A DATAFLOW DESCRIPTION OF ACC-JPEG CODER

Tarek OUNI *Khaled JERBI* *Mohamed ABID*

CES Lab. National Engineering school of Sfax
email: $tarek.ouni@gmail.com$ $khaled\_jerbi@yahoo.fr$ $mohamed.abid@enis.rnu.tn$

ABSTRACT

Video codec standards evolution raises two major problems. The first one is the design complexity which makes very difficult the fast implementation of coders. The second is the computing capability demanding which requires complex and advanced architectures. To decline the first problem, MPEG normalized the Reconfigurable Video Coding (RVC) standard which allows the reutilization of some generic image processing modules for advanced video coders. However, the second problem still remains unsolved. Technology development becomes incapable to answer the current standards algorithmic increasing complexity. In this paper, we propose an efficient solution for the two problems by applying the RVC methodology and its associated tools on a new video coding model called Accordion based video coding. The main advantage of this video coding model consists in its capacity of providing high compression efficiency with low complexity which is able to resolve the second video coding problem.

Index Terms— JPEG, Accordion, RVC-CAL, Data flow computing, CAL compiler

1. INTRODUCTION

During most of two decades, MPEG has produced several video coding standards such as MPEG-2, MPEG-4, AVC and SVC. However, the past monolithic specification of such standards (usually in the form of C/C++ programs) lacks flexibility. Such specification does not allow to use the combination of coding algorithms from different standards enabling to achieve specific design or performance trade-offs and thus fill, case by case, the requirements of specific applications. So as to overcome the intrinsic limitations of specifying codecs algorithms by using monolithic imperative code, Caltrop Actor Language (CAL) [1] [2], has been chosen by the ISO/IEC standardization organization in the new MPEG standard called Reconfigurable Video Coding (RVC). RVC-CAL standard is developped in the ptolemy2 project [3]. It is supported with a complete framework called OpenDF [4] and a recently developed compiler called Open RVC-CAL Compiler (Orcc) [5] allowing users to define a multitude of codecs, by combining together actors (called coding tools in RVC) from the MPEG standard library written in CAL [1] [2], that contains video technology from all existing MPEG video past standards (i.e. MPEG- 2, MPEG- 4, etc.). CAL is used to provide the reference software for all coding tools of the entire library. The originality of this paper is the application of the CAL and its associated tools on a new video coding model called Accordion based video coding [6] [7] [8]. The main advantage of this video coding model consists in its capacity of providing high compression efficiency with low complexity. Such advantage comes from the original idea that consists in applying a particular 3D scan, called Accordion, on temporal frames generated by a temporal video decomposition which is able to transform a set of video frames sequence to a high correlated spatial representation called IACC. The high correlation of the IACC representation, which is actually originated from the video temporal correlation, is easy and efficiently exploited by any still image coder. It was shown in [8] that such coder could produce a high compression ratio (close to Inter compression models such as MPEG) with low computational requirements (close to Intra compression models such as M-JPEG). Additionally, this model offers the possibility of easy reutilizing different mature image compression components for video compression purpose. The ACC-JPEG coder, as an example of Accordion based video coding model, consists in associated the so called Accordion process to the JPEG standard image coder [6]. We try to prove, via this coder, the efficiency of such model, its low computational requirements and the ease of its implementation which become much easier with adopting the RVC framework. Actually, we propose -via this work- an easy and practical solution for fast designing and implementing an efficient video coder.

In section 2, we present the Accordion based coding principle and we give an overview on the RVC-CAL methodology. Section 3 analyses through the proposed CAL based implementation the ACC-JPEG coder performances (coding and decoding speed, memory consumption). Section 4 concludes.

2. BACKGROUND

In this section we review the Accordion based coding approach. Then we details the data flow implementation and the RVC CAL methodology.

2.1. Accordion

The idea behind the Accordion approach relies on the hypothesis saying that the video stream contains more temporal redundancies than spatial ones [9], [10]. In order to take advantages from this hypothesis, the idea consists in trying to put pixels -which have a very high temporal correlation - in spatial adjacency. Thus, video data will be presented with high correlated form which exploits both temporal and spatial redundancies in video signal with appropriate portion that put in priority the temporal redundancy exploitation.

2.1.1. Accordion Representation

The input of our encoder is the so called video cube (GoF), which is made up of a number of frames. This cube will be decomposed into temporal frames which will be gathered into one 2D representation. Temporal frames are formed by gathering the video cube pixels which have the same column index. These frames will be scanned while reversing the direction of odd frames in order to more exploit the spatial correlation of the video cube frames extremities. This representation transforms temporal correlation of the 3D original video source into a high spatial correlation in the 2D representation ("IACC") [6] [7] [8]. Figure 1 illustrates the principle of this representation.

