features the following modules:

- CIM: The built-in camera interface module (CIM) can connect to a CMOS or CCD type image sensor. The CIM can be configured to convert Bayer pattern RGB or CCIR 656 protocol data into planar format and with the cooperation of the on-chip MAE Back End it can demosaic the data stream into YUV format that required by the video encoder.
- LCD: The Au1200 integrated LCD controller has the capabilities necessary for driving the latest industry standard 1-8 bit grayscale or 4-24 bit color LCD panels. The controller performs the basic memory based frame buffer to LCD panel data transfer through use of a dedicated DMA controller with double buffering support. Spatio-temporal dithering (frame rate modulation) is also supported for STN type LCD panels. The LCD controller supports four moveable overlay window, each of them can be configured to use double buffering, alpha blending, and 256-entry 32bpp palette, making the architecture capable to run user friendly graphical interfaces.
- MAE: The media acceleration engine (MAE) supports inverse quantization (IQ), inverse discrete cosine transform (IDCT), motion compensation, image scaling, image post filtering, and color space conversion. MAE operates as two independent parts, the front end and the back end. The front end includes the IQ, IDCT, and motion compensation. The back end part is responsible for the scaler/filter and color space conversion.
- AES: The on-chip 128-bit AES cryptography engine can be applied to encrypt or decrypt the specified data packet, allowing the Au1200 platform to build simple secure applications.

Although the Au1200 processor is designed for multimedia client applications, the presence of the camera interface, the features of MAE Back End module, and the realtively high execution capacity of the main core make the processor available for building secure video phone systems with QCIF video resoliton at 30 frames per second.

3 MPEG-4 Video decoding

MPEG-4 Simple Profile decoding procedure consists on bit stream parsing, Variable Length Decoding (VLD), inverse DC and AC prediction, Inverse Scanning, Inverse Quantization (IQ), Inverse Discrete Cosine Transformation (IDCT), Motion Compensation (MC) and Video Object Plane (VOP) reconstruction (Fig. 2):

- The video stream is parsed into motion and texture substreams for macroblocks.
- Motion substream describes the movement vector of the macroblock, and the decoding process copies the corresponding portion of the reference frame to the actual frame
- Texture substream holds the data that describes the macroblock completely (in the case of I-frames), or compensates the pixel differences of the moved macroblock (in the cases of P and B-frames)

Although the decoding process has high resource comsumption, the on-chip Media Accelerator Engine of the Alchemy Au1200 processor can dramatically decrease the exection time. The main features of the MAE Front End, designed to help the deconding task (Fig. 3):

- Supports up to 720 x 480 - 30 fps decoding

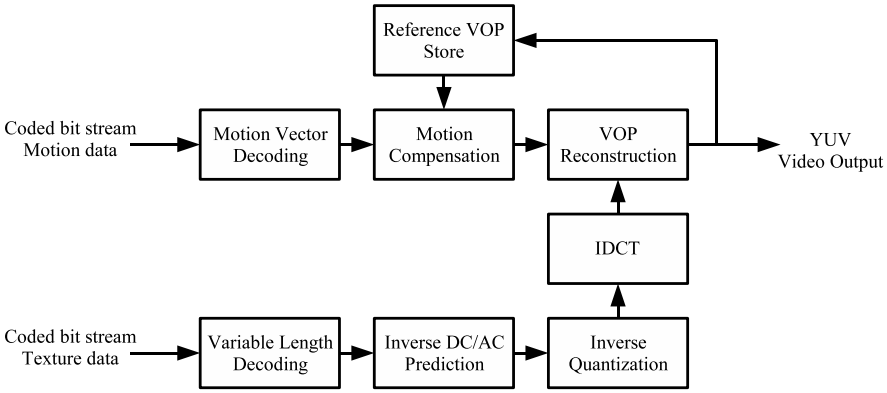

Figure 2: Diagram of MPEG-4 Simple Profile Video Decoding

- Flexible inverse quantization implementation
- Inverse discrete cosine transform
- Motion compensation for I, P, and B frames
- Support for 1, 2, and 4 motion vectors
- Support for interlaced tools (field prediction)
- Full, half, and quarter pel motion compensation
- WMV9 smoothing and in-loop deblocking filters

As the comparison of the figures of the MPEG-4 SP decoding (Fig. 2) and the MAE capabilities (Fig. 3) shows, the decoding process simplifies to the parsing of the video stream for motion vector and texture coefficients, and configuring the MAE registers.

