**I****NTERNATIONAL ORGANIZATION FOR STANDARDIZATION**

**ORGANISATION INTERNATIONALE DE NORMALISATION**

**ISO/IEC JTC1/SC29/WG11**

**CODING OF MOVING PICTURES AND AUDIO**

**ISO/IEC JTC1/SC29/WG11**

**MPEG2014/N14991**

**Strasbourg, France**

**October 2014**

|  |  |
| --- | --- |
| **Status:** | **Approved** |
| **Title:** | **White Paper on Reconfigurable Media coding** |
| **Source:** | **Video** |
| **Authors** | **Marco Mattavelli, Mickaël Raulet** |

# Abstract

The objective of this white paper is to provide an overview of the MPEG Reconfigurable Media Coding framework. Initially started with the intention of covering only MPEG video compression technology, it now includes also 3D Graphics compression standards. The framework is composed by normative elements and by non normative tools. Normative is the standard library of functional units that can be connected into various decoder configurations building high level dataflow based functional specifications of decoders . Non normative tools provide simulation, analysis and executable code synthesis capabilities for the generation of implementations running on different platforms of any RMC specification.

# Objectives and Vision of the Reconfigurable Media Coding Framework

The Reconfigurable Media Coding Framework, in short RMC, started with the objective of overcoming the major limitations on the way MPEG technology was specified. After the first MPEG standards [1], MPEG-1 and MPEG-2 the interoperability problem changed from the need of adopting a common standard to the necessity of being able to support an increasing number of standards and their different variants called profiles [2]. Another problem that the classical way of specifying standards presents, is the essential monolithic structure and supporting formalism. Even if all standards present a common structure the monolithic and sequential way of expressing the specifications, make difficult to identify common and different components. Another limitation of sequential code is that it implies a specific choice of the sequence of operation and that does make explicit if other choices can be made without affecting the implicit data dependencies. Moreover, the sequential assumption was not a drawback in the sequential processor era, but is not the more appropriate starting point when multicore and manycore are becoming the more common implementation choices. Starting from such observations MPEG committee started to develop this new form of specification for video codecs by standardizing what was referred to as:  "Reconfigurable Video Coding" (RVC) framework [4], [5]. Then the RVC framework has then been extended by including 3D graphic coding and, thus has changed its name in RMC where M stands for "Media", looking forward to include audio and possibly systems components in the future. Although some years have passed since the first components of RMC have been developed there is still the room of extending the framework to new MPEG standardization efforts, as it has been done for the recent HEVC specification [3] and for improving the performance and functionality of the non normative tools supporting simulation analysis and direct implementation synthesis. Indeed, besides the goal of unifying the specification of the variety of coding standards another interesting and in some ways very innovative objective of RMC is to try to narrow the gap between the algorithmic (i.e. standard) specification and the implementation of the applications. Such gap not only constitutes a serious impediment for efficient development of implementations, but the augmented complexity of the new generation of video codecs, and the increasing heterogeneity of processing platforms that may include GPUs make it ever and ever wider.

The fact that a specification such as RMC does not imply a specific processing architecture (the single processor), but abstracts from it and results to be portable on any combination of architectures is a very attractive feature nowadays. Moreover, since the RMC framework is now covering not only video coding as at the beginning, but also 3D graphics compression a particular attention is given to the fact that RGC specifications should result appropriate for GPUs platforms that are the privileged processing platforms for graphics application implementations. This objective has been reached by using a lower level modular design for the development of the components of the RGC library if compared to level used to build the video compression components.

