XML-Driven Bitstream Extraction Along the Temporal Axis of SMPTE's Video Codec 1

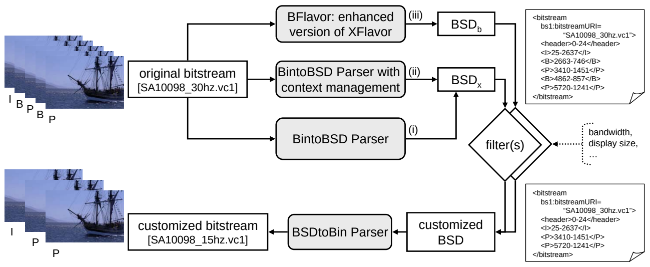
W. De Neve[1], **D. De Schrijver**[1], **D. Van Deursen**[1], **and R. Van de Walle**[2]

[1] Ghent University - IBBT, ELIS, Multimedia Lab
Gaston Crommenlaan 8 bus 201, B-9050 Ledeberg-Ghent, Belgium
[2] Ghent University - IBBT - IMEC, ELIS, Multimedia Lab
Gaston Crommenlaan 8 bus 201, B-9050 Ledeberg-Ghent, Belgium

Abstract The MPEG-21 Multimedia Framework sets out to transparently deliver digital media content to any device over any network. One of the tools that underlies this vision is the MPEG-21 Bitstream Syntax Description Language (MPEG-21 BSDL). This language sits at the core of an XML-driven architecture for media content customization, hereby allowing to tackle the huge diversity in the present-day terminal and network technology. In this paper, a discussion is provided on how Video Codec 1 (VC-1) encoded bitstreams can be fitted in such an adaptation framework. More precisely, we chart a few different strategies to automatically create XML-based descriptions of the high-level structure of VC-1 compliant bitstreams in order to steer a desired adaptation along their temporal axis. Some performance measurements in terms of computational times, file sizes, and memory consumption are presented as well.

1 Introduction

In August 2005, Video Codec 1 (VC-1) reached the Final Committee Draft (FCD) status with the C24 Technology Committee of the Society of Motion Picture and Television Engineers (SMPTE), in which it is officially referenced as SMPTE standard 421M [1]. This specification for digital video coding not only sits at the core of the Windows Media Series (Windows Media Video 9 is Microsoft's implementation of VC-1 [2]), but it is also included as a mandatory video compression format for the next-generation of High-Definition DVDs by both the Blu-Ray Disc Association and the DVD Forum (together with H.262/MPEG-2 Video and H.264/AVC). Hence, there is a good chance that VC-1 will be used in diverse usage environments, thus making it relevant to gain an insight into a framework that allows to realize an efficient and transparent customization of VC-1 compliant bitstreams. In this research, an XML-driven architecture is envisaged for the adaptation of VC-1 encoded bitstreams in the temporal domain in order to meet the constraints of a certain usage environment (user characteristics and preferences; device and network capabilities).



```
<bitstream
   bs1:bitstreamURI=
          "SA10098_30hz.vc1">
   <header>0-24</header>
   <I>25-2637</I>
   <B>2663-746</B>
   <P>3410-1451</P>
   <B>4862-857</B>
   <P>5720-1241</P>
</bitstream>
```

```
<bitstream
   bs1:bitstreamURI=
          "SA10098_30hz.vc1">
   <header>0-24</header>
   <I>25-2637</I>
   <P>3410-1451</P>
   <P>5720-1241</P>
</bitstream>
```

Fig. 1 XML-driven multimedia content adaptation

The outline of the paper is as follows. In Section 2, background information is provided on the concepts of XML-driven content adaptation. The main characteristics of VC-1 are elaborated from a high-level point of view in Section 3. Section 4 discusses some performance results as obtained for the adaptation of VC-1 encoded bitstreams. Finally, Section 5 concludes this paper.

2 XML-Driven Content Adaptation

Scalable coding is supposed to pave the way for several new multimedia architectures. They should make it possible to take into account the heterogeneity in the current plethora of devices and networks. In order to deliver scalable media in a diverse environment, it is important to be aware of the need of complementary logic that makes it possible to exploit the scalability properties of the compressed bitstream. This process typically involves the removal of certain data blocks and the modification of certain high-level syntax elements. One way to realize this scenario is to rely on automatically generated XML-based descriptions that contain information about the high-level structure of scalable bitstreams. In a next step, these structural metadata can be transformed to reflect a desired adaptation of a scalable bitstream, and can subsequently be used to automatically create an adapted bitstream. Typically, only a limited knowledge is required in order to generate an XML-based Bitstream Structure Description (BSD). As such, a BSD acts as an abstraction of the compressed bitstream.