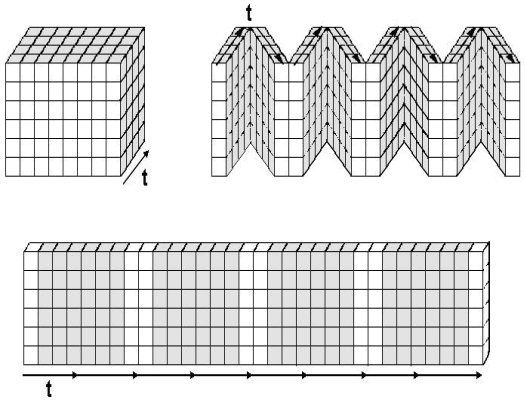

Fig. 1. Accordion principle

2.1.2. Accordion Algorithm

In the following we present the algorithms corresponding to Accordion representation of a sequence of video frames. The input of this algorithm, called ACC, has as input a group of pictures (I_K, for $0 \leq K \leq N-1$) called GOP (Group of Pictures) and as output the resulting IACC image.

The inverse algorithm, denoted ACC-1, has as input the image IACC and as output the set of images I_K, for $0 \leq K \leq N-1$.

Let us note that :

```
Inputs : I0, , IN-1

Outputs : IACC
  For x from 0 to (L x N)-1
    For y from 0 to H-1
      If (x div N mod 2 !=0) Then
        n=(N-1)    (x mod N)
      Else
        n= x mod N
      End
 IACC (x,y)= In (x div N,y)
    End
  End
```

Fig. 2. Accordion algorithm

```
Inputs : IACC
Outputs : I0, , IN-1

 For n from 0 to N -1
    For x from 0 to L -1
       For y from 0 to H -1
         If (x mod 2=0) Then
           XACC= (N -1)    n +(x*N)
         Else
           XACC= n+ (x*N)
         End
       In(x,y)=IACC (XACC,y)
       End
    End
 End
```

Fig. 3. inverse accordion algorithm

- L and H are respectively the length and the height of the video source frames.
- N is the number of frames of a GOF.
- IACC(x, y) is the pixel intensity with the coordinates x, y according to accordion representation repair.
- In(x, y) is the intensity of pixel situated in the Nth frame in the original video source.

2.1.3. Video Coding Model

In this part, we present the coding diagram based on the Accordion representation. First, the video encoder takes a video sequence and passes it to a frame buffer in order to construct volumetric images by combining N frames into a stack. Then, the obtained stack will be transformed to form the accordion representation (IACC). Here N is the constructed stack depth (N is 8 in our experiments). Next, each IACC will be divided into N N blocks to be processed furthermore by the eventual used 2D transform. The encoder block diagram of the Accordion based compression algorithm is in Figure x.

This Accordion based video coding model as it is illustrated in figure x shows a great flexibility with different possibilities of extensions and reutilization of existing image processing tools leading to designing various versions. In this paper, as it was introduced above, we are interested by the JPEG version known as ACC-JPEG.

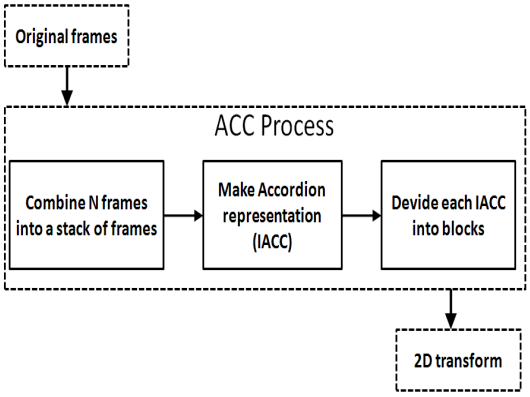

Fig. 4. Accordion based video coding model

2.2. Dataflow implementation

In the following we present the dataflow programming based on the Caltrop Actor Language and the implementation generation using the back-ends of Open RVC CAL compiler [5].

2.2.1. CAL Programming

The execution of an RVC-CAL code is based on the exchange of data tokens between computational entities called *actors*. Each actor is independent from the others since it has its own parameters and finite state machine if needed. Actors are connected to form an application or a design, this connection is insured by FIFO channels. This connection is modeled using XML based dialects as Xml Dataflow Format (XDF). These languages also provide the possibility to include parameters when instantiating an actor. Consequently, the same actor may be instantiated several times with different parameters. We currently use the *Graphiti*[3] tool to manage XDF and NL graphs.

