

RECONFIGURABLE VIDEO CODING: OBJECTIVES AND TECHNOLOGIES

Christophe Lucarz, Ihab Amer, and Marco Mattavelli

Microelectronic Systems Laboratory, Ecole Polytechnique Fédérale de Lausanne (EPFL)
CH-1015 Lausanne, Switzerland
{christophe.lucarz, ihab.amer, marco.mattavelli}@epfl.ch

ABSTRACT

The main objective of the MPEG Reconfigurable Video Coding (RVC) standard is to establish a framework for a more flexible usage of standard video coding technology. The framework not only supports multiple standards and new coding configurations, but also provides an incremental and modular approach to innovation in video compression development and design. This paper provides an overview of the main objectives of RVC, standard accompanied with a presentation of the components of the framework for both normative parts and supporting tools useful for the final implementation of RVC codecs. These elements include: the Video Tool Library (VTL), the new standard RVC-CAL language used for the specification of the library, the Bitstream Syntax Description (BSD) used for the specification of the compressed bitstreams, as well as the Functional unit Network Description (FND) that constitutes the specification of a modular library. Technologies and tools that support the RVC standard are also briefly introduced.

Index Terms— MPEG, Reconfigurable Video Coding, RVC, CAL, BSD, FND, VTL

1. INTRODUCTION

Recent advances in digital video hardware/software, together with the emergence of international standards for digital video compression, have led to a wide variety of digital video products [1]. Digital video compression has become an essential component of broadcast and entertainment media [2]. However, digital video applications still need to satisfy a multitude of growing and stringent requirements in order to achieve the desired quality for real-time applications. Although the evolution of video coding standards in the last two decades has produced outstanding results, it has not been able to address all of these application requirements [3]. Hence, continuous improvements in digital video coding are required to help narrowing the gap between the users' demands and the capabilities and performances of transmission networks, storage devices, and terminals. This fact increases the necessity of continuous releases of standards for

compressed video representation with aggressively increased coding efficiency and enhanced robustness to network environments [4].

The main aim of a video coding standard is to define the syntax as explicitly and unambiguously as possible without necessarily indicating practical constraints that a designer should take into account [5]. Hence, typical systems' specifications get composed of multi-format algorithms and protocols with unbounded complexities. This would enable a designer to produce the codec supporting any desired functionality as well as any desired trade-off between compression performance and implementation complexity. Unfortunately, the conventional process of selecting such algorithms and protocols is hardwired into the normative descriptions of the codec, or at lower level, into a predefined number of choices, known as "profiles", codified within each standard specification. Currently, the large number and variety of available coding tools makes it inadequate to follow on with such paradigm. On one hand, it is difficult to identify the optimal combinations of tools prior to, or soon after, the release of the standard. On the other hand, it is often not possible to identify all of the application scenarios in which a codec will be used, at the time of its release. Nor it is feasible to provide a normative profile for every scenario [6].

Nowadays, there is an increasing requirement for decoding platforms capable of supporting multiple codecs with multiple profiles. Although many of these codecs share common and/or similar coding tools, there is currently no standard way to exploit such commonalities at the level of the specification or the implementation. This problem grows as new standards are released and legacy formats continues to be supported [7]. Ideally, implementers of a standard should be able to select arbitrary combinations of the available tools, in the way that best matches the requirements of each application. The challenge with this approach is ensuring interoperability [6]. These considerations led to the birth of the MPEG RVC, a new standard currently under development by ISO/IEC that aims at providing a framework that allows for dynamic development, implementation, and adoption of standardized video coding solutions with features of higher flexibility and reusability. In this paper, an overview of the RVC

standard is provided, surveying the main related tools and supporting technologies.

The remainder of the paper is organized as follows: Section 2 describes the concept of the MPEG RVC standard, followed by a description of its components and basic terminologies in Section 3. Section 4 briefly overviews the supporting tools and technologies, and finally section 5 concludes the paper.

2. CONCEPTS OF MPEG RVC

The MPEG RVC framework aims at offering a more flexible use and faster path to innovation of video coding standards. An additional challenge taken by the RVC framework is to provide a high level specification formalism that constitutes a starting point model for the direct software and hardware synthesis. Moreover, the RVC framework intends to overcome the lack of interoperability between various video codecs that are deployed into the market. Unlike previous standards, RVC does not itself define a new codec. Instead, it provides a framework to allow content providers to define a multitude of different codecs, by combining together blocks, or Functional Units (FUs), from a standard modular library (VTL). Such possibility clearly simplifies the task of designing future multi-standard video decoding applications and devices by allowing software and hardware reuse across video standards. The MPEG RVC framework defines two standards: ISO/IEC23001-4 (or MPEG-B part 4), which defines the overall framework as well as the standard languages that are used to describe different components of the framework, and ISO/IEC23002-4 (or MPEG-C part 4), which defines the library of video coding tools employed in existing MPEG standards.