During recent years, several languages have been developed that provide solutions for discovering the structure of a binary multimedia resource in order to generate its XML description and for the generation of an adapted multimedia resource using a transformed description by relying on format-agnostic software. For instance, one can think of the MPEG-21 Bitstream Syntax Description Language (MPEG-21 BSDL) and the Formal Language for Audio-Visual Object Representation, extended with XML features (XFlavor). MPEG-21 BSDL allows to express the structure of a media resource by using a modification of the W3C XML Schema language; XFlavor allows the same by relying on the principles of object-oriented programming languages.

Both technologies can be used as stand-alone tools; however, we have developed a joint approach to unify the two solutions and to combine their strengths. More precisely, the processing efficiency and flow-control flexibility provided by XFlavor on the one hand, and the ability to create compact BSDs by making use of MPEG-21 BSDL on the other hand, can be considered key to our motivation for the development of a novel bitstream structure description language. The latter is called BFlavor (BSDL + XFlavor), and is the result of a modification of XFlavor in order to be able to output BSDL compatible descriptions [3] [4]. To be more specific, BFlavor allows to describe the structure of a media resource in a C++-alike manner. It is subsequently possible to automatically create a BS Schema, as well as a code base for a parser that is able to generate a BSD that is compliant with the BS Schema. This implies that the generated BSDs can be further processed by the upstream tools in a BSDL-based adaptation chain (such as a generic BSDtoBin Parser; see further).

Figure 1 shows a high-level overview of a BSDL-oriented content adaptation chain. It illustrates three different approaches to create BSDL compliant BSDs: (i) by using the BintoBSD Parser as available in the MPEG-21 reference software package (version 1.2.1.); (ii) by relying on an optimized BintoBSD Parser, hereby using non-normative extensions to the BSDL Schema language in order to intelligently guide the tool with the context management (i.e., the in-memory representation of the BSD) [5]; and finally, by making use of BFlavor. In short, BFlavor and the extended version of MPEG-21 BSDL are two different approaches for circumventing the inefficient performance behavior of the format-agnostic BintoBSD Parser as available in the reference software. This is due to the XPath evaluation mechanism, used for getting access to context information (i.e., information already retrieved from the bitstream). The behavior in question will also be made clear by the performance results that are presented in Section 4. Note that our extensions to BSDL are under consideration by the MPEG consortium for standardization at the time of writing.

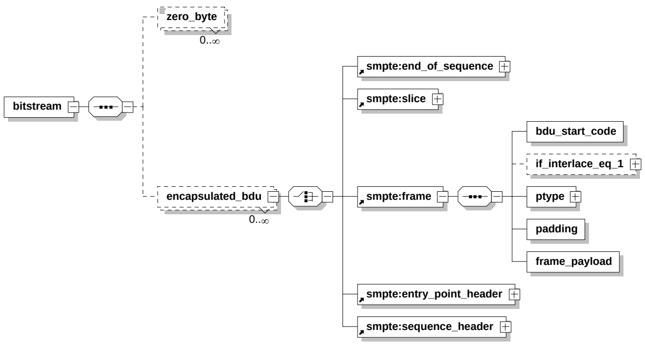


Fig. 2 Simplified VC-1 bitstream structure overview

3 Under the Hood of VC-1

SMPTE's VC-1 is an emerging standard for the coding of digital consumer video: pictures are represented in the YCbCr color space with 4:2:0 sampling, hereby using eight bits per component. The specification defines three profiles [1]. The Simple Profile targets low-rate internet streaming and low-complexity applications such as playback of media on Personal Digital Assistants (PDAs). The Main Profile aims at high-rate internet applications such as TV/Video-On-Demand over IP. The Advanced Profile (AP) focuses on broadcast applications, such as digital TV, HD DVD for PC playback, or HDTV. It is the only profile that supports interlaced content and the use of slices. In addition, the bitstreams that are compliant with this profile are self-contained; their decoding is not dependent on information that has to be conveyed by an external transport mechanism such as a file container. The latter observation does not hold true for bitstreams that are in line with the Simple and Main profile (due to another design philosophy). Their decoding requires Decoder Initialization Metadata (DIM). These metadata items have to be made available to a decoder prior to the start of the decoding process. For instance, the profile and level used have to be communicated to a decoder by external means in case of the Simple and Main profile, while this information is readily available for a decoder in case of AP compliant bitstreams. Hence, this explains why we have chosen to only describe the high-level structure of bitstreams that are satisfying the constraints of VC-1's most complex profile.

