Framework For Efficient Cosimulation And Fast Prototyping on Multi-Components With AAA Methodology: LAR Codec Study Case

Erwan Flécher, Mickaël Raulet, Ghislain Roquier, Marie Babel, Olivier Déforges

FRAMEWORK FOR EFFICIENT COSIMULATION AND FAST PROTOTYPING ON MULTI-COMPONENTS WITH AAA METHODOLOGY: LAR CODEC STUDY CASE

Erwan Flécher, Mickaël Raulet, Ghislain Roquier, Marie Babel, Olivier Déforges

IETR CNRS UMR 6164/Image and Remote Sensing Group, INSA of Rennes
20 avenue des buttes de coësmes, 35043 Rennes cedex, France
Email: {eflecher, groquier}@ens.insa-rennes.fr {mraulet, mbabel, odeforge}@insa-rennes.fr

ABSTRACT

Real-time signal and image applications have significant time constraints involving the use of several powerful calculation units. Programmable multi-component architectures have proven to be a suitable solution combining flexibility and computation power. This paper presents a methodology for the fast design of signal and image processing applications. In a unified framework, application modeling, cosimulation and fast implementation onto parallel heterogeneous architectures are enabled and help to reduce time-to-market. Moreover, automatic code generation provides a high abstraction level for users. Finally, the worthwhile nature of Matlab/C language cosimulation is illustrated on a still image codec named LAR.

1. INTRODUCTION

New embedded multimedia systems require more and more computation power. They are increasingly complex to design yet time-to-market is being constantly reduced. Specific dedicated circuits are not compatible with short design timelines or future capacity adjustments. An alternative is provided by adding several software components (DSP, ARM) or hardware components (FPGA). [1] presents a Matlab/C/VHDL based codesign to enable specification, cosimulation and cosynthesis of the system. First, the system is described with multi-language specification and the functional model is developed. Next, language and architecture models are gradually refined and the system is cosimulated for early validation. According to [1], system level specification, cosimultation and prototyping on multi-components are based on four concepts: concurrent execution or parallel computation for a given granularity level, hierarchy of the algorithm description, communication and synchronization between processors. In [2], two final concepts are studied in a multi-language system composed of Matlab/JAVA processes. The developed framework aims to produce distributed applications for heterogeneous Clusters of Workstations (COW). The Matlab application is partitioned and distributed on workstations which are synchronized with Control Flow Graph (CFG) developed in JAVA language. The speed of a fractal-based image processing application is increased using the advantage of multi-languages with coarse-grain distributed parallel computing.

This paper presents a methodology for fast development and implementation of distributed signal and image processing applications on multi-components. Providing reduced time-to-market, *application modeling*, *cosimulation* and *fast prototyping* on heterogeneous architecture are supported in a unified framework. The cosimulation presented here is a concurrent execution on monoprocessor of different models based on heterogeneous languages. Difficulty with cosimulation occurs during interfacing between processes but it can easily be solved given that cosimulation is a case of marginal prototyping on specific heterogeneous architecture. This approach is possible thanks to the abstraction power of the prototyping methodology that initially provides executives in a generic language. Based on the SynDEx tool, our unified framework for rapid application modeling and prototyping methodology is suitable for transformation-oriented systems and heterogeneous multi-component architectures. SynDEx automatically generates synchronized distributed executives in an intermediate generic language. These generic executives have to be translated to be compliant with the processor type or communication medium type so that they automatically become interpretable or compilable codes. In this article, we will focus on this mechanism based on the concept of SynDEx kernels, and detail new developed kernels enabling automatic code generation for cosimulation and prototyping. This paper is organized as follows: section 2 introduces the AAA methodology and SynDEx tool. The executive generation, kernels for multi-language modeling and Matlab/C cosimulation are described in Section 3. Section 4 presents a global methodology for application modeling, cosimulation and prototyping in a unified framework. Section 5 illustrates a methodology with a still image codec named LAR. Finally, our conclusion and future work are described in section 6.

2. AAA FAST PROTOTYPING METHODOLOGY

The aim of the AAA methodology is to find the best match (or adequation) between an algorithm specifying the application and a multi-component architecture. This methodology is based on a graph theory and is used to model the software application and the hardware architecture. Both the software and the hardware are described by distinct graphs. AAA methodology transforms the two graphs thanks to the graph transformations in order to find an optimized implementation that satisfies real-time and memory constraints. As soon as an optimized implementation is determined, a generic script is automatically generated for each processor. Figure 1 illustrates inputs/outputs of the matching step that is included in the global sequential state diagram of our framework.

