Overview of Bitstream Syntax and Parser Description Languages for Media Codecs

Hyungyu Kim and Euee S. Jang

Department of Electronics and Computer Engineering, Hanyang University / Seoul, Republic of Korea
dmlab.khg@gmail.com, esjang@hanyang.ac.kr

***** Corresponding Author: Euee S. Jang

Received April 15, 2013; Revised April 26, 2013; Accepted May 13, 2013; Published June 30, 2013

** Regular Paper*

***Abstract*:** This paper reviews various bitstream syntax and parser description (BSPD) languages used in MPEG multimedia standards. Traditionally, the bitstream syntax and semantics have been described in human-readable form. Several languages have been developed to describe the bitstream syntax using a computer-readable language and allow the automatic generation of bitstream parsing function from the description. The languages were designed for different objectives and applications but have a range of commonalities in functionality. The aim of this paper is to provide a historical overview of BSPD languages. The background and target application of the BSPD languages are reviewed. In addition, the technical features of each languages, including the linguistic basis (e.g., XML-based) and parser generation method, are discussed and evaluated. In addition, previous studies based on each language are introduced and categorized according to their objectives. Finally, the relevant technical issues that suggest the direction of the future researches are reported.

***Keywords*:** Bitstream syntax, Bitstream parser, MPEG, MSDL, DIA, RMC, BSD

1. Introduction

In digital multimedia applications, a range of media codecs including MPEG-2 [1], MP3 [2], or JPEG [3] are being used in many fields including digital broadcasting, DVD, music player, and internet media. The codec includes the encoder and decoder. For handshaking between them, it is necessary to have a common rule for the structure of the bitstream, which is an output of the encoder and an input of the decoder. The rule is called the bitstream specification, which consists of bitstream syntax and bitstream semantics. The bitstream syntax describes the order of information arranged in a bitstream. In particular, the length of the bits to represent each compressed data and the way of organizing the data within the bitstream are defined in the bitstream syntax. Bitstream semantics explain the contextual meaning of each data, which includes the use of each data in the bitstream and the procedure for how to parse the data.

A range of methods have been used to describe the bitstream syntax and semantics. For example, the notation of a bitstream using figure- or table-based specification has often been used in the international standards developed by the Moving Picture Experts Group (MPEG), the Joint Photographic Experts Group (JPEG), and International Telecommunication Union Telecommunications Standardization Sector (ITU-T). In addition to such human-readable description methods, computer-readable languages to describe the bitstream syntax and semantics have been developed for easy maintenance and better consistency of the bitstream specification. This paper calls such languages *Bitstream Syntax and Parsing Description* (BSPD) languages.

In MPEG, several BSPD languages have been developed based on the different requirements and applications. In particular, the following international standards include the BSPD technology: MPEG-4 Part 1 System [4], MPEG-21 Part 7 Digital Item Adaptation (DIA) [5], and MPEG-B Part 4 Codec Configuration Representation (CCR) [6]. During the standardization of MPEG-4, the *MPEG Syntactic Description Language* (MSDL) was developed and included in the System part of the standard in response to the increasing need for a

standardized way of describing the bitstream structure. Later, the MPEG-21 DIA standard included a tool for transforming a multimedia bitstream into an Extensible Markup Language (XML) metadata document. For that purpose, a BSPD language was developed to describe the structure of bitstream and XML document simultaneously. MPEG-B Part 4, which is also known as the *MPEG Reconfigurable Media Coding* (RMC) framework, is another MPEG standard that requires the BSPD language. In this framework, a codec is described by a number of modular algorithms and the function of the bitstream parser module is described by the BSPD language.

The aim of this study was to provide an overview of various BSPD languages and relevant research studies. Although the BSPD language is used in many MPEG standards, previous studies focused mainly on the language as an integral part of each standard technology. Very little research has been performed to point out the commonalities and differences of various BSPD languages in different standard fields. Therefore, this paper reviews the various BSPD languages that are related to the MPEG standards. Accordingly, the technical concepts of the BSPD languages are introduced and the research trends based on the languages are reviewed. The features of the BSPD languages are also discussed. For each language, the functionality of the language is introduced and the target application of the language is identified. Furthermore, critical review based on the requirements of the application is conducted.

The remainder of this paper is structured as follows. Section 2 introduces the traditional bitstream specification description methods, as a technical background for the BSPD languages. The currently available BSPD languages relevant to the MPEG standards are reviewed in Section 3. A functional comparison among the BSPD languages is made in Section 4 and this paper is concluded in Section 5 with suggestions for future research directions.

2. Background and the Traditional Description Methods

A range of bitstream specifications are available in a number of coding standards. Despite their diversity, they share some basic concepts of the bitstream description. Fig. 1 shows the bitstream handling process in an encoder and a decoder. Because the bitstream generation and parsing process can be considered a derivation of the *Universal Turing Machine* [7], any bitstream specification can be represented using the following set of information:

- A set of *bitstream syntax elements*, which is a coded representation of various data.
- The *bitstream syntax*, which indicates the length and order of the syntax elements in the bitstream, which are explained by describing the sequential, conditional, or hierarchical appearance of the elements.
- The *bitstream semantics*, which describes the contextual meaning and parsing (and/or generation) process of the bitstream syntax elements.

Traditionally, the bitstream specification in the coding standards has been described mainly in a human-readable way. Although the bitstream syntax has been described in a number of different ways, bitstream semantics have described textually because the semantic meaning should be understood by the humans who implement the codec. In the case of bitstream syntax description, the human-readable description methods can be classified into two categories: *figure-based notation* and *table-based notation*.

Figure-based notation is the bitstream syntax description method used in many technical standards. Among the media coding standards, still image codecs designed by JPEG (i.e., JPEG [3] and JPEG2000 [8]), JBIG [9], and early video coding standards designed by ITU-T (i.e., H.120 [10] and H.261 [11]) have used this notation. In this notation, the bitstream syntax is described using block diagrams and directional graphs. The block diagram mainly describes the length, order, and hierarchy

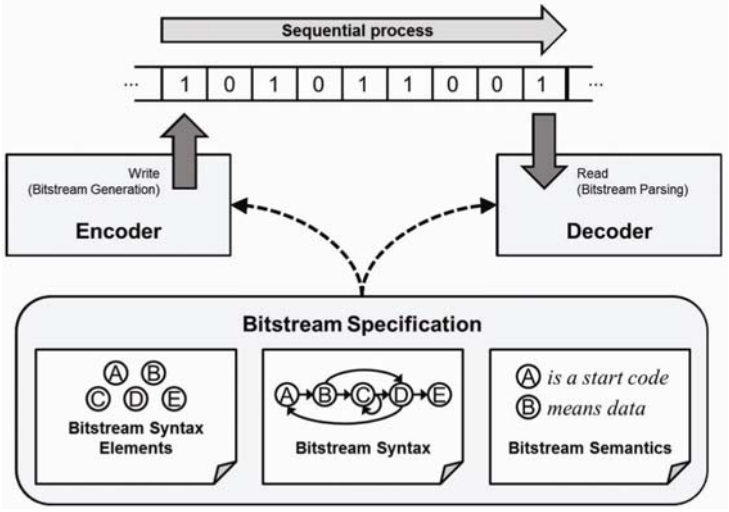

Fig. 1. Bitstream Handling in Codecs.

of the bitstream syntax elements, whereas the graph mainly describes the control flow (e.g., conditional branch or loop) during the bitstream parsing process.

Fig. 2 gives an example of the block diagram, which is a part of Fig. B.4 in [3]. In the block diagram, a bitstream is described as a number of blocks. Each block that is arranged in a sequential order represents a syntax element. The length of each bitstream syntax element is described on the diagram by the width of the block.

The graph has also been used to complement the description of parsing flow for a block diagram. Fig. 3 gives an example of the graph describing parsing flow, which is a part of Fig. B.17 in [3]. The bitstream syntax elements, which are drawn as blocks in the block diagram, are represented by graph nodes. The parsing flow of the syntax elements is described by directional connections. Using the connections, the directional graph can describe a simple conditional branch or loop.

