REAL TIME IMPLEMENTATION OF JPEG2000

Cédric Le Barz, Rachid Elmostadi, Didier Nicholson

Thales Communications, France
135, rue des caboeufs – 92230 Gennevilliers – France
cedric.lebarz@fr.thalesgroup.com, rachid.elmostadi@fr.thalesgroup.com,
didier.nicholson@fr.thalesgroup.com

ABSTRACT

Some applications, like medical or satellite imagery systems for example, require a real time implementation of the JPEG2000 coder. This paper describes a possible implementation on a platform based on the MPC74XX processor, and a complexity study for this platform is presented. After a brief summary of the different components of a JPEG2000 coder, a description of the MPC74XX processor and an example will be given to illustrate its possibilities. Then some measurements and evaluations of the complexity of JPEG2000 will be described.

1. INTRODUCTION

JPEG2000 is a new compression image standard [1] which includes different features, like scalability in resolution or quality, error resilience for wireless transmission and region of interest.
JPEG2000 performances are better than the JPEG one [2], specially for high compression ratio to the detriment of complexity. This complexity is a problem for real time embedded implementation. That's why JPEG2000 algorithm have to be optimized to reduce computation resources and memory requirements. A preliminary study of JPE2000 complexity is presented in this paper. The work presented here has been conducted in the framework of the PRIAM IST European project (Platform for Real-time Interactive Access to Mega-images). This project will demonstrate the properties of the JPEG2000 standard for very large images in the medical and satellite domains.

2. JPEG200 OVERVIEW

The global architecture of the coder [3] is depicted in figure 1. In the case of compression of color images, the first step is a color transformation which performs a decorrelation between color components. Then an intra components decorrelation is performed: on each component a discrete wavelet transformation is realized. This discrete wavelet transform is accomplished by a bi-orthogonal filter bank. Two filter banks are used: either the (5,3) filter-bank [4] with integer taps or either the Daubechies (9,7) filter bank [5] with floating point taps. A quantization is then performed with a central deadzone followed by a bitplane based entropy coder. This entropy coder consists of a coefficient bit modeling and an arithmetic coder called the MQ coder [6]. The coefficient

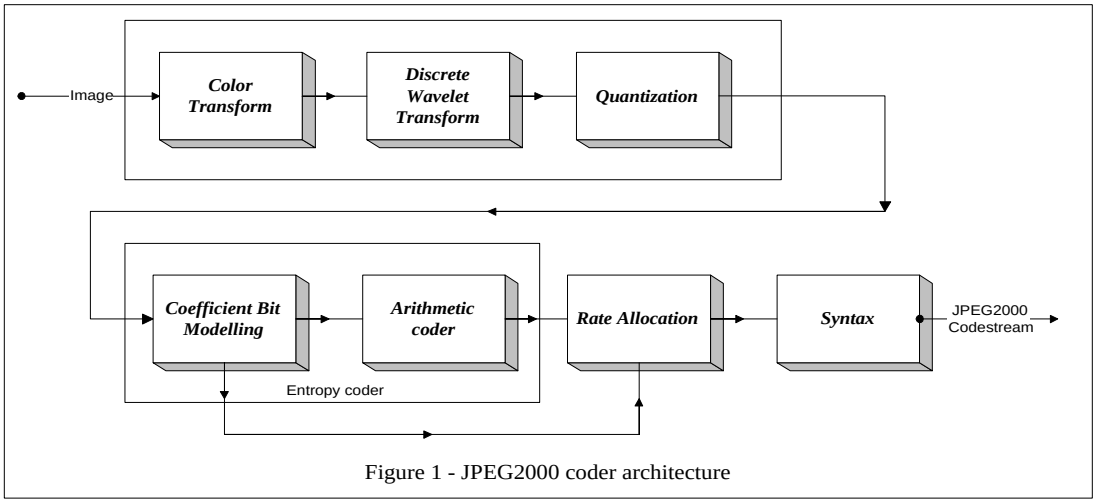

Figure 1 - JPEG2000 coder architecture

bit modeling arrange data to encode first those which contribute to the largest distortion reduction . Then data are arithmetic coded and the rate allocation is performed to select appropriate data in order to achieve the desired rate with a minimum distortion. Finally, markers are added to form the JPEG2000 codestream.