2.1 Adequation Algorithm Architecture (AAA)

The application algorithm is modeled by a Data Flow Graph (DFG) which is an oriented hyper-graph. This means that data dependency demands algorithm operations on multi-components. Each vertex corresponds to an algorithm operation and each edge represents a data transfer between operations. Thus DFG shows the potential parallelism of an

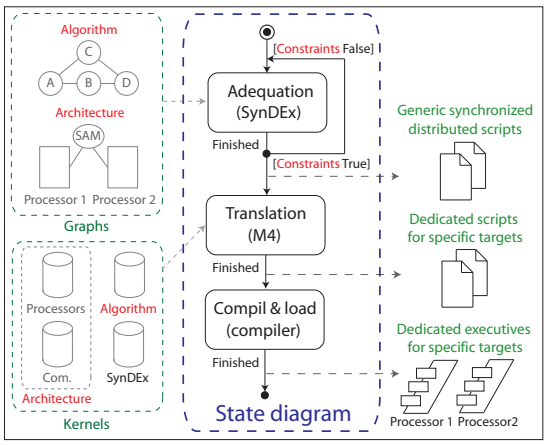


Figure 1: State diagram of the framework based on SynDEx. The simple architecture consists of two processors inter-connnected with a SAM.

algorithm [3].
The architecture is modeled by non-oriented hypergraph in which each vertex is a processor (component) and each hyper-edge represents a communication medium. In this model, a processor consists of one operator and as many communicators as there are connected media. An operator executes operations which are a part of the algorithm and a communicator executes a communication operation when a data transfer is required. In this way, multi-component architecture may be represented by a network of Finite State Machines (FSM) interconnected with communication media (FIFO, shared memories etc.).
Once an optimized implementation has been determined, generic executives are automatically generated for each processor. They consist of a list of macro-instructions which specifies memory allocation, communication sequence(s) and one computation sequence. Macro-executives are generics because they are independent of the programming language. The executives or scripts are transformed later into the appropriate language specified by the different target operators (Matlab/C for cosimulation, C or assembler for DSP, VHDL for FPGA etc.). This is the purpose of section 3.

2.2 SynDEx

SynDEx (Synchronized Distributed Executive) is a system-level CAD software principally designed at the INRIA Rocquencourt Research Unit (France) in association with our image processing lab. This tool supports the AAA methodology for both rapid prototyping and optimized implementation of distributed real-time embedded applications onto multi-component architectures. Although SynDEx is basically a framework for exploring various implementation solutions using optimization heuristics [3], we demonstrate that it can also be used directly for application multi-language modeling, cosimulation and functional checks. In the algorithm graph interface, SynDEx manually allows the distribution and scheduling of DFG on a multiprocessor. It can also be used in multi-language modeling and cosimulation. Indeed, algorithm graph operations are manually clustered and executed with the heterogeneous process located on multi-virtual-processors. The term "Virtual" is used if DFG is distributed on multi-processes but executed on a monoprocessor in order to simulate or cosimulate concurrent execution of the application.

3. GENERIC EXECUTIVE SPECIFICATION

3.1 Translation: automatic executive generation

Each generic script provided by SynDEx is transformed into a compilable or interpretable source file. Inputs/outputs of the so-called "translation" operation are illustrated by Figure 1. The macro-processor transforms the list of macro-instructions contained in the generic script into code for a specific processor target. Since the macro-executive is generic, there is a large number of possible downloadable source codes. Because of this, users have to specify the most appropriate possible translations. Depending on the system development step, two aspects have to be considered. Firstly, languages that are highly target-dependent have to be used for optimized execution. Secondly, fast design and efficient algorithm cosimulation require high-level languages such as Matlab. This being so, the proposed development methodology aims to gradually refine the language model from high-level modeling to executive for a specific target with intermediate multi-language modeling.

