Hindawi Publishing Corporation
Journal of Electrical and Computer Engineering
Volume 2010, Article ID 509394, 7 pages
doi:10.1155/2010/509394

Research Article

Modeling DV/DVCPRO Standards on Reconfigurable Video Coding Framework

Jianjun Li and Esam Abdel-Raheem

Department of Electrical and Computer Engineering, University of Windsor, Windsor, ON, Canada N9B 3P4

Correspondence should be addressed to Esam Abdel-Raheem, eraheem@uwindsor.ca

Received 6 December 2009; Accepted 29 March 2010

Academic Editor: Sos S. Agaian

After more than 20 years, several video coding standards and technologies have been delivered. Less consideration is taken on their commonalities and interoperations. Specification and reference code of case by case is time consuming. The MPEG reconfigurable video coding (RVC) framework is a new standard under development by MPEG. It aims to provide a unified high-level specification of current MPEG video coding technologies. In this framework, the decoder is built as a configuration of video coding tools taken from MPEG toolbox library. Up to now, MPEG-4 simple profile and China audio video coding standard (AVS) decoders have been successfully modeled with RVC framework. In this paper, we examine another video standard, that is, DV/DVCPRO, and model it with RVC-CAL. The flexibility and ease of RVC-CAL is demonstrated as well as the validation of RVC modeling.

1. Introduction

Video coding solutions based on a predefined video coding standard have certain limitations when new standards are being added. How to utilize their commonalities and reduce design time is of great concern. Writing reference code for a new standard starts with scratch, which is time consuming and labor intensive. On the other hand, sequential C/C++ code aiming at functional validation hides intrinsic concurrency and parallelism. Consequently, converting the sequential code into pipeline and multicore process requires architecture rebuilding and rewriting code. In other words, the complex C/C++ specifications no longer constitute a good starting point for modeling the standards. It is preferred to develop a framework to operate at a higher level of abstraction and simplify top-down system development and design. To deal with this issue, MPEG organization launched a new standard called reconfigurable video coding (RVC) in 2006 [1]. Some video coding standards have been successfully modeled with RVC framework [2–4]. In 2007, the work in [2] initiatively modeled the MPEG-4 simple profile with the RVC framework. The work in [3] reported modeling of AVS intra decoder with RVC framework. The work in [4] recently reported that an efficient H.264/AVC baseline encoder had been built by the RVC-CAL dataflow components.

In this paper, we model the other video coding standard, DV/DVCPRO, using the RVC framework. The remaining of the paper is organized as follows: the MPEG RVC framework and DV/DVCPRO standard are introduced in Section 2. The proposed design with RVC framework is presented in Section 3. Section 4 provides the experimental results and analysis. Section 5 concludes this paper.

2. Overview of RVC and DV/DVCPRO Standard

2.1. Reconfigurable Video Coding (RVC). The objective of RVC framework is to describe current and future codecs in a way that makes commonality explicit and reduces the implementation burden for device vendors [2]. The key difference between RVC and conventional codec standards is their conformity point. The conventional codec standards define their conformity point at decoder level whereas MPEG RVC defines it at tool level so that MPEG RVC exhibits much more flexibility. Hence several configurations of components, taken by previous monolithic specifications, are

possible [5]. MPEG RVC framework is initiatively motivated by the following observations.

(1) Supporting multiple standards: video coding standards have been changed for decades. New multimedia devices or development platforms need to support multiple codecs, such as MPEG-1, MPEG-2, MPEG-4, H.264/AVC, and DV/DVCPRO.

(2) Interoperability: commonality and similarity between the standards have not been utilized efficiently. It is urgent to develop a new standard to incorporate the commonality and similarity in order to reduce design time.

(3) Obstacles of current specification are the follows.

(i) The normative specifications (written in generic C/C++) do not expose the potential parallelism which is intrinsic to the algorithms constituted in the codecs. They are excessively large and hard to read.

