IMPLEMENTATION OF JPEG2000 ARITHMETIC DECODER USING DYNAMIC RECONFIGURATION OF FPGA

Sophie BOUCHOUX, El-Bay BOURENNANE and Michel PAINDAVOINE

LE2I, University of Burgundy
21000 DIJON, FRANCE
sbouchou@u-bourgogne.fr, ebourenn@u-bourgogne.fr

ABSTRACT

This paper describes implementation of a part of JPEG2000 algorithm (MQ-Decoder and arithmetic decoder) on a FPGA board using dynamic reconfiguration. Comparison between static and dynamic reconfiguration is presented and new analysis criteria (time performance, logic cost, spatio-temporal efficiency) are defined. MQ-decoder and arithmetic decoder can be classified in the most attractive case for dynamic reconfiguration implementation: applications without parallelism by functions. This implementation is done on an architecture designed to study dynamic reconfiguration of FPGAs: the ARDOISE architecture. The implementation obtained, based on four partial configurations of arithmetic decoder allows reducing significantly the number of logic cells (57%) in comparison with static implementation.

1. INTRODUCTION

Nowadays, FPGAs rising performances enable to increase the design complexity. However, the FPGA logic increasing are correlated to electrical consumption, this factor could not be ignored, specially for embedded applications. The dynamic reconfiguration of FPGAs (partial or global) can be a solution that permits to use a smallest FPGA and to reconfigure it several times during the processing. Consequently an analysis to evaluate advantages and inconvenience of static and dynamic configurations of FPGAs is proposed and some analysis criteria have been defined. This paper proposed to study the dynamic reconfiguration impact on an implementation of the JPEG2000 image compression algorithm. JPEG2000 algorithm is more flexible than baseline JPEG, some parameters are adapted to obtain the better compression in function of images to process (medical, satellite, natural images, text) or of the application (numeric photography, Web images), of course JPEG2000 is also more complex and its implementation require more logic resources. The study is focusing on MQ-decoder of JPEG2000 algorithm, this part is independent of selected parameters and then its implementation can be used in many applications (it is used in the JBIG decoder). Moreover, MQ-decoder is a part of EBCOT, which is the more complex part of JPEG2000. That's why this part has been chosen to be implemented on FPGA and particularly on ARDOISE architecture (designed to study partial fine grained dynamic reconfiguration of FPGAs).

2. CRITERIA ANALYSIS OF STATIC AND DYNAMIC IMPLEMENTATIONS

The analysis of the static implementation will be carried out through the evaluation of two criteria [1]. The first represents the implementation cost and the second its performances. The cost will be designated here by the number of Configurable Logic Block or CLBs (i.e., $Cost_{CLBs} = N_{Stat\_CLBs} = computation\ cells + control\ cells$) used to implement the algorithm. Lower is this number, better is the implementation. The performance criterion can be given by the necessary execution time T_{Stat_exec} to accomplish the application. T_{Stat_exec} will be expressed only by the product of the number of pixels N (image size) to be processed and the iteration period $T_{Stat\_iteration}$ of the algorithm. In the same manner, the dynamic cost is defined by N_{Dynam_CLBs} and the performance criterion by T_{Dynam_exec}. With equal performances the contribution of the dynamic reconfiguration will be given by the ratio:

$$\frac{N_{Stat\_CLBs} - N_{Dynam\_CLBs}}{N_{Stat\_CLBs}} \times 100\% \quad (1)$$

For some applications using dynamic reconfiguration, it may be preferable to have less performance in order to benefit from reductions in cost (in number of CLBs). It is necessary, in this situation, to oppose the reduction in cost to the loss in performances. The spatial efficiency, the temporal efficiency and the spatio-temporal efficiency are defined as:

$$\rho_{spatial} = \frac{N_{actives\_CLBs}}{N_{CLBs}} \quad (2)$$

$$\rho_{temporal} = \frac{T_{exec}}{T} \quad (3)$$

$$\rho_{spatio\_temporal} = \rho_{spatial} \cdot \rho_{temporal} = \frac{N_{actives\_CLBs}}{N_{CLBs}} \cdot \frac{T_{exec}}{T} \quad (4)$$

Case with / without parallelism by functions

The application may (or not) present a pipelined structure and also its execution time on a static FPGA (in static mode) can be less then the image acquisition time ($M.N.T_{Stat\_Iteration} < T$). This case, the static implementation has a spatio-temporal efficiency lower than 1. The spatial efficiency is lower than 1 because all the tasks are loaded into the FPGA at the beginning of the application but only one is active at a time. Moreover, if the static FPGA used is too fast (compared to the application requirements), it will

0-7803-8554-3/04/$20.00 ©2004 IEEE.

be inactive during a fraction of the image acquisition time. Thus, the temporal efficiency obtained is less than 1.