3.2 System level multi-language modeling

Modeling language is used to efficiently describe applications with a high degree of abstraction. This is an important step in application design, providing a reference functional model and while helping to reduce time-to-market. Indeed, this functional model is quickly developed with a high-level modeling language and can be used for early validation including cosimulation and functional check. Based on three criteria, [1] proposes a classification of nine modeling languages in order to facilitate the choice for specific design constraints. The first criterion is the *Expressive power* that takes account of *abstract communication*, *time model*, *computation algorithm*, *specific libraries* and *FSM–exception control*. The second one is the *analytical power* that describes advanced functionalities such as interaction with complementary tools, transformation and format checking. The third one is the *cost of use*, referring to a fast learning stage and ease of use. Of all these nine specification languages, Matlab has several advantages. It is one of the most popular interpreted programming languages, especially in the signal and image processing community. It is a matrix-based language including high-level functions for fast application modeling. Moreover, Matlab's toolboxes provide timing model and generic visualization functions that are appropriate for signal and image processing. [6] presents a survey of Matlab to C language transformations for fast implementation on DSP; it shows that efficient transformation is possible from matlab language. Counterparts of this easy modeling with Matlab are excessive memory allocations, low computional efficiency and lack of communication abstraction, the latter being the most critical drawback especially for multi-language modeling and cosimulation. In order to meet these requirements, recent researches aim to take potential parallel computation into account especially in image applications [2]. Most of them attempt to automatically partition work in order to distribute the application on clustered workstations. In this coarse grain parallel computing situation, data are transferred and synchronized with a network

protocol named Message Passing Interface (MPI).
In section 3.3, the principal Matlab drawback for multi-language modeling and cosimulation is solved. Communication abstraction is realised with a SAM communication model and automatic code generation in the unified SynDEx framework. To accomplish this level of abstraction, kernels for communication and synchronisation between heterogeneous processes have been developed.

3.3 Executive Kernels

For each processor or virtual processor, an executive consists of one *computation task* and as many *communication tasks* as there are connected media. Macro-instruction translations into the target language are contained in dedicated or architecture-based kernels given that definition is not, or not only, application-dependent. Dedicated kernels are used to describe the algorithm specifications such as function-calls. Architecture-based kernels include the translation of both processor and communication media. *Processor component kernels* contain macro-instructions relating to the computation task such as memory allocations or interrupt handling whereas *communication media kernels* contain macro-instructions relating to data transfer and synchronization.

3.3.1 Processor component kernels

In prototyping context described in [5], most of the kernels are developed in C language because that they can be reused for any C programmable device. Indeed, kernels are quite similar for the host (PC) and the embedded processors (DSP). Moreover the gap between two executives resulting from C language or assembly instructions becomes narrow. During the adequation (Fig. 1), compliance with constraints associated with execution time is directly dependenton on the order in which operations are executed on multi-components architecture. Presented in section 4, the proposed methodology shows that multi-language modeling and cosimulation on a monoprocessor are essential during the application design stage. This being so, in a modeling context, constraints associated with the execution time are lifted and operations are manually clustered depending on the type of modeling languages. Similarities between cosimulation on a monoprocessor and prototyping on multi-components are highlighted if we consider that heterogeneous processes are initially distributed on a virtual processor in a first time and are implemented on heterogeneous architecture in a second stage. Consequently, DFG is unchanged and only specific architecture is defined for cosimulation or concurrent simulation execution. Processor component kernels have been developed for the computation task using Matlab language. They provide automatic translation of the macro-instructions included in the computation task such as memory allocations, loop and condition compatible with Matlab language.

3.3.2 Communication link kernels

In AAA methodology [3], two different models are considered as communication media: the SAM (Sequential Access Memory) and the RAM (Random Access Memory). While the first of these defines a basic FIFO-like protocol between two processors or virtual processors, the second works with an undefined number of processors through a shared memory with controled access. The SAM models consider that two processors are synchronized by means of hardware signals. Sequentially, data are pushed by the producer if FIFO is not full and they are used by the receiver if FIFO is not empty. This model requires data to be received in the same order as it is sent. On the other hand, the read and write order need not necessarily be respected in the RAM model but data transfer management is more complex.
Most of the communication kernels proposed in [5] are designed according to the SAM model. Libraries for traditional media such as TCP and BIFO have been created for multi-PC and multi-component transfers. They have been specified in C language for obvious reasons if only fast prototyping with reusable functions are required. For fast application modeling, a higher-level language is considered. Dedicated to Matlab language, a communication medium that relies on a SAM model using TCP protocol and hardware synchronizations has been developed. Included in the SynDEx framework, the system allows communication between two Matlab (*MM*) processes, one Matlab and one C (*MC*) process or *xM* processes based on the prerequisite condition that a TCP based communication kernel has been developed in SynDEx for the process x. More generally, an undefined number of heterogeneous processes (x^j) and Matlab processes (M^i) can be interconnected with the appropriate number of SAM. It should be noted that, data transfers between processes are automatically generated and synchronized by SynDEx using automatic code generation. For users, a high abstraction level allows for less communication protocol knowledge and no restrictions are imposed on $M^i x^j$ process organization.

