Methodology of conception of adaptive multi-constraints system

Tarek FRIKHA, Nader BEN AMOR, Ines BENHLIMA, Kais LOUKIL, Mohamed ABID
Sfax University, National Engineering School of Sfax
CES-Laboratory
Sfax TUNISIA
tarek.1982@gmail.com

Jean-Philippe DIGUET
Lab-STICC
University Bretagne Sud,
Lorient, FRANCE
jean-philippe.diguet@univ-ubs.fr

I. Context:

This paper entitled “Methodology of conception of adaptive multi-constraints system” talks about the adopted methodology to design adaptive multi-constraints system . This system uses the Kais Loukil and Al [1] approach and extends it to run time reconfigurable architectures. This paper is composed by a state of art, the adopted approach, the realized work and finally the future tasks to do.

II. State of Art:

The Embedded multimedia system emerges in different fields. The emergency of multimedia applications particularly in mobile embedded systems puts new challenges for the design of such systems.

The major difficulty is the embedded system’s reduced energy and computational resources that must be carefully used to execute complex applications (Video flow, 3D applications, Video compression applications…) often in unpredictable environments (dust, vibration…). The augmented reality is a very promising 3D embedded multimedia application. It’s based on the addition of specific 3D’s animations on a video flow.

Many fields are concerned by the 3D multimedia application.

In industry, we use it in conception plan of architectural project such France’s Nice Metro [2] project. Many other industrial examples can be cited such as automotive applications. The whole production chain is created virtually and the technicians are formed to understand how to do their job properly. They learn how to find different components and how to add or retire them. It’s also used in reeducation and health domain. The simulation of the chirurgical operation helps doctors to the application of medical theory.

In personal field, the applications are different. It touches the video games in which the user becomes an entire part of the game. He’s projected in an immersion world and he’s reactive with the game environment. Many wars, sports, driving and RPG games are used the AR to make their game real.[3]

The chosen application introduces a 3D object to a video flow. This object is similar to the added objects in Virtual Reality. The described work concerns the 3D objet. We have to adapt our object to the application different constraints.

In the next paragraph, we’ll talk about the adopted approach.

III. Adopted approach :

In this work as cited previously, we were inspired by the Kais Loukil & Al approach. The chosen application for this work is based on adding a 3D object on a video flow.

In his approach, Kais Loukil used a cross layed adaptation as described in figure 1.

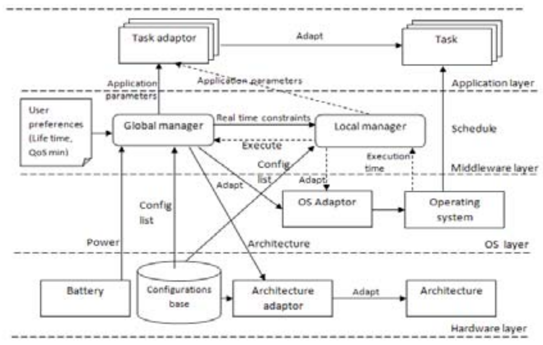


Figure 1: Cross Layer adaptation approach

In his approach, Loukil the global manager coordinates between the hardware, OS and application layers. The local manager is coordinating between the application and the OS layer to guarantee the real time constraint. The task adaptor adjusts the parameters or the algorithm of the task. The OS adaptation adjusts the number of affected CPU cycles for each task while the architecture adaptor deals with the change of hardware architecture system.

The database of the configurations contains a set of characterized configurations (hardware or software) for our system. [1]

