HW/SW Interface Impact on an Adaptive Multimedia System Performance: Case study

Manel Hentati[1], Nader Ben Amor[1], Kais Loukil[1], Mohamed Abid[1], Jean-Philippe Diguet[2]
[1]*Research Unit CES, National school of Engineers of Sfax- Tunisia ,*
[2]*Université de Bretagne Sud, Lorient, France*

Abstract

The embedded systems are more and more complex. They must answer to contradictory constraints. Indeed, these systems integrate new functionalities that require high performance architectures. At the same time, these systems must be flexible and adaptive since they work in a fluctuating environment and have a quite unpredictable workload. In order to answer to all these constraints, we propose a new adaptive system that can adjust its performances according to the context. We are interesting in studying architectural adaptation technique in a reconfigurable platform. We focus in Hardware/ Software interface impact on the system performance. A case study for the application "3D synthesis" is carried out.

Key Words- Architectural adaptation, interfacing, HW accelerator, reconfigurable platform**.**

1. Introduction

The constraints on the design of a system-on-chip (SoC) are very strict. This is due to the evolution of applications, the resources limited energy and the requirements of performance in terms of flexibility, and cost… Various technological solutions were adopted: ASIC, DSP, FPGA…

The Application Specific Integrated Component, called also ASICs[1], is very effective at improving performance, in terms of integration capacity, computing speed, and energy consumption. These specific circuits have also many drawbacks. Added to the high cost of their development, they are not flexible.

The Digital Signal Processors (DSP), which are processors dedicated to processing signal, can effect complex calculations in real time and integrate several instructions in parallel. They allow also a fast access to the data by a particular addressing. But, many DSPs do not support operating systems. They are usually used as specialized coprocessors with a master CPU for global management for the system.

The Application Specific Integrated Processors (ASIP) offers a dedicated hardwired architecture solution so that only a limited number of applications will be able to explore this circuit. However ASIP is no longer able to modify its architecture to address a new class of target applications.

All the previously cited architectures cannot allow an "on-line" modification of the system architecture according to the context. The best current solutions try to adapt the architecture of the multi-media system to the applications constraints. Once the architecture is defined, no mechanism is designed to adapt it during the operations

Currently, the new emerged systems are mobile, battery operated system. They are operating in variable conditions such as limitation of energy resources, fluctuations of the network bandwidth and variable workload due an increasing number of supported applications. Such systems must have online adaptation techniques to address the previously described problem.

Adaptive architectures based on reconfigurable architectures seem to be a promising a solution for such systems. These architectures offer a trade-off between flexibility, performance cost and energy consumption. In this paper, we present experimental results for the validation of a new adaptation technique used for mobile embedded systems.

The paper is organized as follows. In section 1, we discuss the related work. The second section describes the adopted adaptation technique. Section 3 presents a case study. Finally in Section 4, we detail the methods of interfacing accelerator under PACM model and we comment the experimental results.

2. Related work

Adaptation can be located at different levels: application level, operating system level and hardware (HW) level.

For Adaptation at the hardware level, in [2] an adaptive technique was proposed. This technique consists of using the dynamic voltage scaling (DVS) to adjust the speed and the power of the CPU, while being based on the estimation of the workload of time CPU. This technique has several drawbacks: Accesses are not applicable only on the processors which allow a dynamic change of the frequency and supply voltage. Then, with the progressive reduction in the supply voltage of the processors, the DVS becomes less and less effective.

978-1-4244-3321-6/08/$25.00 ©2008 IEEE

Architectural adaptation can also be applied on a proprietary reconfigurable architecture like the Tiled Logic Architecture [3].
For adaptation at the operating system level (OS adaptation), In [4], the authors use a middleware layer to facilitate the adaptation of QoS for the application system which is submitted to the execution time and to the supplied energy constraints. Another technique was proposed in [5] the managers of CPU resources, provide performances guarantees in soft real time. Schedulers further adapt the scheduling policy to handle the variations of application runtime

For adaptation at the application level, many techniques are used. In [6] two approaches are proposed for the deterioration of the quality of an object 3D to satisfy the constraints of resources and network bandwidth. In [7] a method for energy saving for the "decoding MPEG" application using application parameter modifications is presented.

