Towards Multi-Granular RVC Tool Libraries: A Case Study of CAL Transformations on the ISO/IEC MPEG Fixed Point IDCT

Ihab Amer[1, 2]

[1]*Laboratory of Microelectronic Systems (GR-LSM), EPFL*
CH-1015 Lausanne, Switzerland
ihab.amer@epfl.ch

[2]*Advanced Technology Information Processing Systems (ATIPS) Labs*
University of Calgary
2500 University Dr., NW, Calgary, AB, Canada, T2N 1N4
amer@atips.ca

Abstract

Recent advances in digital video hardware, software and coding standards have led to a wide variety of digital video products. However, such continuous evolution leads to the necessity of regular replacements of the available multimedia devices. To avoid this, a new initiative within the MPEG community, namely Reconfigurable Video Coding, has risen to provide the flexible framework that allows for the "simple" realization of highly-reconfigurable video coding solutions, without the need to wait for decades to replace the existing infrastructure. This paper illustrates the different design schemes to build various platform-specific proprietary libraries, at various granularity levels, together with a case study to prove the concept.

1. Introduction

The emergence of international standards for digital video compression (e.g., MPEG-2, MPEG-4 Visual, and Advanced Video Coding (AVC) of the MPEG community) has led to the existence of a wide variety of digital video products. In particular, homes are now equipped with various digital video products such as High-definition Television (HDTV) sets, set top boxes, Personal Video Recorders (PVRs), Digital Versatile Disc (DVD) and Video Compact Disc (VCD) players, laptops, mobile phones, etc. In this context, digital video compression has become an essential component of broadcast and entertainment media. However, digital video (and more generally multimedia) 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. 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. In the meantime, such continuous evolution in the digital video coding field leads to the necessity of regular replacement of the multimedia devices. A fact that is no more acceptable by a typical citizen, especially when considering the unstable economic situation world wide.

Reconfigurable Video Coding (RVC) is a new standard that is under development aiming at providing the suitable framework that allows for the realization of reconfigurable video coding solutions. RVC is supported with many technologies. This paper contributes to the development of such technologies by enhancing the tools to synthesize efficient software or hardware implementations out of the high-level specifications.

The remainder of this paper is organized as follows: Section 2 overviews the RVC standard, followed by Section 3 that introduces the dataflow programming model by CAL. Section 4 introduces the concept of toll libraries' granularity, while Section 5 provides a case study to prove the concept. Finally section 6 concludes the paper.

2. The RVC Standard

The standardization efforts in the video coding field have to guarantee the interoperability. In fact, there exist many technologies that would enable reconfigurable solutions. This includes CPUs, Digital Signal Processors (DSPs), FPGAs, etc. Nevertheless, and in spite of the market demands for reconfigurability, there are still no clear success stories of fully reconfigurable solutions for multimedia applications. This can be explained by the fact that up till recently, there was no framework to enable the existence of such reconfigurable solutions. In fact, the current process of releasing a video coding standard, starting from the rise of the standard's idea, passing through the specification and formalism of the standard, and ending with the physical implementations, is too rigid to allow for the evolution of such reconfigurable solutions. Change of the platform was the solution for any evolution towards new formats or standards. The same concept applied for the video coding standards that time to time were replaced by a new monolithic version. This was the case for MPEG-2, MPEG-4, AVC, and the recent SVC standard. However, such monolithic approach to both content processing and the platform support does not respond anymore to the current demand of adaptation, dynamicity of services and applications, introduction of new multimedia technologies, and to the demands of users.

The observation of such drawbacks opened the door to extend the state-of-the art by raising a new initiative within the MPEG community, namely Reconfigurable Video Coding**,** calling for an "everlasting" video coding standard, that when adopted, will overcome many of the shortcomings of the traditional video coding standards and of the way they are specified and deployed. The MPEG RVC standard provides the flexible framework that allows for the realization of highly-reconfigurable solutions. It provides an incremental and modular approach to innovation in video compression development and design, opening the way for interoperability between various video codecs that are deployed into the market. 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. This offers a more flexible use and faster path to innovation of video coding standards opening the door to the evolutionary introduction of new emerging technologies such as multiview, 3–D TV services that can dynamically develop in terms of P2P clusters without the need to wait for decades before all system providers and users will be forced to replace the existing distribution, broadcast and terminals infrastructure.

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 standardized modular Video Tool Library (VTL).

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. Thus, when finalized, the RVC is expected to **s**everely reduce the time gap between proposing a new idea or concept in video coding and implementing it in practical devices and systems [1].

