A codesign synthesis from an MPEG-4 decoder dataflow description

Nicolas Siret, I. Sabry, Jean François Nezan, Mickaël Raulet

A codesign synthesis from an MPEG-4 decoder dataflow description

Nicolas Siret* **, Ismaïl Sabry*, Jean François Nezan** and Mickaël Raulet**
*Lead Tech Design, F-35043 Rennes, France
**IETR/INSA. UMR CNRS 6164, F-35043 Rennes, France

***Abstract*— The elaboration of new and innovative systems such as MPSoC (MultiProcessor System on Chip) which are made up of multiple processors, memories and IPs lies on the designers to achieve a complex codesign work. Specific tools and methods are needed to cope with the increasing complexity of both algorithms and platforms. Our approach to design such systems is based on the usage of a high level of abstraction language called RVC CAL. This language is dataflow oriented and thus points out the concurrency and parallelism of algorithms. Moreover CAL is supported by the OpenDF simulator and by two code generators called CAL2C (software generator) and CAL2HDL (hardware generator). The MPEG expert group has recently elaborated the Reconfigurable Video Coding (RVC) standard which defines the RVC CAL language as reference for MPEG video decoder descriptions. This paper introduces the opportunities to design an innovative system involving hardware and software IPs, embedded processors and memories from a CAL model. Practical results on a FPGA are provided with a codesign solution of an MPEG4 Simple Profile (SP).**

***Index Terms*— Dataflow, Cal, Reconfigurable Video Coding, MPEG, codesign, hardware, software.**

I. Introduction

There are nowadays a lot of multimedia devices, for instance mobile computer, smartphone or mobile phone. Video standards (e.g. MPEG-1, MPEG-2, etc.) must be fitted to each device. A problem commonly faced by the designers is the lack of re-usability and genericity of the code provided by the standard (usually a C/C++ monolithic specification). To overcome this limitation, a new standard called Reconfigurable Video Coding (RVC) [1], [2] has been recently standardized by the Moving Picture Expert Group (MPEG). An RVC decoder is so built by carefully connecting elementary blocks called Functional Units (FUs). An RVC decoder could decode existing MPEG standards like MPEG-4, AVC or SVC and even future standards. This decoder should also be compliant for hardware and software code generation and so, should be fitted to embedded systems. Currently the most ended (and stable) description for RVC approach is an MPEG-4 SP decoder provided by the MPEG RVC experts group. An AVC decoder is also under development as part of the MPEG RVC standardization [3].

Specific architectures (e.g. low power) need to be design for hardware processing. The evaluation of possibilities and capacities on FPGA and ASIC is one important way to validate the interest of this standard. This paper is the first which introduces the method to design an embedded codesign decoder (i.e. embedded processors, IPs) from RVC descriptions and which provides practical results. The paper is organized in three parts: the section II introduces the RVC standard and its related tools, the section III presents the MPEG-4 SP decoder and the codesign methodology, the section IV outlines practical results and perspectives of future work.

II. RVC standard and tools

A. The RVC standard

The MPEG RVC Framework is currently under development by the MPEG committee as part of MPEG-B and MPEG-C standards [4]. It aims at providing a framework allowing dynamic development, implementation and certification of existing or new video coding solutions. Moreover, it involves abstract model, higher flexibility and reusability features in order to efficiently support multiple codec configurations and to facilitate innovation in codec design. An abstract decoder, shown in figure 1, is built as a block diagram expressed with functional unit Network Language (FNL) [1]. Each block is defined with processing entities called Functional Units (FUs) [2] and connections which represent dataflows between FUs. The MPEG-B part 5 Bitstream Syntax Description Language (BSDL) [5] describes the syntax of the bitstream the RVC decoder.

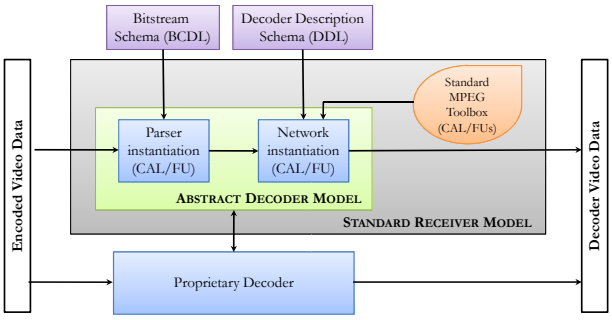

Fig. 1. Graphical representation of the conceptual process of deriving a RVC abstract decoder model

RVC provides both a normative standard library of FUs and a set of decoder descriptions expressed as networks of FUs. Such a representation is modular and helps the reconfiguration of a decoder by modifying the topology of the network.