The combination of the block diagram and the parsing flow graph provides an easy-to-understand explanation of the bitstream structure. The order of the bitstream syntax elements can be shown clearly and the hierarchical structure of the bitstream syntax can be described easily. On the other hand, it is difficult to describe complicated bitstream syntax structure using the block diagram and graph. For example, the syntax description using a block diagram describes the syntax elements with the blocks representing chunks of 4-, 8-, or 16-bits in the JPEG standard. The meaning of each bit in the blocks should be described separately in a textual and ad hoc manner. Therefore, complicated bitstream syntax consists of syntax elements with a range of bit lengths (e.g., bitstream syntax of MPEG-4 Visual) is quite difficult to describe using the block diagram. Moreover, the description for parsing flow is scattered in the block diagram, graph, and textual semantics description, making the specification lengthy and complex. Therefore, the figure-based notation is not feasible for describing long and complex bitstream syntax, such as that of a video codec.

On the other hand, table-based notation is another description method for bitstream specification. This can support a highly complex bitstream structure. The table-based notation was used initially in the MPEG-1 [12] video coding standard and has been used continuously in the most MPEG multimedia standards and recent ITU-T video coding standards owing to its strong description capability. The standards include MPEG-1 Part 2 Video [12], MPEG-2 Part 2 Video [1], MPEG-4 Part 2 Visual [13], MPEG-4 Part 10 Advanced Video Coding (AVC)/H.264 [14], and MPEG-H High Efficiency Video Coding (HEVC) [15].

Fig. 4 gives an example of the bitstream syntax description using the table-based notation. The table shown in Fig. 4 was excerpted from ISO/IEC 14496-2 Section 6.2.2 [13]. In the table-based notation, the order of the bitstream syntax element was considered in sequential order starting from the top of the table. Each row in the table represents a bitstream syntax element or a C-like pseudo-code that indicates parsing flow. The conditional branch (i.e., if-else or switch-case), loop (i.e., for or while), and function call can be described using pseudo-code. A remarkable feature of the table-based notation is the table that describes the bitstream syntax as well as some semantic information for syntax elements. For each bitstream syntax element, its bit length and its data type (e.g., bit string or integer) are annotated in the same row, as shown in Fig. 4.

The table-based notation has several benefits. First, the

Fig. 2. Example of a block diagram.

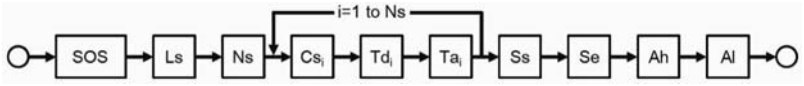

Fig. 3. Example graph describing a parsing flow.

VisualObjectSequence() {	No. of bits	Mnemonic
do {		
visual_object_sequence_start_code	32	bslbf
profile_and_level_indication	8	uimsbf
if (profile_and_level_indication == 11100001-11101000) {		
next_start_code_studio()		
extension_and_user_data(0)		
StudioVisualObject()		
} else {		
while (next_bits()== user_data_start_code){		
user_data()		
}		
VisualObject()		
}		
} while (next_bits() != visual_object_sequence_end_code)		
visual_object_sequence_end_code	32	bslbf
}		

Fig. 4. Example of table-based notation.

table-based notation is easy to design and maintain. The structure of the bitstream can be understood easily and the bitstream syntax can be modified conveniently row-by-row. Moreover, the table-based notation incorporates some semantic information into the bitstream syntax description, allowing a highly comprehensive description of the bitstream specification. In addition, the table is much more space-efficient than the figure-based notation. The stacked rows in the table can efficiently describe a number of bitstream syntax elements in sequential order. This is a significant merit when describing long and complex bitstream syntax, such as for a video codec.

On the other hand, the table-based notation has some limitations, even though it is a very efficient tool for describing the bitstream syntax. One significant problem is that the bitstream parsing process cannot be understood solely by a table-based description. Some portion of the bitstream parsing process, e.g., entropy decoding, should still be explained by the textual bitstream semantics or the reference software implementation.

Another problem is the consistency between bitstream specification and actual codec implementation. Theoretically, the bitstream parsing process implemented in the reference software should be identical to the textual bitstream specifications. On the other hand, frequent technical modifications during the standardization process would cause mismatch between them. Therefore, human effort is needed to restore and maintain the consistency. Effort is also needed not only for consistency but also for the accuracy of the bitstream specification. If there is an error in the textual bitstream specification, it is difficult to detect until a problem occurs during the bitstream parser implementation. Similarly, the aforementioned drawback regarding the table-based notation can be found in any type of human-readable description method. Hence, technical solutions to improve the flaws of the traditional methods are needed.

3. MPEG Standards and BSPD Languages

As discussed above, the human-readable methods for the bitstream specification have certain limits on helping the developers implement the bitstream parsing functionality. Computer-readable description method for the bitstream specification, i.e., *Bitstream Syntax and Parsing Description* (BSPD) language, can overcome the limits of the human-readable methods.

The concept of BSPD language is to have a computer-readable language that describes the bitstream specifications (and parsing process if possible). The language allows easy development, maintenance, and implementation of bitstream specification through the computer readability. Fig. 5 shows the concept of BSPD language. A bitstream specification is defined by the developer using a BSPD language and can be debugged conveniently with the support of the corresponding compiler. Furthermore, the specifications written in BSPD language can be used to automatically generate a bitstream parser in codec implementation through translation or run-time interpretation.

In MPEG, three standards support the BSPD language: MPEG-4 Part 1 System [4], MPEG-21 Part 7 Digital Item Adaptation (DIA) [5], and MPEG-B Part 4 Codec Configuration Representation (CCR) [6].

3.1. MPEG-4 Systems

MPEG-4 Part 1 System [4] is one of the first attempts to adopt the concept of BSPD language. MPEG-4 was designed to be an “object-based” codec: in MPEG-4, video or audio is considered one of the *objects* to build multimedia scenery. When drafting the requirements for MPEG-4, the experts in MPEG predicted that the new standard will require a very complicated bitstream structure to contain a range of multimedia objects. They were also concerned about the future maintenance of the bitstream syntax when a new profile or object is developed and included in the codec newly. Owing to these considerations, the need for a language to describe the bitstream syntax of the new codec was included in the requirements for MPEG-4 Systems standard [16].

In these requirements, three levels of decoder programmability were defined as levels 0, 1, and 2. Level 0 indicates a decoder that is not programmable, similar to

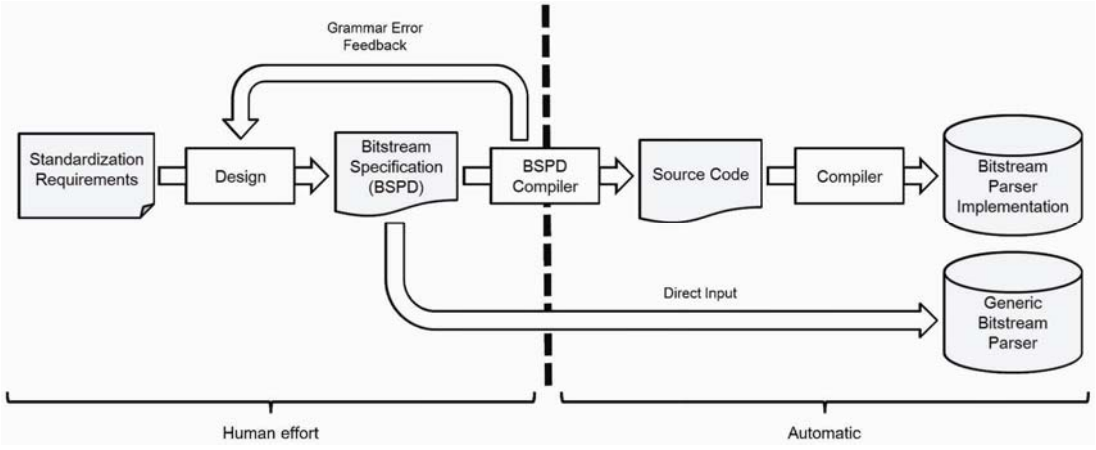

Fig. 5. General concept of BSPD.

the conventional codec implementations; level 1 means a decoder that can be configured from a predefined set of standardized tools according to a given script; and level 2 indicates a decoder that can be extended by downloading new decoding tools on-the-fly. When standardization began, the target of standardization was to achieve the support for level 1 programmability in MPEG-4. To reach the goal, several conceptual languages to describe the MPEG-4 system were specified in the requirements [16-17]. The Syntactic Description Language was one of the several conceptual languages and has become the *MPEG Syntactic Description Language* (MSDL).