Executing an actor is based on firing elementary functions called *actions*. This action firing may change the state of the actor. An action may be included in a finite state machine or untagged. An untagged action is higher priority than FSM actions. An RVC-CAL dataflow model is shown in the network of figure 5.

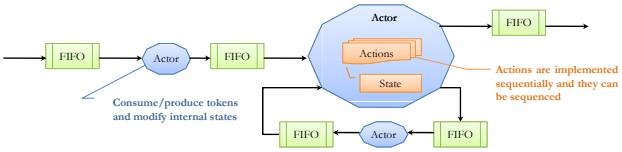

Fig. 5. CAL actor model

[3]http://sourceforge.net/projects/graphiti-editor

When an action is fired, it consumes token streams from input ports and produces token streams to output ports.

We consider a simple *clip* actor that clips the consumed tokens values greater than 255 to 255 and those less than 0 to 0. This algorithm is used in the IDCT2D process of MPEG4 decoders and it is applied on the image 8x8 macro blocks. The associated RVC-CAL code is shown in figure 6 and thus the input streams are:

INPUT1: $INPUT_0, INPUT_1 \ldots INPUT_{63}$

the output stream is:

OUTPUT: $OUTPUT_1, OUTPUT_2 \ldots OUTPUT_{63}$

such as, for an integer k, $OUTPUT_k$ = clip($INPUT_k$)

In this case, the firing rule is the presence of 64 tokens at least in the input FIFO and the availability of 64 memory cells at least in the output FIFO.

```
actor clipActor ()
(int size=8) INPUT1 ==> int(size=8) OUTPUT:

 clip: action INPUT1:[ x ] repeat 64 ==>
 OUTPUT:[[
  if x[i] > 255 then
   255
  else
   if x[i] < 0 then
    0
   else
    x[i]
   end
  end : for int i in 0 .. 63 ] ] repeat 64
 end
end
```

Fig. 6. RVC-CAL example algorithm of clip actor

2.2.2. CAL compiling

Orcc is a CAL compiler that takes in the front-end a set of actors and a graph (see example of graphiti generated graph in Figure 7) that specifies the connexion between theses actors. After some modifications in the middle-end, Orcc applies a string template on the intermediate representation (IR) to generate a chosen back-end this principle is detailed in Figure 8.

Several existing back-ends more or less mature are developped and maintained. We cite: C, C++, Xlim, Verilog, VHDL, Promela, Java, LLVM etc. In this paper we use the C back-end because it is the most efficient and also because a C implementation is the easiest to debug and validate.

3. IMPLEMENTATION AND RESULTS

To implement the ACC-JPEG coder using RVC methodology, we start from an RVC-CAL design of the JPEG codec (see http://orc-apps.sourceforge.net/). The design is shown in Figure 9.

We notice that the design of Figure 9 represents the highest granularity. It means that for instance decoder for example, the content is not a CAL code processing the whole de-

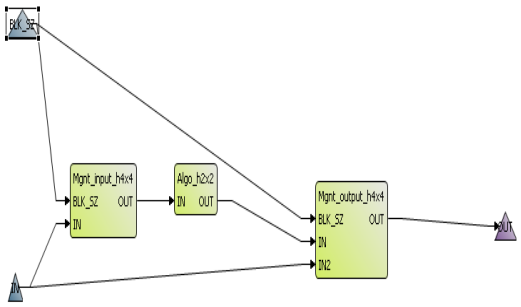

Fig. 7. XDF Graph example

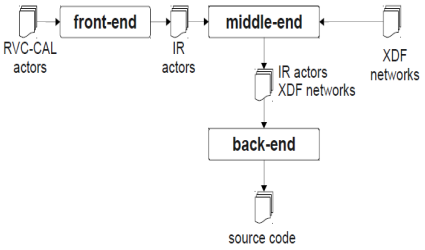

Fig. 8. Orcc compilation principle

coding but a set of actors containing each one a CAL code inside as presented in Figure 10.

The next step consists in integrating the accordion algorithm in the *RawYCbCr* actor and the inverse accordion int the *YCbCrToMB* actor as presented in Figure 11.

We applied the C backend of Orcc on the RVC-CAL JPEG design and we obtained a set of .C files corresponding to every actor of the design and a top file that manages the actors scheduling and the FIFO data exchange. We used the Cmake tool to make a project and generate a solution for C compilers.

Table 1 gives information about the proposed design cost (in terms of implementation simplicity).