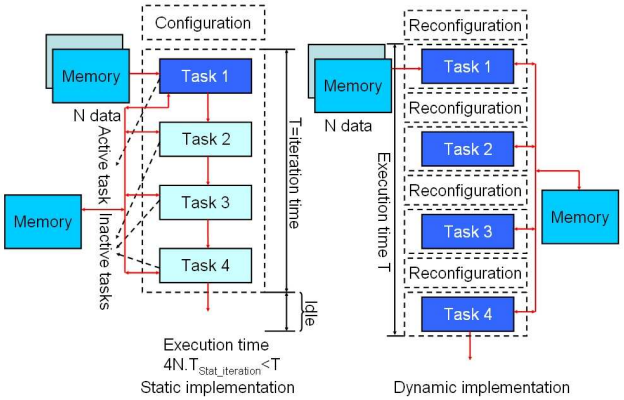

Fig. 1. Static and dynamic implementation with spatio-temporal efficiency lower than 1.

2.1. Discussion of static implementation:

The cost and the performance for the static implementation are given by equations:

$$\begin{aligned} Cost_{Stat} &= N_{Stat\_CLBs} \qquad (5)\\ &= \sum_{i=1}^{M} N_{Task_i\_CLBs} + N_{Control\_CLBs} \end{aligned}$$

$$\begin{aligned} Perf_{Stat} &= T_{Stat\_exec} \qquad (6)\\ &= M.N.T_{Stat\_iteration} + T_{idle} = T \end{aligned}$$

One can notice the presence of T_{idle} in the relation $Perf_{Stat}$. This idle time is taken into account in the static performance because it corresponds to the mismanagement of the available time.

2.2. Discussion of dynamic implementation:

In this case, it will be shown that (by using dynamic reconfiguration) logic resource decrease and processing frequency increase, with the same performances than in the static case. The iteration frequency of the dynamic implementation $f_{Dynam\_iteration}$ can even be lower than the iteration frequency of the static implementation. A faster dynamic FPGA can be also chosen (for such application) so as to decrease the number of necessary CLBs. The cost and the performance for the dynamic implementation are defined by equations:

$$\begin{aligned} Cost_{Dynam} &= N_{Dynam\_CLBs} \qquad (7)\\ &= \max_{i=1}^{M}(N_{Task_i\_CLBs} + N_{Control\_CLBs}) \end{aligned}$$

$$\begin{aligned} Perf_{Dynam} &= T_{Dynam\_exec} = T_{Stat\_exec} = T \qquad (8)\\ &= M.N.T_{Stat\_iteration} + T_{idle}\\ &= M.N.T_{Dynam\_iteration} + M.T_{Config} \end{aligned}$$

$$G_{CLBs} = \frac{N_{Stat\_CLBs} - \max_{i=1}^{M}(N_{Task_i\_CLBs})}{N_{Stat\_CLBs}} \times 100\% \qquad (9)$$

The necessary iteration period is such that:

$$T_{Dynam\_iteration} = T_{Stat\_iteration} + \frac{T_{idle}}{M.N} - \frac{T_{Config}}{N} \qquad (10)$$

The idle time of the static implementation has a positive effect on the iteration period of the dynamic implementation. If the relation (10) is verified, then the dynamic iteration frequency can be weaker than the static iteration frequency.

$$\begin{aligned} (T_{idle})_{Stat} &\geq M.T_{Config} \qquad (11)\\ \Rightarrow T_{Dynam\_iteration} &\geq T_{Stat\_iteration} \end{aligned}$$

One manages thus to obtain a gain in area and in frequency by the use of the dynamic reconfiguration.

3. MQ-DECODER IN JPEG2000 NORM

Standard JPEG2000 [2] can be divided into several successive blocks as shown in figure 2.

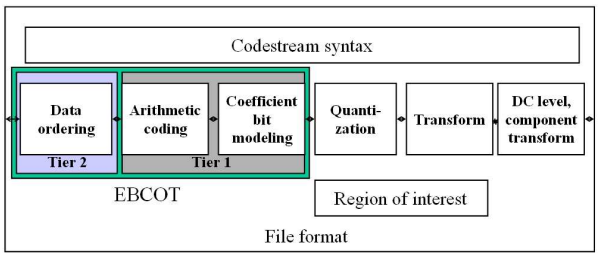

Fig. 2. JPEG2000 specification block diagram.