3. ALTIVEC

3.1. Description

MPC74XX processor 0 is based on a Power PC architecture which is a RISC microprocessor. A vectorial processing unit called AltiVec has been added. This unit is a Single Instruction Multiple Data (SIMD) unit which allows to compute simultaneously several data held in a vector in parallel and in a single instruction. The AltiVec unit operates on 128 bits register: a vector may hold sixteen chars, eight shorts, four long or four single precision floating points.
The MPC74XX processor consists of 8 different execution units. It is a superscalar architecture that can dispatch, execute and complete instructions simultaneously.

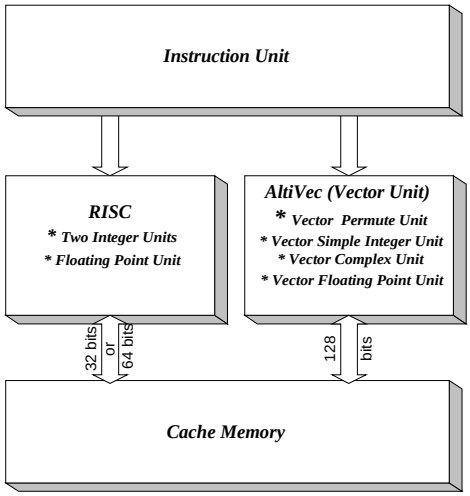

Figure 2 - MPC74XX architecture

Different units are:

- Load Store Unit (LSU)
- Floating point Unit (FPU)
- Two Integer Units (IU)
- A Vector Unit (VU), which is composed of different sub-units:
    - Vector Permute Unit (VPU)
    - Vector Simple Integer Unit (VSIU)
    - Vector Complex Integer Unit (VCIU)
    - Vector Floating Point Unit (VFPU)

Available processors are described below.

Reference	Frequency Processor (MHz)	Bus frequency Data	L1 cache (KB)	L2 cache	L3 cache
7400	350, 400, 450, 500	100/125 64 bits	32 / 32	512,1024 or 2048	
7410	400, 450, 500	100/125 64 bits	32 / 32	512,1024 or 2048	
7550	533, 667, 733, 867	133 MHz 64 bits	32 / 32	256 on chip	1024 or 2048

Table 1 - MPC74XX processor features

3.2. Example

In this section, you will find an example to illustrate the benefit of the AltiVec unit of the MPC74XX. The chosen example is the reversible color transformation (RCT).

This function performs the reversible colour transformation and a DC shift in order to centre the values around 0. The transformation is done judging from the following equations:

$$A0 = \left\lfloor \frac{R + 2 \cdot G + B}{4} \right\rfloor - 128$$

$$A1 = R - G$$

$$A2 = B - G$$

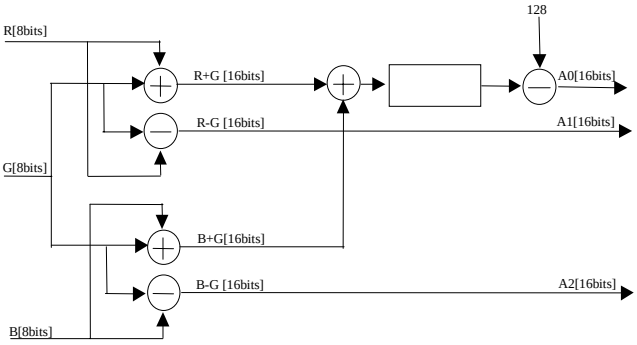

Figure 3 – Reversible Color transformation

To take advantage of the AltiVec unit, vectorisation must be realised. To process this color transformation, it is quiet simple. Instead of processing one A0 data (or A1 data or A2 data) thanks to one R data, one G data and one B data, eight A0 data (or A1 data or A2 data) are processing at the same time. Thus, comparing to a classical DSP, a gain of approximately 8 may be achieved, because vectorial operations required the same number of cycles as a RISC.

