VIDEO DECODER RECONFIGURATIONS AND AVS EXTENSIONS IN THE NEW MPEG RECONFIGURABLE VIDEO CODING FRAMEWORK

Dandan Ding, Lu Yu

Zhejiang University, China
Institute of Information
and Communication Engineering
{vickyddd, yul}@zju.edu.cn

Christophe Lucarz, Marco Mattavelli

Ecole Polytechnique
Fédérale de Lausanne, Switzerland
Microelectronic Systems Laboratory (GR-LSM)
{christophe.lucarz, marco.mattavelli}@epfl.ch

ABSTRACT

Multimedia devices are now required to support multiple coding standards. Supporting seamlessly both interoperability between standards and flexibility for application specific optimizations is a great challenge for current video coding technology. After a brief description of the new MPEG Reconfigurable Video Coding (RVC) framework, this paper describes possible decoder reconfigurations within this framework. The essential idea behind this framework is to reuse as most as possible the algorithms or architectures which are common to several different standards and to reconfigure video decoders in a flexible way at the coding tool level. A coding tool is an encapsulated piece of algorithm. Reconfiguration can address specific optimization objectives such as improvement in colour reproduction or higher performance at high bitrate. These simple examples show that the tool level definition of the video tool library is flexible enough to support the incremental introduction of new coding algorithms, the usage of algorithms taken from different video standards (i.e. AVS is provided in one example), and the possibility of high level reconfigurations. Thus, this paper demonstrates that the RVC framework offers a great flexibility in selecting coding tools for decoder reconfigurations to satisfy a wide variety of different applications.

***Index Terms*—** Reconfigurable Video Coding (RVC), MPEG-4, AVS, reconfiguration, coding tool

1. INTRODUCTION

MPEG has produced several video coding standards such as MPEG-1, MPEG-2, MPEG-4, and multimedia technologies such as MPEG-7, MPEG-21 and MPEG-A. Some other non-MPEG standards like VC-1 (Video Codec 1) and AVS (Audio Video coding Standard of China) are also under development. Nowadays, multimedia devices are usually required to support more than one of those standards. However, the current monolithic specification of standards (usually in C/C++) lacks flexibility and does not allow the combination of coding algorithms from different standards in order to achieve specific design or performance trade-offs and fill the requirements of the system. For example, not all coding tools defined in a profile@level of a specific standard are required in some application scenarios. For given application, codecs are either not exploited at their full potential or require unnecessarily complex implementations. For example, MPEG-4 AVC includes different syntax elements which are unnecessary in specific applications. However, a conformant decoder has to support all of them and may results in a non-efficient solution. Another example is the case of IPTV applications: existing video content is mainly coded using MPEG-2 whereas many service providers prefer MPEG-4 AVC for the higher compression efficiency. Transcoding between MPEG-4 AVC and MPEG-2 requires additional resources in the implementation. RVC offers a mechanism to build flexible decoder reconfiguration with a set of coding tool from different existing standards.

This paper presents the new specification formalism adopted in the RVC framework. Based on this formalism, designers can build data flow models of their algorithm with a set of self-contained modular elements coming from a standard library (Video Tool Library (VTL)). Thanks to the compactness and the intrinsically concurrent language used to define the models, RVC provides a great flexibility for the engineers to design video codecs. With a set of coding tools, the designers can explore rapidly the design space by interconnecting these different blocks to build new decoding solutions. Two different examples of decoder reconfiguration using the RVC framework are presented in this paper. The first example shows the reconfiguration of MPEG-4 Simple Profile (SP) decoder using the Inverse Quantization (IQ) and Inverse Transform (IT) tools taken from AVS standard. The second example shows the extension, currently not supported by any MPEG-4 profile, of the standard MPEG-4 SP decoder to support 4:2:2 and 4:4:4 subsampling pattern configurations.

