Toward a Graphical Tool for Image and Video Processing Embedded Systems Design

Noureddine ZHAR, Mohamed AIT ALI, Mohsine ELEULDJ
Laboratoire Système d'Information et Répartition
Ecole Mohammadia d'Ingénieurs
Rabat, Morocco
{Zhar, aitali, eleuldj}@emi.ac.ma,

***Abstract*—In this paper we identify the requirements of a design tool for the implementation of image and video processing algorithms in hardware platforms such as FPGA or ASIC. We discussthe advantages and weaknesses of some existing design languages. Finally, we propose our solution, in compliancewith specified requirements, which intends to bypass the shortcomings of existing languages by providing a high-level of abstraction through two kinds of diagrams; structural diagram and filter edition diagram. It also allows a formal verification and automatic code generation for an ASIC or a FPGA implementation.**

Index Terms—image processing; real-time; embedded system; design;

I. INTRODUCTION

The growth of image resolution, the nature and diversity of associated applications require the implementation of complex image processing algorithms. Artificial vision is widely involved in many industrial, medicinal, telecommunication and defense applications. A sequential implementation of these algorithms has quickly shown its shortcomings, particularly for real-time applications. This is due to the importantquantity of data to be processed and the severe temporal constraints that characterize such systems.

A hardware implementation thatallowsparallelizingsome parts of the processing and reducingthe execution time and the consumption of resources seems to be the best solution to meet time and budget requirements[1]. Thanks to its ability to permit parallelism and to support different modes of operation on a single hardware substrate FPGA is considered as the most suitable platform for digital signal processing, including image processing [1].

In this case, the main difficulty is to convert the sequential algorithmstoparallel implementations. This task isa real challenge for software designerwho have no hardware knowledge. Moreover, video processing applications are often subject totime constraints. Input devices provide different types of data, compressed with various compression techniques, in a specific frame rate. In such a case, the processing has to be done in accordance with the data type and rate. These constraints are associated with memory bandwidth limitation and resources access conflicts.To manage this situation, the designer should be able to define clearly the system behavior by describing its functional structure and showing how data will be accessed by different functional blocks.

A high-level design tool for real-time image and video processing in hardware platforms should be a practical solution to overcome these difficulties and reduce development time. This solution should meet requirements associated with embedded and real-time systems constraints.

This paper intends to pinpoint requirements for a real-time image and video processing in embedded systems design tool. For this purpose, we present advantages and shortcomings of some existing design languages. We describe here a panel of desirable characteristics of such a tool. Lastly, we propose VIP DESIGN, a graphical language for image and video processing embedded systems design, which aims at providing solution to identified weaknesses.

II. CRITICISM OF AVAILABLE LANGUAGES

Many languages and tools are currently available to fulfill hardware systems designer needs. Some of them try to describe hardware architecture in a low-level of abstraction. Some others were created to raise the abstraction level by adding more extensions to C-like languages. However, this kind of languages remains difficult to be mastered by a large community of software developers because of integration ofsome hardware-specific paradigms. Others prior works suggest visual environments to design hardware systems. These environments increase significantly comprehensibleness of the system architecture but remain linked to hardware specificities.

A. Hardware Description Languages

Hardware Description Languages (HDL) were developed to remedy to the problematic related to the growth of electronic circuits complexity that can no longer be described by schematics. HDLs are software-like languages which intend to describe organization of components on a digital circuit. The most known languages in this family are Verilog[2] and VHDL[3].VHDL was designed to be similar to ADA[4] programming language[4] and is strongly-typed. Verilog has a style close to C programming language [5] using a preprocessor. Both of them offer possibility to simulate architectures before their fabrication.

In the VHDL case, "entity" is the representation of an electronic component. The declaration of the "entity" allows

978-1-4673-2679-7/12/$31.00 ©2012 IEEE

defining input and output ports. The component behavior is specified by an architecture bodyto design a basic operation, the designer needs as a matter of fact to define the entity comprising ports and then its behavior.