The main strength of RVC is that, unlike current video coding standards, where decoders used to be rigidly specified, a description of the decoder is associated to the encoded data, enabling reconfiguration and instantiation of the appropriate decoder. Figure 1 provides a conceptual diagram of the abstract idea behind the RVC initiative [REF MPEG].

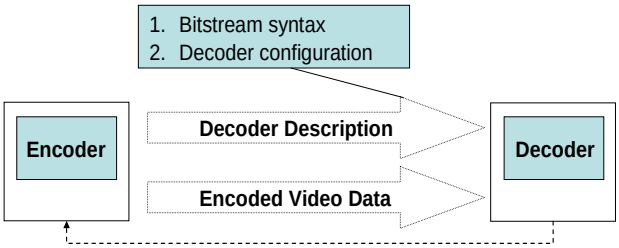

Figure 1: Conceptual diagram of RVC

An RVC decoder (MPEG–B) is composed of coding tools described in VTLs according to the decoder description. The MPEG VTL is described by MPEG–C (specified using RVC–CAL), while other proprietary VTLs can be also used. 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.

The abstract decoder description includes two main types of data:

1. *Bitstream Syntax Description (BSD)*, which describes the structure of the bitstream. The BSD is written in RVC–BSDL. It is used to generate the appropriate parser to decode the corresponding input encoded data [3]-[4].
2. *FU Network Description (FND)*, which describes the connections between the coding tools (i.e. FUs). It also contains 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). The syntax parser (built from the BSD), together with the network of FUs (built from the FND), form a CAL model called the Abstract Decoder Model (ADM) [5]-[6].

Device manufacturers are capable of providing any alternative proprietary implementations of the standard library that are optimized for their particular platform. A few tools are already available, and others are in development to directly synthesize the ADM into both hardware (HDL) [7] and/or software (C, C++…) [6].

3. CAL Dataflow Programming Model

A dataflow program is described by a directed graph, where the nodes represent computational units and the arcs represent the flow of data [8]. This makes it perfectly suitable to model video coding systems, which are typically described by diagrams of blocks connected by arcs that denote the flow of data. In addition, unlike sequential programming, dataflow programming provides simple, understandable, and powerful abstractions that allow the specification of as much or as little parallelism as is required. One of the major challenges taken by the RVC framework is to provide a new high-level specification model enabling direct and efficient software and hardware synthesis. This objective has been pursued by using the CAL actor dataflow model as the core computational model of the RVC standard specification. It is chosen to be the language of the reference model that specifies the normative I/O behavior of the modules of the RVC library of FUs. A CAL actor is a strongly encapsulated computational entity with input and output ports, internal state and parameters [9].

RVC-CAL is a subset of the original CAL language and is normalized by ISO/IEC as a part of the RVC standard. It slightly restricts the data types, operators, and features that could be used in the original CAL language. The main reason of such restrictions is to simplify the development of synthesis tools supporting both HW and SW synthesis.

Figure 2 illustrates the principles of the CAL dataflow programming model. 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 channels. 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/or 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, where 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, in which each actor can be specified as a network of actors [10].

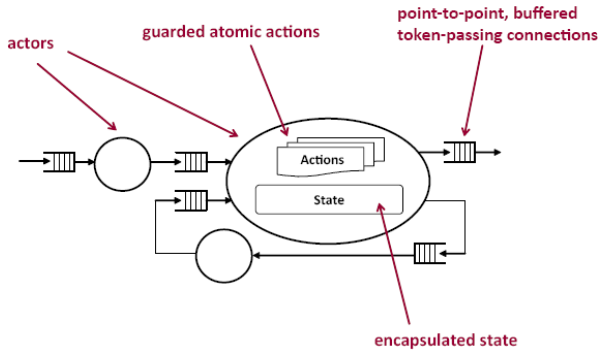

Figure 2: The CAL dataflow programming model

4. Granularity of the Tool Libraries

As the name implies, the ADM is intended to be as high-level as possible, mainly concerned with the *behavioral* description of the different modules of the decoder. The proprietary libraries play a vital role in mapping the ADM to a decoder implementation that meets the constraints of the target platform and the requirements defined by the designer for a specific application. Such libraries contain the proprietary implementation of the FUs composing the ADM. By definition, the usage of proprietary FUs should not alter the behavior of the overall decoder model. Instead, it allows it to execute efficiently on specific target platform(s).

Hence, there is a clear need for the existence of advanced methodologies and tools targeting the

implementation of ADMs on various types of platforms. Such methodologies should be able to impose a set of operations on the FUs that composes the ADM, in a development process that transforms it to another set of functionally-equivalent FUs, which are more appropriate for the target platform. These various operations on the FUs are performed according to the characteristics of the target platform as well as the required degree of parallelism.