(ii) The reference code written in complex sequential C/C++ is labor intensive to transform to Verilog or the new generation codes of multicore platform stream processor.

Thus, the goal of the MPEG RVC standard is to offer a high-level algorithm model to innovate MPEG standards in a way that is competitive in current dynamic environment, thereby enabling MPEG to continue serving the needs of the industry in terms of video coding standards. An additional challenge taken by MPEG RVC is to provide an easy-going model for efficient hardware and software synthesis. The following three components are mainly included in RVC framework.

(i) Video coding tools library (VTL): the normative library is specified by textual specification and corresponding reference software, written with RVC-CAL language [6] to specify each library component.

(ii) Function unit (FU) network language (FNL): the normative language is extensible markup language (XML) dialect. It specifies decoder configuration and interconnected network and parameterization of standard library components.

(iii) RVC bitstream description language (RVC-BSDL): the normative language describes the syntax of a new configuration of an MPEG RVC decoder.

Unlike other video coding standards, MPEG RVC decodes the configuration information before the video bitstream. To decode a video bitstream, the decoder needs to know (a) how to parse the bitstream and (b) how to decode these elements. The MPEG RVC decoding engine receives RVC-BSDL and FNL specifications in compressed form. The decoder composition module generates a decoding solution (an actual video decoder) based on the RVC-BSDL and FNL specifications. It makes use of selected FUs from VTL and connects them according to FNL specification. Once the decoding solution has been generated, it can then decode the bitstream. This approach has a number of potential benefits. A decoder can be modified to decode a different format by sending new RVC-BSDL/FNL descriptions and enabling efficient support for multiple coding formats. Moreover, nonstandard coding format can be supported provided it uses FUs available to the decoder (i.e., FUs in the decoder's VTL).

2.2. DV/DVCPRO Standard. Digital video (DV) is a digital video format created by a group of companies (led by Sony, JVC, Panasonic and other producers), and launched in 1995 [7, 8]. DV refers to the compression format employed to capture, edit and store video footage. Moreover, DV is also applied to cameras that record using mini-DV tape.

Two standards [7, 8] define the data structure for the interface of DV-based digital audio, subcode data, and compressed video at different bit rates: the DV standard includes both 525/60 and 625/50 systems, in which the numeric values "525" and "625" refer to the number of the horizontal sync lines while the numeric values "60" and "50" indicate the field rate. The DVCPRO standard now includes 1080/60i, 1080/50i, 720/60p, and 720/50p systems, in which the numeric values "1080" and "720" refer to "1920 × 1080" and "1280 × 720" image sampling structure, respectively, while the values "50" and "60" refer to the field/frame rate and the letters "i" and "p" indicate the field/frame type. There are three types of compressed bit rates defined in DV/DVCPRO standard: 25 Mbps, 50 Mbps, and 100 Mbps, as shown in Table 1. This expansion allows the DVCPRO format to support not only a high-quality program production system but also the next generation of broadcast system. A broadcast system based on DVCPRO's 25 Mbps, 50 Mbps and 100 Mbps compression is the most efficient, practical, and widely supported method of providing today's requirements.

The number of channels of the compressed DV stream is assigned to 1, 2, and 4 corresponding to DV/DV25, DVCPRO50, and DVCPRO100 formats, respectively [7, 8]. Each channel is further divided into 10 sequences for 525/60 system and 12 sequences for 625/50 system. Each sequence consists of header, subcode, video auxiliary data (VAUX), audio, and video sections as shown in Figure 1. Each section is comprised of numbers of digital interface (DIF) blocks. The DIF block is the basic element of DV data structure and each DIF block consists of a 3-byte ID and 77 bytes of data. DIF data bytes are numbered 0 to 79. The type of DIF block is assigned by its ID and the DIF data part (play load) presents the parameters of the DV decoder.