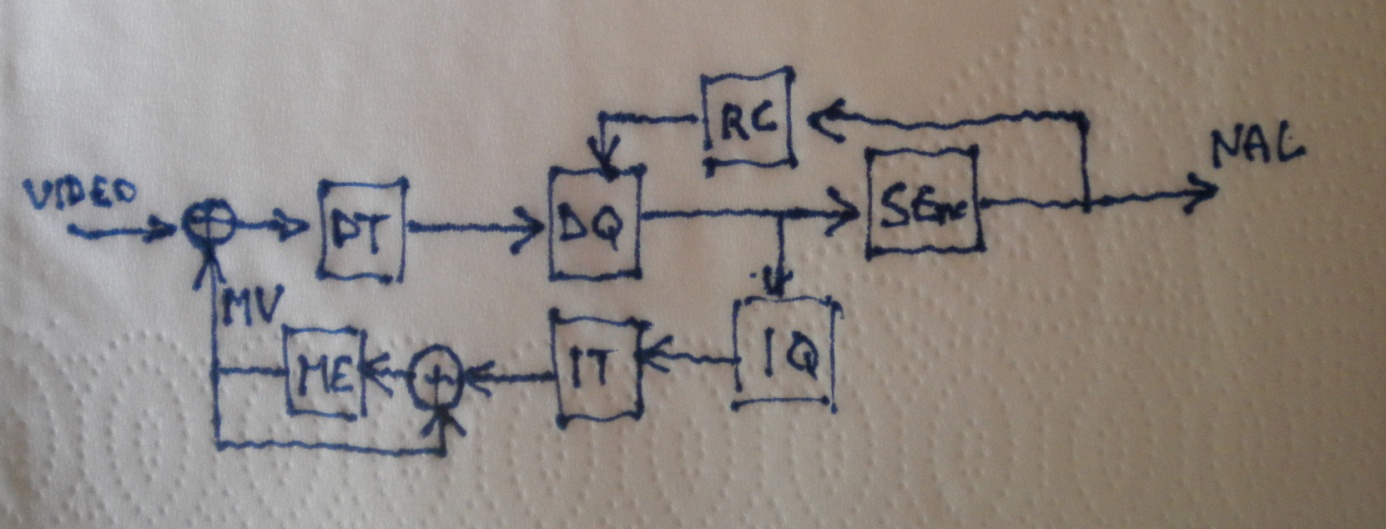


Figure 1: Example of the specification of a video coding algorithm that can be obtained by a video engineer during lunch-time. As can be noticed, it is in the form of "boxes" that represent processing units and arrows that represent flows of data. No control loops of sequential programming languages or abstract objects, classes or methods are used.

# RMC concepts and functionalities

The possibility of providing a modular and unified specification of video and graphic compression standards without the drawbacks of monolithic and sequential reference software packages relies on being able of building “composable” components.  Such components should also be able of encapsulating the specific processing algorithm and explicitly express input and output data streams that are the only form in which they can communicate with each other. Such analysis yielded the concept of "functional unit" (FU - Figure 2) build as a self-contained processing algorithm with specific inputs, outputs, data streams and encapsulated and not shared internal states.

|  |  |
| --- | --- |
| **FU Name** | Algo\_IS\_ZigzagOrAlternateHorizontalVertical\_8x8 |
| **Description** | This module inverts the one-dimensional array of coefficients ordered in zigzag (AC\_PRED\_DIR=0), alternate vertical (AC\_PRED\_DIR=1) or alternate horizontal (AC\_PRED\_DIR=2) scan to 2D raster order. It inputs a list of 64 integer coefficients (one per 8x8 block) and outputs the ordered list of integer according to the value of the token AC\_PRED\_DIR. |
| **Profiles@levels supported** | MPEG-4 SP |
| **Input** | |
| **Name** | **Token** |
| AC\_PRED\_DIR | ACPRED\_DIR token |
| QFS\_AC | AC token |
| **Output** | |
| **Name** | **Token** |
| PQF\_AC | AC token |
| **Package** | |
| package org.sc29.wg11.mpeg4.part2.sp.texture | |

Figure 2: An example of an abstract FU definition taken from the MPEG MTL.

Another important concept was the definition of the granularity of the algorithms contained into the FUs.  This is a very important choice for achieving the objectives that a unified modular specification intends to reach. A decoder in RMC could be viewed as an FU with one input (for example, the binary bitstream) and three outputs (for example, the three YUV pixel streams). However, it is clear that such granularity implies very large FU and would clearly result to be an obstacle to the achievement of the goals of the RMC framework, that is, to define a unified library containing FUs that can be reused and reconfigured for specifying and implementing the different coding standards. In the development of the standard RMC video tool library (ISO/IEC 23002-4) the choice of the granularity level was guided by the attempt of trying to extract the essential characteristics of algorithms as naturally seen from a signal processing point of view of the codec.

For the 3DG library and HEVC extensions, also other considerations such as processing complexity and explicit data or pipeline parallelism have been considered.