The paper is organized as follows: section 2 introduces

This work was supported by the project of Natural Science Foundation of China under contract no. 90207005 and the working group of Audio Video coding Standard of China (AVS).

the essential components of the RVC framework. Section 3 briefly introduces the AVS standard and shows how one can use the concept of RVC framework to build a new decoder with better performance at high bitrate by using coding tools from two different standards (MPEG-4 and AVS). Section 4 shows the second reconfiguration example of MPEG-4 SP decoder providing higher colour rendering accuracy. Section 5 discusses the advantages of such an approach. Finally, section 6 concludes the paper.

2. THE MPEG RECONFIGURABLE VIDEO CODING (RVC) FRAMEWORK

The concept and objective of the RVC framework were firstly presented in [1] under the name Reconfigurable Media Coding (RMC). RVC is a framework which allows the definition of a multitude of different codecs, by combining together coding tools from a standard tool library. The essential elements of RVC framework are:

- the standard VTL which contains video coding tools, also named Functional Units (FU). A data flow language called CAL [2] is used to describe the algorithmic behaviour of the FUs. These latter are self-contained and communicate with the external world only by means of input and output ports from which FUs receive data tokens (input ports) and generate output data tokens (output ports).
- a language called Decoder Description Language (DDL) used to specify a decoder configuration made up of FUs taken from the VTL and the connections between those FUs.
- a MPEG-21 Bitstream Syntax Description Language (BSDL) schema which describes the syntax of the bitstream that the reconfigured decoder has to decode.

Based on the essential elements mentioned above, the components and processes that lead to a new decoder reconfiguration are:

- a Decoder Description (DD) written in DDL describing the architecture of the decoder, in terms of FUs and their connections.
- an Abstract Decoder Model (ADM), a behavioural model of the decoder composed of the syntax parser automatically generated from BSDL schema, FUs from the VTL and their connections.
- the final decoder implementation that is either generated by substituting any proprietary implementation, conformant in terms of I/O behaviour, of the standard FUs, or obtained directly from the ADM by generating SW and/or HW implementations by means of appropriate synthesis tools [3, 4].

A brief description of DDL and BSDL which are used to instantiate the Decoder Description (DD) and the associated BSDL schema necessary to generate the corresponding syntax parser is provided below.

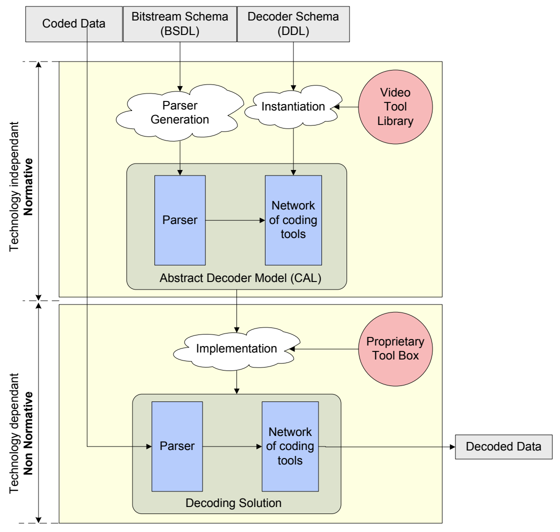

Fig. 1. Components and processes for deriving and Abstract Decoder Model and a platform dependent implementation in the RVC framework

2.1. Decoder Description Language (DDL)

The language used for the description of the decoder is an XML dialect which describes the interconnection of different FUs from VTLs. In RVC, the DDL is made available in the video bitstream and is used at the terminal to instantiate the decoder. The specific ”system-level” mechanisms applicable in different application scenarios are under development and are not described here. An example of a decoder description in DDL as result of reconfiguration is reported in section 4.