The development process consists of modifying the internal structure of the FU that may be composed of one or more actors. Such transformation operation results in a functionally-equivalent (in terms of input/output behavior) FU(s), which is more adapted to the target platform architecture. Splitting or merging actors within the set of FUs are examples of transformation operations that allow exposing more or less parallelism respectively. One distinguishing feature of the dataflow model is that it is able to expose different levels of parallelism by representing algorithms in the form of graphs. For example, as an extreme, parallelism may almost diminish from the system if the decoder model is composed of a single actor that fires its atomic actions sequentially. On the other hand, parallelism can be more exposed if the decoder model is composed of reasonably sized networks that are composed of several simple actors.

The definition of an "appropriate model" depends on the target application. For instance, if the target application scenario is a programmable hardware processing element (i.e. an FPGA), it is advisable to expose the maximum (and at the same time reasonable) level of parallelism/concurrency in the decoder model, and vice versa if the target application scenario is a single sequential processing unit.

Discrete Cosine Transform (DCT) is the primary transform coding tool for many image and video coding standards such as JPEG and MPEG–2. The most commonly used block size in most of the video and image coding standards is 8×8, due to the reasonable compromise it presents between compressibility and computational requirements. The Inverse DCT (IDCT) reconstructs a block of image samples (pixels) from an array of DCT coefficients (usually quantized and scaled) [11]-[13].

Due to its existence in many video decoders, the MPEG-standardized 2D fixed-point 8×8 IDCT module has been chosen as the case study to observe the effects of the granularity of the tool library on the implementation performance targeting various platforms. Figure 3 summarizes the performed experiment. CAL2C and CAL2HDL are available tools that synthesize the RVC specifications to software and hardware respectively [6]-[7].

5. Case Study – MPEG Fixed Point IDCT

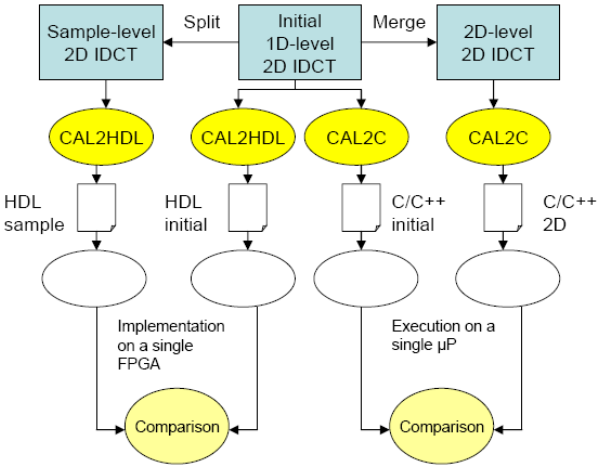

Figure 3: Testing scheme to examine the effect of granularity levels of the video tool libraries

In the initial (or reference) CAL model, the granularity is said to be one dimensional "1D" as the 2D IDCT is performed separably, by applying the MPEG standardized fixed-point 1D-IDCT to the coefficients, first in the horizontal direction, followed by a transpose operation, and then finally another MPEG standardized fixed-point 1D-IDCT in the vertical direction is applied. Figure 4 shows a block diagram of this "1D" CAL model. Each of the 1D-IDCT modules is described as a single actor, composed of three actions. The first action is to read 8 coefficients; the second action performs all the required operations, while the third action writes back the 8 coefficients.

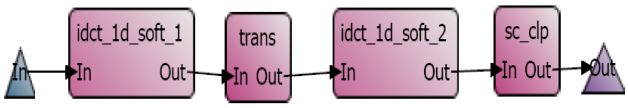

Figure 4: Block diagram of 2D IDCT with 1D granularity level

CAL transformations are applied to this model to change the granularity of actor(s), to either the "2D" granularity or to the "sample" granularity.

From "1D" to "2D" Granularity

This transformation is represented by *merging* actors into a single "2D-IDCT actor". Due to its static behaviour, it is possible to automatically merge actors into a single actor without changing the structure of actions and tokens.

The Cal2C tool is applied to both models (the initial and the transformed one) to generate C-code. The performance is then compared by executing the code on a single-processor platform. The results are normalized according to the initial CAL model ("1D"). The achieved speedup is around 2.5. This is mainly due to the savings in the overhead of static scheduling of actors, which is more efficient than the dynamic scheduling (actors are translated by threads).

From "1D" to "sample" Granularity

CAL transformations are applied to the reference model to transform it to another model with "sample"-level granularity of actor(s). This transformation is done mainly by *splitting* each "1D-IDCT" actor into a network of 12 actors. This transformation intends to exploit the parallelism intrinsic in the algorithm. Figure 5 shows the top level of the resulting CAL model.