Such requirements have naturally lead to what in literature is referred to the concepts of a dataflow network of actors for their component and composability properties [6]. Since dataflow networks exists in several forms, each of them characterized by different models of computation (MoC) and different degree of expressiveness (i.e. the capacity of concisely express different classes of algorithms), the choice taken by RMC was to use the so called “asynchronous data process network with firing”, expressed by a formal dataflow programming language. The choice of an asynchronous MoC is motivated by the capability, that differently from the traditional reference SW, abstracts from time and does not impose any non-necessary ordering or scheduling of the operations than the one strictly necessary of the algorithmic data dependencies. The choice of the class of “data process networks” was motivated by the requirement of being capable of expressing dynamic behaviors very common in video compression standards and that cannot be directly expressed by more restricted MoCs referred to as "static" dataflow models.



Figure 3. Example of RVC-CAL operators expressing dynamic dataflow actors. The formal specification of RVC-CAL dataflow operators is available in Annex B of the ISO/IEC 23001-4 standard.

The use of actors with internal encapsulated states and a set of associated firing rules was motivated by the large expressivity achievable by the dataflow program, and by the use of a formal language that directly capture the dynamism and the expressivity by its native operators. However, beside these interesting properties, a formal language (Figure 3) enables the RMC specification to remain independent from a specific dataflow simulation tool and, particularly interesting for a standard specification, to remain independent from implementation platforms, that conversely would have needed to be associated to the RMC standard framework. In such a dataflow model FU connections need only to be lossless and order preserving.

**Package** org.sc29.wg11.mpeg4.part2.sp.texture;

**actor** Algo\_IS\_ZigzagOrAlternateHorizontalVertical\_8x8 ()

**uint**(**size**=2) AC\_PRED\_DIR, **int**(**size**=SAMPLE\_SZ) QFS

==>

**int**(**size**=SAMPLE\_SZ) PQF :

**int**(**size**=7) Scanmode[3][64] =

[

[

0, 1, 5, 6, 14, 15, 27, 28,

2, 4, 7, 13, 16, 26, 29, 42, //0:63 zigzag

3, 8, 12, 17, 25, 30, 41, 43,

9, 11, 18, 24, 31, 40, 44, 53,

10, 19, 23, 32, 39, 45, 52, 54,

20, 22, 33, 38, 46, 51, 55, 60,

21, 34, 37, 47, 50, 56, 59, 61,

35, 36, 48, 49, 57, 58, 62, 63

],

[

0, 4, 6, 20, 22, 36, 38, 52,

1, 5, 7, 21, 23, 37, 39, 53, //64:127 Alt vert

2, 8, 19, 24, 34, 40, 50, 54,

3, 9, 18, 25, 35, 41, 51, 55,

10, 17, 26, 30, 42, 46, 56, 60,

11, 16, 27, 31, 43, 47, 57, 61,

12, 15, 28, 32, 44, 48, 58, 62,

13, 14, 29, 33, 45, 49, 59, 63

],

[

0, 1, 2, 3, 10, 11, 12, 13,

4, 5, 8, 9, 17, 16, 15, 14, //128:191 Alt Horiz

6, 7, 19, 18, 26, 27, 28, 29,

20, 21, 24, 25, 30, 31, 32, 33,

22, 23, 34, 35, 42, 43, 44, 45,

36, 37, 40, 41, 46, 47, 48, 49,

38, 39, 50, 51, 56, 57, 58, 59,

52, 53, 54, 55, 60, 61, 62, 63

]

];

**action** AC\_PRED\_DIR:[ mode ], QFS:[ qfs ] **repeat** 64 ==>

PQF:[[qfs[Scanmode[mode][i]] : **for** **int** i **in** 0 .. 63]] **repeat** 64

**end**

**end**

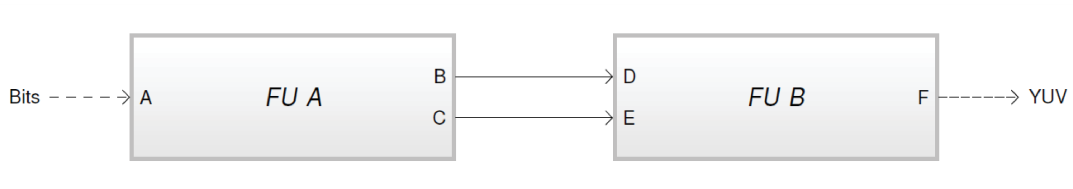
Figure 4: Example of the Functional Unit specification of Algo\_ISZigzagOrAlternateHorizontalVertical\_8x8 FU expressed by using the RVC-CAL language.