The DV/DVCPRO standard only has intra (I) frame and its process is straightforward. Figure 2 illustrates the procedure of the complete process. The decoder generates the video output by the following processes: variable length decoder (VLD), inverse quantization (IQ), weighting, inverse discrete cosine transform (IDCT), block deshuffling, and upsampling while the audio output is generated by data mapping and data deshuffling units. One of the most important differences between DV/DVCPRO and MPEG compression is that the audio and video data of DV are mixed into DIF blocks. In the 1080-line system, video data, audio data, and subcode data in one video frame are processed

Table 1: Formats of DV standards.

DV-Standards (Types)	Size	Format	Channels	Sequences	DIF-Size (bytes)	System	DataRate (Mb/s)	Specification
DV-NTSC	720 × 480	4:1:1	1	10	120000	60 Hz	25	IEC61834
DV-PAL	720 × 576	4:2:0	1	12	144000	50 Hz	25	IEC61834
DV25-NTSC	720 × 480	4:1:1	1	10	120000	60 Hz	25	SMPTE314
DV25-PAL	720 × 576	4:1:1	1	12	144000	50 Hz	25	SMPTE314
DVCPRO50-NTSC	720 × 480	4:2:2	2	10	240000	60 Hz	50	SMPTE314
DVCPRO50-PAL	720 × 576	4:2:2	2	12	288000	50 Hz	50	SMPTE314
DVCPRO100P720P-NTSC	960 × 720	4:2:2	2	10	240000	60 Hz	100	SMPTE370
DVCPRO100P720P-PAL	960 × 720	4:2:2	2	12	288000	50 Hz	100	SMPTE370
DVCPRO100HD-NTSC	1280 × 1080	4:2:2	4	10	480000	60 Hz	100	SMPTE370
DVCPRO100HD-PAL	1280 × 1080	4:2:2	4	12	576000	50 Hz	100	SMPTE370

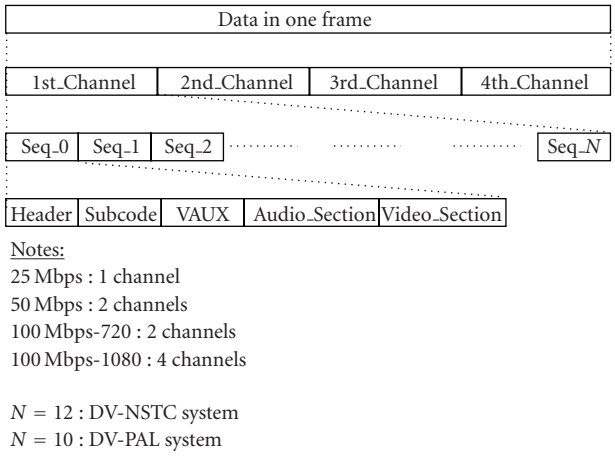


Figure 1: DV data type.

in each frame. In the 720-line system, these data are spread into two video frames. To process the 720-line system in the same way as the 1080-line system, they are processed within one frame duration of the 1080-line system. The audio data, corresponding to one video frame in the 1080-line system and two video frames in the 720-line system, are defined as an audio processing unit [8].

3. Modeling and Design

As stated above, DV/DVCPRO is intraframe only video standard without bidirectional (B) and progressive (P) frames so that its operation is not as complex compared to other standards, such as MPEG-4 and H.264. However, there are challenges when modeling DV/DVCPRO with RVC-CAL for the following reasons: (1) How to efficiently partition the FUs while considering the features of DV data. (2) Video and audio data are shuffled in DV compressed data. (3) There are more than nine types of video formats as well as various processing modules, such as 8-8 inverse discrete cosine transform (IDCT), 2-4-8 IDCT, scanning, weighting, and de-shuffling. At first, the partition of DV FUs is important for RVC-CAL modeling. Efficient FU partition, utilizing available FUs, is able to save design time and improve performance. Based on the features of DV/DVCPRO processing, our proposed RVC modeling mainly contains three parts as shown in Figure 3: Parser FUs, VLD and IDCT FUs, and Deshuffling FUs. This partition divides DV/DVCPRO into reasonable function blocks in order to minimize the number of tokens between FUs and actors while considering the reuse of available MPEG-RVC FUs. For example, 8-8 IDCT and 2-4-8 IDCT are separated in order to use the available MPEG-4 8 × 8 IDCT FU. The audio process is separated from other FUs since no reference FUs are available.