2.2. Bitstream Syntax Description Language (BSDL)

BSDL is also an XML dialect to describe the syntax of binary bitstreams. In MPEG-21, it is used to provide syntax description at different level of detail (i.e. GOPs, frames, slices…) enabling partial decoding of bitstreams for Digital Item Adaptation purposes. In RVC, with special extensions and restrictions, it is used to provide all information necessary for the complete parsing of any compliant bitstream. Thus BSDL provides a way to create schema for bitstreams and to instantiate new parsers that, obviously, cannot be present in the standard VTLs, but can be instantiated from the BSDL schema for a new decoder configuration. With minor extensions and restrictions to the standard MPEG-21 BSDL, RVC BSDL is able to fully describe the structure and syntax of a bitstream and thus is a good specification for the parsing procedure. An example of BSDL used for extending MPEG-4 SP is reported in figures 4, 6. A systematic procedure for the automatic synthesis of RVC syntax parsers in the same form of the FUs

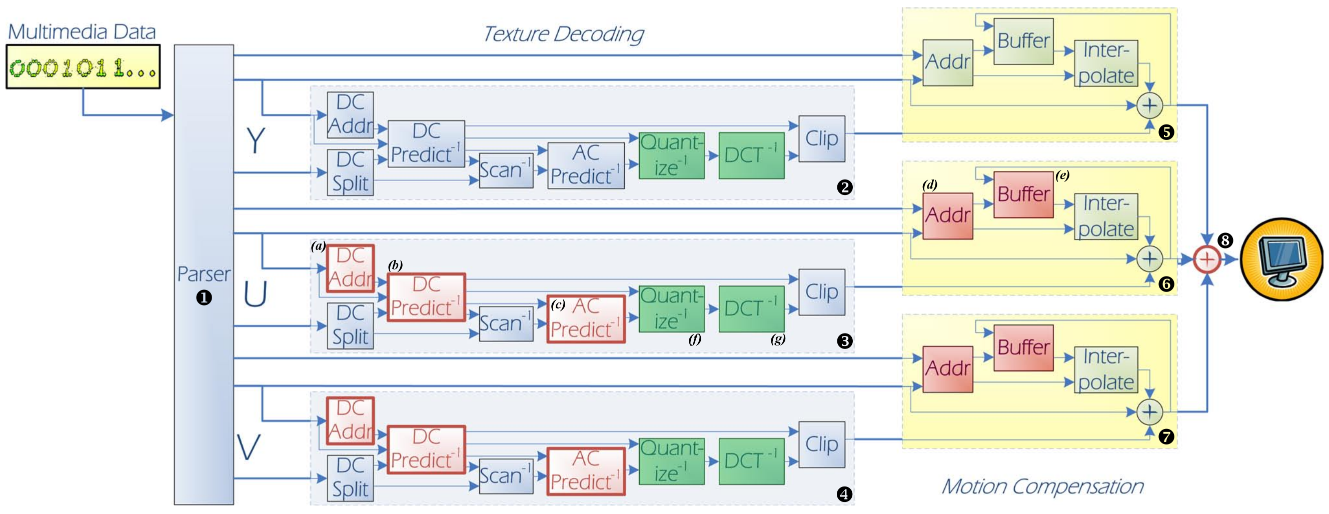

Fig. 2. The MPEG-4 SP decoder model in the RVC framework

from VTL can be found in [1, 5].

3. DECODER RECONFIGURATION BASED ON MPEG-4 SP USING INVERSE QUANTIZATION AND TRANSFORM FROM AVS

3.1. Audio Video coding Standard of China

Audio Video coding Standard of China (AVS) [6] is a new compression standard developed by AVS Workgroup of China. AVS Part 2 is addressing high-definition, high-density storage media and digital video broadcasting applications and was published as national standard for China in February 2006. Integer transform, intra and inter-picture prediction, in-loop deblocking filter and context-adaptive two dimensional variable length coding (CA-2D-VLC) are the key compression tools of AVS [7].

The Inverse Quantization (IQ) stage in AVS is characterized by a Quantization Parameter (QP) for which the reconstructed coefficients are obtained through a 1-D lookup only. In MPEG-4 Simple Profile (SP), two inverse quantization methods are used and DC coefficients of intra coded blocks are inverse quantized in a different manner. AVS implies only one quantization type and processes coefficients of the whole block in the same way.