It has to be remarked that the adoption of a dataflow model is a significant departure from the traditional model of computation based on imperative sequential specifications. In fact it offers, in addition to the modularity and encapsulation, the new features that provide the possibility of expressing the explicit parallelism of algorithms and the portability to heterogeneous platforms. However, a data-flow specification of a decoder remains a high level abstraction of a decoder implementation, neglecting several implementation details such as buffer size and management, scheduling and timing of the firing that consume and generate the input and output data exchanged among FUs called "tokens", so that such details can be defined according to specific implementation objectives.

For these reasons the RMC standard is structured into two distinct parts: the first is the standard reference for the framework (ISO/IEC 23001-4) in which the definition of the formal dataflow language RVC-CAL, the Functional unit Network Language (FNL) and the associated informative annexes are included, and the second is the part for the video and graphics coding toolbox (ISO/IEC 23002-4) that contains the specification of the FUs that are used to build a video and a graphic decoder Configuration. In addition the standard framework also specifies the decoder description language used to describe a FU network configuration (Figure 4) and the language that specifies bitstream syntax and implicitly the associated parsing process. The standard toolbox includes video and graphics coding FUs [7], [8] and a simulation model with several decoder configurations providing the complete specification of the more used existing standard video and graphics decoders.



Figure 5. Graphic representation of the RVC framework. The normative standard component of the framework are represented in the upper part above the dashed line, the non normative implementation process is represented in the lower part of the picture.



**<Input src=" FU\_A " src -port ="A"/>**

**<Instance id=" FU\_A ">**

**<Class name =" Algo\_Example1 "/>**

**</ Instance >**

**<Instance id=" FU\_B ">**

**<Class name =" Algo\_Example2 "/>**

**</ Instance >**

**<Connection src=" FU\_A " src -port ="B" dst=" FU\_B " dst -port ="D"/>**

**<Connection src=" FU\_A " src -port ="C" dst=" FU\_B " dst -port ="E"/>**

**<Output src=" FU\_B " src -port ="F"/>**

Figure 6: Example of an MPEG RMC network specification composed of two Functional Units "A" and "B" and the relative network description using the FNL XML dialect specified in Annex A of ISO/IEC 23001-4.

# RMC supporting tools

The development of multimedia applications is a time-consuming and error-prone task with an ever-increasing complexity of the algorithm over the years. The development of multimedia applications is even getting harder with the emergence of parallel platforms. Consequently, the need for efficient development methods and tools is becoming increasingly important to meet the time-to-market requirement:

*Assisted writing of the applications:* The development of applications is made easier with an Integrated Development Environment.

*Easy validation of the code:* A fast functional verification is made possible using an integrated simulator.

*Develop once, run everywhere:* The embedded trans-compiler (Orcc) is able to generate both hardware and software code from a single description that can be executed on large panel of platform thanks to the availability of dedicated runtime libraries.

Orcc (short for Open Rvc-Cal Compiler) is an integrated tool for the development, simulation and code generation of RMC specifications. It is delivered with an entire Eclipse-based Integrated Development Environment (IDE), making user friendly the development of RMC-based applications. This environment is composed of two dedicated editors handling both actor programming and network designs:

1. At the beginning, an intuitive graph editor that enables fast and easy building of the actor network using visual programming is used. A few mouse clicks are sufficient to create a node and assign it to an existing component from the project, or to create an edge that represents the communication channels between two nodes. The editor also supports hierarchical representations, assigning a whole subnetwork to a graph node, and hierarchical navigation, opening a subnetwork with a simple click on a graph node.
2. In a second phase when the datflow network is built, a full-blown RVC-CAL editor with advanced features, such as syntax coloring, content assist and code validation for supporting the development of the actors. Orcc implements all features expected for a modern and efficient Domain-Specific Language editor. The development environment is able to parse the actors and build the intermediate representation on-the-fly, in a incremental fashion, allowing fast simulation and compilation.

Additionally to the editors functionality, Orcc provides a complete Java-based simulator which enable the developers to quickly test their applications without taking in consideration low-level details relative to the target platform, but only the correctness of the algorithm specification. The simulator can be launched directly from eclipse to execute any RVC-CAL application. Indeed, the simulator simply interprets the intermediate representation of networks and actors, but it is also able to perform all basic interactions required to perform a functional validation, such as displaying text, images or videos to the screen.

Orcc is delivered with a compiler that allows to write a single description of the application to target a variety of executing platforms such as General-Purpose Processors, FPGAs, embedded processors and so on.