3.1. RVC Parser FUs. The parser FUs unit tries to decode DV parameters for the following using. Unlike other video data, DV data have strict DIF with a size of 80 bits and data location, which makes the design simpler than others. The RVC-CAL parser unit consists of ten interacting actors. At first, the "Serial" actor reads the input DV data and makes them in the form of tokens. The following actors consume the incoming tokens according to different ID types: Header (ID = 000), Subcode (ID = 001), video auxiliary data (VAUX, ID = 010), and audio auxiliary data (AAUX, ID = 011). When these actors consume the incoming tokens, the parsing process is considered complete. "Header" actor produces the DIF sequence number, DIF block number, and channel identification tokens while "Subcode" actor generates time code (TC) and binary group (BG) package tokens. The actors "VAUX" and "AAUX" are further separated into " VAUX source pack (VS), VAUX source control pack (VSC), AAUX source pack (AS), and AAUX source control pack (ASC)" actors according to the different incoming tokens. The frame type, display type, and decoded type are parsed by "VSC" and "VS" actors while audio-compressed mode and sampling frequency are parsed by "ASC" and "AS" actors. The actual

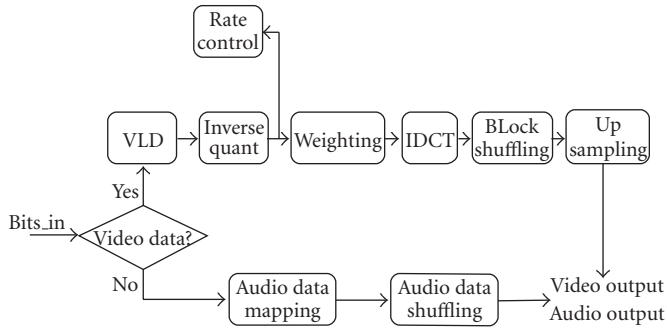


Figure 2: DV decoder data processing block diagram.

compressed video and audio data tokens are produced by the "A/V data" actor.

3.2. RVC VLD and IDCT FUs. The variable length decoder (VLD) actor of DV standard is different from MPEG standard. As shown in Figure 3, three passes have been adopted for the DV VLD. The procedure of VLD is as follows: first, passing the bitstream data into the first pass; second, passing the remaining data into the second pass if there are surplus data remained in the current DIF block; third, passing the remaining data into the third pass if there are surplus data remained in current MB; finally, the VLD process is terminated whether there are data remained or not. The VLD table is automatically generated using similar procedure in [9].

The IDCTs of DV standard have two modes: 8-8-IDCT and 2-4-8-IDCT. They are selectively used to optimize the data-reduction process, depending upon the degree of content variations between the two fields of a video frame. Based on 8-8 DCT and 2-4-8 DCT modes, there are two modes of scanning and weighting, accordingly. The 8-8 scanning IDCT mode is kind of zigzag scanning, which is similar to MPEG-4, hence, this allows one to make little modification and reuse the FU from MPEG-4. The 2-4-8 mode needs further modifications from MPEG-4. In this mode, one 8×8 macroblock (MB) is divided into two vertical 8×4 subblocks so that the scanning order has to be modified accordingly. Therefore, the left top pixel of the upper subblock is scanned at first and followed by the left top pixel of the lower subblock. The DCT coefficients are weighted by a quantizer matrix. The different quantizer matrices are set for different luminance and color signals [7, 8]. Also, different DV formats have different quantizer matrices and therefore different weighting values. DV DCT coefficients are quantized to within 9-bit words in order to limit the amount of data in one video segment to five compressed MBs.

