EXPLORING JPEG-2000 ENTROPY CODER IMPLEMENTATIONS ON XILINX VIRTEX-II PRO PLATFORMS

O.Hammami, R. Benmouhoub and I.Aouadi
ENSTA
32 Bvd Victor 75739 Paris France
hammami@ensta.fr

ABSTRACT

The JPEG-2000 image compression standard is increasingly gaining widespread importance. The rich variety of features makes it highly suitable for a large spectrum of applications but at the same time its associated complexity makes it hard to optimize for particular implementations. One of the key step during the processing is entropy coding which takes about 70 % of the execution time. We propose in this paper an analysis and hardware design of this entropy coder in the system framework of the xilinx virtex-II pro chips.

1. INTRODUCTION

Image compression is essential in many applications ranging from PDA to satellite transmissions. The complexity of image compression algorithms as well as the various constraints on their execution and their performance requirements makes it difficult to find a one size-fits-all implementation.

The JPEG-2000 standard defines an ISO/IEC standard for the compression of still images [1]. Several features have been desired when designing the standard: (1) superior low bit rate performance (2) continuous tone and bi-level compression (3) progressive transmission by pixel accuracy and resolution (4) lossless and lossy compression (5) random code stream access and processing (6) robustness to bit-errors (7) sequential build-up capability. As described Figure 1 a JPEG-2000 encode is composed of several operations: (1) component transform (2) wavelet transform (3) quantization (4) entropy coder(tier 1 coding) (5) stream creation(tier 2 coding).

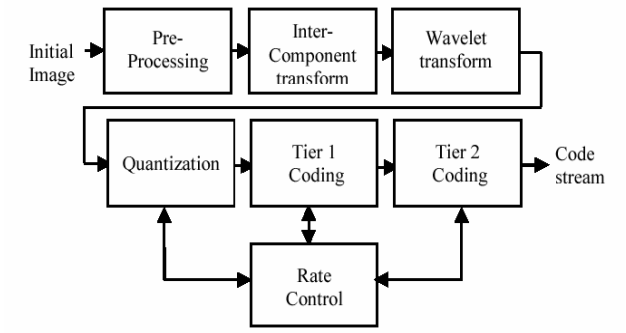

Figure 1 – JPEG 2000 encoding process

The component transform provides decorrelation among image components in order to improve compression. This step is followed by a wavelet transform of two possible types (9/7, 5/3) the first (9/7) provides the highest compression while the second 5/3 allows for lower complexity and lossless compression Both provide lower resolution images and spatial decorrelation of the images. Wavelet transform is followed by quantization with the purpose of finding a trade-off between rate and distortion. Similar to JPEG wavelet coefficients can be divided by a different value for each subband. The coefficient bit modeling step, context model step cluster the bits of the quantized wavelet coefficients into groups with similar statistics to ease the task of the arithmetic coder so that it can efficiently compress them. Each bit plane of a coefficient is processed by one of three coding passes. The last major step is the arithmetic coding of the coefficients. JPEG-2000 uses the MQ binary arithmetic coder. Finally the output of the arithmetic coder is collected into packets. Possible applications of JPEG-2000 include satellite imagery, digital camera, internet video, and mobile phone/PDA. An extension of JPEG-2000 is the Motion-JPEG2000 (MJP2). Our previous studies [4,5] on the JPEG-2000 reference software [2,3] optimization demonstrated the need for large images to rely on hardware accelerator for the most computationally intensive part of the coding process that is the entropy coder.

2. COEFFICIENT BIT MODELLING (CBM)

The components consist of a two-dimensional array of samples. Each component is divided into tiles corresponding to the tiling of the reference grid. These tile-components are coded independently. Each tile-component is wavelet-transformed into N_L decomposition levels (noted $r = 0,1,\ldots,N_L$) which are related to resolution levels. Each resolution level consists of either the HL, LH, and HH subbands from one decomposition or the N_LLL subband (lowest resolution level $r = 0$). Precinct and code-blocks are defined at the resolution level and subband. Consequenlty they can vary over tile-components. Precincts are defined so that code-blocks fit neatly with each other. Subbands are partitioned into rectangular code-blocks for the purpose of coefficient modelling and coding. The code-block size is the same from all resolution levels. Code-blocks are rectangular in shape and their nominal size is a free parameter of the coding process subject to certain constraints: (1) the nominal width and height of a code-block must be an integer power of two (2) the product of the nominal width and height must not exceed 4096. Code-blocks are decoded a bit-plane at a time starting from the most significant bit-plane with a non-zero element to the least significant bit-plane. For each bit-plane in a code-block a special code-block scan pattern is used for each of the three coding