B. Functional Units (FUs)

Two kinds of FU can be distinguished in a video processing. The first one involves almost algorithmic video decoding

processes (i.e. *IDCT, inverse-quant, etc.*) which are usually reusable between decoders. Conversely, the second one contains data management processes (i.e. *parser, multiplexer, etc.*) which usually have to be adapted to each specific codec. The FUs which compose the RVC standard library are built from the model of computation and normative I/O specified in RVC CAL. RVC CAL [6], [7] is a subset of the CAL Actor Language, which is a dataflow oriented language. Each FU corresponds to a RVC CAL actor contained in the Standard MPEG Toolbox.

Figure 2 introduces a network of actors. An actor is a computational entity with interfaces (input and output ports), internal states and parameters. Actors are completely independent from each others and they can interact by exchanging data (called tokens) along channels. During a process (also referred as firing an action), the actor consumes input tokens, produces output tokens and changes its internal state. Action guards, action scheduling with Finite State Machine (FSM) or action priorities are the controlling structures that constrain the selection of actions to fire.

Fig. 2. A network of CAL actors

C. CAL tools

CAL is supported by the OpenDF [8] simulator[1] and by two code generators called CAL2C [9] and CAL2HDL [10] which has been designed to work on an Eclipse environment. CAL2C converts CAL actors into C code which can be compiled on any processor [11] including embedded processors, DSPs, ARMS. CAL2HDL is a subset of OpenDF plugin. It converts a CAL model into HDL code which can be implemented on Xilinx FPGA. For an automatic generation of the parser, a BSDL to CAL translator is also under development as part of the OpenDF effort [5].

III. Codesign methodology

A. The MPEG-4 SP decoder

As shown figure 3, the MPEG-4 SP decoder is compound with three main functional units' blocks: a *parser*, a *texture decoder* and a *motion compensator* (i.e. *acdc* and *idct2d*). Furthermore all of these FUs are themselves composed of algorithmic video actors or data management processes actors.

Motion compensation and texture decoding are classical processes in a video decoding application. They handle the video data provided by the *parser* to realize a macroblock decoding of each image. On the contrary the *parser* which

[1]Open dataflow sourceforge project : `http://opendf.sourceforge.net/`.

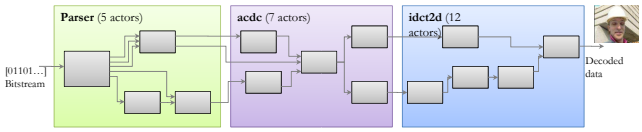

Fig. 3. The CAL MPEG-4 SP decoder desciption

analyzes the incoming bitstream and extracts the video data is specific to the standard. The *parser* is the most complex actor (1389 lines), in fact it is compound with more than forty actions sequenced with guards and complex FSM.

B. Design process

Figure 4 introduces the four steps involved in the design process.

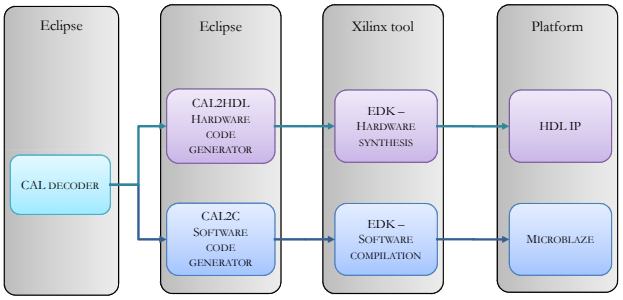

Fig. 4. Four steps to design : CAL research and developement, code generation, code synthesis and onchip validation