4 MPEG4 Video Encoding

MPEG-4 Video Encoding creates the video stream from the video input stored in YUV format. Figure 4 shows the simplified diagram of the MPEG-4 Simple Profile encoding process:

- The first step is to decide the frame type (I or P).
- In the case of P frames the motion vectors for MacroBlocks should be computed. This step requires the presence of the reference frame that needs to be created in the same way as the decoder.
- Motion Compensation step calculates the differences between the corresponding region of the reference frame and the current frame.
- DCT step transforms these values into the frequency domain, and Quantization step cuts the not coefficients by reducing the storing precision. This cutting step determines the compression of the encoding of the texture data.
- The quantized values are scanned in a special (zigzag) order and then coded with VLE (Variable Length Encoding) and Huffman encoding.
- Controlling the compressing rate can be achieved by configuring the quantization or by adjusting the frame type decision (P-frames requires less bits than I-frames).

4.1 Algorithmic Level Optimization

The process of algorithmic optimization involves changing the algorithms in high-level language (usually C) to reduce the computations. Apart from implementing optimal algorithms, optimization techniques like loop unrolling, loop distribution and loop interchange are used.

4.1.1 Motion Estimation

Profiling the encoding, it can be observed that motion estimation requires major portion of the processing power. Besides the diamond searching [4] and the fast three-step searching algorithms [5], an adaptive rood pattern searching method [6] is implemented. The fast and effective execution of this task is the key point of the encoding process.

- SAD value calculation: The sum of absolute differences can be calculated using the SAD value corresponding the previous adjacent position: The absoulte difference values that belong to the block of previous position but not the actual position, should be stubstraced, while the values that belong the actual position but not the previous position, should be added to the previous SAD value to form the actual SAD value.
- Half pixel Motion Vector Estimation: This involves the refining the integer pixel motion vector to half pixel accuracy. Without finding the SAD value corresponding to all possible half-pixel motion vectors around the integer pixel motion vector, the algorithm described in [7], reduces the computational

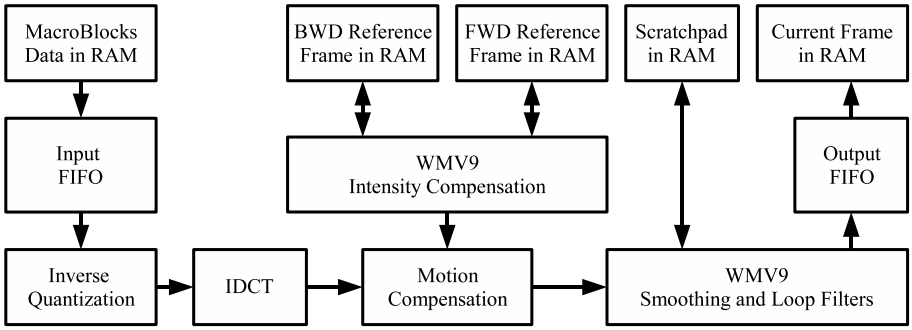

Figure 3: RMI Alchemy Au1200 Media Accelerator Engine Front End Video decoding implementation. The input of the decoding system is stored in the RAM as a list of MacroBlock descriptors containing motion vectors, texture coefficients and some additonal configration entries. Motion compensation can use both backward and forward reference frames, and some features for Windows Media 9 are also implemented.

complexity by minimum 25%, and the change in PSNR is less than $\pm$ 0.03 db.

4.1.2 DCT and Quantization

- Prediction of not coded blocks: The SAD value calculated for 8x8 blocks is applied to predict whether the DCT coefficients will be all zeroes after the quantization step. If the SAD value is less than a threshold then those blocks are treated as not-coded blocks. If the threshold value is high, then more blocks are predicted as not-coded blocks and computational complexity reduces but mis-prediction increases leading to quality degradation.
- Multiplication Instead of Division using Table lookup: On Au1200 processor the execution time for integer division operation is about 35 cycles, while a multiplication operation takes one cycle. In quantization, one division operation is required for each coefficient, this division operation could be replaced by multiplication with the reciprocal of the denominator without loss of precision using lookup table and constant data memory required is less.
- Last Position: Last position of the block is the position of the non-zero coefficient, after which all the AC coefficients have zero value. Last position of every block can be computed during quantization. Knowing last position results reduced procession time, especially when calculating Coded Block Pattern (CBP) and VLE.