passes. Each coefficient bit in the bit plane appears in only one of the three coding passes called significance propagation, magnitude refinement, and cleanup. For each pass contexts are created which are provided to the arithmetic coder, CX, along with the bit stream CD. Each bit-plane of a code-block is scanned in a particular order. Starting at the top left the first 4 coefficients of the first column are scanned followed by the first four coefficients of the second column and so on, until the right side of the code-block is reached.

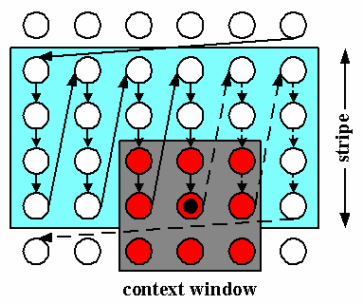

Figure 2 : Scan Order and Templates for Context Selection

The scan then returns to the left of the code-block and the second set of four coefficients in each column is scanned.The process is continued to the bottom of the code-block. If the code-block height is not divisible by 4, the last set of coefficients scanned in each column will contain fewer than 4 members. The main passes are described in the figures below.

Algorithm 1 Significance pass algorithm.

```
1:  for each sample in code block do
2:    if sample previously insignificant and predicted to become significant
      during current bit plane then
3:      code significance of sample /* 1 binary symbol */
4:      if sample significant then
5:        code sign of sample /* 1 binary symbol */
6:      endif
7:    endif
8:  endfor
```

Figure 3: Significance Pass Algorithm

Algorithm 2 Refinement pass algorithm.

```
1:  for each sample in code block do
2:    if sample found significant in previous bit plane then
3:      code next most significant bit in sample /* 1 binary symbol */
4:    endif
5:  endfor
```

Figure 4: Refinement Pass Algorithm

Algorithm 3 Cleanup pass algorithm.

```
1:  for each vertical scan in code block do
2:    if four samples in vertical scan and all previously insignificant and un-
      visited and none have significant 8-connected neighbor then
3:      code number of leading insignificant samples via aggregation
4:      skip over any samples indicated as insignificant by aggregation
5:    endif
6:    while more samples to process in vertical scan do
7:      if sample previously insignificant and unvisited then
8:        code significance of sample if not already implied by run /* 1 binary
          symbol */
9:        if sample significant then
10:         code sign of sample /* 1 binary symbol */
11:       endif
12:     endif
13:   endwhile
14: endfor
```

Figure 5 : Cleanup Pass Algorithm

3. ARITHMETIC CODER (AC)

The basic entropy encoder block diagram is shown below.

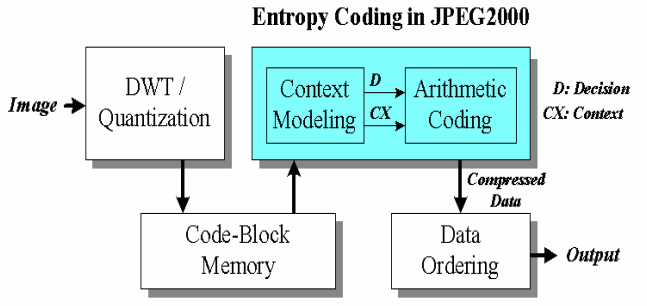

Figure 6 : Entropy Coding in JPEG 2000

Inputs of the arithmetic coder are composed of a pair a decision (D) and a context (CX) processed together to produce the compressed data.

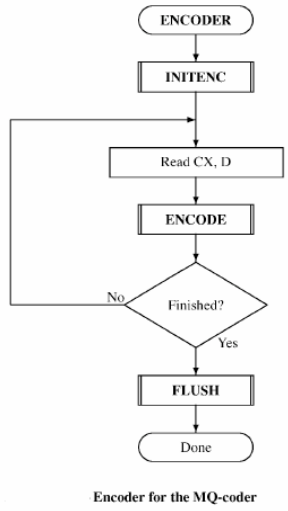

Figure 7: Main algorithm

Compressed image data. CX selects the probability estimate to use during the coding of D and represents a label for a context. The encoder will iterate on (D,CX) pairs till all pairs are read.