Actors of the MPEG-4 decoder are compliant with CAL2HDL [12] [10] and CAL2C [9]. In effect some algorithms involved in video decoding are usually fitted to hardware or software processing. As a consequence, it is necessary in a first step, to specify the CAL blocks on the network which are fitted to hardware or software processes. In this case, *acdc* and *idct2d* actors involve almost parallel processes and are adapted to hardware. On the opposite, the *parser* block performs full sequential operations (e.g. reading input bit, storing input bit, looking for start code ...) and thus, is fitted to software processing. The second step consists in the hardware code generation, the mapping between the generated IP and the embedded processors and the code synthesis. Indeed, once the hardware code is generated, it is necessary to connect the MPEG-4 decoder IP with processors or others hardware IPs which will send him the MPEG-4 video data and display the decoded video. In a third step it is essential to program the processors with a scheduler and the generated software code. The last step consists in implementing and testing the MPEG-4 codesign decoder on the platform.

C. Hardware and software processing

The MPEG-4 decoder sources are generated thanks to the CAL2HDL. Each CAL actor is transformed into a verilog file and the top level into a vhdl file. Inputs and outputs (I/Os) compulsory connected on the top are *Data (I/O), Send (I/O), Acknowledgment (I/O), ready (I/O), clock* and *reset*. The tool used to implement the hardware code generated is Xilinx EDK,

thus embedded processors which realize the software part are Microblazes. To connect the generated HDL decoder with a microblaze, a study of options point out that an efficient way consists in encapsulating the IP as a coprocessor. Then the IP can be plug to the processor thanks to a Xilinx proprietary bus called Fast Simple Link (FSL). FSL is a unidirectional fast communication channel bus used by Xilinx to connect there processors to peripherals. As shown figure 5, the link between FSL bus and decoder I/Os has been made in accordance with the FSL standard thanks to logical operators (e.g. and, or, when, etc.).

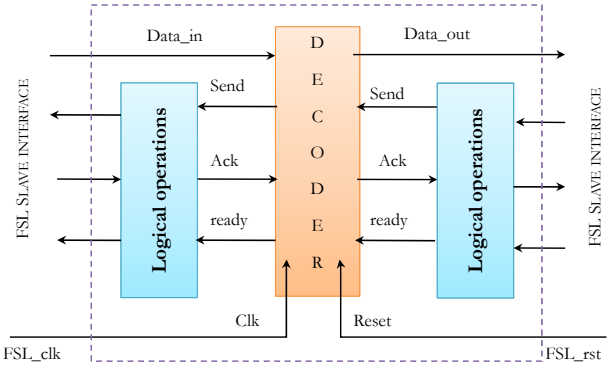

Fig. 5. The generated decoder can be connect with a FSL link thanks to logical operators

Before code synthesis, Xilinx peripherals have to be connected as introduces figure 6.

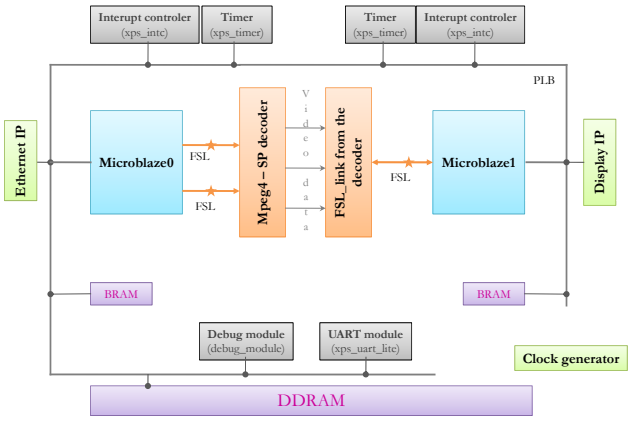

Fig. 6. The whole system needed to decode video, notably the decoder and two embedded processors

The whole system is divided into two parts. The first part is composed of three elements: the Ethernet IP which manages the Ethernet link between a network and the system, the first Microblaze which orders the Ethernet IP, realizes the software part and transmits MPEG-4 video data through two FSL links and the MPEG-4 decoder. The second part is also composed of three elements: a YUV2RGB (i.e. conversion and display) IP, a second microblaze which read YUV data from the decoder and send them to the YUV2RGB IP and a coprocessor called "*FSL link from the decoder*". The "*FSL link from the decoder*" coprocessor formats the video data to make them suitable with the FSL bus. Moreover it is compulsory to overcome a limitation of the FSL link indeed a FSL link can be connected with a single Microblaze. Due to the platform limitation (figure 9), software applications and related data are stored on the external DDRAM. **Thus, processors must regularly make external memory access to complete their treatment**.