The 8x8 pre-scaled Integer Inverse Cosine Transform (IICT) used in AVS provides a unique specification of the finite precision implementation and yields significant saving in processing complexity compared to the traditional Discrete Cosine Transform (DCT). IICT features are particularly interesting for low-end processors. A block diagram of the pre-scaled IICT is shown in Figure 3. It shows that the inverse scaling stage has been moved to the encoder side and has been combined with forward scaling into a single process [8].

Although the Inverse Quantization (IQ) and Inverse Transform (IT) algorithms are completely different between AVS and MPEG-4, they both use 8x8 blocks and have the same input and output token requirements. Thanks to the flexibility of the RVC frameworks, original MPEG-4 SP FUs can be easily replaced by the new FUs coming from AVS. This example shows the great reconfiguration potential of decoders defined within the RVC framework by replacing seamlessly the existing FUs by new ones which are more efficient. Obviously it should be noticed that decoder reconfigurations are not restricted to quantization and/or transform blocks of AVS and MPEG-4, but are extensible to all the existing video coding tools. The RVC framework provides a very interesting framework in which dynamic reconfigurations of decoders is straightforward.

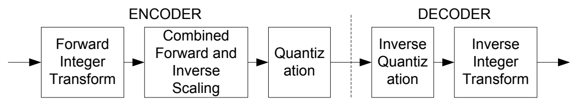
ENCODER
DECODER
Forward Integer Transform
Combined Forward and Inverse Scaling
Quantization
Inverse Quantization
Inverse Integer Transform

Fig. 3. Block diagram of pre-scaled Integer Inverse Cosine Transform (IICT) in AVS

3.2. Decoder reconfiguration

Concerning the coding tools, IQ and IT (FUs (f) and (g) on Y, U and V channels on Figure 2 are taken from AVS and the rest of the FUs comes from MPEG-4 SP. By combining those FUs, the resulting decoder can be seen as a hybrid configuration of MPEG-4 and AVS.

The bitstream description (written in BSDL) must be slightly modified from the MPEG-4 SP in order to be compatible with the new reconfigured decoder. The quantization precision of MPEG-4 SP decoder ranges on a 5 bits scale. In the reconfigured decoder, the quantization precision must be extended to 6 bits in order to apply the quantization algorithm of AVS. Thus, the elements of syntax `quant_precision`, `vop_quant` and `quant_scale` in the new bitstream description must be extended to 6 bits. The syntax elements `vop_quant` and `quant_scale` are both defined as VOPQuantType whose length in bits is equal to `quant_precision`. Figure 4 shows a fragment of the BSDL schema corresponding to the newly reconfigured decoder. It illustrates the necessary changes in the bitstream description corresponding to MPEG-4 SP in order that it is compatible with the new IQ and IT stages of the reconfigured decoder.

```
<xsd:schema>
  [...]
  <xsd:annotation><xsd:appinfo>
    <bs2x:variable name="m4v:quant_precision" value="6"/>
  </xsd:appinfo></xsd:annotation>
  [...]
  <xsd:element name="vop_quant" type="VOPQuantType"/>
  [...]
  <xsd:element name="quant_scale" type="VOPQuantType"/>
  [...]
  <xsd:simpleType name="VOPQuantType">
    <xsd:restriction base="bs1:b9">
      <xsd:annotation><xsd:appinfo>
        <bs2:bitLength value="$m4v:quant_precision"/>
      </xsd:appinfo></xsd:annotation>
    </xsd:restriction>
  </xsd:simpleType>
  [...]
</xsd:schema>
```

Fig. 4. Modification of the BSDL schema for the new decoder

3.3. Experimental results after decoder reconfiguration

Apart from showing the elegant reconfigurable features of RVC, it is necessary to look at the results and notice what are the advantages and the drawbacks of such a reconfiguration. This section presents the performance of the reconfigured decoder. The quantization precision has been extended to 6 bits in both the variable length encoder and decoder in order to make the encoding and decoding processes compliant. A reconfigured bitstream is produced and is provided as an input to the reconfigured decoder.