3.3. RVC Deshuffling FUs. The RVC Deshuffling FUs contain video and audio data rearrangement actors. "MB_Mapper" actor designates the correspondence between video DIF blocks and compressed macro blocks. "Video_Deshuffling" actor defines the correspondence between compressed macro-blocks and the video segment. The video segment consists of five MBs which are assembled from various areas within the video frame. Note that the NSTC system follows different MB mapping and video shuffling rules.

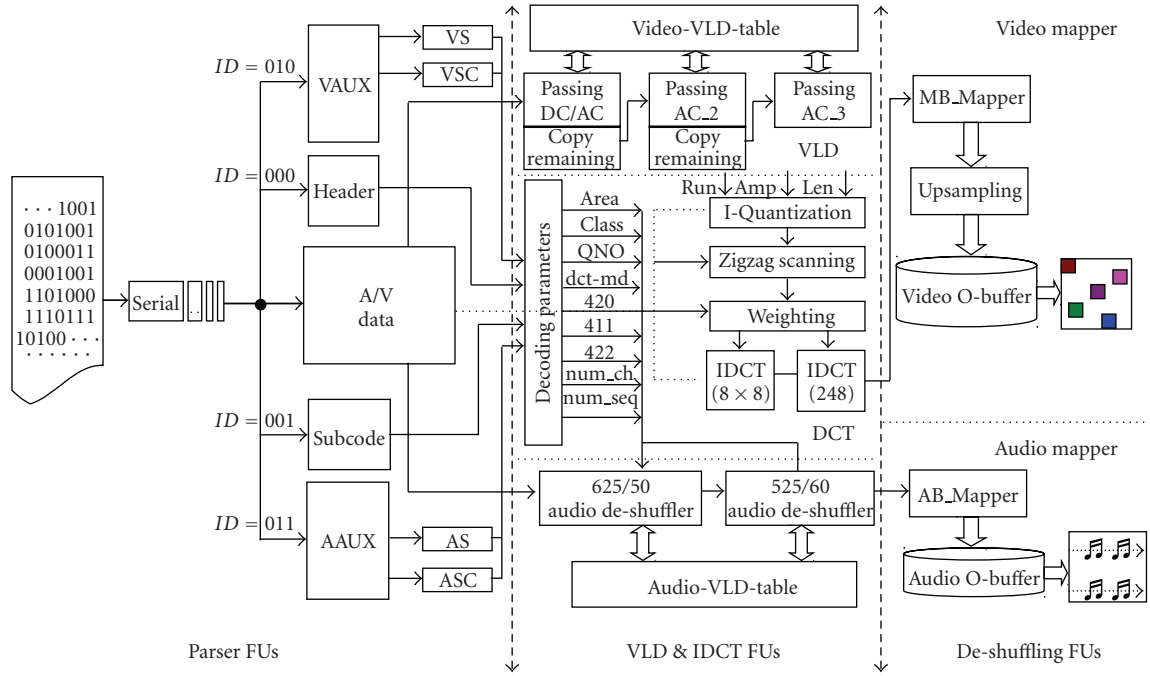
The "AB_Mapper" actor outputs the decoded audio data according to the DV audio block mapping rule which is defined in DV/DVCPRO standard [8]. Note that NSTC and PAL system have different audio mapping rules to be followed.

4. Simulation and Analysis

The DV/DVCPRO decoder is successfully modeled in RVC-CAL simulation environment. CAL is not only a description formalism but is supported by a simulation environment portable on all platform supporting JAVA virtual machine (VM). Two modeling and simulation environments: Ptolemy II [10] and Moses [11], were used in the proposed simulation. However, these two environments are currently updated by ORCC [12] and OpenDF [13], that can also be used and it is expected that they would provide the same results.