4.1.3 Bitstream creation

Variable Length Encoding: The last position of the non-zero coefficient calculated during the quantization is available, so there is no need to check the values in each step.

4.2 Architecture Level Optimizations

- Inverse DCT and Inverse Quantization: The Au1200 processor built-in MAE Front End is designed to accelerate the video decoding. In the encoding process this engine can be applied to reconstuct the reference frame, significantly reducing the computational time.
- Reducing the function call overhead: Intensive loops having function calling inside in the loop are converted into loops containing the called code. Additionally, functions invoked from only one location of the code can be declared to inline, this reduces the function call overhead with out increasing the code size and without disturbing the code readability.
- Bitstream packing: Writing variable length bit sequeces into a specified bit position is an essential step of the encoding process. Reimplementing this operation in assembly can speed up the method and execution time can be decreased by 50%.

REFERENCES

[1] M. Verification, M. IEC, J. WG, and M. No. Mpeg-4 video verification model, 1997.

[2] Yen-Kuang Chen And. Implementation of real-time mpeg-4 fgs encoder.

[3] R. S. V. Prasad and K. Ramkishor. Efficient implementation of mpeg-4 video encoder on risc core.

[4] S. Zhu and K-K Ma. A new diamond search algorithm for fast block-matching motion estimation. *IEEE Trans. Image Process.*, 9(2):287–290, February 2000.

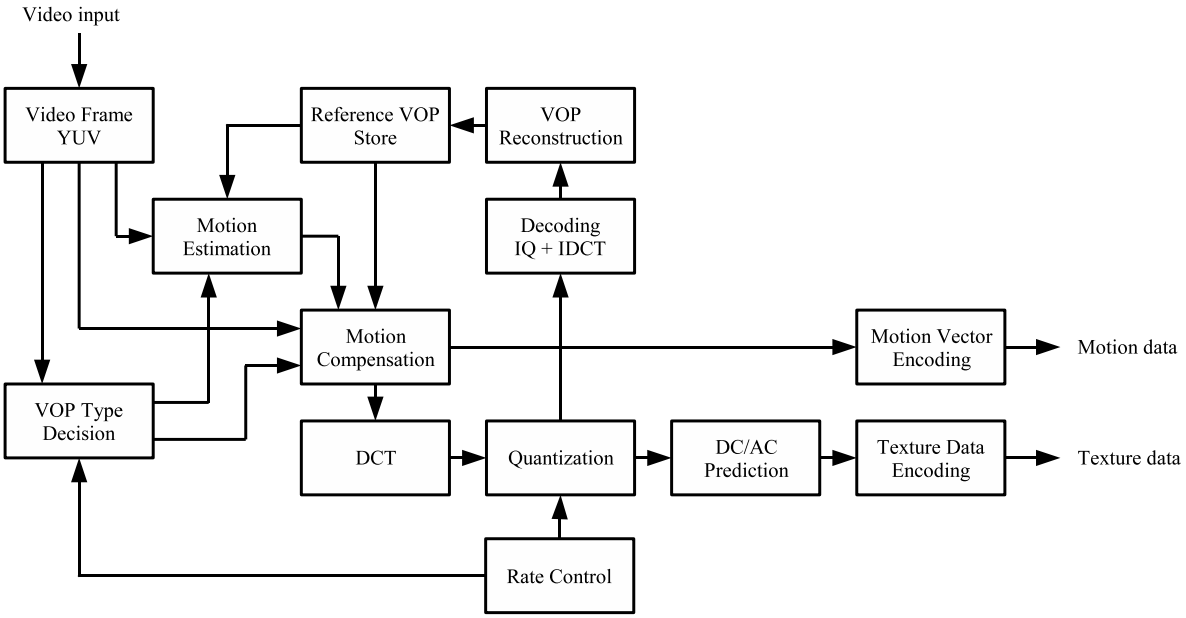

Figure 4: Simplified diagram of MPEG-4 Simple Profile Video Encoding

[5] B. Zeng R. Li and M.L. Liou. A new tree-step search algorithm for block-matching motion etimation. *IEEE Trans. Circuits Syst. Video Technol.*, 3(4):438–443, August 1994.

[6] Y. Nie and K-K Ma. Adaptive rood pattern search for fast block-matching motion etimation. *IEEE Trans. Image Process.*, 11(12):1442–1448, December 2002.

[7] PSSBK Gupta and Ramkishor Korada. Mpeg-4 video encoder on adi blackfin dsp for digital imaging applications.