Original image is divided into tiles after components transform (see figure 2). A discrete wavelet transformation is then performed on each tile component and a set of two-dimensional sub-band signals is obtained, each representing activity of signal in various frequency bands, at various spatial resolutions. All resolution levels are then divided into code-blocks. Then, coefficients of these code-blocks are quantized and separated into bit planes. Starting with the most significant bit plane, they are coded according to their "significance" and their context following three successive passes (significance, magnitude refinement and cleanup passes). At this point, resulting bits from these passes and associated contexts, are sent to an arithmetic coder. Finally, these coded data are formatted, following the syntax defined in the standard to form the final bitstream. These last three modules form EBCOT (Embedded Block Coding with Optimisation Truncation). EBCOT is composed of two parts: "Tier 1" and "Tier 2". The part selected for implementation is "Tier 1", i.e., coefficients bit modelling and entropic coding. Procedures corresponding to this part of algorithm are found in [3]. The block diagram of the MQ-Decoder is shown with more details in figure 3.

After initialization, all elements of the bitstream are decoded in one of the three decoding passes which are performed successively. Contexts, i.e. state of significance of eight nearest neighbors of processed bit, are initialized and modified in decoding passes by "decoder of bit plane with adaptive context". There are 19 possible values for the context: 9 for significance pass (from 0 to 8), 5 for coding of the sign (from 9 to 13), 3 for refining pass (from 14 to 16) and 2 for cleanup pass (17 and 18). During

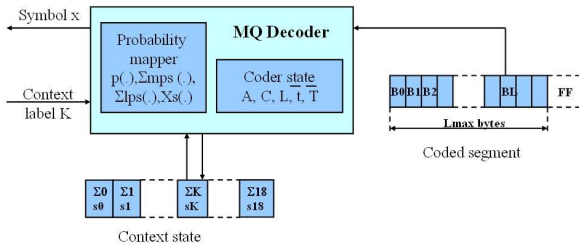

Fig. 3. MQ-Decoder schema.

these decoding passes, one procedure is called regularly: the MQ-Decode procedure. This procedure is based on the probability of contexts obtained in preceding passes.

4. ARCHITECTURE USED FOR IMPLEMENTATION: ARDOISE

In order to implement this application, architecture named ARDOISE (Architecture Reconfigurable Dynamically Oriented Image and Signal Embedded) has been done. The ARDOISE project was lead within several French laboratories. The main goal is to evaluate the performance of an architecture dedicated to the real time image processing based on FPGA dynamic reconfiguration (global or partial) [5]. The selected FPGA in this architecture is an ATMEL FPGA of the AT40K40 family. This FPGA family has been selected in view of its very precise reconfigurations: each CLB of the FPGA is configurable independently of others. For all others FPGAs families (for example XILINX), basic element of configuration is column of CLBs (named "frame" by XILINX).

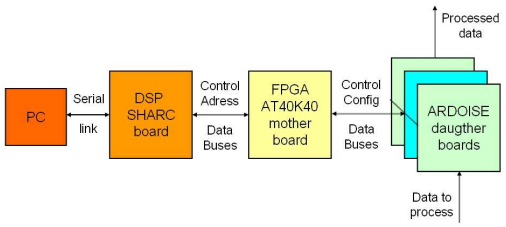

Fig. 4. Different boards of ARDOISE architecture.

DSP board is used as an interface between mother board based on an FPGA and computer. It is used to transmit configuration files to daughter boards. The main task of FPGA mother board is to manage the configuration of different daughter boards.

5. IMPLEMENTATION OF MQ-DECODER ON ARDOISE

This part of JPEG2000 algorithm requires many resources (high number of operations and memory accesses for one processed bit [7]). In proportion, EBCOT requires 70% of total computing time of decoding algorithm and wavelets transformation 20%, remaining 10% correspond to all others processes. The first part of implementation is related to MQ-Decode procedure. Indeed, this procedure is the basic element of EBCOT. It is used in different decoding passes. This procedure owns 3 distinct phases: a first phase of initialisation, MQ-Decode phase and a normalization's phase of registers. The second part of implementation is related to the use of MQ-Decode procedure in three passes of arithmetic decoder. Indeed, it is always necessary to have values related to contexts. Then, it is judicious to store these values in internal RAM memory of FPGA, and not in external one, because repeated accesses to extern storage elements are expensive both in time and in logic resources, as well as in current consumption. Thus, this part will always be inside FPGA. It will remain set while the rest of FPGA is reconfigured. Implementation on the ARDOISE board of this part of algorithm requires 4 configurations. The first one (see figure 6) corresponds to initialisation of MQ-Decoder and significance states for each element of the code-block. It is used only one time at the beginning of the decoding passes of code-block. The second one corresponds to first pass of decoding (significance), the third correspond to second pass (magnitude refinement) and the fourth corresponds to last pass (cleanup). During processing, values of contexts stored in internal RAM memory of FPGA are preserved. In these configurations, except initialisation, MQ-Decoder is always implemented in FPGA.