MSDL, which is also known as *Formal Language for Audio-Visual Representation* (Flavor), is a language developed at Columbia University, NY, United States. The origin of this language is a Perl script called *mkvlc*, which is a project to automatically generate a variable length decoder from a given Huffman table [18]. This effort has advanced toward the generic description language for the entire bitstream syntax and eventually proposed to MPEG as MSDL.

Fig. 6 gives an example of bitstream specification written in MSDL [19], which describes the *macroblock* syntax description of MPEG-4 Visual [12]. The language is designed to be an extension of the table-based bitstream syntax notation in the MPEG-2 standard [1]. The overall grammar of MSDL follows the grammar of C++ and Java [20]. Therefore, the language natively supports C-style operators and control flow statements (e.g., if, for, while). In addition to the C-based grammar of the table-based notation, MSDL also includes an object-oriented concept.

In MSDL, each bitstream syntax element is defined as a variable declaration with an extended grammar to annotate the bit length. Such an extended variable is called a *parsable variable* in MSDL. The hierarchical structure of a bitstream can be described using the *classes* that consist of parsable variables. The object-oriented concept of MSDL supports the inheritance and polymorphism of the classes in a C++ or Java style. In addition, MSDL provides built-in functionality to describe the entropy coded bitstream syntax elements. MSDL supports variable length coding using a Huffman table and binary arithmetic coding. Even more complex algorithms can be described using the *verbatim code* feature, which allows C++ or Java code to be incorporated directly within a MSDL description.

Several studies related to MSDL have been reported. These studies include (1) introduction to MSDL [17, 20-22], (2) examination of the automatic generation of entropy decoder in MSDL [23-24], and (3) extension of MSDL (Flavor) for use in the Digital Item Adaptation (DIA) framework, which is XFlavor [25-26] and BFlavor [27-29].

MSDL is a powerful language that enables a comprehensive description of the bitstream specification. The bitstream parsing process can be described on its own without textual annotations using MSDL. In particular, MSDL provides a unique grammar to describe the parameters and tables for the entropy coding and decoding process, even though the supported coding algorithms are limited. Because of this feature, MSDL overcomes the

```
class MACROBLOCK(int vop_coding_type, bit interlaced, bit field_prediction,
                 bit fcode) {
    int i;
    int cbp[6];
    if (vop_coding_type != VOP_I)
      bit(1) not_coded;
    if (!not_coded || vop_coding_type == VOP_I) {
      if (vop_coding_type == VOP_I) MCBPC(TABLE_B_6) mcbpc;
      if (vop_coding_type == VOP_P) MCBPC(TABLE_B_7) mcbpc;
    }
    int derived_mb_type = mcbpc.mb_type;

    cbp[5] = (mcbpc.cbpc & 0b10) >> 1;
    cbp[6] = (mcbpc.cbpc & 0b01);

    if (derived_mb_type == 3 || derived_mb_type == 4)
      bit(1) ac_pred_flag;
    int(TABLE_B_8) cbpy;

    cbp[0] = (cbpy & 0b1000) >> 3;
    cbp[1] = (cbpy & 0b0100) >> 2;
    cbp[2] = (cbpy & 0b0010) >> 1;
    cbp[3] = (cbpy & 0b0001);

    if (derived_mb_type == 1 || derived_mb_type == 4)
      int(TABLE_6_32) dquant;
    if (interlaced)
      INTERLACED_INFORMATION(vop_coding_type, derived_mb_type,
                             field_prediction) interlaced_info;
    if ((derived_mb_type == 0 || derived_mb_type == 1)
        && vop_coding_type == VOP_P) {
      MOTION_VECTOR(MV_F, fcode);
      if (interlaced && interlaced_info.field_prediction)
        MOTION_VECTOR(MV_F, fcode);
    }
    if (derived_mb_type == 2) {
      for (i=0; i<4; i++)
        if (!cbp[i])
          MOTION_VECTOR(MV_F, fcode);
    }
    for (i=0; i<BLOCK_COUNT; i++) {
      BLOCK(i, derived_mb_type, cbp);
    }
}
```

Fig. 6. Example of MSDL (Flavor) description.

drawbacks of the traditional human-readable methods and significantly helps the codec designers and software developers. Another merit of MSDL is its encoder support. The language is designed not to be dependent on either the encoder or decoder. Therefore, the same MSDL document can be used for the bitstream generation and bitstream parsing process without modification.

On the other hand, despite the aforementioned strengths of MSDL, the language is not being popularly used, which was not expected in the beginning. The bitstream specification written in MSDL can be found in few MPEG standards including MPEG-4 Part 1 System [4] and [30-32]. On the other hand, most MPEG coding standards including video and audio codecs still use the table-based notation. Many standard developers remain in a traditional way because such a fundamental change in standardization methodology was not urgent.

3.2. Digital Item Adaptation

MPEG-21 Digital Item Adaption (DIA) framework [5] is a framework to cope with the increasing diversity of terminal environments for multimedia playback. For example, while a high resolution video stream is suitable for playback on television or a PC, it is unsuitable for mobile devices that require a narrower network bandwidth, smaller power consumption and smaller screen resolution. The DIA framework resolves this problem by adapting the multimedia data to fit in the target environment. Fig. 7 presents a simplified illustration of the adaptation process in the DIA framework.

During the adaptation process, multimedia data in a bitstream is parsed and reproduced into a metadata format written in XML. The XML metadata works as an intermediate format between the source bitstream and adapted bitstream. To allow a bitstream to be converted to an XML document and vice versa, the BSPD language is in demand because designing a DIA engine for every codec is not feasible and the engine should be generic (i.e., format-agnostic). A method to inform the generic engine of the structure of the bitstream to be parsed is needed. Owing to the peculiar functional demand, a BSPD language for the DIA framework should fulfill the following technical requirements:

- The language should describe the structure of the bitstream and XML document at the same time (Figs. 7(a), (c), and (e)).
- The language should describe the parsing and generation process of bitstream at the same time (Figs. 7(b) and (d)).

Accordingly, the BSPD languages developed for the DIA framework have a strong relationship with the XML schema [33] technology. This paper reviewed two appropriate BSPD languages available that can be used in the DIA framework: *Bitstream Syntax Description Language* (BSDL) and *Flavor* (MSDL).

BSDL is a language to represent a bitstream content using an extended XML schema. This language was first specified within MPEG-21 Part 7 [5] and later separated as an individual standard, MPEG-B Part 5 [34]. In BSDL, the XML schema grammar feature is extended by adding various XML attributes related to the bitstream structure and bitstream parsing. Fig. 8 gives an example of BSDL description from the ISO/IEC 21000-7 Section 4.2.2 [5]. Each bitstream syntax element is represented as an XML element declaration in BSDL. In addition to the element declaration based on the XML schema grammar, BSDL adds several attributes to annotate bit length, data type, and control flow statements. The control flow in the bitstream syntax can be defined using BSDL-2 grammar, which includes attributes to annotate the condition or loop for elements or segments. In the control flow statements, XPath expression [35] is used to declare and reference the variables.