Numbers of DV formatted sequences are tested by the proposed design, such as Foreman, Mobile and Calendar, and Driver and Flower. To verify the DV audio decoding, the tested video sequences are combined with raw audio data. The encoded DV format sequences can be generated by FFMPEG [14] reference code. We compare the results from the proposed design with RVC-CAL and ones from FFMPEG reference decoder both in video and in audio output. No differences are found. Both the decoded video and audio output are able to be played back successfully. Therefore, the feasibility of DV modeling with RVC-CAL is verified. The experimental analysis is made in the following aspects:

4.1. Reusability of MPEG-4 FUs. One of the major advantages of RVC framework is to reuse available FUs and reduce design time. As stated above, MPEG-4 simple profile has been successfully developed by MPEG organization. Some FUs are available to be used or can be modified for reuse. Unlike H.264/AVC or AVS standard, the DV/DVCPRO standard has less similarity with MPEG. Therefore, some new FUs have to be redesigned, however, some MPEG-4 FUs can



DIF-digital video digital interface format
MB-macroblock
AB-audio block unit
A/V-audio video
VAUX-video auxiliary
VS-VAUX source
VSC-VAUX source control
AAUX-audio auxiliary
AS- AAUX source
ASC-AAUX control
QNO-quantization number step
dct-md-DCT mode ch-channel
Seq-sequence

Figure 3: DV-FU partition and modeling.

Table 2: FUs reusability of DV.

No.	Function Block	FU	Reused	New	Modified	Comments
1	Parser	ParserHeader		√		Header, Subcode, VAUX, AAUX
2		Passing		√		3 Passing for DC and AC,Remains
3		I-Quant	√			Reuse MPEG-4
4	Decode_Video	Scaning			√	Zagzig scanning for 8 × 8 and 2-4-8
5		Weighting			√	Weighting for 8 × 8 and 2-4-8 mode
6		IDCT			√	IDCT for 8 × 8 and 2-4-8 mode
7	Video_De-shuffling	Video_Mapping		√		Mapping DIFs with MBs
8		UpSampling		√		Upsample 420 to 422, or 411 to 422
9		Audio_PAL_Deshuffling		√		Deshuffling PAL Audio DIFs
10	Decode_Audio	Audio_NTSC_Deshuffling		√		Deshuffling NTSC Audio DIFs
11		AB_Mapping		√		Mapping Audio Blocks for Output

Table 3: Comparison between C code and RVC-CAL.

Name	C code	RVC-CAL	Reduction (%)
Video process	3718	2927	21.3
parser	—	978	—
VLD	—	365	—
IQ	—	202	—
Scanning	—	166	—
Weighting	—	178	—
IDCT-8 × 8	—	229	—
IDCT-248	—	281	—
Video deshuffler	—	144	—
Upsampling	—	256	—
Video mapper	—	128	—
Audio Process	835	451	46
Audio deshuffler	—	238	—
Audio block mapper	—	213	—
Total	4553	3378	25.8

still be modified to be used in order to save design time. As shown in Figure 3, modified IDCTs from MPEG-4 simple profile have been used in the proposed design while the inverse quantization FU is much similar as MPEG-4. We just reuse it as normal. The statistic reusability table is listed on Table 2.

4.2. Reduction of Design Overhead. Another advantage of RVC framework is that it has high abstract and concise code representation. Table 3 shows the lines of code (LOC) compared with reference code reported in [14]. The numbers show that the RVC-CAL modeling has an average 25.8% less LOCs than the reference C code from FFMPEG. We cannot compare the development time in detail because it is hard to know how long writing the reference code has been taken. However, writing the RVC-CAL code only takes 3 months for a middle-level CAL programmer, which is more efficient than writing C code.

4.3. Efficient Code Transformer. The last advantage of RVC framework is that it automatically targets both software and hardware with supporting tools. An automatic CAL-to-C code generator has been developed in [15] while CAL-to-VHDL code generator is successfully developed in [16]. It is reported that the automatically generated VHDL is not only four times faster in development time, but it is also more efficient in execution time. The main reason is attributed to RVC-CAL being using dataflow methodology instead of direct VHDL register transfer level (RTL) design.