**Efficient dataflow trans-compiler** The Orcc's compiler is able to translate a unique high-level dataflow program, written in RVC-CAL, into an equivalent description in both hardware and software languages for various platforms. A specific compiler back-end has been written to tackle each configuration case:

1. *software* back-ends that generates C/C++ programs with multi-core abilities usable on most of the programmable processors.
2. *hardware* back-ends using well-known High-Level Synthesis tools to generate synthesizable HDL code for FPGA and ASIC implementations.

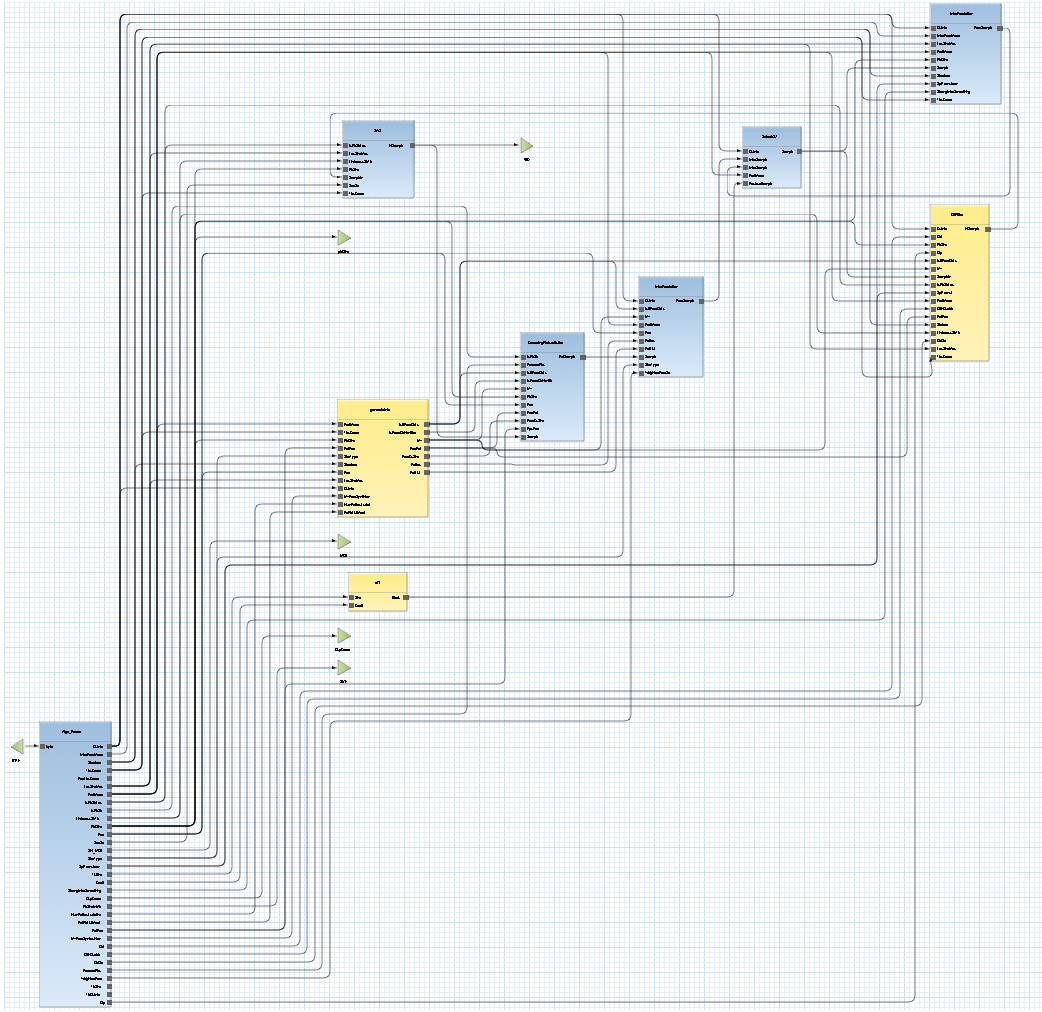


Figure 7: Example of the HEVC hierarchical dataflow diagram (top view) as available in the grpahical interface of Orcc.

The Orcc compilation framework for RMC-specifications is also completed by other tools for performance analysis, design space exploration and HW generation and optimization to build a complete system design environment for heterogeneous systems. A graphic representation of the system design flow of an RMC specification is provided in Figure 8. In such picture the functionality of the design flow are labelled with their dependencies and mapped into the corresponding tool environment. Orcc provides dataflow program development functionalities and simulation capabilities (top section of the design flow) and SW generation (right bottom part of the flow). The main tools that complete Orcc to build a complete system design are: Turnus, a design space environment integrated as Plug-in of Orcc Eclipse environment and Xronos an HDL synthesis tool (left bottom part of the design flow). Both of them are available as open source tools.