4. PLATFORM BASED DESIGN

The implementation of the JPEG-2000 entropy coder could have been targeted to DSP [11] or to coprocessor based executing environment following by this way the traditional accelerator approach [12-17]. We decided to think platform from the start and thus to embed the entropy coder in a platform based environment. The selected platform is the Xilinx Virtex-II platform [20]. This platform is composed by hard IPs (IBM PowerPC CPU core), soft IPs (IBM CoreConnect infrastructure) and targeted towards FPGA since it is all mapped on the Xilinx Virtex-II pro family of devices.

The Xilinx Virtex-II pro devices are new devices composed of several resources as described in Table 1.

Feature	XC2VP4	XC2VP7
Logic cells	6,768	11,088
BRAM (Kbits)	504	792
18x18 multipliers	28	44
Digital Clock Management Blocks	4	4

PowerPC processor	1	1
Max User I/O	348	396

Table 1 – Virtex II Pro Devices

The Virtex-II pro represents a powerful chip for system implementation.

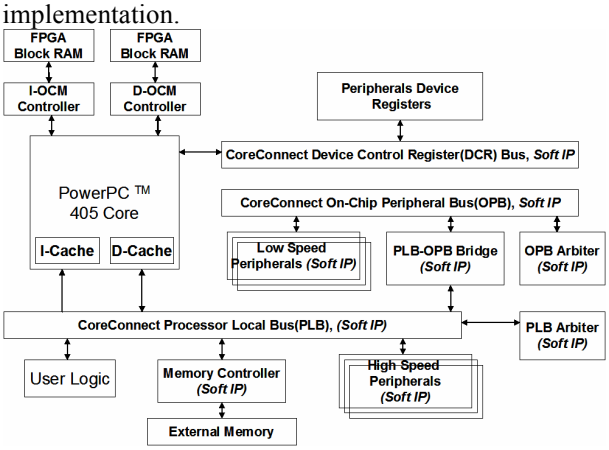

Figure 8 – Virtex II Pro Architecture

The design flow is based on the EDK tool which requires a specification of the architecture and will generate the associated vhdl file for the hardware part and an elf program for the software part. The inclusion of a user core is done through user core templates which are available in various versions (master/slave) for the OPB bus [27]. In order to evaluate our design we used the Memec board.

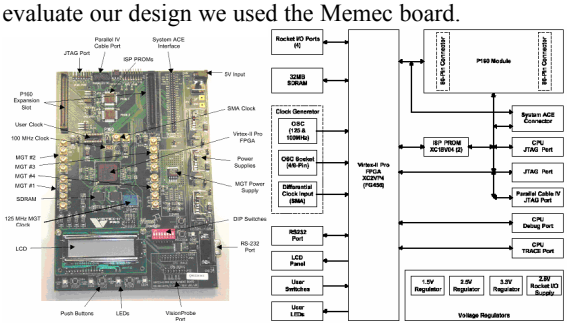

Figure 9 – Memec board

The board utilizes either the Xilinx XC2VP4-7FG456CES or the XC2VP7-7FG456CES and includes 8Mx32 SDRAM memory, three clock sources, and RS232 port and additional circuitry to develop a complete system.

5. HW/SW PARTITIONNING

In the above described platform design framework we conducted hw/sw partionning of the entropy coder. We assume that at start as in a non pipelined JPEG2000 flow all code blocks of an image are available at once for processing at the entropy coding step and are delivered by the quantification stage. We evaluated 4 versions as described in the following table.

Versions	CBM	AC
version 1	soft	soft
version 2	hard	soft
version 3	soft	hard
version 4	hard	hard

Table 2 – hw/sw partitionning versions

In version 1, the entropy coder is a full software implementation executing on the PowerPC cpu core. Data is accessed throught the data cache and comes from the external SDRAM. In version 2 the CBM is implemented as a user core on the OPB bus and interact with the AC implemented in software and executing on the PowerPC. Version 3 reverse the roles and the AC is implemented as a user core on the OPB bus while the CBM executes as software on the PowerPC. This natural partitioning between the two mains modules of the entropy coder is also based on data dependency between both modules. Finally in version 4 both the CBM and the AC are implemented as a single user core on the OPB bus.