5. Conclusions

In this paper, an RVC-CAL modeling of DV/DVCPRO video coding standard has been featured and hence the validation of RVC modeling has been proved. Based on the features of DV/DVCPRO standard and RVC framework, RVC FUs partition has been described. The functions of important actors and tokens are also explained in detail. The experimental result illustrates the advantages of using MPEG RVC as a new video coding standard. Writing RVC-CAL takes less time than writing reference C code or VHDL code, which allows developers to concentrate on function optimization rather than coding skills. Moreover, RVC framework enhances the interoperation between standards. Not only can available function units be reused, but new function units and algorithms, or even rebuilt standards, can also be incorporated into the RVC framework. Finally, RVC-CAL adopts the parallelism and dataflow programming methodology, which is closer to hardware implementation than sequential process.

Acknowledgments

This work is supported by the University of Windsor, Ontario, Canada (2009).

References

[1] E. S. Jang, J. Ohm, and M. Mattavelli, "Whitepaper on Reconfigurable Video Coding (RVC)," ISO/IEC JTC1/SC29/WG11, January 2008.

[2] C. Lucarz, M. Mattavelli, J. Thomas-Kerr, and J. Janneck, "Reconfigurable media coding: a new specification model for multimedia coders," in *Proceedings of the IEEE Workshop on Signal Processing Systems (SiPS '07)*, pp. 481–486, October 2007.

[3] D. Ding, H. Qi, L. Yu, T. Huang, and W. Gao, "Reconfigurable video coding framework and decoder reconfiguration instantiation of AVS," *Signal Processing: Image Communication*, vol. 24, no. 4, pp. 287–299, 2009.

[4] H. Aman-Allah, K. Maarouf, E. Hanna, I. Amer, and M. Mattavelli, "CAL dataflow components for an MPEG RVC AVC baseline encoder," *Journal of Signal Processing Systems, Springer*, Published online July 2009, In press.

[5] J. Li, D. Ding, C. Lucarz, S. Keller, and M. Mattavelli, "Efficient data flow variable length decoding implementation for the MPEG reconfigurable video coding framework," in *Proceedings of the IEEE Workshop on Signal Processing Systems (SiPS '08)*, pp. 188–193, October 2008.

[6] J. Eker and J. Janneck, "CAL language report," Technical Memo UCB/ERL M03/48, University of California at Berkeley, December 2003.

[7] SMPTE 314M-1999 Data Structure for 25 and 50 MB/s, SMPTE, 1999.

[8] SMPTE 370M-2006 Data Structure for 100 MB/s, SMPTE, 2006.

[9] C. Lucarz, J. Li, D. Ding, and M. Mattavelli, "Automaticgeneration of RVC Parser from BSDL Syntax Description: Variable Length Decoding," ISO/IEC JTC1/SC29/WG11 MPEG/M15163, Antalya, Turkey, January 2008.

[10] http://ptolemy.eecs.berkeley.edu.

[11] http://www.tik.ee.ethz.ch/~moses/.

[12] http://sourceforge.net/projects/orcc/.

[13] http://sourceforge.net/projects/opendf/.

[14] http://ffmpeg.org/.

[15] G. Roquier, M. Wipliez, M. Raulet, J. W. Janneck, I. D. Miller, and D. B. Parlour, "Automatic software synthesis of dataflow program: an MPEG-4 simplpe profile decoder case study," in *Proceedings of the IEEE Workshop on Signal Processing Systems (SiPS '08)*, pp. 281–286, July 2008.

[16] J. W. Janneck, I. D. Miller, D. B. Parlour, G. Roquier, M. Wipliez, and M. Raulet, "Synthesizing hardware from dataflow programs—an MPEG-4 simple profile decoder case study," *Journal of Signal Processing Systems, Springer*, Published online July 2009, In press.