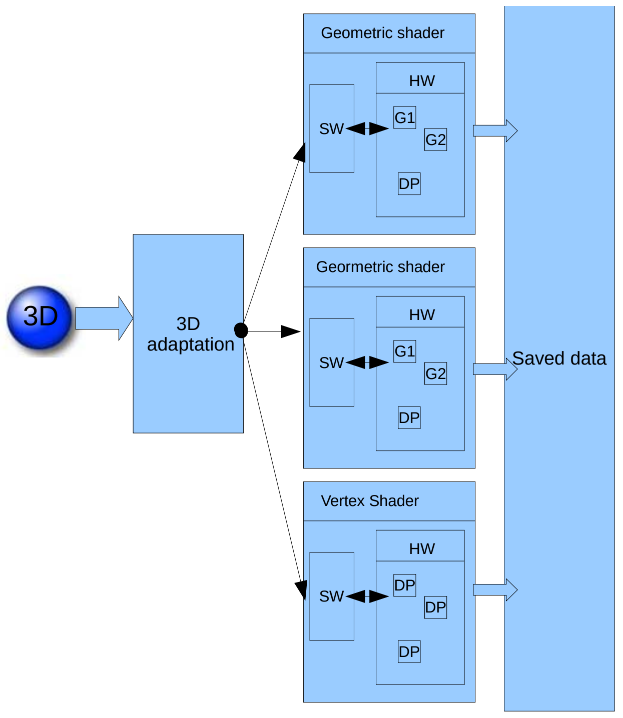
We need to add a network adaptation. Due to the environmental noises, we can have network problems. This problem consists on the broadband decrease. To remedy to this problem the use of a dynamically partial reconfiguration is the technical adequate chosen network layer. One of the most important problems of the Loukil &Al approach is that he had to implement all the bitstreams in the FPGA. These implemented architectures requires many FPGA spaces and consumes energy that's why we have to use an architecture based on dynamically partial reconfiguration to optimize the consumptions.

IV. The realized work:

a. The adapted architecture:

The virtual reality application which is the chosen application consists on adding a 3D object to the video flow. This application will be finalized by a demonstrator.

This demonstrator is composed by two parts: a transmitter and a receiver. A camera is used for video acquisition. This camera transmits a video flow to the transmitter. The transmitter is composed of a video flow transformation bloc and a MJPEG coder (embedded in a first ML 507 FPGA board) which is used to compress the video. The 3D animations specifications are multiplexed with the encoded video. They are sent over the TCP IP network using an XML file. At the reception, the video is decoded; 3D animations are computed using XML specifications and mixed with the decoded file. The figure 2 reperesents the 3D adaptation technique. According to the 3D object characteristics we add the appropriate hardware blocs. The final data are saved on a memory blocs. The blocs are the geometric shader and the vertex shader. The 3D application characteristics are described in the next part.



SW : Software, HW : Hardware, G Geometric, DP: Dessine Polygon

Figure 2: Architecture used accelerators

b. The 3D application characteristics:

We use a 3D application available as a C code. In this application the object rotates around different axis. Due to its complexity, the software application version can be displayed but are so slow.

To hardware candidate functions, we analyze the functions and particularly used arithmetic operations that consume the major part of execution time.

The 3D application study divides the function in two important parts:

- Geometric shader: describes the geometrical characteristics such as rotations, translations, scale …
- Vertex shader: describes how to fill the pixel colors.

The figure 3 represents the application code functions. The C code is profiled to know which functions are the greediest in term of time and cycles consumption. After this profiling we found that calculnormal (Normal) et Transformation (Transform), dessine poly (DP) and barycentre (bary) are used 3*nb of polygons time and are have the most important time of execution. That's why we'll transform them into hardware accelerators. These accelerated functions are reconfigured dynamically depending on the 3D application characteristics. The future works will talk about the dynamically partial reconfiguration.