A simplified overview of the high-level structure of a VC-1 bitstream, in line with the possibilities of the AP, is provided in Figure 2. A BSD for a particular VC-1 bitstream is given in Figure 3. It is clear that a VC-1 bitstream consists of a number of Encapsulated Bitstream Data Units (EBDUs). Such units can carry compressed picture data (frame and slice EBDUs), as well

```
<bitstream bs1:bitstreamURI="./SA10098_30hz.vc1">
   <encapsulated_bdu>
      <sequence_header>
         <bdu_start_code>0000010F</bdu_start_code>
         <profile>3</profile>
         <level>2</level>
         <!-- ... -->
      </sequence_header>
   </encapsulated_bdu>
   <encapsulated_bdu>
      <entry_point_header>
         <bdu_start_code>0000010E</bdu_start_code>
         <broken_link>0</broken_link>
         <closed_entry>0</closed_entry>
         <!-- ... -->
      </entry_point_header>
   </encapsulated_bdu>
   <encapsulated_bdu>
      <frame>
         <bdu_start_code>0000010D</bdu_start_code>
         <ptype>
            <I>6</I>
         </ptype>
         <padding>0</padding>
         <frame_payload>143 17753</frame_payload>
      </frame>
   </encapsulated_bdu>
   <!-- ... -->
</bitstream>
```

Fig. 3 Simplified bitstream structure description

as header information (sequence and entry-point header EBDUs). A sequence-level header contains parameters that are used to decode a sequence of compressed pictures. The entry-point header has two purposes. First, it is used to signal a random access point: it guarantees that subsequent pictures can be decoded. Second, it is used to signal changes in the coding control parameters that are enabled for a particular entry point segment.

Further, the VC-1 specification also defines five types of pictures [1]. An I picture (intra coded picture) is coded using information only from itself; all its macroblocks are intra coded. A P picture is coded using motion compensated prediction (MCP) from past reference pictures. It can contain macroblocks that are inter or intra coded. A B picture is coded using motion compensated prediction from past and/or future reference pictures; macroblocks can be inter or intra coded. A BI picture is a B picture that only contains intra coded macroblocks. It cannot be used for predicting other pictures (see further). A Skipped picture is a P picture that is identical to its reference picture; its reconstruction is equivalent to copying the reference picture, implying that no further data is transmitted (similar to pseudo-skipped pictures in H.262/MPEG-2 Video and Non-coded Video Object Planes (N-VOPs) in MPEG-4 Visual).

B pictures in VC-1 are not used as a reference for subsequent pictures. They are placed outside the decoding loop, allowing shortcuts to be taken during their decoding without causing drift or long-term visual artifacts (temporal scalability). Intra coded B pictures (BI pictures) are also allowed in VC-1. They typically occur at scene changes where it is more economical to code the data as intra rather than P or B. This picture type is distinguished from true I pictures by disallowing them to be referenced by other pictures. This allows a decoder (e.g., on a constrained device) to omit decoding them. It also allows an encoder to choose a lower quality setting or quantization step size to encode them. Finally, an AP

Table 1 Bitstream characteristics

#Pic.	#EBDU	#seq	#entry	#I	#P	#B	#Skipped
150	1866	1	2	1	41	74	34
301	3531	1	2	1	79	150	71
450	5244	1	2	2	119	224	105
601	7084	1	2	2	155	300	144
750	8762	1	3	3	186	374	187
900	10498	1	3	3	225	449	223
1800	20994	1	6	6	450	898	446
2700	31490	1	8	9	675	1347	669
3600	41986	1	12	12	900	1796	892
4500	52482	1	15	15	1125	2245	1115
9000	104962	1	30	30	2250	4490	2230

bitstream offers a trivial form of spatial scalability: the coded size in pixels can be changed at entry points. This provides an encoder with the ability to alter the coded picture size and thus the bit rate.