The advantage of BSDL is its strong compatibility with XML because the main objective of BSPD language in the DIA framework is the generation of XML metadata from a bitstream. An XML metadata document generated using BSDL clearly shows the structure and hierarchy in a very natural XML format, which is highly human-readable. This can be manipulated easily by various applications. In addition, the backward compatibility as an XML schema allows the schema file written in BSDL to be used not only within the DIA framework but also with ordinary XML applications to validate the XML structure of a BSD.

On the other hand, BSDL has some weaknesses. BSDL is designed to describe high-level bitstream syntax and is not intended to express complicated bitstream structures, such as entropy coding. Another problem is that the bitstream parsing speed of BSDL is slow. Several articles highlight the exponential slowdown of the bitstream parsing software based on BSDL, which was caused by the inefficient design of the XPath-based bitstream content handling model [28, 36-39].

Many attempts have been made to extend the functionality of Flavor (MSDL) to support the DIA framework. Such efforts result in the extensions of Flavor: *XFlavor* [25, 26] and *BFlavor* [27-29].

XFlavor was introduced in 2002 [25]. The language was not targeted for the MPEG-21 DIA framework. Although developers were aware of the standardization of MPEG-21 DIA, which was ongoing at that time, the language was aimed for broader target applications where

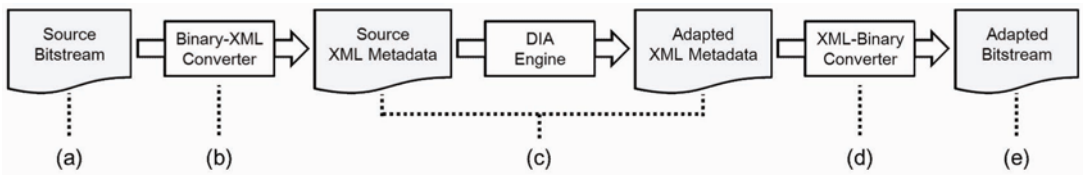

Fig. 7. DIA Framework and BSPD Language.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema
  targetNamespace="urn:mpeg:mpegb:example:AVC"
  xmlns:bs1="urn:mpeg:mpeg21:2003:01-DIA-BSDL1-NS"
  xmlns:bs2="urn:mpeg:mpeg21:2003:01-DIA-BSDL2-NS"
  xmlns:avc="urn:mpeg:mpegb:example:AVC"
  xmlns:xs="http://www.w3.org/2001/XMLSchema"
  elementFormDefault="qualified"
  bs2:removeEmPrevByte="000003 0000"
  bs2:rootElement="avc:Bitstream"
  bs2:bsdlVersion="ISO/IEC 23001-5">

  <xs:import namespace="urn:mpeg:mpeg21:2003:01-DIA-BSDL1-NS"
             schemaLocation="../Schemas/MPEG-B-BSDL-1.xsd"/>

  <xs:element name="Bitstream">
    <xs:complexType>
      <xs:sequence>
        <xs:choice maxOccurs="unbounded">
          <xs:element ref="avc:seqParameterSet"/>
          <xs:element ref="avc:pictParameterSet"/>
          <xs:element ref="avc:sliceIDR"/>
          <xs:element ref="avc:sliceNonIDR"/>
          <xs:element ref="avc:otherNAL"/>
        </xs:choice>
      </xs:sequence>
      <xs:attribute ref="bs1:bitstreamURI"/>
      <xs:attribute ref="bs1:bsdlVersion"/>
    </xs:complexType>
  </xs:element>

  <xs:element name="seqParameterSet"
        bs2:ifNext="0000000107" bs2:ifNextMask="FFFFFFFF1F"
        type="avc:NALUnitType"/>

  ...