The main strength of RVC is that, unlike the traditional video coding standards, where decoders used to be rigidly specified, a description of the decoder is associated to the encoded data, enabling a reconfiguration and instantiation of the appropriate decoder. Figure 1 shows three different decoder types that can be constructed within the RVC framework.

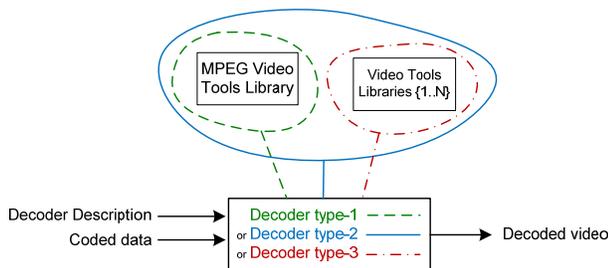


Figure 1: Different decoder types within RVC

All the three types of decoders are compliant with the RVC framework and constructed using the specification formalisms (i.e. languages) standardized by MPEG-B.. A

Type-1 decoder is constructed using the FUs within the MPEG VTL only. Hence, this type of decoder conforms to both the MPEG-B and MPEG-C standards. A *Type-2 decoder* is constructed using FUs from the MPEG VTL as well as one or more proprietary libraries (VTL 1-n). This type of decoder conforms to the MPEG-B standard only. Finally, A *Type-3 decoder* is constructed using one or more proprietary VTL (VTL 1-n), without using the MPEG VTL. This type of decoder also conforms to the MPEG-B standard only.

MPEG VTL is normatively specified using RVC-CAL (see section 3.1). An appropriate level of granularity for blocks within the standard library is important, to enable efficient reuse within the RVC framework. If the library is too coarse, modules will be too large to allow reuse between different codecs. On the other hand, if the granularity is too fine, the number of modules in the library will be too large for an efficient and practical reconfiguration process, and may obscure the desired high-level description and modeling of the RVC decoder.

In the RVC framework, the receiver gets the Decoder Description that fully specifies the architecture of the decoder. In order to instantiate the decoder, the receiver needs libraries of building blocks specified by MPEG-C. Figure 2 illustrates how decoders are built from the Decoder Description, which includes two main types of data:

- The **Bitstream Syntax Description (BSD)**, which describes the structure of the bitstream. The BSD is written in RVC-BSDL (see section 3.2).
- The **FU Network Description (FND)** describes the connections between the coding tools (i.e. FUs). It contains also the values of the parameters used for the instantiation of the different FUs composing the decoder. The FND is written in FU Network Language (FNL) (see section 3.3)

The model instantiation consists of assembling the video decoder by picking up FUs from the MPEG VTL according to the FND provided in the RVC bitstream. The assembly results in the so called Abstract Decoder Model (ADM), which is composed of the syntax parser (built from the BSD) and the network of FUs (built from the FND).

The decoder implementation process consists of implementing a true decoding solution from the ADM. Device manufacturers are thus capable of providing any alternative proprietary implementations of the standard library that are optimized for their particular platform (i.e. proprietary video tool box). Several tools are already available, and others are in development to directly synthesize the ADM into both hardware (HDL) and/or software (C, C++...) implementations (see section 4).

3. COMPONENTS OF THE RVC FRAMEWORK

This section provides an overview of the constitutive components of the RVC framework, briefly describing the languages used for each component.

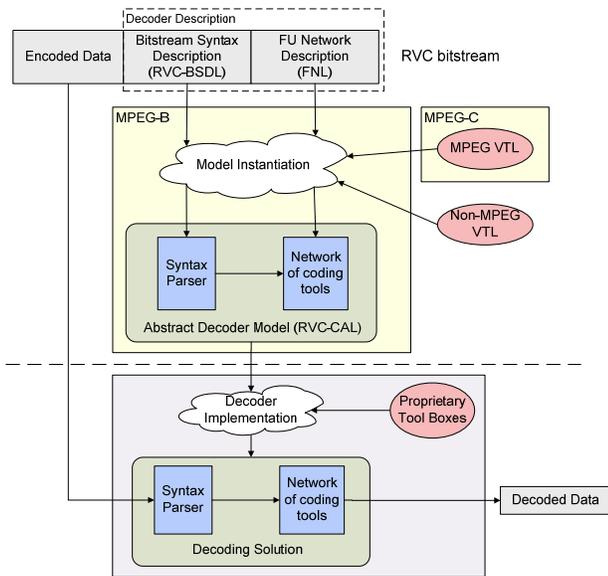


Figure 2: Overview of the RVC framework

3.1. Dataflow Actor-Oriented Language

CAL is a dataflow and actor oriented language that has been recently specified as a subproject of the Ptolemy project at the University of California at Berkeley. The final CAL language specification has been released in December 2003 [8]. CAL models different algorithms by using a set of interconnected dataflow components called *actors* (see Figure 3).