4 Simulation Results

Some performance measurements are presented in this section. We are hereby targeting an off line scenario that requires the elimination of B and Skipped pictures in a bitstream that is compliant with VC-1's AP. Other use cases in the compressed temporal domain can be devised as well, such as content summarization and scene selection. The measurements were done on a PC having an Intel Pentium M 1.6 GHz CPU and 512 MB of system memory at its disposal. SAXON8 was used in order to apply Extensible Stylesheet Language Transformations (XSLTs) to BSDs. They make it possible to eliminate the B and Skipped pictures in the XML domain. An artificial set of test bitstreams was generated (by using the BSD approach) from a representative conformance test bitstream (SA10098.vc1; AP@L1; 6000kbps; 720x480; 30Hz; slice coding). This is due to the fact that a software encoder, able to output elementary bitstreams, was not available (yet) at the time of writing; only a reference decoder and a set of conformance bitstreams were at our disposal in the public domain. The characteristics of the resulting bitstreams are summarized in Table 1. Version 1.2.1 of the MPEG-21 BSDL reference software was used. The memory consumption was registered by relying on JProfiler 4.0.2.

The most important observations will now be put forward. Table 2 contains the times needed to generate a BSD by relying on the different approaches. BintoBSD$_r$ denotes the tool as available in the reference software. BintoBSD$_m$ refers to our modified version that supports the BSDL extensions for achieving a usable adaptation framework. PaU stands for Parse Unit (i.e., an EBDU). It is clear that the BintoBSD$_r$ Parser cannot be used in practice. It is characterized by a decreasing parsing speed and an increasing memory consumption (see Table 5), due to the fact that the entire BSD is kept in memory in order to allow the evaluation of arbitrary XPath expressions. The other strategies result in satisfactory processing speeds (i.e., faster than real time).

Table 2 BSD generation speeds

#Pic.	BintoBSD$_r$ (PaU/s)	BintoBSD$_m$ (PaU/s)	BFl. (PaU/s)	XFl. (PaU/s)
150	24.4	186.9	1304.6	162.5
301	15.7	200.6	1411.3	163.7
450	11.6	207.9	1530.4	181.8
601	9.0	212.3	1613.7	189.5
750	7.5	212.1	1615.7	174.2
900	6.4	214.4	1629.0	169.0
1800	3.3	215.1	1658.5	153.3
2700	1.3	215.2	1656.4	164.5
3600	-	204.9	1673.6	186.2
4500	-	201.6	1678.0	186.1
9000	-	197.1	1660.4	172.6

Table 3 Adaptation and bitstream generation times

	XSLT (s)			Bitstream Generation (s)		
#Pic.	BSDL	BFl.	XFl.	BSDL	BFl.	XFl.
150	0.5	0.5	60.7	0.7	0.7	16.8
301	0.7	0.7	716.1	0.8	0.7	28.5
450	0.9	1.0	-	0.8	0.9	-
601	1.1	1.2	-	1.0	1.0	-
750	1.2	1.5	-	1.1	1.1	-
900	1.5	1.7	-	1.2	1.3	-
1800	2.3	2.6	-	2.0	2.2	-
2700	2.9	3.3	-	3.4	3.2	-
3600	3.8	4.5	-	4.2	4.2	-
4500	4.4	5.1	-	5.8	5.6	-
9000	8.4	9.3	-	11.9	32.3	-

Table 4 File sizes

	Original Files (MB)				Transformed Files (MB)			
#Pic.	Bitstr.	BSDL	BFl.	XFl.	BSDL	BFl.	XFl.	Bitstr.
150	4.2	0.5	0.6	88.7	0.3	0.3	27.9	1.3
301	7.7	1.0	1.2	161.9	0.5	0.6	47.3	2.3
450	10.9	1.5	1.7	228.1	0.7	0.9	-	3.2
601	14.1	2.0	2.3	295.1	0.9	1.1	-	4.0
750	17.5	2.5	2.9	366.8	1.1	1.4	-	4.9
900	21.1	3.0	3.5	442.1	1.4	1.7	-	6.0
1800	42.1	6.0	7.0	884.2	2.7	3.4	-	12.0
2700	63.2	9.1	10.5	1326.3	4.1	5.1	-	18.0
3600	84.2	12.1	13.9	1768.5	5.4	6.8	-	24.0
4500	105.3	15.1	17.4	2210.6	6.8	8.4	-	30.0
9000	210.6	30.3	34.9	4421.1	13.6	16.9	-	60.0