6. RESULTS

We generated the 4 versions in the framework of the EDK 3.2.2 tool. Percentage of resources consumption for all 4 versions is described in the following figure.

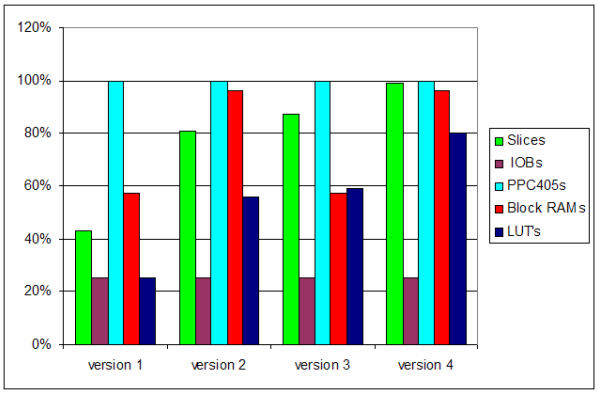

Figure 10 - Area/Resources Results

Version 1 obviously consumes the least as the implementation is done in software. However, the associated coreconnect infrastructure consumes 40% of the slices. Version 4 consumes the maximum percentage of resources as all the entropy coder is implemented in hardware while still using the powerpc for data transfers with the external memory. This also implies that in order to implement several entropy codeurs to exploit concurrent code block processing macroparallelism [15] one need to use a larger virtex II pro device than the 2vp7. Version 2 consumes more BRAM but less slices than version 3. In order to evaluate the performance of the 4 versions of the implementation we generated several code blocks from the Lena image to use as test inputs. The performance metric is the number of cycle for all the code blocks. Due to low number clock cycles of the version 4 the values are multiplied by a factor of 100. So the version 4/100 is the original version values multiplied by 100.

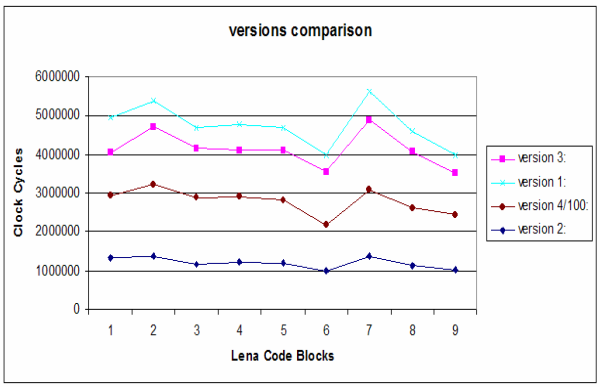

Figure 11 – Performance Results

Clearly version 4 outperforms the three other versions. The tight coupling between the coefficient bit modeling and the arithmetic coder translates in version 2 and 3 by regular data exchanges over the coreconnect structure (PLB, OPB, bridge) and therefore communication between the two modules affect performance. The nature of the entropy coder algorithm does not allow for burst transfers and in that sense burst transfer capabilities of the coreconnect structure cannot be exploited.

7. CONCLUSION

In this paper we described an exploration of hw/sw partitionning implementation for JPEG-2000 entropy coder on Xilinx Virtex-II Pro platforms. A data dependency analysis supports the use of macroparallelism at the block coder level in order to improve code block throughput during the entropy coding phase. The software alone implementation turns out to be the worst due mainly to the simple architecture of the PowerPC cpu core. The all hardware solution is the best and still brings added value with regard to an isolated IP core because it carries an analysis of its implementation in a system framework and the software part running on the PowerPC for data exchange with external devices. Fully mixed hw/sw partitioning brings mixed results. The reason comes from the fact that due to the heavy coupling and data exchange rate between CBM and AC a significant part of the execution time is spent in data transfers throught the CoreConnect structure. Besides, although the PLB monitor is part of the IBM CoreConnect in ther ASIC context it is not part of the Virtex-II Pro context.

8. REFERENCES

[1] D.Taubman and M.W.Marcellin, « JPEG2000 – Image Compression Fundamentals, Standards and Practice », Kluwer Academic Publishers, Nov. 2001.

[2] M.D.Adams, *"Jasper Software Reference Manual"*, V.1.500.4, ISO/IEC JTC1/SC29/WG1 (ITU-T SG8) N2415, Dec. 2001.