HDLs contributedconsiderably to raise the abstraction level in comparison with schematics.However, they remain maladjusted to implement complex algorithms such as those of image and video processing. In spite of their flexibility and their powerfulness in hardware describing, they are difficult to program and they provide no image processing specific operations.

B. C-like languages

In order to cope with the increasing complexity of designing digital circuits with HDLs, some solutions attempt to raise the abstraction level. In this context, some languages, based on standard C or C++, try to reduce the development cycle for hardware systems. SystemVerilog[6], SystemC[7] and Handel-C[8] are C-like languages providing additional libraries and extensions to include ability to describe hardware aspects and to allow parallel programming. The main aim of all of these languages is to increase productivity and to make hardware design more accessible to software designer.They allow the designer to break away from the physical structure of the system and focus on its behavior. Closed to the C language, widely used, these languages are easily mastered. These proprieties reduce greatly the conception and development time of hardware systems. However, this gain in terms of time and simplicity is combined with losses in performance due to the automatic translation of Handel-C code in VHDL or EDIF that requires a manual refinement to optimize the algorithm implementation on hardware. Also, these languages don't provide specific features for image and video processing systems design and they don't allow representing concurrence aspects.When working in a team project, using textual languages to design complex architecture don't facilitate communication between the team members.

C. Visual tools for hardware design

In order to bypass limitations of textual languages, there are moves to create visual tools (languages and environments) to raise the abstraction level, increase solutions visibility and reduce time to market. Some available languages are based on UML[9] notation. Others are completely designed to fit hardware design requirements,like Catapult-C[10], and specially image processing needs.

1) UML profile for hardware design

UML profiledefines a "Domain Specific Modeling Language" without contradicting the semantics of UML by adding concepts relating to a particular field, by changing the representation of these concepts, defining constraints applied to the associations between these concepts and by adding constraints on the use of certain concepts or not depending on the context and the identification of semantic variation points[9].

SysML [11] is an UML profile for modeling systems. It incorporates additional concepts adapted to the design of embedded and real-time systems. These concepts, which are sometimes inadequate to the spirit of UML and suitable to the representation of hardware systems, remain inapplicable in the case of a complete automated development flow [12].

Other works have attempted to use the standard notation of UML, particularly those of the activity, composition and deployment diagrams, to model the behavior, composition and functional block for hardware systems [12]. The model thus produced can be subject to transformation and generation rules to obtain a descriptive code. This language does not provide sufficient flexibility to define other specific concepts such as data types.

2) VERTIPH

VERTIPH [13] (Visual Environment for Real-Time Image Processing in Hardware) is a design environment for image processing applications for real-time systems. It offers three views covering different aspects of an image processing system: An architectural view, a computational view and a scheduling and resource sharing view. This language attempts to meet the specificity of image processing applications to be implemented on FPGAs in terms of data types, reuse of primitive functions specific to this area and the graphical representation of competitive and sequential execution of different functional blocks of the designed system. However, although it is consideredas a high-level design environment which doesn'trequire hardware knowledge, this tool requires a perfect mastery of resource sharing concepts and a good ability to handle and define data types at very low level. This specific knowledge makes the use of this tool inaccessible to developers' community unaccustomed to such concepts. VERTIPH doesn'tintegrate verification and validation of the designed model before the implementation phase, considered expensive in terms of time and cost.

III. REQUIREMENTS OF A GRAPHICAL LANGUAGE FOR IMAGE AND VIDEO PROCESSING IN EMBEDDED SYSTEMS

The study of a large set of available languages andtools used in hardware system design and image/video processing techniques leads to define requirements to be observed when developing a graphical language adapted to this field.

A. To focus on functional structure

In most cases, an image processing specialist has no knowledge about hardware systems. They develop algorithms without consideration to how it will be implemented in circuits. A high-level language should allow to the designer to focus exclusively on functional structure of the solution.

B. To integrate specificities for image and video processing field

Most available languages and tools for hardware systems designseek a general purpose and provide no specific operations for image/video processing needs.An image/video processing applications manipulate several specific data-types (image, pixel, row, stream…). The whole of data-flow in such application is submitted to a succession of operations called *filter*.A specific high-level language for image/video processing in hardware systems ought to permit manipulation of operations and data type relevant to this area in order to reduce the gap between design and implementation levels.