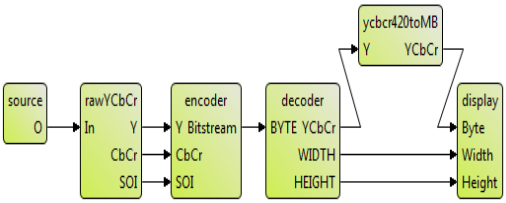

Fig. 9. JPEG reference design

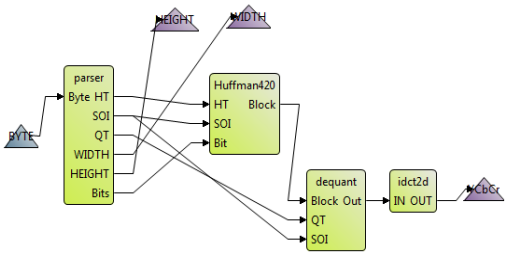

Fig. 10. Finer granularity of JPEG decoder

decoders	level	actors	Parser size LOC CAL	Parser size LOC C
M-JPEG	2	6	184	979
ACC-JPEG	2	7	184	979
MPEG4	3	27	1285	4720

Table 1. Complexity comparison between CAL and C descriptions

The design is evaluated with CIF and Q-CIF video sequences. Originally, the video frames look like Figure 12. By adding the accordion algorithm, we obtained the frames of Figures 13, 14, 15, 16. Figure 17 shows a comparison in terms of rate-distortion between the ACC-JPEG, MJPEG and MPEG-4. The results shown are obtained with the sequence 'Hall Monitor' (CIF, 25Hz). We considered a general processor as an implementation target. The main configuration features of this target are presnted in table 2

processor	Intel dual core CPU
frequency	2,5 GHz
RAM	2 Go
operating system	Microsoft windows XP Professionnel

Table 2. Implementation target features

Some results about the architecture performances (processing speed) and requirements in term of (memory and resources) are given in tables 3, 4, 5. All results are recorded in the same experimental conditions given in table 2.

Tables 3, 4 shows the coding frames frequencies recorded for CIF and QCIF videos. We can notice that the ACC-JPEG has almost the same processing (encoding and decoding) speed than M-JPEG coder.

Table y illustrates the ACC-JPEG memory requirements in comparison with other standards coders.

The memory requirements can be divided into two by eliminating intermediate tables reserved for the construction of the Accordion Presentation and thus it will match the theoretical memory requirements as it was estimated in [8]. Even processing speed can also be improved and tends perfectly to the M-JPEG processing speed by adjusting the state ma-

	encoding FPS	
resolution	ACC-JPEG	M-JPEG
CIF (288x352)	178	182
QCIF (144x176)	748	760

Table 3. Throughput frequency ACC-JPEG VS M-JPEG encoders

	decoding FPS	
resolution	ACC-JPEG84	M-JPEG
CIF (288x352)	152	156
QCIF (144x176)	690	702

Table 4. Throughput frequency ACC-JPEG VS M-JPEG decoders

chine which was initially designed for M-JPEG and then not perfectly adapted to the own ACC-JPEG constrains. Such state machine involves large timeouts resulting to the need to buffering of frames for the set up of the Accordion representation.

4. CONCLUSION

In this paper a CAL description of a new video coder is evaluated. The presented work constitutes a demonstration of an efficient video coder fast implementation. Regarding the proposed video coder tradeoff between performances and complexity of the proposed video coder and the tradeoff between efficiency and cost of the implementation methodology, the presented work could be considered as an economic and efficient solution for many video applications including embedded systems, digital multimedia devices, networks sensors which requires low computational processing and low time to market. Further orientations will include the implementation of the second Accordion-based-coder called ACC-JPEG 2000- proposed in [7] as a more efficient version than one presented in this paper.

component	memory consumption in Ko	
Image size	CIF	QCIF
ACC RawYCbCr	304,128xGOF	76,032xGOF
Encoder	3,614	3,614
Decoder	66,080	33,240
YCbCr420ToMB	304,128xGOF	76,032xGOF
FIFO	65,536	32,768
Total	591,788	184,054

Table 5. JPEG memory consumption

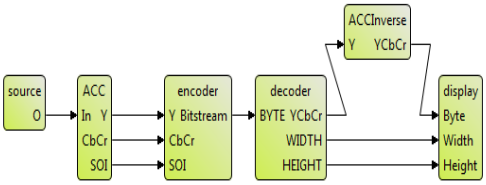

Fig. 11. ACC-JPEG design

Fig. 12. Normal frame

Fig. 13. Frame 1

Fig. 14. Frame 2

5. REFERENCES

[1] J. Eker and J. Janneck, "CAL Language Report," Tech. Rep. ERL Technical Memo UCB/ERL M03/48, Univer-