Altivec implementation:

Step 1:

- load sixteen red pixels of 8 bits in a vector Tmp_Vector_0

- load sixteen green pixels of 8 bits in a vector Tmp_Vector_1
- load sixteen blue pixels of 8 bits in a vector Tmp_Vector_2

Step 2:

- Cast the sixteen red pixels of 8 bits in two vectors R0_R7 and R8_R15. Two vectors of eight red pixels of 8 bits are got.
  - R0_R7=(vector signed short)vec_mergeh(Null_Vector,Tmp_Vector_0)
  - R8_R15=(vector signed short)vec_mergel(Null_Vector,Tmp_Vector_0)
- idem for the green and blue pixels to get G0_G7 and B8_B15

Step 3:

- compute R+G
  - RG_07_Sum=vec_add(R0_R7,G0_G7)
- compute A1=R-G
  - A1_07=vec_sub(R0_R7,G0_G7)
- compute B+G
  - BG_07_Sum=vec_add(B0_B7,G0_G7)
- compute A2=B-G
  - A2_07=vec_sub(B0_B7,G0_G7)
- compute A0=(R+G+B+G)/4
  - A0_07=vec_sra(vec_add(RG07_Sum, BG07_Sum),Shift)
- idem for A0_815, B0_815 and C0_815

Thus to compute sixteen A0 data, sixteen A1 data and sixteen A2 data, 3 loads (vec_ld), 6 casts (vec_merge), 4 additions (vec_add), 2 shifts (vec_sra) and 4 subtractions (vec_sub) are required. This represents 3x8+6+4+2+2=38 cycles which are equivalent to 38/16≈2.4 cycles per pixel. Note that data are assumed to be in the L2 cache. Dedicated operations exist to prefetch data in the cache.

4. PERFORMANCES EVALUATION AND MEASUREMENTS

Benchs have been realised on a Power Mac under MAC OS 9 and using as compilator CodeWarrior 6.0. The MPC74XX runs at 450MHz with 100 MHz bus SDRAM. The test images are “Lenna” 512x512 pixels on 8 bits (one component gray image) and “Lenna” 512x512 pixels on 24 bits (three components color image).

4.1. Discrete wavelet transform 5X3 and quantization

The following figures are the results of benchs performed on “Lenna” one component image.

Decomposition level	Time (ms)	Cycles per pixel
1	3.1	5.3
2	4.0	6.9
3	4.5	7.7
4	4.7	8.1

4.2. Discrete wavelet transform 9X7 and quantization

The following figures are the results of benchs performed on “Lenna” one component image.

.

Decomposition level	Time (ms)	Cycles per pixel
1	5.0	8.6
2	6.4	11.0
3	7.3	12.4
4	7.6	13.0

4.3. Color transformation (RCT), discrete wavelet transform 5X3 and quantization

The following figures are the results of benchs performed on “Lenna” three components image.

Decomposition level	Time (ms)	Cycles per pixel
1	18.9	32.4
2	23.5	40.3
3	24.2	41.5
4	24.5	42.1

4.4. Color transformation (ICT), discrete wavelet transform 9X7 and quantization

The following figures are the results of benchs performed on “Lenna” three components image.

Decomposition level	Time (ms)	Cycles per pixel
1	30.2	51.9
2	37.6	64.6
3	38.7	66.5
4	39.4	67.6

4.5. Entropy coding

Benchs for the entropy coder have been made for data that have been filtered by the (5,3) filter-bank. Size of code-bloc were 32X32. At a first stage, the vectorial capacities of the MPC74XX have not been used. The entropy coder includes the coefficient bit modeling and the arithmetic coder. These two algorithmic modules are made essentially of non systematic processing which disallow an easy vectorisation. Nevertheless, a first analysis shown

that part of the processing, like coefficient bit modelling, should offer vectorisation possibilities. The following results correspond to an optimised implementation using only classic processor instructions.