The overall decoding performance is evaluated under the following test conditions: I frames only, progressive sequence coding, VOL frame rate equal to 30, a sequence of 200 frames. In the MPEG-4 SP decoder, the quantizer range is equal to 5 bits, IDCT is used and the values of QP are $\{1, 2, 3, 5, 7, 11, 16, 21, 26, 31\}$. In the reconfigured decoder, the quantizer range is equal to 6 bits, IICT from AVS is used and the values of QP are $\{1, 9, 15, 22, 29, 36, 43, 50, 57, 63\}$.

Figure 5 reports the PSNR versus Bitrate curve. At low and medium bitrates, the two decoders yield similar performance. However, performance of the two decoders differs at high bitrate. The reconfigured decoder outperforms gradually the MPEG-4 SP as the bitrate increases.

In this example, coding tools from AVS are used in the reconfigured decoder which yields both complexity reduction

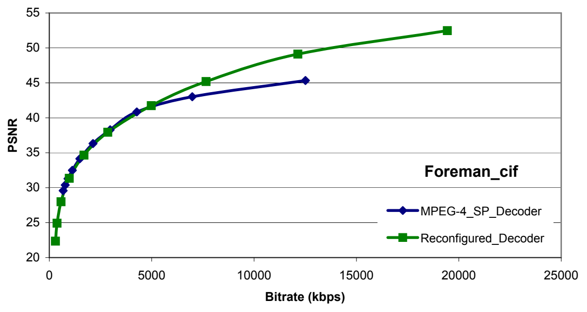

Fig. 5. Quality of the decoding versus Bitrate for the original MPEG-4 SP decoder and reconfigured decoder with the "Foreman_cif" sequence

and improvement of performance in specific bitrate ranges. Obviously, other reconfigurations can be achieved by selecting coding tools from different standards, such as intra prediction, half/quarter-pixel precision interpolation, MV prediction in order to reach specific complexity vs performance tradeoffs.

4. EXTENDING THE CHROMINANCE SUBSAMPLING PATTERNS OF THE MPEG-4 SP DECODER

This section presents the second example of decoder reconfiguration. The MPEG-4 SP decoder supports only 4:2:0 chrominance subsampling patterns. In order to improve chrominance rendering accuracy, it might be useful in some applications to use a MPEG-4 SP decoder which is capable of handling the 4:2:2 or 4:4:4 subsampling patterns. In the RVC framework, the reconfiguration of such decoders from the initial 4:2:0 configuration to 4:2:2 or 4:4:4 configurations is extremely simple.

4.1. Modification of the BSDL schema

The BSDL schema specifies the structure of the bitstream. Figure 6 shows a fragment of BSDL schema describing encoded luminance and chrominance 8x8 blocks.

The element "yBlock" is the same for the three subsampling configurations whereas elements "uBlock" and "vBlock" are different according to the different subsampling patterns configurations. In figure 6, modifications in the BSDL description of the new extended configurations are reported as comments to show the simple changes required by the 4:2:2 and 4:4:4 extensions. In the BSDL schema, "yBlock" element comprises 4 blocks in all configurations. For the chrominance components, "uBlock" and "vBlock" elements are composed of one block in the 4:2:0 configuration, of two blocks in the 4:2:2 configuration and of four blocks in the 4:4:4 configuration .

```
<xsd:group name="Blocks">
 <xsd:sequence>
 <!-- 420, 422 and 444 configurations -->
 <xsd:element name="yBlock" type="BlockType" bs2:nOccurs="4"/>

 <!-- 420 configuration -->
  <xsd:element name="uBlock" type="BlockType" bs2:nOccurs="1"/>
 <!-- bs2:nOccurs equals to "2" for the 422 config -->
 <!-- bs2:nOccurs equals to "4" for the 444 config -->

 <!-- 420 configuration -->
 <xsd:element name="vBlock" type="BlockType" bs2:nOccurs="1"/>
 <!-- bs2:nOccurs equals to "2" for the 422 config -->
 <!-- bs2:nOccurs equals to "4" for the 444 config -->
 </xsd:sequence>
</xsd:group>
```