</xs:schema>
```

Fig. 8. Example of the BSDL schema document.

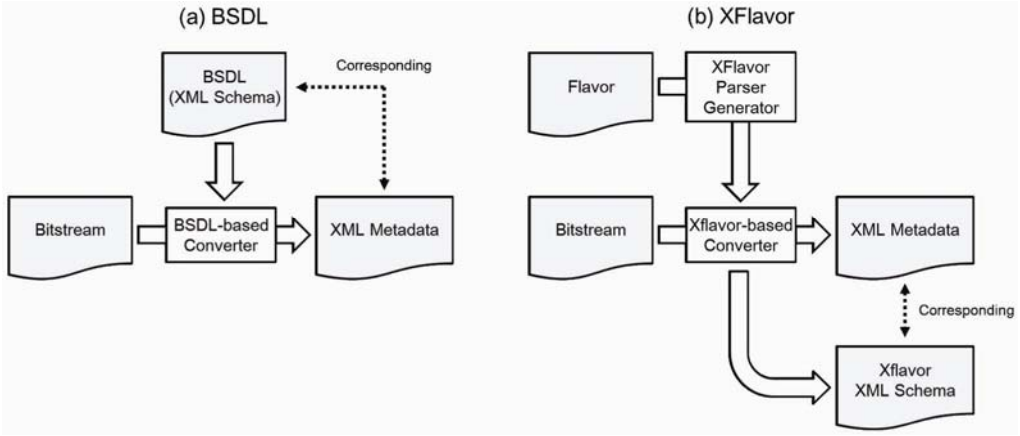

Fig. 9. Comparison of the bitstream processing chain of BSDL and Xflavor.

XML representation of a multimedia bitstream is in demand. The language, Flavor, itself was not changed. The extension was made mainly for the parser generation software for Flavor.

Fig. 9 compares the bitstream processing chain of BSDL and XFlavor. XFlavor-based XML metadata generator is composed automatically by the XFlavor parser generation software. This makes the XML converter dedicated to a specific format of bitstream. Each design has pros and cons. The BSDL-based chain is simpler and the BSD generation and validation of the XML document can be performed using the same BSDL document. On the other hand, XFlavor-based design requires more steps for the same process but provides a faster converter.

BFlavor is an approach to harmonize the merits of XFlavor and BSDL [27-29]. BFlavor combines the processing chain of two languages. The bitstream parsing and XML generation is performed using Flavor-based software. On the other hand, the adaptation process and bitstream generation is done using BSDL software. For handshaking between the two language systems, XFlavor is extended and modified to generate BSDL-compliant XML metadata and XML schema written in BSDL.

The advantage of BFlavor against BSDL is the fast bitstream parsing speed. In reference [29], the performance of BFlavor-based parser was evaluated against the BSDL-based parser. According to the simulation results described in Table 1 of [29], BFlavor-based DIA software has a significant advantage on the performance in the execution time and memory consumption. One of the most notable

results was the simulation with a bitstream containing a 300 second-long video coded in MPEG-4 AVC/H.264. The BFlavor parser generated an XML metadata document in 9.4 seconds, whereas the BSDL parser took more than one day for the same process.

3.3. Reconfigurable Media Coding

The continuous development in multimedia compression technology has resulted in an increasing number of coexisting codecs and an increasing complexity of each codec. One of the responses to such a technical trend was the MPEG-4 system standard. The MPEG-4 was one of the first media coding standards based on an object-oriented concept by considering a codec as a set of modularized coding tools. The MPEG experts attempted to describe the relationship between the tools, i.e., the structure of a codec, through programmable languages in MPEG-4 Systems. In the Systems requirements [16], several system description languages were suggested to describe the architecture, algorithm, and bitstream syntax of the new codec. Nevertheless, the programmability of codec was not actually realized in the MPEG-4 standard except for the MSDL.

MPEG Reconfigurable Media Coding (RMC) framework, which is also known as Reconfigurable Video Coding (RVC) framework, is a recent technical approach to achieve a similar goal. Fig. 10 presents the overall concept of the MPEG RMC framework. RMC is a framework to define a standard method for describing the structure of a codec in a modular way. The RMC standard specifies a number of modularized coding tools, such as inverse transform or de-blocking filter, as *functional unit* (FU). A codec can be composed by connecting the FUs within a processing network based on a dataflow model.

The MPEG RMC framework is specified over two MPEG standards: MPEG-B Part 4 Codec Configuration Representation (CCR) [6] and MPEG-C Part 4 Media Tool Library (MTL) [40]. CCR defines the overall functionality of the RMC framework and specifies normative languages to describe the structure of a codec. MTL defines a set of FUs by specifying the internal algorithm and input/output behavior of each FU.

To provide programmability for the bitstream parser FU, the RMC framework requires a BSPD language. Although the programmability of the decoding process can be achieved by a flexible combination of FUs, the programmability of the bitstream syntax can only be assured by changing the internal behavior of the bitstream parser FU. Therefore, the bitstream parser FU is allowed to be generated from a given script written in BSPD language, whereas the internal algorithms of the other FUs are fixed in the MTL standard.

Bitstream parser generation for the RMC framework can occur in two ways. Figs. 11(a) and (b) shows the

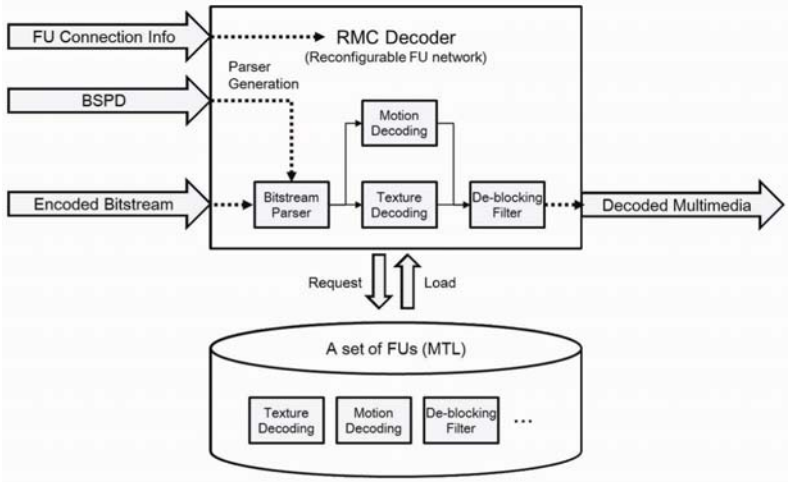

Fig. 10. Concept of the MPEG RMC framework.

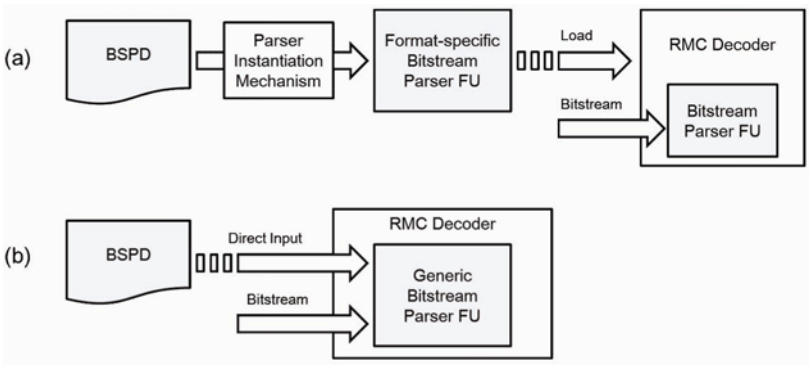

Fig. 11. Parser instantiation method (a) and generic parser method (b).

parser instantiation method and *generic parser* method, respectively. The parser instantiation method suggests that an executable and format-specific bitstream parser FU is derived from a given BSPD. The generic parser method indicates that there is a generic bitstream parser FU that receives a BSPD as an input to configure the parsing process in run time.

Considering its technical usage, the BSPD language for the RMC framework should be equipped with the following functionalities:

- The language should be able to describe not only the bitstream syntax, but also the semantic decoding process of the bitstream.
- The language should be designed to build a bitstream parser. In particular, the language should support at least one of either the parser instantiation method or generic parser method.
- The language should be able to describe how to distribute the data read from the bitstream to the other FUs composing the rest of a codec.

In chronological order, three BSPD languages were developed for the RMC framework: *Compact Decoder Description Language* (CDDL), *BSDL for RVC* (RVC-BSDL), and *XML Script-based Parser Description Language* (XS-PDL).

First, CDDL is a language that has been proposed to the MPEG RMC standardization activity in 2005 [41]. The language is designed to realize a profile-free codec design along with the RMC framework [42]. The language was proposed to MPEG as one of the responses to the call for proposal for RMC standardization [43]. The language is a script language that consists of several "tables" written in different grammars. Four tables are used to describe the bitstream parsing process: (1) *Syntax Element Table* (SET), (2) *Syntax Rule Table* (SRT), (3) *Control Signal and Context Information Table* (CSCIT), and (4) *Default Value Table* (DVT). Each table is written in a suitable form of script for the representation of each type of data.

Fig. 12 gives an example of a script written in CDDL. The bitstream syntax is described in CDDL using a *Finite State Machine* (FSM) [41]. The bitstream syntax elements are described as FSM states in the SET and the transition rules among the syntax elements are described in the SRT. CSCIT is used to declare the variables during the bitstream parsing process. DVT is used to annotate the entropy decoding parameters (e.g., Huffman table). CDDL supports entropy decoding through a predefined function for variable length decoding.

The design of CDDL is aimed to use a generic parser method. The CDDL is a procedural language like other programming languages rather than a description language. The description itself is designed to be executable by an interpreter in run time. In the RMC framework, a generic bitstream parser FU equipped with a CDDL interpreter can process a specific bitstream in run time by receiving a corresponding CDDL description as an input. The language can also be fit with the parser instantiation method, even though the possibility has not been tested.

CDDL has several advantages. The key strength of CDDL is the compact description scheme. The language is

```
SET
{
        [Read]                  ("READ P1 > P2;"),
        [Read with BITSIZE]     ("(V1=P1);(V2=0);
                                  WHILE(V1!=0){ (V1=V1/2); (V2++); }
                                  READ V2 > P2;
                                "),
        [Marker Bit]            ("READ 1;"),
        [Read, isSame]          ("READ 32 B > V10; (P1=(V10==P2));"),
        [Is user data ot not]   ("SEEK 32 B > P1;"),

        [VOS end code]          ("READ 32 B > V10;(P1=((V10==433) || (EOF)));"),
        [User data start code]  ("SEEK 32 B > V10;(P1=(V10==434));"),
        [User data]             ("READ 32 B > V10; SEEK 24;
                                  WHILE (V10 != 1) {READ 8 > V10;PUSH P1 V10;SEEK 24;}"),
        [VO start code]         ("READ 32 B > V10;(P1=(V10==437));"),
        [Video start code]      ("READ 32 B > V10;(P1=((V10 >= 256) && (V10 <= 287)));"),

         ...
}


SRT {
("
        R1();
        (C67=(C24*C25/256));