All the approaches previously cited are local and cannot allow a mixed adaptation. In this paper, we study the validation of a more global approach based on a multilevel adaptation technique.

3. Proposed approach

In [8], a more global adaptation technique is presented. It manages three main constraints: lifetime, real time constraint and the quality of service QoS Lifetime is related to battery's system remaining energy. Real time constraint can be modelled with the execution time of the target application (noted as Texe). QoS is related to the user satisfaction of the received quality (video rate, 3D game fluidity). For example, if the system is efficient, it deals easily with complex applications and the resulted QoS will be high.

As shown in figure 1, the adopted approach uses three main activities. The first one is the observation task. It permits the follow the evolution of the systems parameters (Lifetime, QoS and Texe). The second one is the adaptation task. It uses classic regulation algorithms. Its output is reconfiguration order. The third task applies the required reconfigurations. Reconfigurations are made using a set a predefined and offline characterized configurations. A configuration can be seen as a state of the system that corresponds to an amount of energy consumption, a Texe and a QoS level. All the configurations execute the same target application with different schemes corresponding to different application parameter (or algorithm) and/or different architectures solutions (case 1). Moving from a configuration to another can be made through application parameter (such as search window size in MPEG application). It can be made also through architecture modification such as moving from a pure software version to a mixed hardware/software version (case2) with high-performance but less energy efficient. For energy saving purpose, the system can use low performance configurations. High performance configurations can be used for complex applications. Those configurations are offline characterized: the energy consumption, the Texe and QoS values are known previously using several tests.

The configurations number must be chosen carefully. A too big number increase the overhead of the adaptation approach (associated energy cost, complexity of regulation algorithm…). A too small one leads to a system not dynamic enough that can not address a great number of real situations.

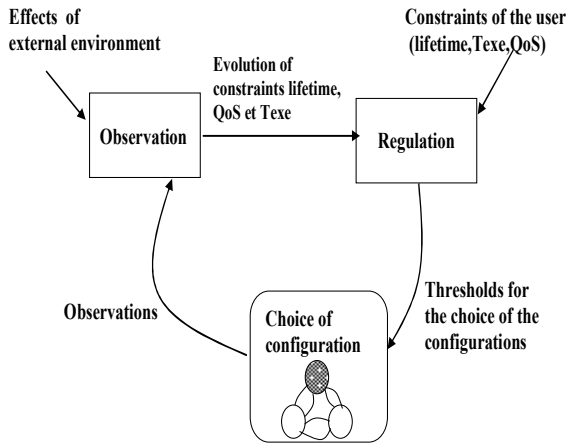


Figure 1. Synoptic diagram of the system of adaptation to multiple constraints

For a same mixed hardware/software configuration, different communication schemes can be used. They correspond to various types of interconnections techniques (processor/memory, peripheral processor/and processor accelerators). In this paper, we are interested in the adaptation possibility at the interconnect level. It is a finer granularity adaptation that corresponds to case 2.

This requires a study performance of the various interfacing techniques (their performance and energy consumption)

The adopted architecture model is given in the figure 2. It is named the PACM (Processor, accelerator, coprocessor, memory) model. It is a general model easy to implement on a large set of various platforms. It gives more both flexibility and adaptability to the system.
The figure 2 presents the PACM model [9]. Basically, our architecture is built around a processor core (for example NIOS, ARM, LEON…), a standard on chip bus (Amba, Avalon, IBM CoreConnect…) to communicate between processor and blocks. The main processor can be enhanced with dedicated hardware accelerators and coprocessors. Their use depends on the application

constraints and requirements. The PACM model uses also specific memory blocks to facilitate communication.