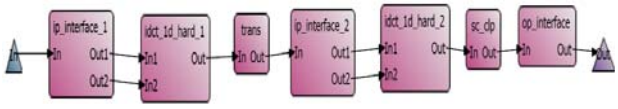

Figure 5: Top-level view of 2D IDCT with sample granularity level

While the second level of the model that shows the composition of each 1-D IDCT from smaller actors is shown in Figure 6.

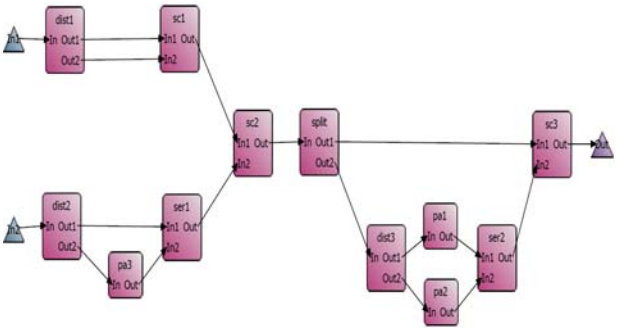

Figure 6: Second-level view of 2D IDCT with sample granularity level

The effect of the CAL transformations is further emphasised by the results obtained on a single FPGA target platform. The Cal2HDL tool is used to translate the CAL model into the HDL model. Then the generated HDL model is synthesised and mapped onto a Xilinx Virtex 5 XC5VLX50T FPGA. The results are normalized according to the initial CAL model ("1D"). In case of "1D to sample" transformation, the throughput is increased by 176.6% compared to the synthesised 1D CAL model. This is mainly because the target FPGA platform exploits the parallelism in the sample model more efficiently. On the other hand, the synthesised sample model consumes 252.7% more Slice Registers, and 19.58% more Slice LUTs than the initial model. The increase in the required hardware resources is due to the overhead obtained by the finite state machines and the interfaces between the multiple actors.

6. Conclusion

In this paper, MPEG-RVC is introduced with an emphasis on the various possible granularity levels of the video tools libraries. A case study of the ISO/IEC MPEG fixed point 2D IDCT is introduced. The results helped in proving some concepts such as the preference to represent the decoder model as a multitude of networks, each composed of simple actors, when targeting HW platforms where exposing high level of parallelism is desirable. While on the other hand, when targeting single-processor SW platforms, it is preferable to represent the decoder model as complex actors with complex state machines

7. Acknowledgements

The author would like to thank Ghislain Roquier, Christophe Lucarz, Marco Mattavelli, and Mickael Raulet for all their valuable contributions to obtain and report this work. The author would also like to thank the GR-LSM lab at EPFL as well as ATIPS Labs at the University of Calgary for funding this research.

8. References

[1] C. Lucarz, I. Amer, and M. Mattavelli, "Reconfigurable Video Coding: Concepts and Technologies", initially accepted in IEEE International Conference on Image Processing, Special Session on Reconfigurable Video Coding, Cairo, Egypt, November 2009.
[2] ISO/IEC N10165, "Text of ISO/IEC FDIS 23001-4: Codec Configuration Representation".
[3] ISO/IEC 23001-5, "Bitstream Syntax Description Language".
[4] 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.
[5] 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.
[6] 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.
[7] 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 Workshop on Signal Processing Systems, October 2008.
[8] S. Bhattacharayya, G. Brebner, J. Janneck, J. Eker, C. Von Platen, M. Mattavelli, and M. Raulet, "OpenDF – A Dataflow Toolset for Reconfigurable Hardware and Multicore Systems", First Swedish Workshop on Multi-Core Computing, MCC 2008, Ronneby, Sweden, November 27-28, pp. 43-49.
[9] J. Eker and J. W. Janneck, "Cal language report," University of California at Berkeley, Tech. Rep. UCB/ERL M03/48, December 2003.
[10] I. Amer, C. Lucarz, M. Mattavelli, G. Roquier, M. Raulet, O. Deforges, and J.-F Nezan, "Reconfigurable Video Coding: The Video Coding Standard for Multi-core Platforms", initially accepted in IEEE Signal Processing Magazine, Special Issue on Signal Processing on Platforms with Multiple Cores.
[11] I. E. G. Richardson, Video CODEC Design: Developing Image and Video Compression Systems, John Wiley & Sons Ltd., April 2002.
[12] A. Tamhankar, and K. R. Rao, "An Overview of H.264/MPEG–4 Part 10," Proceedings of EURASIP Conference on Video/Image Processing and Multimedia Comm., Vol. 1, pp. 1-51, July 2003.
[13] R. Gonzalez and R. Woods, Digital Image Processing, Addison-Wesley Pub (Sd), March 2002.