C. To describesystem behavior

A specific high-level language should provide tools to represent all of the states and transition events of the system. It could also permit to control the execution flow by a set of assignment and conditional statements. Also, the designer should be able to graphically express logic and arithmetic operations.

D. To allow reuse

A large community of developers is working every day to create new algorithms and to develop new techniques for image processing systems. A designer is always looking to reuse what has already been developed by himself or by another designer in order to reduce development time. A design tool of image processing systems must offer to its user the ability to reuse the already developed algorithms and techniques.

Also, image and video processing applications use often some common routines. Windows filters, data buffering and lookup table appear in almost image and video processing algorithms[14]. A graphical language for image and video processing purpose should include such routines as primitives.

E. To manage resource access

Themaingoalofa hardware implementationofimage processingalgorithmsis toparallelizesome orallof the processing. In this case, the designer has to manageconflicts over accessto shared resources. Thisneed arises especially for areal-time processingof a video streamwhere severalcomponentsof the systemwill try toread or writein the samememory space.

A graphical languagefor designing thesesystemsmustenable the managementofthis kind of conflictsby providing toolsfora clear representationof shared resources usage.

F. To express time constraints

The increasing complexity of algorithms and the large volume of data handled in the image processing present a serious challenge which requires a considerable computation time to accelerate this complex process. Parallelizing tasks cannot be designed without an associated model of time. In a real-time context, the adherence to these constraints is as important as the accuracy of the obtained result. So, it is essential to associate each operation to one or more time constraints.

G. To represent clearly pipelined operation

Parallelization of tasks involves increased knowledge of their schedule to determine what will be executed in a competitive manner and whatwill be executed sequentially. A hardware systems designer looks for a tool to represent clearly this aspect. Textual programming languages, even when providing statements to support parallelization of tasks, cannot allow a clear representation of tasks scheduling. A graphical tool for designing hardware systems should offer to the designer the ability to define unambiguously the scheduling of various operations that the system has to perform. This will also help to avoid conflicts related to resources access.

H. To allow formal verification

Obtaining a productive model goes through rigorous verification of the validity of this model compared to its defined specifications. A graphical language for designing hardware systems should allow the verification and the validation of the producedmodels. This assumes that the semantics of this language is well defined and does not cause confusion. Models produced can be injected into a formal verification process to ensure compliance with constraints.

I. To be easy to learn and to use

Learning and handling time is an important criterion for choosing a design tool. Reducing this time may be obtained by the simplicity, intuitiveness and the availability of documentation.

The ability to document produced models contributes to make them Useful, to serve as an effective means of communication and to ease maintenance and improvement of the designed solution.

IV. VIP DESIGN

In order to dispel difficulties associated with the use of textual languages and provide answers to the limits of visual languages described in the previous section, we suggesta graphical language for image and video processing in embedded systems design called VIP DESIGN.

A. Appraoch

VIP DESIGN proposes an approach based on a graphical representation of structural and behavioral aspects of designed system while abstracting details about the physical architecture. This is possible through two types of diagrams. Embedded and real-time image processing system has input interfaces for data acquisition and an output interface to deliverits returned result. The processing can be described as an arrangement of several sequential or competitive filters which communicate through channels or shared memory.

The first diagram is called *Structural Diagram*. Itspurpose is to model the system structure through a hierarchical composition of all necessary functions. A second diagram can describe each basic element of this hierarchical structure. This diagram called*Filter Editing Diagram* details the processing functions through a description of elementary operations and their scheduling.