Fig. 6. Modification of the BSDL schema for the new decoder

4.2. Modification of the decoder configuration

Since the bitstream syntax descriptions of the new extensions are different from the bitsteam syntax description of the original MPEG-4 SP, the new decoder configuration must contain a new parser and new FUs. How to synthesize a new parser from BSDL has been outlined in [1] and is further detailed in [5]. The other changes in the network of FUs described in the Decoder Description (DD) (figure 7) consist in replacing eight FUs (over the 38 FUs constituting the complete decoder) by new ones and modifying one parameter in the top level FU of the motion compensation network.

Concerning the Texture Decoding part, network no.2 on figure 2 (instance no.2 in the DDL description, "intra_FUs_16x16_Y") remains the same for the three configurations. The networks no.3 and 4 on figure 2 (instance no.3 and 4 in the Decoder Description, "intra_FUs_8x8_C") handle one block for the U and V components for the 4:2:0 configuration, whereas for the 4:2:2 and 4:4:4 configurations, such networks must handle two and four chrominance blocks respectively. Thus, FUs constituting these networks must be replaced by the appropriate versions which are available in the standard MPEG VTL. Such FUs are instantiated from the VTL using the "class" keyword in DDL. For the 4:2:2 configuration, instances no.3 and 4 are replaced by the network "intra_FUs_16x8_C", and for the 4:4:4 configuration, by "intra_FUs_16x16_C". The networks "intra_FUs_16x16_C", "intra_FUs_16x8_C" and "intra_FUs_8x8_C" are composed of eight actors. Only three of them make the difference between the three networks. (see Figure 2). Only "DC_Addr" (a), "DC_Predict-1" (b) and "AC_Predict-1" (c) FUs must be replaced in these networks in order to obtain the new reconfigured decoder.

Concerning the motion compensation part, networks no.5, 6 and 7 (instances no.5, 6, 7 in the Decoder Description) are built from the "motion" network. The reconfiguration consists in modifying the "LAYOUT" parameter among values 1 (one block), 2 (two blocks) and 4 (four blocks). On figure 2, in the "motion" network, only the actors "Addr" (d) and "Buffer" (e) use the "LAYOUT" parameter.

In order to reconstruct correctly the image, the FU no.8 on figure 2 (instance no.8 in the Decoder Description) must be replaced by the right FU according to the configuration. The MPEG VTL contains three FUs: "merger_420", "merger_422" and "merger_444".

Figure 7 is a fragment of the Decoder Description (in DDL) of the decoder of figure 2. The parameters given to each FU are not reported on the figure because of the lack of place.

5. DISCUSSION

This second example shows that the extensions of MPEG-4 SP decoder to 4:2:2 and 4:4:4 subsampling patterns are simple and straightforward by carrying out high level operations in the RVC framework. In data flow models, substitutions of FUs are easy and very localized in the decoder architecture. Minor modifications in the Decoder Description (DD) and the BSDL schema are required to extend the existing MPEG-4 SP decoder so that it supports the 4:2:2 and 4:4:4 subsampling patterns.

In general such changes in a classical decoder specification based on C/C++ reference SW would have required much more modifications. The first challenge would have been to localize in the entire program (approximately 40 000 lines of code) the lines of codes which must be modified. Such task is not as simple as with the RVC model in which the graphical representation of the FUs and their connections gives a rapid overview of the structure of the algorithm. The use of global variables may also constitute a big problem when modifying code in C/C++. In RVC models, global variables do not exist. It makes the algorithm much easier and safer to modify. Moreover, dealing with fewer lines of code than the corresponding C/C++ reference software is another advantage of the RVC models. For example, the MPEG-4 SP decoder is described in approximately 3000 lines code in CAL instead of the 40000 lines of code in the C reference software version [1].