[3] ISO/IEC 9945-1: Information Technology – JPEG 2000 image coding system – Part 5: Reference Software, 2002.

[4] I. Aouadi and O.Hammami, 'Microarchitectural Characterization of JPEG-2000 Software', in Proc. of the 9th International Workshop on Systems, Signals and Image Processing 7-8 November 2002, Manchester, UK.

[5] O.Hammami, E.Zheng and I. Aouadi, ' Performance Evaluation of JPEG-2000 on VLIW Microarchitectures', in Proc, of IEEE International Conference on Microelectronics, Dec.11-13, 2002.

[6] Virtex-II Platform FPGA Handbook (http://support.xilinx.com/xlnx/xweb/xil_publications_index.jsp)

[7] FPGA Compiler II/FPGA Express VHDL Reference Manuel

[8] Synthesis and Simulation Design Guide (http://support.xilinx.com/support/sw_manuals/xilinx4/)

[9] Po-Chyi Su; Kuo, C.-C.J: Information embedding in JPEG-2000 compressed images; Circuits and Systems, 2003. ISCAS '03. Proceedings of the 2003 International Symposium on , Volume: 3 , May 25-28, 2003 Page(s): 802 -80

[10] Chung-Jr Lian; Kuan-Fu Chen; Hong-Hui Chen; Liang-Gee Chen: Analysis and architecture design of block-coding engine for EBCOT in JPEG 2000; Circuits and Systems for Video Technology, IEEE Transactions on , Volume: 13 Issue: 3 , March 2003 Page(s): 219 -230.

[11] Ramamurthy, S.; Madhavankutty, S.; Meena, V.; Gupta, R.: JPEG-2000 on an advanced architecture, multiple execution unit DSP; Circuits and Systems, 2002. ISCAS 2002. IEEE International Symposium on , Volume: 4 , 26-29 May 2002 Page(s): IV-325 -IV-328 vol.4.

[12] Hong-Hui Chen; Chung-Jr Lian; Te-Hao Chang; Liang-Gee Chen : Analysis of EBCOT decoding algorithm and its VLSI implementation for JPEG 2000; Circuits and Systems, 2002. ISCAS 2002. IEEE International Symposium on , Volume: 4 , 26-29 May 2002 Page(s): IV-329 -IV-332 vol.4.

[13] Martina, M.; Masera, G.; Piccinini, G.; Vacca, F. ; Zamboni, M : Reconfigurable coProcessor based JPEG 2000 implementation; Electronics, Circuits and Systems, 2001. ICECS 2001. The8th IEEE International Conference on , Volume: 3 , 2-5 Sept. 2001Page(s): 1227 -1230 vol.3.

[14] Kuan-Fu Chen; Chung-Jr Lian; Hong-Hui Chen; Liang-Gee Chen: Analysis and architecture design of EBCOT for JPEG-2000; Circuits and Systems, 2001. ISCAS 2001. The 2001 IEEE International Symposium on , Volume: 2 , 6-9 May 2001 Page(s): 765 -768 vol. 2.

[15] D.Taubman, E.Ordentlich, M.Weinberger G.Seroussi, Embedded Block coding in JPEG 2000, Signal processing Image Communication 17 (2002) 49-72.

[16] www.amphion.com

[17] www.alma.com

[18] C.Delhomme, 'Implementation Optimisee du Codeur Entropique JPEG-2000 sur Pentium 4'' , Rapport de Projer Personnel en Laboratoire, UEI, ENSTA, Juillet 2003.

[19] Memec : V2Pro Board User Guide 2 2. october, 2002.

[20] Xilinx : Virtex-II Pro Platform FPGA Handbook; October 2002.

[21] Xilinx : PowerPC Processor reference guide; Embeded Developpement Kit (v3.2); February 20, 2003.

[22] Xilinx : Processor ip reference guide; EDK v3.2; June, 2003.

[23] Xilinx : Getting started with EDK (v3.2); May 12, 2003.

[24] User Core Templates refernce Guide, Xilinx Corp., April 2003.

[25] The CoreConnect Bus Architecture, White Paper, IBM Microelectronics, 2001. http://www-3.ibm.com/chips/products/coreconnect/

[26] Using the PLB Performance Monitor , Application note, IBM PowerPC Family, Sept. 12, 2002.

[27] On chip Peripheral Bus , Architecture Specifications, v.2.1., IBM Microelectronics 2001.