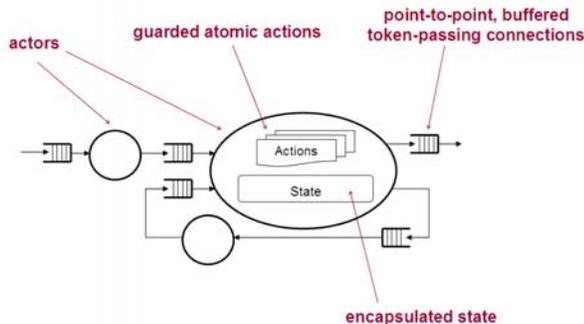


Figure 3: The CAL Model of Computation

An actor is a modular component that encapsulates its own state. The state of any actor is not shareable with other actors. Thus, an actor cannot modify the state of another actor. Interactions between actors are only allowed through input and output ports. The behavior of an actor is defined in terms of a set of *actions*. The operations an action can perform are to *consume* input tokens, to *modify* internal state and to *produce* output tokens. The topology of a set of interconnected actors constitutes what is called a *network* of actors. The transitions of an actor are purely sequential: actions are fired one after another. At the network level, the actors can work concurrently, each one executing their own

sequential operations. CAL allows also hierarchical system design. Each actor can be specified as a network of actors.

The following points summarize some of the features of the CAL language that makes it highly suitable to model complex signal processing systems:

Parallelism Scalability: Writing programs such that their parts execute concurrently without much interference is one of the key problems in scaling traditional imperative programs. Encapsulated actors allow exposing more parallelism as applications grow in size.

Modularity: The strong encapsulation of actors along with their hierarchical structure offers high potential of parallelism. Thus, the internal specification of any actor can be modified without impacting other actors.

Scheduling: Unlike procedural programming languages, where control flow (sequence of execution of instructions) is tightly coupled with the algorithm, the actor model allows more flexibility in the scheduling process of the model (i.e. determining the order of execution of the actions) by allowing for various scheduling schemes depending on optimization criteria.

Portability: For many highly concurrent programs, portability has remained an elusive goal, often due to their sensitivity to timing. The “untimedness” and asynchrony of dataflow programming offers a solution to this problem.

Adaptivity: Each atomic firing of an action triggers the actor in a well-defined state. Thus, every actor of the model lies in a known state at any point of time. This is an important feature knowing that the success of the dataflow programming model depends on its ability to be reconfigured dynamically.

3.2. Bitstream Syntax Description Language

The description of the bitstream syntax used for the encoded data is written in RVC-BSDDL which is a sub-set of the BSDL language [9]. The reason why BSDL has been restricted is to be sure that there is only one way to describe bitstreams without losing any power of the language. This description is passed to the RVC decoder in order to generate the appropriate parser which in turn decodes the corresponding input encoded data. BSDL was found to be the most suitable, because:

- it is stable and defined by an international standard [9]
- its XML-based syntax integrates well with FNL
- parsers may be derived by transforming the BSD using standard tools such as XSLT [10]

For further details about BSD and RVC-BSDDL, the reader can refer to [10].

3.3. FU Network Language

The FU Network Language is an XML dialect that describes an interconnected network of standard library components (FUs), which together represent a complete decoder. FNL

provides a facility for declaring parameters, and passing parameters to the FUs. This is useful for declaring values that are constant for a particular instantiation of an FU, but may vary between different instantiations. For further details about FND and FNL, the reader can refer to [11].

4. RVC SUPPORTING TECHNOLOGIES

Many technologies are introduced to support the RVC standard. This includes (but is not limited to) the tools for validating the BSDL schema versus a bitstream instantiation [10], the CAL parser from a validated BSDL schema description [10], the simulator that both verifies and validates the behaviour of the ADM [12], tools that automatically translate the ADM to C [13] and HDL [14] code, targeting both software and hardware platforms respectively. The scheduling of the ADM (i.e. the CAL model) is also an important issue, and interesting works are in progress [15]–[16].

Furthermore, the existence of RVC encoding tools supports the evolution of the RVC standard by providing a complete framework for testing the developed VTL modules. Figure 4 provides an MPEG RVC typical Encoding/Decoding Scenario [17].