B. Structural Diagram

Structural diagrammetamodel is based on this set of rules:

- An image processing system consists of a set of input interfaces, a set of output interfaces, a serie of filters applied to a data stream, clocks and communication channels;
- One or several filters run at a specified clock rate;
- Each filter consists of an input stream of data, an output, one or more parameters and a computing entity;
- A computing entity contains instruction blocks executed in parallel or sequentially,

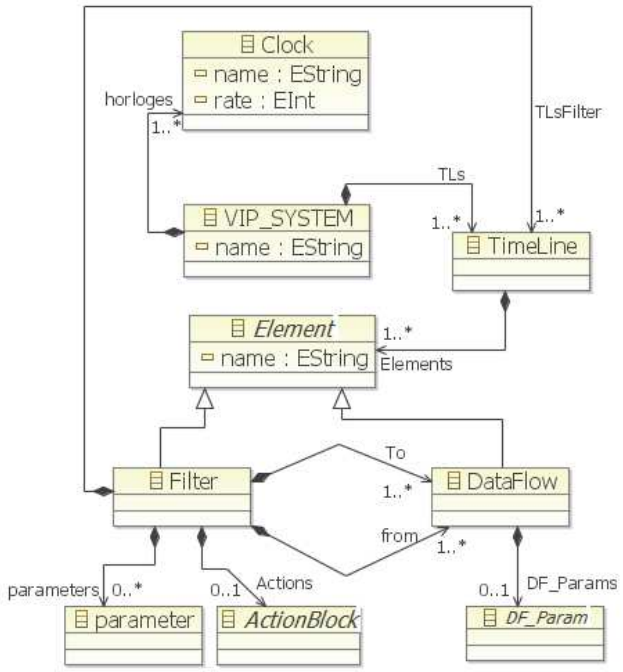

Figure 1. Partial view of VIP DESIGN metamodel

C. Filter Editing Diagram

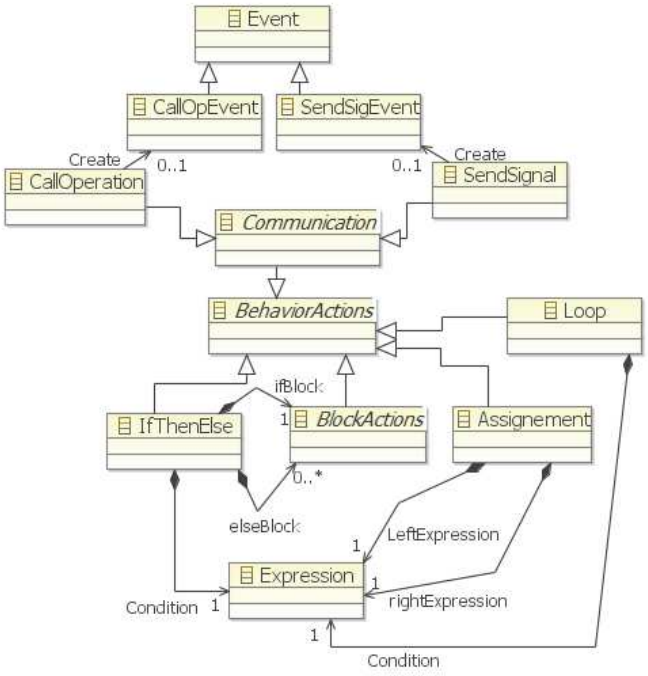

Figure 2. Package "ACTION"

The behavior of an embedded and real-time image processing system can be described by defining three basic aspects:

- An accurate expression of structural (conditional branches, loops), arithmetic and/or logic operations executed by the system;
- Scheduling of these operations;
- Defining rules to manage competitive access to shared resources;

1) Expression of filter operation : A clear expression of the system operations requires an action language using a concrete syntax with a sufficient level of accuracy to enable an unambiguous code generation. For this need, we used the package "Action" of the intermediate modeling language COCODEL [15].

2) Scheduling : The increasing complexity of algorithms and the large volume of data handled in the image processing present a serious challenge which requires a considerable computation time. Parallelizing tasks represents an effective solution to obtain more efficient in execution speed.