4. METHODOLOGY FOR SIGNAL AND IMAGE PROCESSING APPLICATION DESIGN

This section presents an extension of the methodology for the fast development of applications over heterogenous architectures proposed by [4]. The authors have demonstrated that SynDEx may be used as the front-end of the process for the design of fast signal and image processing applications. *Functional check*, *emulation* and *fast prototyping* on heterogenous architecture have been defined in [4]. In this paper we improve the methodology by adding *application modeling*, *cosimulation* and enriched *functional check* which are supported in the same SynDEx-based framework. Illustrated by Figure 2, this new methodology is described with a cycle of five sequential steps. The principle is to gradually refine the language model starting from functional modeling of the application until reaching final multiprocessor implementation. Naturally, the more efficient the computational power, the more the simulation capacity is reduced.

4.1 Algorithm design

In this step a Data Flow Graph (DFG) is created to provide a high-level application description which will be reused especially in the fast prototyping step, in accordance with AAA methodology. In a second stage, a reference functional model is developed with a high-level modeling language and will be used in subsequent steps for early validation. Matlab has been retained because it has several advantages for fast application modeling, especially for signal and image processing. In the SynDEx framework, automatic code generation provides a Matlab code (mono-process) for use on a single PC. In this way, users can design each Matlab function associated with each operation or vertex of its DFG.

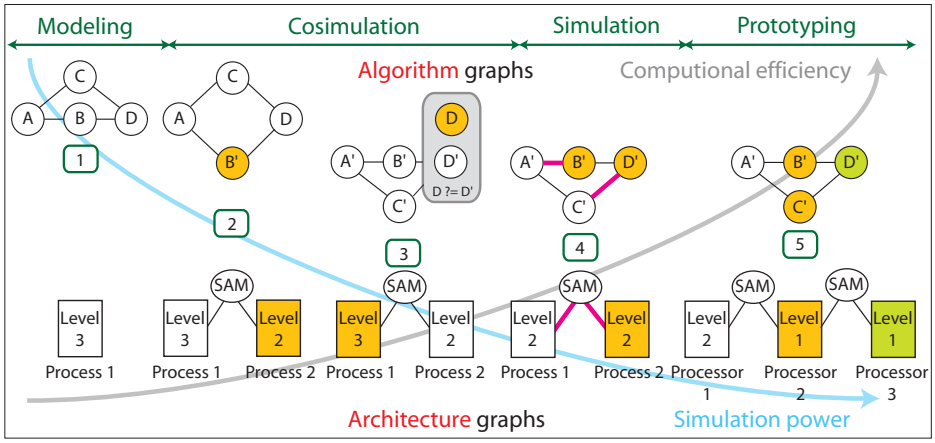


Figure 2: Methodology cycle including five sequencial steps: application modeling (1), cosimulation (2), functional check (3), emulation (4) and fast prototyping (5)

4.2 Heteregeneous modeling & cosimulation

As far as mono-PCs are concerned, cosimulation has been defined as concurrent execution of several models based on heterogeneous languages [1] but parallel execution on multi-PCs can be carried out very easily with SynDEx. Note that the final aim is to provide an application execution that meets difficult time constraints, especially in image processing. This means that language model refinement is often required and intermediate multi-language modeling is necessary in order to provide progressive steps in the application design process. Moreover multi-language modeling, cosimulation and functional check (respectively steps 2 and 3 in Fig. 2) allow early validation. For example, functional check can be easily carried out between an operation developed in C language and the result provided by the reference functional model specified in Matlab. In section 5, C/Matlab cosimultation is illustrated with a still image coder named LAR. This example shows that automatic code generation allows the use, in a C application, of generic vizualisation functions efficiently developed in Matlab. The last two last steps are similar to the ones proposed in [4].