These experiments show the great flexibility of the RVC framework enabling designers to reconfigure easily video decoders by replacing existing FUs of an existing decoder with new and more efficient coding tools. The usage of the RVC framework makes designers focus on the coding tool level definition of decoder algorithms in order to improve reusability and exchangeability of FUs. The success of the RVC framework will make obsolete the traditional MPEG codec-level definition.

6. CONCLUSION

This paper presents two examples of video decoder reconfiguration within the RVC framework. The first example describes a reconfigured decoder by combining FUs from MPEG-4 Simple Profile and AVS. The result is a performance improvement in the medium high bitrate range as well

```
<Network xmlns="decoder_420"
[...]
<Port kind="Input" name="mpeg"/>
<Port kind="Output" name="video_Y"/>
<Port kind="Output" name="video_U"/>
<Port kind="Output" name="video_V"/>

<Instance id="1">
 <Class name="parser"> <QID> <ID id="network"/> </QID>
 </Class>
 [...]
</Instance>

<Instance id="2">
 <Class name="intra_FUs_16x16_Y"> <!-- 420, 422 or 444 config -->
  <QID> <ID id="network"/> </QID>
 </Class>
 [...]
</Instance>

<Instance id="3">
 <Class name="intra_FUs_8x8_C"> <!-- 420 configuration -->
 <!-- the Class name is "intra_FUs_16x8_C" for 422 config -->
 <!-- the Class name is "intra_FUs_16x16_C" for 444 config -->
  <QID> <ID id="network"/> </QID>
 </Class>
 [...]
</Instance>

<Instance id="4">
 <Class name="intra_FUs_8x8_C"> <!-- 420 configuration -->
 <!-- the Class name is "intra_FUs_16x8_C" for 422 config -->
 <!-- the Class name is "intra_FUs_16x16_C" for 444 config -->
  <QID> <ID id="network"/> </QID>
 </Class>
 [...]
</Instance>

<Instance id="5">
 <Class name="motion">
  <QID> <ID id="network"/> </QID>
 </Class>
 <Parameter name="LAYOUT">
  <!-- 420,422 or 444 configurations-->
  <Expr kind="Literal" literal-kind="Integer" value="4"/>
 </Parameter>
 [...]
</Instance>

<Instance id="6">
 <Class name="motion">
  <QID> <ID id="network"/> </QID>
 </Class>
 <Parameter name="LAYOUT">
  <!-- 420 configuration -->
  <Expr kind="Literal" literal-kind="Integer" value="1"/>
  <!-- the value equals to "2" for 422 config, "4" for 444 config-->
 </Parameter>
 [...]
</Instance>

<Instance id="7">
 <Class name="motion">
  <QID> <ID id="network"/> </QID>
 </Class>
 <Parameter name="LAYOUT">
  <!-- 420 configuration -->
  <Expr kind="Literal" literal-kind="Integer" value="1"/>
  <!--value equals to "2" for 422 config, "4" for 444 config-->
 </Parameter>
 [...]
</Instance>

<Instance id="8">
 <Class name="merger_420"> <!-- 420 configuration -->
 <!-- The Class name is "merger_422" for the 422 configuration -->
 <!-- The Class name is "merger_444" for the 444 configuration -->
   <QID> <ID id="FU"/> </QID>
</Class>
</Instance>

<Connection src="" src-port="mpeg" dst="8" dst-port="in8"/>
<Connection src="8" src-port="out" dst="1" dst-port="BITS"/>
<Connection src="1" src-port="BTYPE_Y" dst="2" dst-port="BTYPE"/>
<Connection src="1" src-port="B_Y" dst="2" dst-port="QFS"/>
<Connection src="1" src-port="BTYPE_U" dst="3" dst-port="BTYPE"/>
<Connection src="1" src-port="B_U" dst="3" dst-port="QFS"/>
<Connection src="1" src-port="BTYPE_V" dst="4" dst-port="BTYPE"/>
<Connection src="1" src-port="B_V" dst="4" dst-port="QFS"/>
<Connection src="2" src-port="f" dst="5" dst-port="TEX"/>
<Connection src="3" src-port="f" dst="6" dst-port="TEX"/>
<Connection src="4" src-port="f" dst="7" dst-port="TEX"/>
<Connection src="1" src-port="MV_Y" dst="5" dst-port="MV"/>
<Connection src="1" src-port="BTYPE_Y" dst="5" dst-port="BTYPE"/>
<Connection src="1" src-port="MV_U" dst="6" dst-port="MV"/>
<Connection src="1" src-port="BTYPE_U" dst="6" dst-port="BTYPE"/>
<Connection src="1" src-port="MV_V" dst="7" dst-port="MV"/>
<Connection src="1" src-port="BTYPE_V" dst="7" dst-port="BTYPE"/>
<Connection src="5" src-port="VID" dst="" dst-port="video_Y"/>
<Connection src="6" src-port="VID" dst="" dst-port="video_U"/>
<Connection src="7" src-port="VID" dst="" dst-port="video_V"/>
</Network>
```