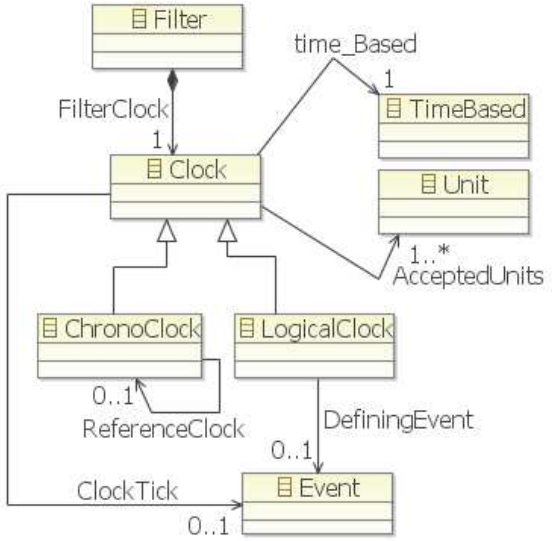

Figure 3. Partial view from VIP DESIGN time model

This parallelism can not be designed without an associated model of time.Therefore, VIP DESIGN language manages different real-time constraints related to image processing via a TIME package facilitating the addition of temporal information to model elements.This package allows to manipulate the values of temporal parameters and markup language elements with temporal information used later by tools for simulation, performance analysis, verification, validation and analysis of schedulability. In fact the components of our language are related to one or more clocks which gives the possibility to use multiple temporal repositories in the same VIP DESIGN model and divides time into a succession of discrete instants for modeling parallel processing, concurrency, and support the design of distributed and multi-clocks electronic systems.

D. *Formal analysis framework*

We integrated real-time constraints modeling related to image processing in the TIME package. To address these constraints, formal techniques have gained much attention since they provide fundamental techniques to analyze,to validate and to transformsystems in a provably sound way. For that reason, we provide a verification framework to ensure the respect of time constraints at model level.

Our verification framework is based on an existing model checking, named CADP-toolbox[16]. We opted for model checking rather than theorem proving because of possibility to automatically check behavioral and timed properties. The question that we are answering is: "once we model our image processing algorithm for FPGA implementation, how can we check the respect of behavioral and timed requirements before going any further and generating the implementation code?"

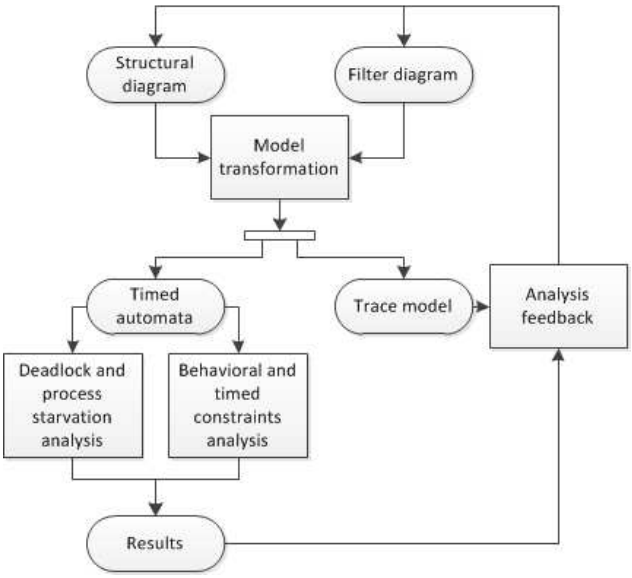

Figure 4. Formal analysis process overview

To answer this question, we integrated a formal verification framework into our design tool. This framework takes as input both of the structural and the filter editing diagrams, and produces as a result a formal model expressed in the form of a set of timed automata. We defined an ad-hoc domain-specified transformation language in terms of Ecore metamodel and define a Model-to-Model transformation chain. From the structural diagram, our verification framework generates a timed automaton that represents the flow of data through computing entities to check the absence of deadlocks and process starvations. From Filter Editing Diagram, we generate a timed automaton to check the respect of behavioral properties according to structural and arithmetic and/or logic operations. Model transformations used to generate formal artifacts from Structural and Filter Editing diagrams generate a trace model for each transformation. The trace model is used later to trace back verification result to give a diagnosis support for the designer. Figure 4 shows an overview of the analysis process of our verification framework.

V. APPLICATION : BLOBS DETECTION IN A VIDEO STREAM

In this section, we present briefly a blob detection algorithm to illustrate how we can easily implement a hardware implementation of an image processing algorithm.