        WHILE(1 == 1) {
                IF(C63==1) {
                        R3();
                }
                ELSE IF(C63==2) {
                }
                ELSE IF((C63==3) || (C63==4)) {
                        END;
                }
                ELSE {
                        ERR;
                }
        ...
}
```

Fig. 12. Example of the CDDL description.

dedicated for use in the RMC framework. Accordingly, the description scheme is strictly constrained in a small size of the written description. Moreover, the BSPD description written in CDDL can be translated to a binary form to achieve further compactness. The directly executable grammar of CDDL is preserved in binary form. The preserved structure allows a faster interpretation of CDDL than the script format in the generic bitstream parser FU.

Another remarkable strength of CDDL is that it has a wide scope of description. CDDL is designed to be both a BSPD language and a description language for the entire codec. The execution order of the FUs (i.e., the decoding process) can be also described in CDDL using tables. The decoding process is actually driven by the bitstream parsing process in CDDL. This feature is derived from the early design of the RMC framework, which has a highly centralized structure with a central processor called the *Global Control Unit* [44]. The CDDL has a very strong ability of description compared to the other BSPD languages.

On the other hand, those strengths of CDDL can turn into weaknesses at the same time. First, to obtain compactness, the grammar of CDDL becomes a quite abstract and unique syntax. This is less compatible with previously available languages and description methods, even though the grammar is not difficult to learn. Second, the parser-driven description scheme for the decoding process of CDDL inevitably restricts the parallelism in the decoder implementation. Although it is not a significant problem as a BSPD language, it has become an obstacle to its acceptance in the standard. Currently, the development of CDDL is discontinued. On the other hand, some features of the language, such as FSM-based description of bitstream syntax, are inherited to a recently developed BSPD language, XS-PDL.

Next, RVC-BSDL is a relatively newer language to the MPEG RMC standardization than CDDL. The language is the current normative language specified in the MPEG-B Part 4 standard [6]. Fig. 13 gives an example of the RVC-BSDL description. The language is a derivative of BSDL, which was previously introduced for the DIA framework. The initial idea of RVC-BSDL emerged from the fact that the bitstream processing chain of BSDL, which was shown in Fig. 9, has some similarities to the bitstream parsing functionality required by the RMC framework. If a BSDL-based parser can be modified to interact with other FUs, it can work as a bitstream parser FU within the RMC framework. In addition, it was the latest stable MPEG standard to describe the bitstream specification at that time.

RVC-BSDL is proposed as a BSPD language that supports the RMC framework by the parser instantiation method [45, 46]. In reference [46], the language was claimed to be automatically translated to an executable source code written in CAL Actor Language (CAL), which is the language used to specify the internal functions of the FUs in the RMC framework. The proposed translation process is run using XML Stylesheet Language Transform (XSLT) technology [47]. The instantiated parser FU does not contain entropy decoding functions. Instead, the entropy-coded bitstream syntax elements are parsed by the separated entropy decoding FUs, which are pre-defined in the MTL standard and connected to the bitstream parser FU.

For that purpose, from the conventional BSDL, two extensions to support the RMC framework are added. A new XML datatype is added to describe the entropy coded bitstream syntax elements and a new attribute is added for the XML element declaration to describe the interaction between the to-be-generated bitstream parser FU and the other FUs. On the other hand, several restrictions to the

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema
  xmlns:m4v="MPEG-4_Visual"
  xmlns="MPEG-4_Visual"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema"
  xmlns:bsl="urn:mpeg:mpeg21:2003:01-DIA-BSDL1-NS"
   xmlns:bs2="urn:mpeg:mpeg21:2003:01-DIA-BSDL2-NS"
  xmlns:rvc="urn:mpeg:2006:01-RVC-NS"
  targetNamespace="MPEG-4_Visual"
   elementFormDefault="qualified"
  bs2:rootElement="m4v:Bitstream">

  ...

  <xsd:group name="bitstream">
    <xsd:sequence>
      <xsd:sequence>
        <xsd:element name="bitstreamObject" minOccurs="0" maxOccurs="unbounded"
         bs2:startContext="'pu'" bs2:stopContext="'old_pu'"
         bs2:redefineMarker="'tmp_pu' 'old_pu' 'pu' 'tmp_pu'">
      </xsd:sequence>
      <xsd:choice>
        <xsd:element ref="m4v:visual_object_sequence" minOccurs="0" bs2:ifNext="&visObjSeqSC;"/>
        <xsd:element name="VO" type="VideoObjectType" minOccurs="0" bs2:ifNext="&voSC;"/>
        <xsd:element name="visual_object_sequence_end_code" type="m4v:StartCodeType"
         minOccurs="0" bs2:ifNext="&visObjSeqEC;"/>
        <xsd:element name="GoVOP" type="GroupOfVideoObjectPlaneType" bs2:ifNext="&goVopSC;"/>
        <xsd:element name="VOP" type="VideoObjectPlaneType" bs2:ifNext="&vopSC;"/>
      </xsd:choice>
    </xsd:sequence>
  </xsd:group>

  ...

</xsd:schema>
```

Fig. 13. Example of RVC-BSDL Description.

BSDL grammar are enforced to simplify the translation process from RVC-BSDL to CAL. The complete list of the restrictions can be found in the Annex C of the MPEG-B Part 4 standard document [6]. The remainder of the description scheme is not changed from the original BSDL.

The merit of using RVC-BSDL is its compatibility with the conventional technologies. The derivation from BSDL was preferred over having a new language (e.g., CDDL) because BSDL is a stable technology specified by an international standard. Moreover, the XML-based format of RVC-BSDL is helpful for integrating the bitstream syntax description with other metadata [48].

On the other hand, there are some flaws in RVC-BSDL. Despite the promising proposals and articles [45, 46, 48, 49], the research regarding the process to automatically generate a bitstream parser FU from a given RVC-BSDL description is still ongoing and the result is currently unavailable in the standard. Another problem is that the structure of RVC-BSDL is unstable as an XML-based language. An XML-based language should have its own XML schema to guide the users of the language. On the other hand, one study [50] reported that the textual specification of RVC-BSDL described in the MPEG-B Part 4 standard document is in a form that is difficult to describe using the XML schema. This problem draws a veil over the merit of RVC-BSDL as an XML-based language.

Finally, XS-PDL is the most recently developed BSPD language in MPEG. The language was developed at Hanyang University to provide the bitstream parser generation method for the RMC framework using a proprietary BSPD language. Fig. 14 gives an example description written in XS-PDL [19]. The design principle of XS-PDL is to combine the advantages of previously developed languages, CDDL and RVC-BSDL [19]. The language defines the bitstream syntax using an FSM concept and describes the FSM in XML format.

For the bitstream parser generation in the RMC framework, XS-PDL was designed to support the generic parser method. A generic bitstream parser FU for XS-PDL was implemented, and its parsing performance was tested in reference [19].The test result against the MPEG-4 Visual reference software showed that the bitstream parsing speed of the XS-PDL generic bitstream parser is 85.3% on average. This can be considered a reasonable speed for practical use in run time.

A key strength of XS-PDL is its flexible description ability. For example, an XS-PDL description might contain references to the plug-in functions performing complex bitstream parsing algorithms (e.g., entropy decoding). Such an extensible scheme allows an arbitrary bitstream parsing tool to be integrated within a bitstream specification written in XS-PDL without having a lengthy algorithmic description. In addition, the XML-based grammar can be integrated easily with other XML metadata formats. Currently, studies to make the language compatible with RVC-BSDL are ongoing in MPEG [19, 51].

4. Summary and Future Visions

Table 1 presents a functional comparison of the BSPD languages reviewed in this paper. The comparison shows that the currently available BSPD languages are complementary to one another. In addition, the functionalities of the BSPD languages have a tendency to be dependent on its target application, i.e., its corresponding MPEG standard.

For example, the BSPD languages designed for the DIA framework can be used in both the encoder-side and decoder-side at the same time, whereas the languages for

```
<?xml version="1.0" encoding="utf-8"?>
<BSD name="MPEG4SP" xsi:noNamespaceSchemaLocation="xspdl.xsd"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <control>
    <FSM name="VS">
      <states>
        <!-- Start code detection -->
        <FSM_call name="start_code.s1" callee="next_start_code" />

        <branch condition="start_code==0x1B0" next="VOS_header" /> <!-- VOS -->
        <branch condition="0x11F >= start_code and start_code >= 0x100" next="VO" />
        <branch condition="start_code==0x1B1" next="END" /> <!-- VOS end code -->
        <FSM_call callee="fatal_error" />

        <!-- VOS start code -->
        <parsing name="VOS_header" />
        <parsing name="profile_and_level_indication" bits="8" />
        <FSM_call callee="USER_DATA" />

        <!-- Visual Object-->
        <FSM_call name="VO" callee="VO" />

        <parsing bits="0" bytealign="true"/>
        <plugin name="isEOF" callee="EOFPlugin" />
        <branch condition="isEOF==0" next="start_code.s1" />

        <parsing name="END" final="true" />
      </states>
    </FSM>

  ...

</BSD>
```

Fig. 14. An example of XS-PDL Description.