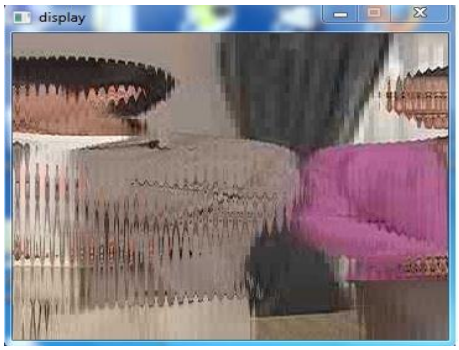

Fig. 15. Frame 3

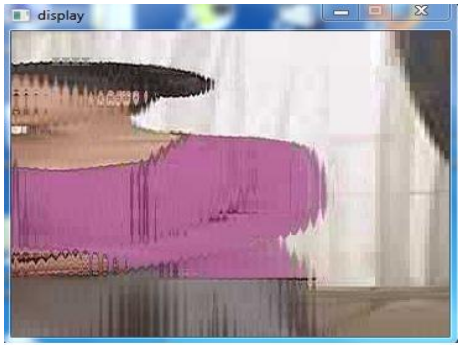

Fig. 16. Frame 4

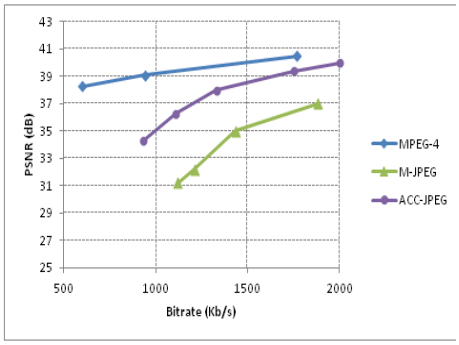

Fig. 17. Performance comparison

sity of California at Berkeley, Dec. 2003.

[2] ISO/IEC FDIS 23001-4: 2009, "Information Technology - MPEG systems technologies - Part 4: Codec Configuration Representation," 2009.

[3] C. Brooks, E.A. Lee, X. Liu, S. Neuendorffer, Y. Zhao, and H. Zheng (eds.), "PtolemyII - heterogeneous concurrent modeling and design in java (volume 1: Introduction to ptolemyII)," Technical Memorandum UCB/ERL M04/27, University of California, Berkeley, CA USA 94720, July 2004.

[4] S Bhattacharyya, G Brebner, J Eker, J Janneck, M Mattavelli, C von Platen, and M Raulet, "OpenDF - A Dataflow Toolset for Reconfigurable Hardware and Multicore Systems," First Swedish Workshop on Multi-Core Computing, MCC , Ronneby, Sweden, November 27-28, 2008, 2008.

[5] Jörn W. Janneck, Marco Mattavelli, Mickael Raulet, and Matthieu Wipliez, "Reconfigurable video coding a stream programming approach to the specification of new video coding standards," in *MMSys '10: Proceedings of the first annual ACM SIGMM conference on Multimedia systems*, New York, NY, USA, 2010, pp. 223–234, ACM.

[6] Tarek Ouni, Walid Ayedi, and Mohamed Abid, "New low complexity dct based video compression method," in *Proceedings of the 16th international conference on Telecommunications*, Piscataway, NJ, USA, 2009, ICT'09, pp. 202–207, IEEE Press.

[7] Tarek Ouni, Walid Ayedi, and Mohamed Abid, "New non predictive wavelet based video coder: Performances analysis," in *Image Analysis and Recognition*, Aurlio Campilho and Mohamed Kamel, Eds., vol. 6111 of *Lecture Notes in Computer Science*, pp. 344–353. Springer Berlin / Heidelberg.

[8] Tarek Ouni, Walid Ayedi, and Mohamed Abid, "A complete non-predictive video compression scheme based on a 3d to 2d geometric transform," *International Journal of Signal and Imaging Systems Engineering*, vol. 4, no. 3, pp. 164–180, 2011.

[9] Huifang Sun Yun Q . Shi, "Image and video compression for multimedia engineering fundamentals, algorithms, and standards," in *CRC Press*, 1999.

[10] Aaron A.M. Gokturk, S.B., "Applying 3d methods to video for compression," in *Digital Video Processing (EE392J) Projects Winter Quarter*, 2002.