Fig. 7. Fragment of the Decoder Description of the reconfigured decoder

as a complexity reduction in implementation. The second example shows the reconfiguration potential of existing MPEG profiles to support new chrominance subsampling patterns. It is shown that the RVC framework is flexible enough to combine the existing coding tools with new FUs or FUs coming from different standards (for example AVS). Compared with the traditional codec level definition of video coding standards, the coding tool level specification of decoders within RVC allows a great flexibility in selecting coding tools and creating an ad-hoc decoder in order to satisfy different applications constraints.

7. REFERENCES

[1] Christophe Lucarz, Marco Mattavelli, Joseph Thomas-Kerr, and Jörn Janneck, “Reconfigurable Media Coding: A New Specification Model for Multimedia Coders,” in *IEEE Workshop on Signal Processing Systems*, 2007, pp. 481–486.

[2] Johan Eker and Jörn Janneck, “CAL Language Report,” 2003, ERL Technical Memo UCB/ERL M03/48.

[3] Matthieu Wipliez, Ghislain Roquier, Mickaël Raulet, Jean-Francois Nezan, and Olivier Déforges, “Code generation for the MPEG reconfigurable video coding framework: from CAL actions to C functions,” in *IEEE International Conference on Multimedia & Expo (ICME)*, Hannover, Germany, 2008.

[4] Jörn W. Janneck, Ian D. Miller, Dave B. Parlour, Marco Mattavelli, Christophe Lucarz, Matthieu Wipliez, Mickaël Raulet, and Ghislain Roquier, “Translating Dataflow Programs to Efficient Hardware: an MPEG-4 Simple Profile Decoder Case Study,” in *Design, Automation and Test in Europe (DATE)*, Munich, Germany, 2008.

[5] David Li, Dandan Ding, Christophe Lucarz, Samuel Keller, and Marco Mattavelli, “Validation of bitstream syntax and synthesis of parsers in the MPEG Reconfigurable Video Coding Framework,” in *IEEE Workshop on Signal Processing Systems*, 2008.

[6] China Audio and Video Standard (AVS), “GB/T 20090.2/-2006: Information technology & Advanced coding of audio and video Part2: Video,” .

[7] L. Yu, F. Yi, J. Dong, and C. Zhang, “Overview of AVS-video: tools, performance and complexity,” 2005, vol. 5960 of *Presented at the Society of Photo-Optical Instrumentation Engineers (SPIE) Conference*, pp. 679–690.

[8] Ci-Xun Zhang, Jian Lou, Lu Yu, Jie Dong, and Wai-Kuen Cham, “The technique of pre-scaled integer transform,” in *Circuits and Systems, 2005. ISCAS 2005. IEEE International Symposium on*, 2005, pp. 316–319 Vol. 1.