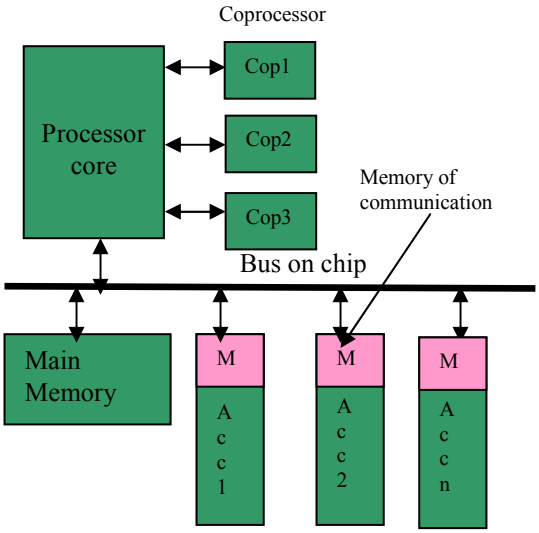


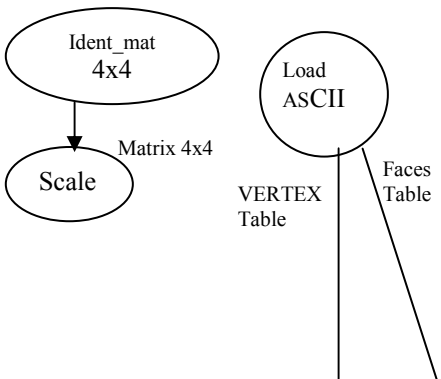
Figure 2. PACM architecture

4. Case study

We choose as a case study the 3D image synthesis. It is a complex application more and more used in mobile systems.

4.1. 3D synthesis overview

The figure 3 shows the tasks graph associated with this application. The entry of this graph is a set of local coordinates of the various vertices (or polygons) summits that constitute the 3D object. Those coordinates are computed using a local coordinate space related to the 3D object. Those coordinates are saved in text file and are extracted by "Load ASCII" task. The Functions "Scale", "Rotation", "Translation" allow the animation of the 3D object (zoom and motion). The function "transformation" allows the transformation of local coordinates of the 3D object to the scene space (called as world space) [10]. The function "normal_Cal" allows calculating the polygons normal. It is used for light computation. The "tri_face" function determines the faces visible to the observer and eliminates the hidden faces. "Draw_Poly" and "Object_Poly" are used for 2D screen visualization.



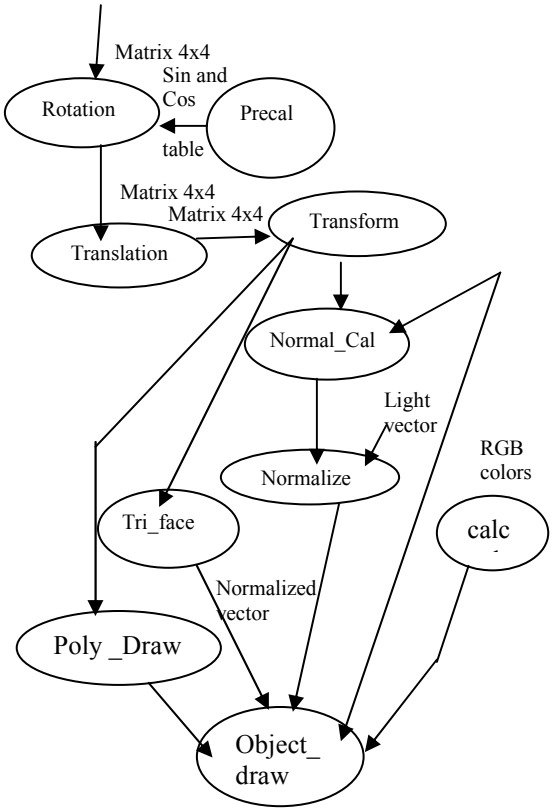


Figure 3. Task graph of the synthesis 3D application

This application is written in C for PC platform. It allows visualization of a 3D image with a variable number of vertices and using two light computation algorithm (flat and Gouraud). Those two parameters will be used to build various configurations as explained in the following sections.

4.2. 3D synthesis application analysis

The original code was modified and adapted to the NIOS II compiler.