The C code provided by CAL2C is almost fully synthesizable. CAL actors are transformed into C files scheduled with a SystemC process. As a consequence it is necessary to modify the C scheduler using FIFOs (e.g. structure of table type) to send variables or parameters between actors in accordance with the RVC CAL model of computation. The work to do is to call all the necessary function in the scheduler and to read or write data into the correct FIFOs. The structure of the program is presented figure 7.

```
// FIFOs definition (for one fifo)
struct fifo fifo1 = {sizeof(char), size, array};
struct fifo *parser = &fifo1;

// Main
int main() {
  // various initialization
  init();

  // Infinite loop
  while(1) {
    Receive_TCP();
    Serialize();
    Parser();
    FSL_transmission();
    }
  }
```

Fig. 7. Structure of the scheduler code

In first, MPEG-4 data are read from the Ethernet IP and are written in the input FIFO of the parser. This one achieves its own process without any intervention from the designer. In the second place, YUV data has to be sent on the FSL bus. This is made by reading the output FIFO of the parser and making a basic writing operation, (i.e. *write_data_into_fsl(data, slot)*).

IV. Results on the MPEG-4 SP decoder

Results introduce figure 8 and figure 9 are obtained without any optimization of the CAL description provided by the MPEG RVC group. H.decoder is made up of the hardware *decoder* and the hardware *parser*. H.system consists of the whole system presented figure 6 including the MPEG-4 decoder (with the *parser*). Cod.decoder is made up of the hardware *decoder* without the hardware *parser*. Cod.system consists of the whole system presented figure 6 including the MPEG-4 decoder (without the hardware *parser*). For decoders, the slices occupation, the internal RAM (iRAM) occupation and the maximum frequency are computed by the ISE tool and the Frames per second (FPS) are calculate thanks to Modelsim. For the whole systems, results are computed by the EDK tool (e.g. slice, iRAMs, and frequency) and by experimentations on the Microblaze (e.g. FPS).

Comparing the results highlights a profit in term of frequency performance and slices occupation. FPS results for the

Project	H.decoder	H.system	Cod.decoder	Cod.system
Slices:	26%	80%	22%	74%
iRAM:	8%	33%	8%	30%
frequency:	55 MHz	40 MHz	55 MHz	50 MHz
FPS:	58	4	58	0,77

Fig. 8. Results obtained on the Xilinx ML402 evaluation platform (FPGA: Virtex4 - XC4VSX35)

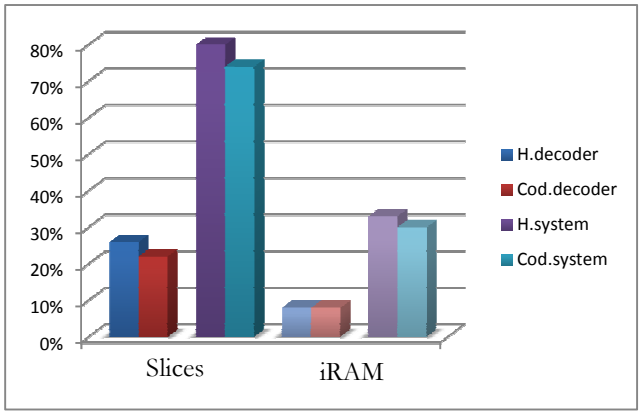

Fig. 9. Slices and iRAM utilization for both the decoders and the whole systems

hardware solution and the codesign solution seem disappointing but are actually encouraging. In fact the highly generalist architecture of the Microblaze (hardware, codesign), the hardware aspect of the RVC CAL *parser* (codesign) and the lack of pipelining (codesign) explains the FPS loss. More precisely, in both the codesign and the hardware solution, Microblaze performances are highly strained by the capabilities of the platform (e.g. one instruction per cycle, 100MHz operating frequency and some regularly external memory access, see subsection III-C) and by the necessary use of a kernel to manage the application. Moreover, for the codesign system other significant confines appear. On the one hand there is no pipelining, so the Microblaze has to manage four processes: receiving data from the Ethernet IP, serializing the data (e.g. byte to bits), parsing the data and transmitting them to the FSL. On the other hand the ***parser* provided with the MPEG-4 SP decoder has been designed for a hardware processing**. Indeed, it is made up of a sizeable FSM (more than fifty states in the C code) and many actions (more than one hundred in the C code) which achieve a binary process. At this moment the hardware aspect of the RVC CAL *parser* makes it very complex and difficult to change, a prospect is to write a generic *parser* suitable for software processing to replace the current one and to measure the benefits.