Table 5 Peak system memory consumption

	BSD Generation(MB)				XSLT	BSDtoBin
#Pic.	BintoBSD$_r$	BintoBSD$_m$	BFl.	XFl.	(MB)	(MB)
150	6.6	1.8	0.7	0.7	3.1	0.9
301	12.2	1.7	0.7	0.8	4.7	1.1
450	19.0	1.7	0.7	0.8	8.8	0.8
601	33.8	1.8	0.7	0.8	8.7	1.0
750	40.0	1.8	0.7	0.9	17.3	1.1
900	61.0	1.9	0.7	0.9	17.0	1.0
1800	65.0	1.9	0.7	0.9	41.0	1.3
2700	-	1.8	0.7	1.4	33.4	1.3
3600	-	1.9	0.7	1.6	65.4	1.2
4500	-	1.8	0.7	1.8	61.6	1.1
9000	-	1.9	0.7	1.9	123.5	1.3

Table 3 summarizes the times needed for customizing the different BSDs and for deriving a tailored child bitstream. The measurements make clear that the transformation of XFlavor BSDs fails in this regard. As illustrated by Table 4, this is due to the large size of those BSDs: in order to allow a maximized transformation space, an XFlavor BSD has to contain all bitstream data. As such, it does not act as an additional metadata layer on top of the compressed bitstream (as is the case for the BSDL and BFlavor BSDs). Note that the generated BSDs can be compressed efficiently as well (i.e., compression ratios of up to 98% can be achieved when using common compression software). Finally, Table 5 gives an overview of the memory consumption of the different tools involved. The BintoBSD$_m$ Parser, as well as the BFlavor and XFlavor parsers, are characterized by a constant and low memory usage. The generic BSDtoBin Parser shows a low and constant memory usage too (results shown for BFlavor BSDs). The XSLT engine requires an increasing amount of memory (an entire DOM tree is needed; results shown for BFlavor BSDs). This illustrates the necessity to pay attention to the transformation technology used in practical situations.

5 Conclusion

In this paper, several languages were discussed that provide a solution for translating the structure of a VC-1 bitstream into an XML description, and for the generation of an adapted bitstream using a transformed description. The performance measurements illustrate that BFlavor (BSDL + XFlavor) and our extended version of MPEG-21 BSDL offer an elegant and practical solution for the description-driven customization of VC-1 bitstreams in the temporal domain.

Acknowledgement: The research as described in this paper was funded by Ghent University, the Interdisciplinary Institute for Broadband Technology (IBBT), the Institute for the Promotion of Innovation by Science and Technology in Flanders (IWT), the Fund for Scientific Research-Flanders (FWO-Flanders), the Belgian Federal Science Policy Office (BFSPO), and the European Union.

References

1. "Proposed SMPTE Standard for Television: VC-1 Compressed Video Bitstream Format and Decoding Process," document 421M, SMPTE, New York, USA, August 2005.
2. S. Srinivasan, P. J. Hsu, T. Holcomb, K. Mukerjee, S. L. Regunathan, B. Lin, J. Liang, M.-C. Lee, J. Ribas-Corbera, "Windows Media Video 9: overview and applications," *Signal Processing: Image Communication*, vol. 19, pp. 851–875, 2004.
3. W. De Neve, D. Van Deursen, D. De Schrijver, K. De Wolf, R. Van de Walle, "Using Bitstream Structure Descriptions for the Exploitation of Multi-layered Temporal Scalability in H.264/MPEG-4 AVC's Base Specification," *Proc. of PCM 2005*, Springer-Verlag, pp. 641–652, Chejudo, 2005.
4. D. Van Deursen, W. De Neve, D. De Schrijver, R. Van de Walle, "BFlavor: an optimized XML-based framework for multimedia content customization," *Proceedings of PCS 2006*, Accepted for publication, China, 2006.
5. D. De Schrijver, W. De Neve, K. De Wolf, R. Van de Walle, "Generating MPEG-21 BSDL Descriptions Using Context-Related Attributes," *Proceedings of the 7th IEEE ISM conference*, pp. 79–86, USA, 2005.