This application gets in input a grayscale video stream at a constant rate (23 images /seconde). It furnishes in output an array containing blobs sizes and coordinates.The first image is stored in a RAM block as reference image. Each new image is subtracted from a stored reference image. A pretreatment routine eliminate residual noise by applying successively a blur 5×5and a threshold filters. After this pretreatment we obtain a mono-bit image on which we apply our blob detection algorithm. Figure 5 shows the processing sequence implemented entirely on FPGA.

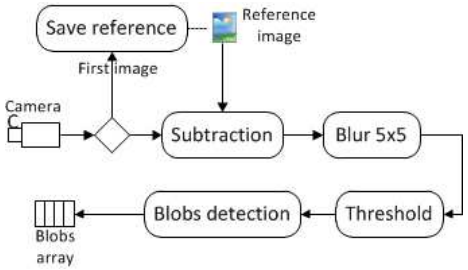

Figure 5. Synoptic solution scheme

To implement the solution described above, we define a system (SYSTEM). This system has an input stream *Image* and a Boolean one called *first* injected into a conditional block to redirect stream flow to the initialization phase or to the processing one. The system provides a list of spots (Blobs list) in output. This processing will run while the video stream is available in input. Each stage of processing can be parallelized internally while the computational tasks are independent. Diagrams are produced using a graphical tools created with eclipse GMF [17]. We use ACCELEO [18] to define transformation and generation rules to produce Handel-C code.

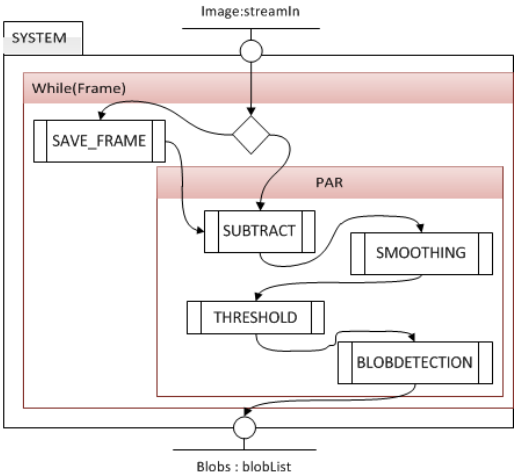

Figure 6. Solution designed with VIP DESIGN

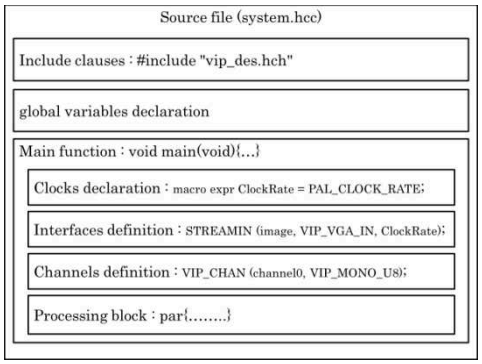

Figure 7. Main source file pattern

TABLE I. FPGA RESOURCE CONSUMPTION BY FILTERS

FPGA resource consumption	Filters			
	SUBTRA.	*THRESH.*	*SMOOTH.*	*BLOBS.*
CLBs blocks	326	294	652	1830
Block RAM's	1	1	2	4

Generated code can be easily handled by software developers thanks to its similarity with high level programming languages. However, the final solution implemented on FPGA doesn't reach a satisfactory level of optimization, as shown in table I, particularly in terms of occupation rate.

VI. CONCLUSION

Several programming languages are used to describe hardware systems. However, the increasing complexity of algorithms and the need to reduce development time motivate the rise of graphical tools to raise the abstraction level and overcome hardware concepts difficulties. The study of some available tools allows us to pinpoint their weaknesses and advantages in order to identify a range of suitable characteristics to develop a specific-domain design tool.