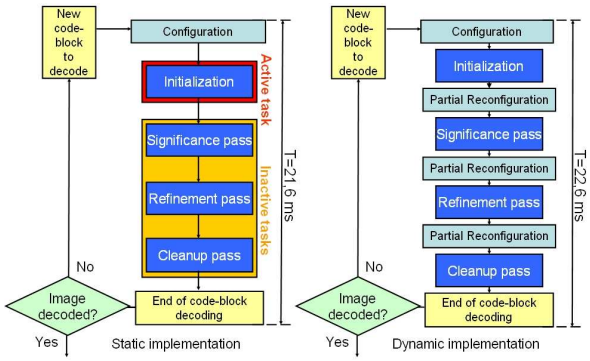

Fig. 5. Temporal representation of 4 partial configurations.

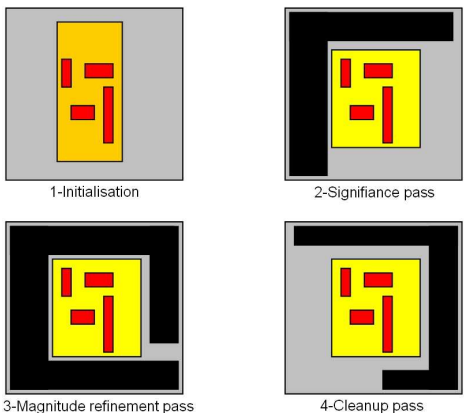

Fig. 6. Spatial representation of 4 partial configurations.

MQ-Decoder was implemented in dynamic reconfiguration (2 partial configurations) on ARDOISE architecture (1362 CLBs used out of 2304 and 21 blocks RAM used out of 144 for the first configuration, i.e. initialisation of MQ-Decoder phase, and 1447 CLBs used out of 2304 and 21 blocks RAM used out of 144 for the second configuration, i.e. main part of MQ-Decoder). The remain part "Tier 1" of EBCOT, i.e. binary modelling of coefficients

Table 1. *Comparison of results obtained for static and dynamic cases*

	Static configuration	Dynamic configuration	Gain Dynamic/ Static
Cost (CLBs)	5380	2304	57%
Performance	21.6 ms	22.6 ms	- 4.6%
$\rho_{spatial}$	0.78	1	28%
$\rho_{temporal}$	0.18	0.19	5.5%
$\rho_{spatio\_temporal}$	0.14	0.19	35.7%

(three successive passes of decoding), have been implemented by reusing MQ-Decoder implementations. Contexts are initialized at beginning of decoding of each code-block and evolve according to decoded data. It is necessary to preserve implementation of MQ-Decoder during all Tier 1 part of decoding and to use remain area of FPGA to implement with partial dynamic reconfiguration different passes of decoding one after the others. Figure 6 shows the spatial repartition of 4 configurations in FPGA and figure 5 shows the temporal repartition of tasks in static and dynamic cases. The ATMEL AT40K40 FPGA is fully reconfigurable in 337μs [8]. The maximal frequency of use for this FPGA is up to 100MHz [9]. Considered code-blocks have here the size 32×32. So, maximal number of execution cycles and then execution time in function of frequency for each pass are estimated to 721920 for a code-block's complete decoding. With a working frequency of 33MHz, computation time is 21.6ms. If working frequency is 100MHz, computation time is 7.2ms. The total reconfiguration time considered in this case, is 1ms (4 partial reconfigurations). So, execution time obtained for dynamic reconfiguration is: $T_{Dynam_exec} = 8.2$ms (at 100MHz) and $T_{Dynam_exec} = 22.6$ms (at 33MHz). In static implementation case, computation time is identical to computation time for dynamic reconfiguration because all is sequential in passes. In this case, there is no parallelism by functions and only benefit with dynamic reconfiguration is FPGA area's gain. One can consider that total area of the AT40K40 FPGA is used with dynamic reconfiguration and number of CLBs needed for static implementation of arithmetic decoder can be estimated. The cost for static implementation is obtained as follow: $Cost_{Stat} = N_{Stat\_CLBs} = 1362 + 1447 + 3 \times (2304 - 1447) = 5380$ CLBs (so, at least 3 AT40K40 FPGAs are needed for complete implementation in static case, i.e., 6912 CLBs). The dynamic implementation's cost is: $Cost_{Dynam} = N_{Dynam\_CLBs} = 2304$ CLBs. The gain in CLBs obtained by using dynamic reconfiguration is given by : $G_{CLBs} = 57\%$. With dynamic reconfiguration, CLBs' gain obtained in this particular case is 57% in comparison with static implementation of this application. The value T = 40ms (standard video) is chosen as a reference to calculate the efficiency for each case. The working frequency is 33MHz and the maximal working frequency is f_{max} = 100MHz. In this particular case, efficiency of implementation based on dynamic reconfiguration is better than efficiency of implementation based on static configuration.