To identify the critical (often-called) functions which require a hardware implementation, it is necessary to analyze the 3D synthesis application using profiling technique and high level complexity analysis to each function. As we plan the use of the NIOS embedded processor for the configurations set up, we adopt its associated profiling tool "NIOS IDE". It has a "Performance Counter" that determines the different functions execution time, their percentage as well as the call number of each function. This will enable us to identify the critical functions requiring hardware modules for their execution. The hardware candidate tasks are analysed using a high level codesign tool called "Design Trotter" [11] in order to better analyze their inherent

parallelism and orientation (memory, treatment or control) using specific metrics. Such analysis is useful to define the most suitable implementation for critical functions so that the application-architecture matching goal can be reached. For example, often-called functions with high inherent parallelism are ideal candidates for hardware accelerators. Critical functions with small local data exchange are ideal candidates for coprocessors. Design Trotter tool analyses of separate functions written in C language and computes three metrics. "Gamma" metric is used for inherent parallelism determination. MOM computes the ratio of data exchanges on the total operations executed by the function. MOC metric computes the control operation (do while loops, test operations, switch tests,…) on the total operations executed by the function.

Table 1 shows the profiling results of the synthesis 3D application. Using the task Graph in the figure 3, we can identity the most really most called functions. The most critical functions are, "transform", "Normal_Cal" "Tri_faces", and "Poly_draw". We analyse them using the "Design Trotter" tool which enables to know more precisely their orientations.

Table 1. Profiling result of the 3D synthesis

Section	%	Time(sec)	Occurrences
Transform	6.03	26.875	360
Tri_face	5.92	25.877	360
Normal_Cal	13.5	58.966	360
Translation	0.165	0.726	360
Load_ASCII	0.013	0.0566	1
Objet_draw	77.6	340.79	360
Rotation	0.479	2.103	360
Poly_draw	68.6	301.244	11489
normalize	2.23	9.787	12960
scale	0.144	0.62963	360

Table 2. Results from "Design Trotter" analysis

Function	Gamma	MOM	COM
transform	3.8750	0.3939	0.000
Normal_Cal	3.3244	0.6527	0.0425
Tri_faces	1.8571	0.6666	0.1818
Poly_draw	1.2429	0.8298	0.0038

According to table 2, we note that the "transformation"," normal_Cal", functions have a relatively high value of gamma; therefore they have an important average parallelism. "Tri_face" function has a high value of COM metric and a low value of Gamma metric that is why it is better to implemented in software . It can be enhanced using coprocessors technique. In this paper, we show only the implementation of two functions: the "transform" function and "Normal Calculate" function. The remainder of the application functions is implemented in software. This choice a priori simple is useful to facilitate the set up of the hardware/software interface.

5. Design and interfacing of 3D accelerators

This section presents the design and the interfacing of 3D HW accelerators on a reconfigurable platform. The adopted hardware platform is ALTERA STRATIX development board. It allows the design of a complete PACM (Processor Accelerator Coprocessor Memory) based system using the NIOS II processor and the AVALON bus. The architecture presented in this paper contains HW accelerators which are described in VHDL.

5.1. HW Accelerator Implementation

We realise a system according to the PACM model composed of a NIOS II Processor that communicates with HW accelerator through the Avalon bus. The accelerator treats the data while the Avalon Bus supports the control and runs the data trade off between blocks and memories. The NIOS processor must only produce the signals that determine the behaviour of HW blocs (slave, master) which will take the wished aspect.
To study the impact of the AVALON bus on the system's performance, two interface modules are developed for the same HW accelerator the PIO (Parallel Input/Output) and the user defined interface.

5.1.1. The PIO interfacing method. PIO (Parallel Input/Output) is a component that enables to connect peripheral components to the bus. A PIO provides an interface between the SW part and the user logic. The added accelerator is a slave i.e the main processor gives it the data to proceed and recuperate results at the end of computation.

This system is composed by a processor as a unique master connected to the Avalon bus and a hardware slave accelerator. The hardware accelerator specification has a reading and writing port for processor data exchange.