4.3 Emulation or concurrent execution & simulation

In the step 4 of Fig. 2, the user can easily check his own DFG on a cluster of PCs interconnected by TCP links. This cluster can emulate the topology of an embedded platform. Next DFG developed in the target language is used for the automatic prototyping onto a monoprocessor and chronometrical macros are automatically inserted by the SynDEx code generator. The execution duration associated with each operation is automatically estimated through dedicated temporal primitives and is used to define operation duration in the adequation algorithm.

4.4 Fast prototyping

SynDEx generates a real-time distributed and optimized executive, in which previous chronometrical macros are automatically removed, depending on the target platform. Several platform configurations can be simulated (processor type, their number, but also different media connections).

5. IMAGE PROCESSING APPLICATION

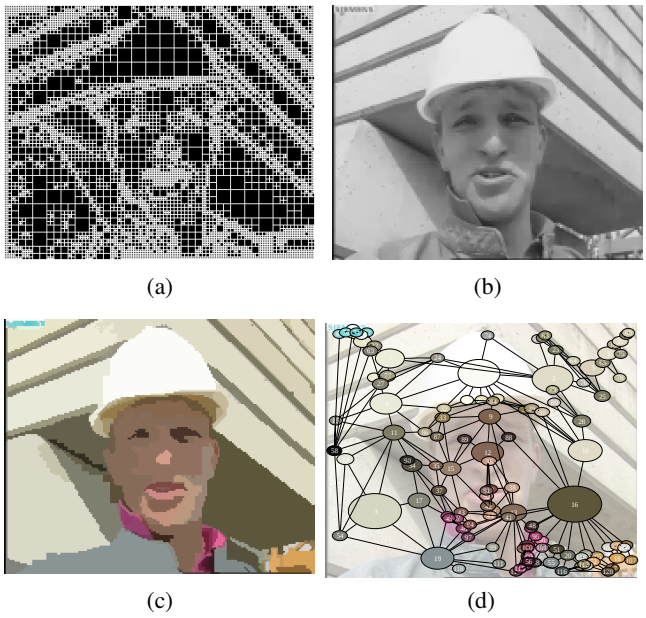
5.1 LAR for still color image coding

Proposed in [7], LAR (Locally Adaptive Resolution) relies on the principle that an image can be described as a superposition of local textures (fine details) over some basic information. With the same scheme, a low bit rate image (basic information) can be transmitted with or without its additional error image. This two-layer, context-based coding approach intrinsically provides at least two levels of progressivity.
Firstly, still image coding step aims to generate a low-resolution image adapted to the Human Visual System. It results from a variable block-size decomposition in which block size depends on local activity. Indeed, homogeneous areas are represented by larger blocks whereas higher resolution is required for textured areas and edges.Without the second layer, the resulting coder named “Flat LAR”, clearly aims to achieve a high compression ratio. The low bit-rate decoded image is visually acceptable thanks to quadtree partitioning and efficient post-processing. For higher bit-rates, the first step can be followed by a refinement layer but only the Flat LAR coder is considered in this paper. Applied to an image of the Foreman sequence, Figure 3.a shows the quadtree partition $QP^{[N_{max}..N_{min}]}$ where N_{max} and N_{min} are the upper and lower limits of square block size and 3.b is the post-processed low-resolution image. As far as this low resolution image is concerned, a low-complexity spatial segmentation algorithm has been developed to achieve a higher compression ratio. Contrary to pixel-based processing with traditional segmentation, in [7] region merging is processed on the grey-level block image and provides a self-extracting hierarchical segmentation. In other words, the segmentation process takes place in both coder and decoder from the highly compressed image. Taking advantage of the YUV correlation, a high compression ratio is obtained using region descriptions to encode the chromatic components.

5.2 C/Matlab cosimulation and compression results

It has been proven that spatial segmentation can be efficiently described with a Region Adjacency Graph. In a RAG, each region is defined with a vertex and for each spatial connexity between two regions a edge is created. We have developed a

Matlab toolbox that includes RAG manipulation and display functions. In the RAG display function, the vertex is represented as an ellipse that can be transformed in accordance with region properties. The ellipse position is given by the region's gravity-center. The ellipse color is the weighted mean color of the blocks included in the region and the ellipse surface is proportionate to region surface. Figure 3.c and 3.d are respectively the region representation and its associated RAG.