Design the MPEG-4 codesign decoder highlights several advantages. The greatest one is the time to design, indeed using the RVC CAL MPEG-4 description and the RVC associated tools reduces the work to do and thus, the time to design from several month to few days. Moreover, despite several strains and no code optimization, results are quite good and opens up many prospects in term of software optimization, logical occupation reducing and dataflow architecture research.

V. Conclusion and prospects

Designing new and innovative embedded systems is more and more complex. The rapidly growth of MPSoC (Multi-Processor System on Chip), which are made up of multiple processors, memories and IPs, poses problems of development and optimization of the hardware and software code. This paper points out another way to program this kind of systems: instead of separating hardware and software developments, the designers can work on the entire system at a higher level of abstraction. The system is seen as a set of independent actors which can be transformed either in hdl code or in C code. The framework of the RVC standard (with the RVC CAL language) uses this new and innovative approach. The codesign method introduces in this paper allow the designer to generate a codesign solution from an RVC CAL decoder.

The results presented are promising, it is possible to generate within a limited time, a complex codesign system. Moreover, its frequency performance, slice occupation and FPS are quite good whereas no optimization has been made. Based on this work, several prospects are under studies notably the building of a generic parser, the enhancement of the tools to improve hardware and software generated code and the deployment of multiple processors (MPSoC) on the FPGA.

References

[1] ISO/IEC Interntional Standard 23001-4: 2009, "Information technology - MPEG systems technologies - Part 4: Codec Configuration Representation," 2009.

[2] ISO/IEC Interntional Standard 23002-4: 2009, "Information technology - MPEG video technologies - Part 4: Video Tool Library," 2009.

[3] J. Gorin, M. Raulet, Y-L. Cheng, H-Y. Lin, N. Siret, K. Sugimoto, and G.G. Lee, "An RVC dataflow description of the AVC constrained baseline profile decoder," in *International Conference on Image Processing (ICIP)*, 2009.

[4] C. Lucarz, M. Mattavelli, J. Thomas-Kerr, and J. Janneck, "Reconfigurable Media Coding: a new specification model for multimedia coders," in *Signal Processing Systems (SiPS)*, 2007.

[5] 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 *Signal Processing Systems (SiPS)*, 2008, pp. 293–298.

[6] S. S. Bhattacharyya, J. Eker, J. W. Janneck, C. Lucarz, M. Mattavelli, and M. Raulet, "Overview of the MPEG Reconfigurable Video Coding Framework," *Springer journal of Signal Processing Systems. Special Issue on Reconfigurable Video Coding*, 2009.

[7] ISO/IEC FDIS 23001-4, "MPEG systems technologies – Part 4: Codec Configuration Representation," 2009.

[8] Shuvra S. Bhattacharyya, Gordon Brebner, Jörn W. Janneck, Johan Eker, Carl von Platen, Marco Mattavelli, and Mickaël Raulet, "OpenDF: a dataflow toolset for reconfigurable hardware and multicore systems," *SIGARCH Comput. Archit. News*, vol. 36, no. 5, pp. 29–35, 2008.

[9] 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 simple profile decoder case study," in *Signal Processing Systems (SiPS). IEEE Workshop on*, 2008, pp. 281–286.

[10] 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," in *Signal Processing Systems (SiPS)*, 2008.

[11] I. Amer, C. Lucarz, G. Roquier, M. Mattavelli, M. Raulet, J.-F. Nezan, and O Deforges, "Reconfigurable video coding on multicore," in *Signal Processing Magazine, IEEE*, 2009, pp. 113–123.

[12] Julien Dubois Richard Thavot, Romuald Mosqueron and Marco Mattavelli, "Hardware synthesis of complex standard interfaces using CAL dataflow descriptions," in *Design and Architectures for Signal and Image Processing (DASIP)*, 2009.