5.1.2. The interfacing method « User defined Interface. This method allows the designer to define its own interface between the user peripheral and the processor NIOS Avalon bus. It is an interface that the user can adapt according to his needs (data size, data transfer mode…). This interface is automatically generated by the Builder SOPC tool included in the NIOS IDE. The added accelerator may be master or slave.
This interface is composed essentially by 4 units (see figure 4): NIOS II processor, a HW component**,** external RAM and an Avalon bus.

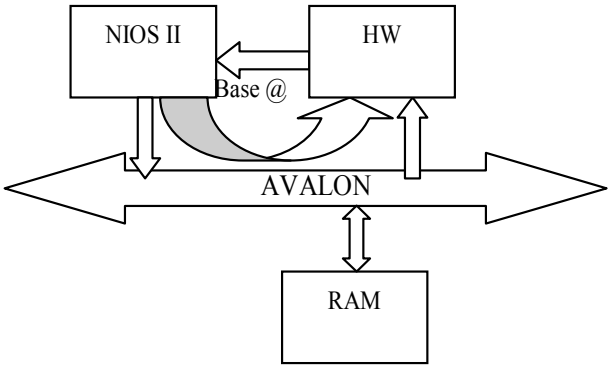


Figure 4. Structure of the master accelerator interface

The master accelerator, which is described in VHDL can reach the memory via a master port with a direct memory access mechanism. The HW module receives base address of RAM from the processor via the slave port, and it accesses in the external RAM to read data via the master port. Computations are done in combinatorial processes. Finally the accelerator accesses again in external RAM via the master port to write the result. This process is repeated until the end of computation. At the end the HW module sends an interruption signal (IRQ) to the processor.

For both of the interface methods, the specific driver is written in C language to pilot the HW accelerator**.**

When the HW accelerator is fully tested, we conduct necessary modifications on the C original code to obtain a new functional code that supports the HW components.

5.2. Results

We validate the HW components and their two associated interfaces using a development board with a STRATIX FPGA device. For validation purpose, we realize a demonstrator for the 3D image synthesis. It uses a Lancelot VGA adaptor (see figure 5)

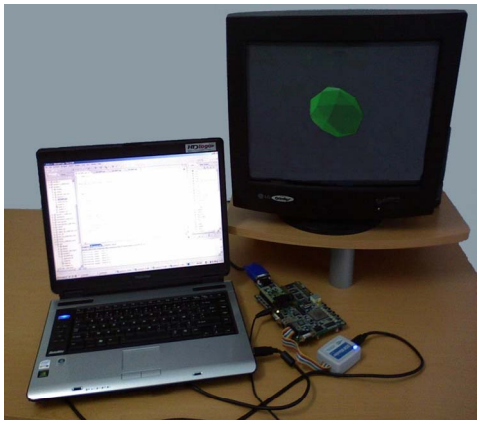


Figure 5. 3D synthesis demonstrator

In table 3 we group the experimental results (FPGA occupation and power consumption.

Table 3. Table of the results

	Total ALUTS	Dynamic power (mW)
slave accelerator (transform)	7.948 (19%)	333.29
master accelerator (transform)	9.402 (23%)	346.22
2 master accelerator (transform and normal_Cal)	13.004 (32%)	419.06

Figure 6 gives the experimental results in term of execution time in second while varying the number of polygons for the developed solutions: software, mixed (slave) mixed (Master), and where the angle rotation of the object is varied from 0 degree up to 259 degrees.