(a) (b)

(c) (d)

Figure 3: (a) Quadtree partition $QP^{[16..2]}$ (b) Decoded image + post-processing (c) Segmentation with 120 regions (d) RAG provided with Matlab

An RAG display function is included in a Matlab toolbox whereas an LAR coder has been developed in C language. The joint use of RAG display and LAR coder provides a particular case of Matlab/C cosimulation. Here, the RAG display function is used to illustrate the segmentation result obtained with the LAR coder. In the SynDEx framework illustrated by Figure 4, images and RAG are displayed with Matlab functions whereas image segmentation and coding are rapidly processed with C language. Concurrent execution is realized using architecture consisting of two PCs, one real and the other virtual. In our framework, a generic matlab function for data read or display can be easily included in a C application thanks to automatic code generation.

6. CONCLUSION

We have presented a framework for the fast development of signal and image applications based on multi-language modeling. Thanks to high-level language modeling, this enables the design of a real-time application with a high level of abstraction. This work is linked to the development of a second generation video coder named "LAR video" that is an extension of the LAR codec for still color images. In this case, an RAG can be used to show time changes in regions in order to evaluate the temporal consistency of spatio-temporal segmentation. Given region movements, it is easy to consider

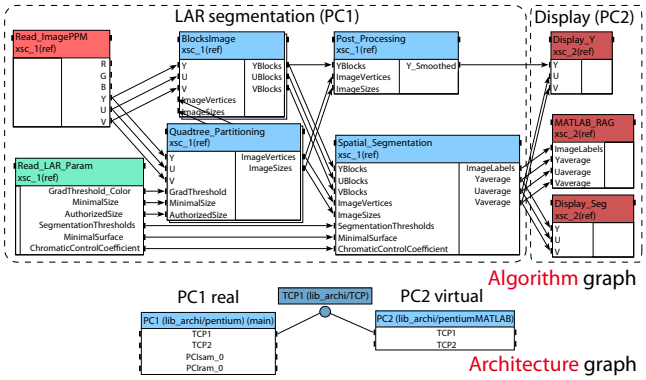


Figure 4: Description of LAR segmentation and display with algorithm and architecture graph interface proposed by SynDEx

the use of a M-RAG in which ellipse axes describe dominant translation movements. Morever, C/Matlab modeling and cosimulation aim to take advantage of the fast matlab modeling that is essential in the development of new algorithms and benefit from the computational efficiency of C language for real-time image sequence decoding.

REFERENCES

[1] A. A. Jerraya, M. Romdhani, Ph. Le Marrec, F. Hessel, P. Coste, C. Valderrama, G. F. Marchioro, J. M. Daveau and N.-E. Zergainoh, "Multilanguage specification for system design," *System-level synthesis*, pp. 103–136, 1999.

[2] E.S. Manolakos, D. Galatopoullos and A. Funk, "Distributed Matlab Based Signal and Image Processing Using JavaPorts," in *Proc. ICASSP 2004*, Montreal, Canada, May 17-21. 2004.

[3] T. Grandpierre and Y. Sorel, "From algorithm and architecture specifications to automatic generation of distributed real-time executives: a seamless flow of graphs transformations," in *First ACM and IEEE International Conference on Formal Methods and Models for Co-Design*, Mont Saint-Michel, France, June 2003.

[4] M. Raulet, M. Babel, J.-F. Nezan, O. Déforges, and Y. Sorel, "Automatic Coarse Grain Partitioning and Automatic Code Generation for Heterogeneous Architectures," in *Proc. SIPS 2003*, Seoul, Korea, August 27-29. 2003.

[5] M. Raulet, F. Urban, J-F. Nezan, O. Déforges and C. Moy, "SynDEx Executive Kernels for Fast Developments of Applications over Heterogeneous Architectures," in *Proc. EUSIPCO 2005*, Antalaya, Turquey, September 2005.

[6] R. Allen, "Compiling high-level language to DSPs," *IEEE Signal Processing Mag.*, vol. 22, no. 3, pp. 47-56, May 2005.

[7] O. Déforges, M. Babel, L. Bédat and J. Ronsin, "Color LAR codec: a color image representation and compression scheme based on local resolution adjustment and self-extraction region representation," *IEEE Transactions on Circuits and Systems for Video Technology*, accepted, 2007.