We suggested VIP DESIGN, a new graphical design tool for embedded and real-time image and video processing systems. VIP DESIGN approach is based on the description of the system through the scheduling of several parallel or competitive filters communicating through channels and running at a speed set by one or different clocks. This is possible by using two kinds of diagram which allow the designer to define the functional structure and the internal behavior of the system. The structural diagram describes the functional hierarchy of the system. Filter editing diagram is used to describe the internal behavior of each filter using an action language. Time constraints are expressed by integrating clauses adapted to real-time system design.However, VIP DESIGN doesn't cover the entire development flow. To this end, the code generated automatically from the produced model must be submitted to a verification and simulation process before its implementation. Actually, we use a tool provided by Celoxica[19].

REFERENCES

[1] W. J. MacLean, «An Evaluation of the Suitability of FPGAs for Embedded Vision Systems,» chez *Computer vision and pattern recognition*, San Diego, CA, 2005.

[2] IEEE, Verilog HDL quick reference manual - standard IEEE1364-2005, IEEE, 2005.

[3] I. o. E. a. E. Engineers, IEEE Standard VHDL Language Reference Manual, IEEE, 2008.

[4] Ada-Europe, ADA reference manual, Language and Standard Libraries, 2006.

[5] Brian W. Kernighan, Dennis M. Ritchie., The C programming language, NJ, USA: Prentice Hall Press, 1998.

[6] Accellera, IEEE Standard for System Verilog: Unified Hardware Design, Specification and Verification Language, 2005: IEEE.

[7] Open Accellera Initiatives, OPEN SYSTEMC LANGUAGE REFERENCE MANUAL, IEEE, 2012.

[8] RG, Handel-C Language Reference Manual, Celoxica Limited, 2005.

[9] OMG, OMG Unified Modeling LanguageTM (OMG UML), Superstructure, Version 2.3, 2010.

[10] MentorGraphics, "Catapult C Synthesis datasheet," Mentor Graphics Corporation, 2010.

[11] S. Partners, "SysML Specification v. 1.0a," 2005. [Online]. Available: http://www.sysml.org.

[12] Tim Schattkowsky, Jan Hendrik Hausmann, Gregor Engels, «Using UML Activities for System-On-Chip design and synthesis,» *Springer,* 2006.

[13] C.T.Johnston, D.G.Bailey, P.Lyons, «A Visual Environment for Real-Time Image Processing in Hardware (VERTIPH),» *EURASIP Journal on Embedded Systems,* p. 1–8, 2006.

[14] JOHNSTON C.A, GRIBBON K.T, BAILY D.G, «Implementing Image Processing Algorithms on FPGAs,» chez *11th Electronic New Zealand Conference*, Palmerston North, 15 november 2004.

[15] M.AIT ALI, M.ELEULDJ, «An intermediate modelisation Language for embedded system developemnt chain COCODEL : Un Langage intermédiaire de modélisation pour une chaine de développement des systèmes embarqués COCODEL,» chez *JODIC'2012*, Rabat, 2012.

[16] J.-C. Fernandez, G. Hubert , A. Kerbrat, L. Mounier , R. Mateescu et M. Sighireanu, «CADP : A Protocol Validation and Verification Toolbox,» chez *CAV '96 : Proceedings of the 8th International Confe- rence on Computer Aided Verification*, London, UK, 2006.

[17] E. Foundation, «Graphical Modeling Framework,» The Eclipse Foundation, [En ligne]. Available: http://www.eclipse.org/modeling/gmp/.

[18] E. Foundation, «Acceleo,» Eclipse Foundation, [En ligne]. Available: http://www.eclipse.org/acceleo/.

[19] Pixel Streams manual, Celoxica, 2001.

[20] Charest, L., Aboulhamid, E.M., «A VHDL/SystemC Comparison in Handling Design Reuse,» 2008.

[21] OMG, UML Action Semantics, November 2001.

[22] OMG, «Mof qvt Technical report,» Object Management Group, 2005.

[23] OMG, Action Semantics for the UML, Request For Proposal, OMG Document, 2003.

[24] Venkateshwar,R., Patil,P., Naveen, A., Muthukumar,V., «Implementation and Evaluation of Image Processing Algorithms on Reconfigurable Architecture using C-based Hardware Descriptive Language,» *International Journal of Theoretical and Applied Computer Sciences,* 2006.