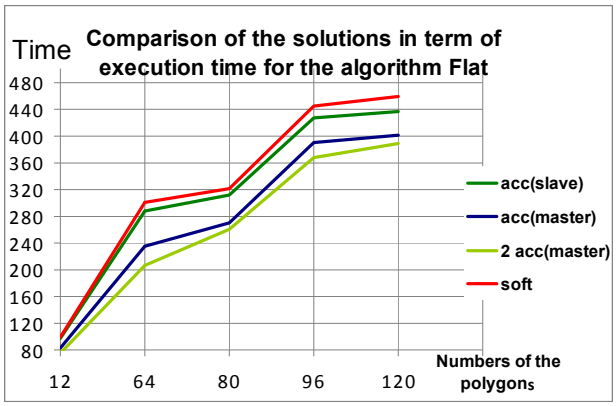


Figure 6. The curves of the execution time variation depending on the numbers polygons

We notice that master accelerators are more efficient than slave one due to high speed DMA-like memory transactions. But they require more power. As shown in figure 6, execution time of the 3D image synthesis depends on number polygons and the type and the number of the HW accelerators. Curves presented in figure 6 can be approximated with linear curves so that simple models can be extracted. Those various model (concerning time execution and power consumption) are high level ones used by the regulation task (figure 1) for adaptation purpose.

6. Conclusion

In this paper, we interested in a hardware adaptation technique. It uses a set of previously set up configurations. Configurations are obtained with various algorithmic and architectural parameters. They are used by a regulation task to adapt the system. A mixed configuration is made by a processor with HW accelerators and with different HW/SW interfaces. We study the interfacing method effect on the performances of the system (consumption, execution time). We used as a case study the 3D image synthesis. We made several HW accelerators (slave and master) that communicate with a host processor with two different interfaces. We realized a demonstrator based on a development board with a VGA adaptor for an easy visual validation. Different models can be carried out from those studies. They can be used for configurations set up. They can be also used by the regulation task. As a perspective, we propose to study other architectural adaptation techniques such as memories management, multi-applications adaptation. We plan also to extend the approach on a dynamic reconfiguration platform.

7. References

[1] Aoudni Y., Abid M,., Gogniat G., Philippe J., "Custom Instruction Integration Method within Reconfigurable SoC and FPGA Devices", IEEE ICM, Dhahran Arabie Saudite December 2006..

[2] R. Melhem, N. AbouGhazaleh, H. Aydin, and D. Mosse, "Power management points in power-aware real-time systems," in Power Aware Computing, R. Graybill and R. Melhem, eds., Plenum/Kluwer Publisher, 2002.

[3] A. Laffely, J. Liang, P. Jain, N. Weng, W. Burleson, and R. Tessier, "Adaptive system on a chip (aSoC) for low-power signal processing," in Proceedings, Thirty-Fifth Asilomar Conference on Signals, Systems, and Computers, Nov. 2001.

[4] J. Flinn, E. de Lara, M. Satyanarayanan, D. Wallach, and W. Zwaenepoel, "*Reducing the energy usage of office applications,*" in Proc. of Middleware 2001, Heidelberg, Germany, Nov. 2001.

[5] A. Bavier and L. Peterson, "The power of virtual time for multimedia scheduling," in Proc. of 10th International Workshop for Network and Operating System Support for Digital Audio and Video (NOSSDAV), June 2000.

[6] W. Van Raemdonck, G. Lafruit, E.F.M. Steffens, C.M. Otero Pérez, R.J. Bril "Scalable graphics processing in consumer terminals" Multimedia and Expo, 2002. ICME '02. Proceedings IEEE International Conference 2002.

[7] M. Mesarina and Y. Turner, "Reduced energy decoding of MPEG streams," in Proc. of SPIE Multimedia Computing and Networking Conference, San Jose, CA, Jan. 2002.

[8] N. Ben Amor., « *Approche de Conception De Processeur De Vision Embarqué* », thèse, l'Ecole Nationale d'Ingénieurs de Sfax, Tunisie ,2005.

[9]Y. Aoudni, K. Loukil , G. Gogniat, J.L. Philippe, M. Abid, "Mapping SoC architecture Solutions for an Application based on PACM Model", ISIE Montreal Canada July 2006.

[10] T. Akenine-Möller, Eric Haines "real time rendering, second" p14 edition 2002.

[11] N. Ben Amor, Y. Moullec, J-Ph. Diguet., J-L. Philippe, M. Abid, "Design *of a multimedia processor based on metrics computation*", Special Issue for "Advances in Engineering Software", volume 36 (p 448–458.,2005.