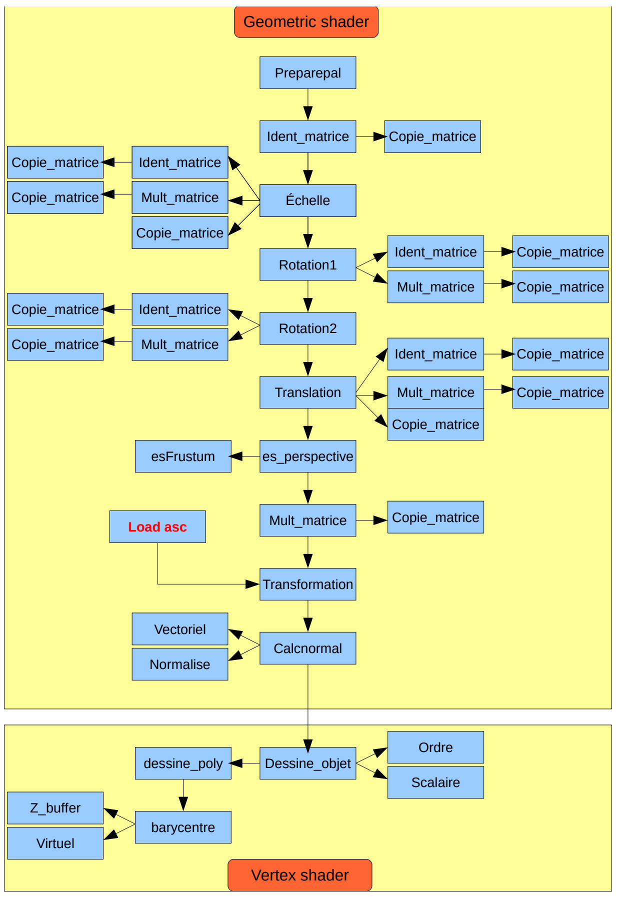


Figure 3: 3D application functions figure

V. The future work:

In this section, we describe the proposed reconfigurable architecture. This architecture will be implemented on the FPGA. We'll compare the static architecture with the dynamic one.

The adopted reconfigurable architecture is described in the figure 5. There are 3 hardware accelerator zones.

The zone 1 (Z1) contains Normal #1 and Transform #1 as described in Section III, C and b). These functions represented the geometric shader. This zone is a permanent one. This zone is attached to the microblaze via FSL. The data will be sent to the zone 2 or zone 3 for vertex shading. The microblaze (µB) is the Xilinx softcore.

The zone 2 is the reconfigured one. It can be used for not only geometric shader but also a vertex one. The zone 2 is like the zone 1 if the application contains many rotations, translations and scaling data. The used FIFO becomes virtual. We don't need to use it. This zone can be also a vertex pipeline. This pipeline needs a FIFO to save the data on it. The Reconfigurable zone needs to have the same input and output despite the configuration type. That's why the FIFO is always used in the second zone. This zone is also connected to the microblaze via FSL. If the Z2 is similar to Z1 (moved object), the FSL send data to microblaze to be treated after that by the zone 3 (Z3). If the Z2 is similar to Z3 (textured object), the output are saved on FIFO.

The third and final zone contains three blocks. The barycenter block. This block determinates the input of the Dessine-Poly (DP) block. These data are saved on FIFO to be ready for the DP treatment. The Z3 or/and Z2 results are saved after treatment on a FIFO. The two blocks can be used in parallel. For this reason we use the FIFO to save data.

The µB*, is used to transfer data from FIFO to VGA IP (IP permitting to display the saved data in the screen) via the Processor Local Bus.

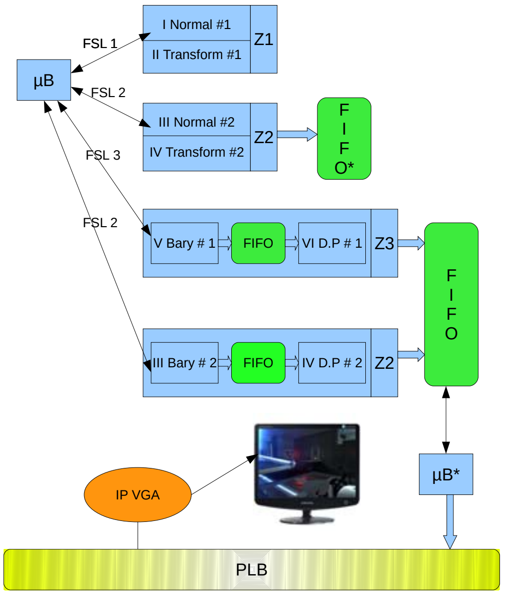


Figure 5: Reconfigurable proposed architecture

This architecture will be implemented to obtain a dynamically partial optimized architecture.

VI. Conclusion:

In conclusion, this paper represents our work which is based on dynamically partial reconfiguration architecture. This architecture is optimized to make us obtain a 3D object which is implemented on a video flow with 25 frames by second. To transfer this data on the network, we'll need to compress it using the H264 algorithm.

References:

[1] Phd, Kais Loukil, "Approche de gestion de performances/contraintes pour les systèmes embarqués temps réel", december 2011

[2] http://www.enodo.fr

[3] P. Fuchs, B. Arnaldi, and J. Tisseau. La réalité virtuelle et ses applications,chapter 1, pages 3–52. Les Presses de l'Ecole des Mines de Paris, 2003.