6. CONCLUSION AND PERSPECTIVES

The presented implementation demonstrates that dynamic reconfiguration allows implementations of complex algorithms on small FPGAs. In this case, the selected algorithm illustrates dynamic reconfiguration advantages in applications without parallelism by functions. Using dynamic reconfiguration to obtain an implementation both MQ-Decoder and arithmetic decoder allows to save logic cells in FPGA (the CLBs' gain of dynamic reconfiguration in comparison with static configuration is 57%). Moreover, the dynamic reconfiguration optimizes the use of FPGA logic resources in time (better spatio-temporal efficiency for dynamic reconfiguration than for static one). Minimizing the logic resources can obviously decrease system price or enables the reuse of this area to implement other parts of JPEG2000. The dynamic reconfiguration is not limited by FPGA family so a comparison with Xilinx products has been performed. Dynamic reconfiguration proposed by Xilinx FPGAs, is less precise (coarse grain reconfiguration) than ATMEL FPGAs (fine grain configuration). However, Xilinx FPGA is proposing families with very high number of logical cells ($\leq$ 8 millions gates). MQ-Decoder implementation was also fully implemented in a Xilinx Virtex II XC2V250 FPGA (approximately eight times the size of AT40K40 FPGA). The implementation can be achieved with 491 slices on a total of 1536, i.e. 32% of the FPGA logical cells. The saved logic resources, will enable several MQ-Decoder blocks to be implemented, then the number of code-blocks processed in parallel is increasing and processing time for the whole image is reduce. Dynamic reconfiguration, for this type of algorithms, can represent a promising solution to reach real time performance or/and reducing production system costs (smaller FPGA).

7. REFERENCES

[1] E. Bourennane, C. Milan, M. Paindavoine, S. Bouchoux, "Real Time Image Rotation using dynamic Reconfiguration," *Real Time Imaging Journal* 8, 277 - 289, 2002.

[2] JPEG2000 Commitee, "JPEG2000 Image coding system," *JPEG2000 final commitee draft version 1.0*, 16 mars 2000.

[3] D. Taubman, M. Marcellin, "JPEG2000 Image compression fundamentals, standards and practice," *Kluwer Academic Publishers*, ISBN 079237519X, Bk&C-Rom edition, November 1, 2001.

[4] D. Taubman, "High performance scalable image compression with EBCOT," *IEEE Transaction on Image Processings*, Vol. 9, N7, 1158 - 1170, July 2000.

[5] D. Demigny, M. Paindavoine, S. Weber, "Architecture Reconfigurable Dynamiquement pour le Traitement Temps Réel des Images," *Revue Technique et Sciences de l'Information*, vol. 18, no. 10, pp. 1087 - 1112, 1999.

[6] K. Andra, C. Chakrabarti, T. Acharya, "A high performance JPEG2000 architecture," *IEEE Transactions on Circuits and Systems for video technology*, Vol. 13, N3, 209 - 218, March 2003.

[7] D. Nicholson, P. Martinez, M. Iregui, J. Corral, "Evaluation de la complexité algorithmique de JPEG2000," *Proceedings CORESA*, October 2000, Futuroscope de Poitiers, France.

[8] Atmel Corporation, "Application Note, ATMEL AT40K FPGAs, AT40K Series Configuration," *http://www.atmel.com.*

[9] Atmel Corporation, "Data Sheet, 5K - 50K Gates Coprocessor FPGA with FreeRAM," *http://www.atmel.com.*