# 

Figure 8: Example of a design flow for the implementation on heterogeneous platforms of an RMC specification and associated supporting tools.

**Turnus design exploration Environment.** The first step of design space exploration is a functional (i.e. high-level and platform-independent) profiled simulation [11, 12]. During this stage, an exhaustive analysis of the design under study is performed leading to the definition of its basic structure and complexity. This initial analysis enables multidimensional design spaces explorations and helps in finding bottlenecks and potentially unexploited parallelism. In literature several different methods have been proposed to measure the complexity of an algorithm in terms of execution of its building blocks. Two main axes are typically defined: (a) the computational load (b) the data-transfers and storage load [13]. In this direction, TURNUS implements a CAL dataflow profiler based on the Open RVC-CAL Compiler (Orcc) simulator [14]. This is an interpreter of an Intermediate Representation (IR) code used in the initial design stages to validate the algorithmic and functional behaviour of a RMC CAL dataflow specification. TURNUS adds profiling information on top of the ORCC simulator: for each executed action both (a) the computational load and (b) the data-transfers and storage load are evaluated. The computational load is measured in terms of executed operators and control statements (i.e. comparison, logical, arithmetic and data movement instructions). The data-transfers and storage load are evaluated in terms of state variables utilization, input/output port utilization, buffers utilization and tokens production/consumption. Moreover, its is possible to extract the causation trace for each run of the simulation.



Figure 9: The TURNUS co-exploration environment. The causation trace graph is used as main tool for exploring the design space. The main features are: fast performance estimation, bottlenecks evaluation, buffer size minimization and optimization, multi-clock domain partitioning.

**References**

1. MPEG homepage, <http://mpeg.chiariglione.org/>
2. T. Wiegand, G.J. Sullivan, G. Bjontegaard, A. Luthra, "Overview of the H.264/AVC video coding standard," Circuits and Systems for Video Technology, IEEE Transactions on, vol.13, no.7, pp.560—576, 2003.
3. G.J. Sullivan, J. Ohm, W.-J. Han, T. Wiegand, "Overview of the High Efficiency Video Coding (HEVC) Standard," *Circuits and Systems for Video Technology, IEEE Transactions on* , vol.22, no.12, pp.1649,1668, Dec. 2012
4. G.-G. Lee, Y.-K. Chen, M. Mattavelli, E.S. Jang, "Algorithm/Architecture Co-Exploration of Visual Computing on Emergent Platforms: Overview and Future Prospects," Circuits and Systems for Video Technology, IEEE Transactions on , vol.19, no.11, pp.1576—1587, Nov. 2009
5. M. Mattavelli, I. Amer, M. Raulet, The Reconfigurable Video Coding Standard [Standards in a Nutshell], Signal Processing Magazine, IEEE 27 (3) (2010) 159 –167.
6. J. Eker, J. Janneck, A Structured Description Of Dataflow Actors And Its Application (May 2003).
7. J. Gorin, M. Raulet, F. Prêteux, “MPEG Reconfigurable Video Coding: from specification to a reconfigurable implementation", Signal processing Image Communication, 2013.
8. C. Tulvan, M. Preda, “3D Graphics Coding in a reconfigurable environment”, Signal processing Image Communication, 2013.
9. https://github.com/turnus
10. https://github.com/orcc/xronos
11. J. Janneck, I.D. Miller, and D.B. Parlour, “Profiling dataflow programs,” in Proceedings of the IEEE International Conference on Multimedia and Expo, 2008, pp. 1065–1068.
12. S. Casale-Brunet, M. Mattavelli, A. Elguindy, E. Bezati, R. Thavot, G. Roquier, and J. Janneck, “Methods to explore design space for MPEG RMC codec specifications,” Journal of Signal Processing Image Communication, Elsevier, 2013.
13. M. Ravasi and M. Mattavelli, “High-abstraction level complexity analysis and memory architecture simulations of multimedia algorithms,”Circuits and Systems for Video Technology, IEEE Transactions on, vol. 15, no. 5, pp. 673 – 684, may 2005.
14. “Open RVC-CAL Compiler (Orcc),” ”http://orcc.

sourceforge.net/”.