Table 1. Functional comparison of the BSPD languages.

Language	Developed for	Functionality						
		Bitstream Syntax Description			Parser Generation Method		Encoder Support	XML-based
		Syntax Elements & Control Flow	Entropy Coding	Extensible Parsing Algorithm	Parser Instantiation	Generic Parser		
MSDL (Flavor)	Generic	✓	(1)	(3)	✓		✓	
BSDL	DIA	✓		(3)		✓	✓	✓
XFlavor	DIA	✓	(1)	(3)	✓		✓	
BFlavor	DIA	✓	(1)	(3)	✓		✓	
CDDL	RMC	✓	(1)			✓		
RVC-BSDL	RMC	✓	(2)	(2)	(4)			✓
XS-PDL	RMC	✓	(2)	(2)		✓		✓

(1) Limited types of entropy coding algorithms are natively supported
(2) Processed by external FUs or functions
(3) By attaching source codes written in C++, Java, or ECMAScript
(4) Investigation is still ongoing

the RMC framework only supports the decoder side. In addition, the DIA-based languages support the description of entropy coded bitstream syntax elements in a limited manner, whereas the RMC-based languages provide extensible support. Such differences in functionality reflect the characteristics of the corresponding standard frameworks, showing a strong relationship between the languages and applications.

In such a technical environment, relatively little interest has been given to improving the technical maturity of BSPD language as an independent technology. On the other hand, as discussed in this paper, the BSPD language is a generic technology that can be applied to any application that requires bitstream parsing (and/or generation) functionality rather than an integral part of a specific standard framework. Future research on the following research topics is expected to result in the functional or performance improvement of BSPD languages:

- ***Performance evaluation methodology for BSPD languages***: Although a qualitative assessment [19, 48] or performance analysis [19, 29] have been attempted in some studies, there is no stable and standardized performance evaluation methodology generally applicable to any BSPD language. A stable test condition (e.g., standardized bitstream input) should be designed to allow an objective analysis and easy comparison of the different BSPD languages.
- ***In-depth analysis on the parser generation methodology***: As described in Section 3.3, there are two major parser generation methods: (1) parser instantiation method and (2) generic parser method. On the other hand, the precise technical trait of each parser generation method has not been investigated experimentally. Therefore, further research is needed to understand the pros and cons of each parser generation method more clearly. One of the potential challenges is the development of a BSPD language that supports both parser generation methods because none of the languages has been proven to support both.
- ***Translation between BSPD languages***: Different BSPD languages share a common set of information, as discussed in Section 2. This commonality suggests that it would be possible to develop a translation process between two different BSPD languages, which might allow many BSPD languages to be used together within a single application. For example, there is an ongoing attempt to establish a translation process between RVC-BSDL and XS-PDL to stabilize the bitstream parser generation process in the RMC framework [51].

5. Conclusion

The BSPD language is a tool for describing a bitstream specification in not only a standardized manner but also in a computer-readable manner. The importance of the BSPD language is ever increasing with the continuously developing multimedia codec standards. Moreover, the BSPD language is a necessary technology for media processing frameworks that require automated bitstream parsing functionality, such as the DIA framework or the RMC framework. In particular, the MPEG RMC is a standard framework, which has strong potential for the basis of the future codec standards [52], may result in an increased demand for the BSPD technology.

On the other hand, there are technical challenges still to be overcome in BSPD technology. Despite the significance of the technology, there has been relatively little interest in research relevant to BSPD languages. As discussed in Section 4, the languages have been developed only as a part of the specific standard framework, and there has been little consideration in improving the functionalities provided by BSPD languages. BSPD languages have strong potential as an independent technology and can be enhanced further by future research.

References

[1] *Information technology—Generic coding of moving pictures and associated audio information—Part 2: Video*, ISO/IEC 13818-2, 2000. Article (CrossRef Link)

[2] *Information technology—Generic coding of moving pictures and associated audio information—Part 3: Audio*, ISO/IEC 13818-3, 1998. Article (CrossRef Link)

[3] *Information technology—Digital compression and coding of continuous-tone still images—Part 1: Requirements and guidelines*, ISO/IEC 10918-1, 1994. Article (CrossRef Link)

[4] *Information technology—Coding of audio-visual objects—Part 1: Systems*, ISO/IEC 14496-1, 2010. Article (CrossRef Link)

[5] *Information technology—Multimedia framework (MPEG-21)—Part 7: Digital Item Adaptation*, ISO/IEC 21000-7, 2007. Article (CrossRef Link)

[6] *Information technology—MPEG systems technologies—Part 4: Codec configuration representation*, ISO/IEC 23001-4, 2011. Article (CrossRef Link)

[7] The Turing Archive for the History of Computing, *Turing's Automatic Computing Engine* [Online]. Article (CrossRef Link)

[8] *Information technology—JPEG 2000 image coding system: Core coding system*, ISO/IEC 15444-1, 2004. Article (CrossRef Link)

[9] *Information technology—Coded representation of picture and audio information—Progressive bi-level image compression*, ISO/IEC 11544, 1993. Article (CrossRef Link)

[10] *Codecs for videoconferencing using primary digital group transmission*, ITU-T H.120, 1993. Article (CrossRef Link)

[11] *Video codec for audiovisual services at p x 64 kbit/s*, ITU-T H.261, 1993. Article (CrossRef Link)

[12] *Information technology—Coding of moving pictures and associated audio for digital storage media at up to about 1,5 Mbit/s—Part 2: Video*, ISO/IEC 11172-2. 1993. Article (CrossRef Link)

[13] *Information technology—Coding of audio-visual objects—Part 2: Visual*, ISO/IEC 14496-2, 2004. Article (CrossRef Link)

[14] *Information technology—Coding of audio-visual objects—Part 10: Advanced Video Coding*, ISO/IEC 14496-10, 2012. Article (CrossRef Link)

[15] *Information technology—High efficiency coding and media delivery in heterogeneous environments—Part 2: High efficiency video coding*, ISO/IEC FDIS 23008-2, 2012. Article (CrossRef Link)

[16] B. G. Haskell, A. Puri, and A. N. Netravali, "MPEG-4 and the Future," in *Digital video: an introduction to MPEG-2*, Springer, 1997, pp. 369-411. Article (CrossRef Link)

[17] A. Eleftheriadis, "The MPEG-4 system and description languages: from practice to theory," In *Proceedings of 1997 Int'l Symposium on Circuits and Systems*, Hong Kong, Jun. 1997, pp. 1480-1483. Article (CrossRef Link)

[18] Multimedia Signal Processing Laboratory, Columbia University, NY (1994). *Mkvlc software* [Online]. Available: Article (CrossRef Link)

[19] H. Kim, S. Kim, S. Lee, and E. S. Jang, "Parser Description-based Bitstream Parser Generation for MPEG RMC Framework," *Signal Processing: Image Communication*, Available online, Sep. 2013. Article (CrossRef Link)

[20] A. Eleftheriadis, "Flavor: a language for media representation," in *Proceedings of the fifth ACM international conference on Multimedia*, ACM, Nov. 1997, pp. 1-9. Article (CrossRef Link)

[21] Y. Fang and A. Eleftheriadis. "A syntactic framework for bitstream-level representation of audio-visual objects," in *Proceedings., International Conference on Image Processing*, Lausanne, Sep. 1996, vol. 2, pp. 421-424. Article (CrossRef Link)

[22] A. Eleftheriadis and D. Hong, "Flavor: a formal language for audio-visual object representation," in *Proceedings of the 12th annual ACM international conference on Multimedia*, ACM, Oct. 2004, pp. 816-819. Article (CrossRef Link)