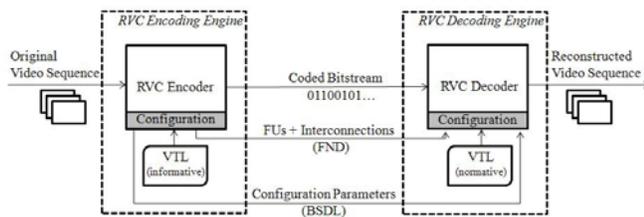


Figure 4: MPEG RVC Encoding/Decoding Scenario

5. CONCLUSION

This paper provides an overview on the RVC video coding standard, describing its objective, and surveying its basic components. When finalized, the RVC is expected to severely reduce the timescale between proposing a new idea or concept in video coding and implementing it in practical devices and systems. Moreover, other proposals to further reduce the time lag between proposing a new coding tool, and practically making use of it are evolving. Dynamically Configurable Video Coding [7] is an example of such initiatives, and more are expected to appear alongside with the development of the RVC standard.

6. REFERENCES

[1] A. M. Tekalp, Digital Video Processing, *Prentice-Hall, Inc.*, 1995.
 [2] I. E. G. Richardson, H.264 and MPEG-4 Video Compression: Video Coding for Next-generation Multimedia, *John Wiley & Sons Ltd.*, December 2003.
 [3] A. Tamhankar, and K. R. Rao, “An Overview of H.264/MPEG-4 Part 10,” *Proceedings of EURASIP*

Conference focused on Video/Image Processing and Multimedia Communications, Vol. 1, pp. 1-51, July 2003.
 [4] “ITU-T Rec. H.264 | ISO/IEC 14496-10 AVC,” *Draft Text of Final Draft International Standard for Advanced Video Coding*, [Online]. Available at: http://www.chiariglione.org/mpeg/working_documents.htm, March 2003.
 [5] I. E. G. Richardson, Video CODEC Design: Developing Image and Video Compression Systems, *John Wiley & Sons Ltd.*, April 2002.
 [6] C. Lucarz, M. Mattavelli, J. Thomas-Kerr, and J. Janneck, “Reconfigurable Media Coding: A New Specification Model for Multimedia Coders,” *Proceedings of IEEE Workshop on Signal Processing Systems*, pp. 481-486, October 2007.
 [7] I. Richardson, M. Bystrom, S. Kannangara, and M. L pez, “Dynamic Configuration: Beyond Video Coding Standards,” *Plenary Presentation, IEEE International System on Chip Conference (SOCC08)*, Newport Beach, CA, September 2008.
 [8] J. Eker and J. W. Janneck, “Cal language report,” *University of California at Berkeley*, Tech. Rep. UCB/ERL M03/48, December 2003.
 [9] ISO/IEC 23001-5, “Bitstream Syntax Description Language.”
 [10] M. Raulet, J. Piat, C. Lucarz, and M. Mattavelli, “Validation of Bitstream Syntax and Synthesis of Parsers in the MPEG Reconfigurable Video Coding Framework,” *Proceedings of IEEE Workshop on Signal Processing Systems*, October 2008.
 [11] Dandan Ding, L. Yu, C. Lucarz, and M. Mattavelli, “Video decoder reconfigurations and AVS extensions in the new MPEG reconfigurable video coding framework,” *IEEE Workshop on Signal Processing Systems*, Washington DC, US : 2008, pp. 164-169.
 [12] Open DataFlow Sourceforge Project: <http://opendf.sourceforge.net/>
 [13] G. Roquier, M. Wipliez, M. Raulet, J. Janneck, I. Miller, and D. Parlour, “Automatic Software Synthesis of Dataflow Program: An MPEG-4 Simple Profile Decoder Case Study,” *Proceedings of IEEE Workshop on Signal Processing Systems*, October 2008.
 [14] J. Janneck, I. Miller, D. Parlour, G. Roquier, M. Wipliez, and M. Raulet, “Synthesizing Hardware from Dataflow Programs: An MPEG-4 Simple Profile Decoder Case Study,” *Proceedings of IEEE*
 [15] J. Boutellier, V. Sadhanala, C. Lucarz, P. Brisk, and M. Mattavelli, “Scheduling of dataflow models within the Reconfigurable Video Coding framework,” *IEEE Workshop on Signal Processing Systems*, Washington DC, US : 2008, pp. 182-187.
 [16] Ruirui Gu, Jorn W. Janneck, Mickael Raulet, and Shuvra Bhattacharyya, “Exploiting Statically Schedulable Regions In Dataflow Programs,” *IEEE International Conference on Acoustics, Speech, and Signal Processing*, 2009.
 [17] I. Amer, H. Aman-Allah, E. Hanna, and K. Maarouf, “Towards a Comprehensive RVC VTL: A CAL Description of an Efficient AVC Baseline Encoder”, initially *Proceedings of IEEE International Conference on Image Processing*, Special Session on Reconfigurable Video Coding, Cairo, Egypt, November 2009.