400 cycles per pixel are required. This figure is obviously dependant of the number of bit-planes to code and of the data which composed the bit-planes. Note that in our case (use of (5,3) filter-bank and no Region Of Interest) the number of processed bit-planes was 10. The maximum number of bit-planes that have to be processed in our case is 10 or 11. Indeed the reversible color transform generates a dynamics increasing of one bit maximum for some components, and the discrete wavelet transform (5,3 filter-bank) generates a dynamics increasing of 2 bits maximum for some subbands.

4.6. Rate allocation

The aim of the rate allocation is to minimize the distortion of a image with the desired rate or minimize the rate with the desired quality. The rate allocation proposed in JPEG2000 standard [1] is based on the Lagrange multiplier.

The complexity evaluation study of the rate allocation method proposed in the standard leads to the figure of 7 cycles per pixel. This figure depends on the code-bloc size, the code-bloc coefficients bit-depth and the number of layers. This figure have been obtained in the case of code-bloc of 32X32, code-bloc coefficients of 11 bits and in the case of one layer.

4.7. Syntax

As well as the rate allocation, a complexity evaluation study leads to the figure of 5 cycles per pixel. For this evaluation, assumptions made about the compression options are the following: one component, one tile, one layer and one precinct.

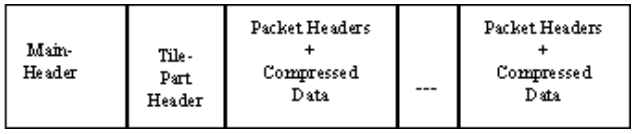

Figure 4: JPEG 2000 Codestream syntax and structure

5.COMPLEXITY REPARTITION

To code a color image (three components) of 512X512 pixels with the kernel 5X3, four decomposition levels and code-bloc of 32X32, the complexity repartition will be the following:

Function	Cycles	Complexity repartition (%)
RCT, DWT5X3 and quantization	42.1*512*512	3.4
Entropy coder	400*512*512*3	95.6
Rate allocation	7*512*512	0.6
Syntax	5*512*512	0.4

5. CONCLUSION

JPEG2000 algorithm despite its compression efficiency has the drawback of processing complexity. As shown in this document, algorithm efforts, taking into account the targeted architecture, have still to be done in order to perform a real time implementation of JPEG2000 for video image size for example. This complexity is mainly in the entropy coding. Other parts of the algorithm suit very well with an optimised implementation on a SIMD processor and particularly with the MPC74XX. Nevertheless, additional performances are required for the entropy coder which remains difficult to be optimised due to its non systematic behaviour. The first step, optimising it firstly using non-vector instructions, as presented in this document, has opened several ways for parts of it to take advantage of the vector unit architecture. This work will be conducted in the future.

6. REFERENCES

[1] ISO/IEC 15444-1/ IUT-T T.800

[2] Diego Santa-Cruz and Touradj Ebrahimi, “An analytical study of JPEG2000 functionalities”, ICIP2000

[3] Majid Rabbani and Rajan Joshi, “An overview of the JPEG2000 still image compression standard”, Signal processing: Image Communication, Eurasip, Volume17 No. 1, pp. 3-48, January2002.

[4] D. Le Gall , A. Tabatabai, “Subband coding of digital images using symmetric kernel filters and arithmetic coding techniques”, Speech Signal Processing, Proceedings of the international conference on acoustics, New York USA, pp761-764, April 1988.

[5] M. Antonini, M. Barlaud, P. Mathieu, I. Daubechies, “Image coding usinf wavelet transform”, IEEE Trans. Image Process. 1 (2), pp205-220, April 1992.

[6] David Taubman, Erik Ordentlich, Marcelo Weinberger, Gadiel Seroussi, “Embedded block coding in JPEG 2000”, Signal processing: Image Communication, Eurasip, Volume17 No. 1, pp. 49-72, January2002.

[7] Motorola Inc., “MPC7400 RISC Microprocessor User’s Manual”, MPC7400UM/D Rev0, March 2000.