[23] Y. Fang and A. Eleftheriadis, "Automatic generation of entropy coding programs using Flavor," in *1998 IEEE Second Workshop on Multimedia Signal Processing*, Redondo Beach, CA, Dec. 1998. Article (CrossRef Link)

[24] D. Hong and A. Eleftheriadis, "Automatic Generation of C++/Java Code for Binary Arithmetic Coding," in *Proceedings of Picture Coding Symposium*, San Francisco, CA, USA, Dec. 2004. Article (CrossRef Link)

[25] D. Hong and A. Eleftheriadis, "XFlavor: bridging bits and objects in media representation," in *Proceedings. 2002 IEEE International Conference on Multimedia and Expo*, Aug. 2002, vol. 1, pp. 773-776. Article (CrossRef Link)

[26] D. Hong and A. Eleftheriadis, "XFlavor: providing XML features in media representation," *Multimedia Tools and Applications*, vol. 39, no. 1, pp. 101-116, Aug. 2008. Article (CrossRef Link)

[27] D. Van Deursen and R. Van de Walle, "BFlavor: a new bitstream structure description language," in *Sixth FirW PhD Symposium*, Nov. 2005. Article (CrossRef Link)

[28] D. Van Deursen and R. Van de Walle, "A real-time XML-based adaptation system for scalable video formats," *Advances in Multimedia Information Processing-PCM 2006, LNCS*, vol. 4261, pp. 339-348, Nov. 2006. Article (CrossRef Link)

[29] W. De Neve, D. Van Deursen, D. De Schrijver, S. Lerouge, K. De Wolf, and R. Van de Walle, "BFlavor: A harmonized approach to media resource adaptation, inspired by MPEG-21 BSDL and XFlavor," *Signal Processing: Image Communication*, vol. 21, no. 10, pp. 862-889, Nov. 2006. Article (CrossRef Link)

[30] *Information technology—Coding of audio-visual objects—Part 12: ISO base media file format*, ISO/IEC 14496-12, 2012. Article (CrossRef Link)

[31] *Information technology—Coding of audio-visual objects—Part 14: MP4 file format*, ISO/IEC 14496-14, 2003. Article (CrossRef Link)
[32] *Information technology—Coding of audio-visual objects—Part 16: Animation Framework eXtension (*AFX*)*, ISO/IEC 14496-16, 2011. Article (CrossRef Link)
[33] *W3C XML Schema Definition Language (XSD) 1.1*, W3C Recommendation, 2012. Article (CrossRef Link)
[34] *Information technology—MPEG systems technologies—Part 5: Bitstream Syntax Description Language (BSDL)*, ISO/IEC 23001-5, 2008. Article (CrossRef Link)
[35] *XML Path Language (XPath) 2.0 (Second Edition)*, W3C Recommendation, 2011. Article (CrossRef Link)
[36] D. De Schrijver, W. De Neve, K. De Wolf, R. De Sutter, and R. Van de Walle, "An optimized MPEG-21 BSDL framework for the adaptation of scalable bitstreams," *Journal of Visual Communication and Image Representation*, vol. 18, no. 3, pp. 217-239, Jun. 2007. Article (CrossRef Link)
[37] W. De Neve, S. Lerouge, P. Lambert, and R. Van de Walle, "A performance evaluation of MPEG-21 BSDL in the context of H. 264/AVC", in *Optical Science and Technology, the SPIE 49th Annual Meeting*, pp. 555-566, Nov. 2004. Article (CrossRef Link)
[38] P. Lambert, W. De Neve, D. De Schrijver, Y. Dhondt, and R. Van de Walle, "Using placeholder slices and MPEG-21 BSDL for ROI extraction in H. 264/AVC FMO-encoded bitstreams," in *E-Business and Telecommunication Networks*, vol. 9, pp. 173-185, 2008. Article (CrossRef Link)
[39] D. De Schrijver, W. De Neve, K. De Wolf, and R. Van De Walle, "Generating MPEG-21 BSDL descriptions using context-related attributes," in *Seventh IEEE International Symposium on Multimedia*, Dec. 2005. Article (CrossRef Link)
[40] *Information technology—MPEG video technologies—Part 4: Video tool library*, ISO/IEC 23002-4, 2010. Article (CrossRef Link)
[41] H. Kim, S. Lee, C. Yie, and E. S. Jang, "Reconfigurable syntax parsing FU design for VCTR," ISO/IEC JTC1/SC29/WG11 M12851, Jan. 2006.
[42] H. Ahn, S. Lee, Y. Cho, B. Min, H. Kim, S. Lee, C. Yie, and E. S. Jang, "VCTR demo scenario on bitstream reconfigurable coding and profile free coding," ISO/IEC JTC1/SC29/WG11 M12557, Oct. 2005.
[43] F. Pereira, "Submissions to the Call for Proposals on Reconfigurable Video Coding (RVC)," ISO/IEC JTC1/SC29/WG11 M13721, Jul. 2006.
[44] "WD 1.0 of Reconfigurable Video Coding (RVC)," ISO/IEC JTC1/SC29/WG11 N8260, Jul. 2006.
[45] J. Thomas-Kerr, "Report on CE on RMC Bitstream Syntax Description," ISO/IEC JTC1/SC29/WG11 M14747, Jun. 2007.
[46] C. Lucarz, J. Thomas-Kerr, M. Mattavelli, and I. Burnett, "A systematic procedure for the generation of a CAL parser from BDSL in the RVC framework - result CE 1.1," ISO/IEC JTC1/SC29/WG11 M14874, Oct. 2007
[47] *XSL Transformations (XSLT) Version 1.0*, W3C Recommendation, 1999. Article (CrossRef Link)
[48] M. Raulet, J. Piat, C. Lucarz, and M. Mattavelli, "Validation of bitstream syntax and synthesis of parsers in the MPEG Reconfigurable Video Coding framework," in *IEEE Workshop on Signal Processing Systems*, pp. 293-298, Oct. 2008. Article (CrossRef Link)
[49] C. Lucarz, J. Piat, and M. Mattavelli, "Automatic Synthesis of Parsers and Validation of Bitstreams Within the MPEG Reconfigurable Video Coding Framework," *Journal of Signal Processing Systems*, vol. 63, no. 2, pp. 215-225, May 2011. Article (CrossRef Link)
[50] H. Kim, S. Lee, J. S. Choi, B. Koo, and E. S. Jang, "On the conformance issues of MPEG-B RVC-BSDL," ISO/IEC JTC1/SC29/WG11 M29257, Apr. 2013.
[51] "Core Experiments in RVC," ISO/IEC JTC1/SC29/WG11 N13568, Apr. 2013.
[52] G. Fernando and R. Glidden, "Proposal to Use RVC in Type-1 (Royalty-Free) MPEG Specification Process," ISO/IEC JTC1/SC29/WG11 M17548, Apr. 2010.

Hyungyu Kim received his M.S. degree at Hanyang University, Seoul, Republic of Korea, in 2007. He is currently a Ph. D. candidate in the Department of Electronics and Computer Engineering at Hanyang University. He is interested in the MPEG Reconfigurable Media Coding framework: his main research topics are the bitstream syntax description and the bitstream parser generation methodology. He has received a Certificates of Appreciation from International Organization for Standardization (ISO) and International Electrotechnical Commission (IEC) in 2011 for his contribution to the development of ISO/IEC 23001-4:2009 international standard.

Euee S. Jang received his B.S. from Jeonbuk National University, Korea and a Ph. D. from SUNY at Buffalo, NY, USA. He is a Professor in the Division of CSE, Hanyang University, Seoul, Korea. His research interests include image/video coding, reconfigurable media coding, and computer graphics objects. He has authored more than 150 MPEG contribution papers, more than 30 journal or conference papers, 35 pending or accepted patents, and two book chapters. Dr. Jang has received three ISO/IEC Certificates of Appreciation for contributions to MPEG-4 development. He received the Presidential Award from the Korean Government for his contribution to MPEG standardization.