Design of Real Time Multiprocessor System on Chip

Kaïs Loukil, Nader Ben Amor, Yassine Aoudni, Mohamed Abid,
CES Laboratory
ENIS National Engineering School
Sfax, Tunisia
Email: Kais_loukil@ieee.org

Abstract— **Actually, multiprocessor architecture is one of the solutions to fulfill the heavy computational requirements of the new applications running on embedded systems such multimedia and 3D games. The design of such systems pose various problems located at different level: architecture topology, lack of multiprocessor RTOS. Hence, we suggest in this paper a new topology of multiprocessor architecture as well as a generic layer of inter-processor communication which allows the adaptation of the single processor operating systems to multiprocessor architectures. Finally, we round off this article by a comparison between some possible architecture for the design of a system. Those experiments are made through the 3D images synthesis application.**

Keywords: Multiprocessor, RTOS, SoC

I. INTRODUCTION

With the progress of the capacity of integration of hundreds of million transistors on one chip and the development of high level design of the embedded processors core, new architectures are now directed towards the integration of several processors on the same chip, like: DSP[1], hardware and software IP[2], memories, shared bus, etc… We, accordingly, speak about multiprocessors systems on chip (MPSoC[3]) [1, 2, 3]. Those systems are one of the solutions to answer the rising complexity of the integrated systems used for applications such as the multi-media applications.

Moreover, the use of real time operating system (RTOS[4]) become essential in the embedded systems for many reasons [4, 5, 6]. First, the integrated systems as we mentioned above are increasingly getting complex. Second, we witness the presence of strong real time constraints. Third, the available resources are limited especially in memory and also in supplied energy, i.e. the computing power is certainly limited. All these factors make the management of the different embedded systems resources (computing resources, energy resource, etc) an activity more and more complicated that require a RTOS.

However, in spite of the large diffusion of multiprocessor systems; it is found that the majority of the existing RTOS do not support multiprocessor architectures wherein the need for finding methods to fulfill the requirements of such systems. Two solutions can be distinguished. The first one consists in developing new operating systems that support multiprocessors architectures. The second (which is adopted in this paper) one consists in extending the existing RTOS by other functionalities so that they can support the multiprocessor architectures.

This paper is organized in the following way. In the section 2, a study of exiting topologies of multiprocessor architecture is presented. The section 3 shows a new topology of multiprocessor architecture. The section 4 is devoted to the presentation of preliminary experiments of the new architecture.

II. TOPOLOGIES OF MULTIPROCESSOR ARCHITECTURES

In order to set up a multiprocessor system, several types of multiprocessor architectures are planned [7]. We give hereafter an overview of those architectures.

A. Shared memory systems

This type of memory has the advantage of allowing an immediate division of the data, facilitating by this way the programming of the software that manages the system (fig 1). But this solution has many drawbacks. Firstly, the number of processors which can be added on the same memory is limited [8]. Secondly, as the memory bandwidth is limited, the rise of the performance of the whole system does not follow linearly the number of processors added. Moreover, even if the communication software is facilitated, it remains the programmer's duty to check the coherence of the data by synchronizing the accesses to the critical data. For example, it should be avoided that two processors can modify the same variable without taking into account the modification of the other processor, for fear of putting in danger the coherence of the data and thus of the result of a computation.

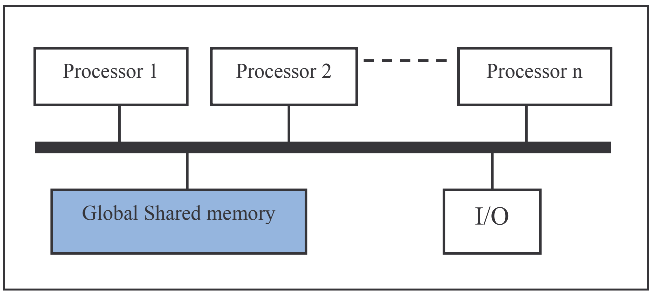

Figure 1. Shared memory systems

[1] *Digital Signal Processor*
[2] *Intellectual Properties*
[3] *MultiProcessor System on Chip*
[4] *Real Time Operating System*

B. *Distributed memory system*

In this case, each processor has its own memory. The modification by one of the processors of its own memory does not have a direct influence on that of the other processors. That, as a matter of fact, presumes to set up an explicit communication between the processors.

This type of architecture (fig 2) has the advantage of allowing a rise of the performances processors/memories more interesting than in the case of the shared memory, but it is the role of the programmer to manage the majority of the details of the communication between the processors. It also makes the complete exchanges of data structures difficult, it poses problems of non-uniform access in time, and it makes the coherence of data harder to maintain.

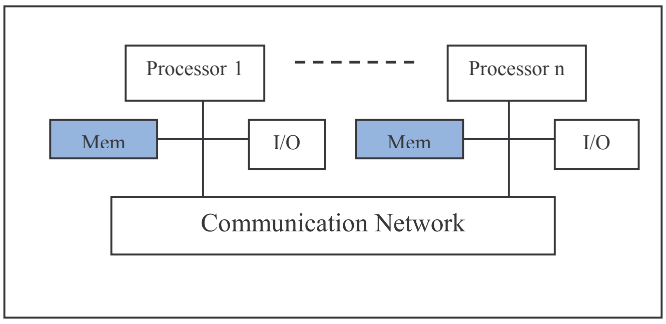

Figure 2. distributed memory system

C. *Distributed shared memory system*

This type of memory is a mixture of the two first types of architectures. In this architecture, there are several groups of processors sharing a global memory thanks to a network. At the same time, every processor has its own local memory. That allows, to a certain extent, to draw the advantages from the two preceding architectures and to reduce their disadvantages [9].

In this model (fig 3), the various tasks share the same addressing report; they can read and write inside in an independent and asynchronous way. That makes it possible to avoid the problem of the communication between the data and the tasks. But the principal disadvantage of this model is that the coherence of the data and the accesses must be managed by the programmer using semaphores or bolts, with the risk of decreasing the performances of the system if the number of messages exchanged is very important. Moreover, each processor uses its own local memory. The communication of the data and the synchronization are made using messages whose format is left to the discretion of the programmer. The various authorities of the application distributed must be synchronized. Indeed, the sending of a message must be the subject of an explicit reception by the recipient.

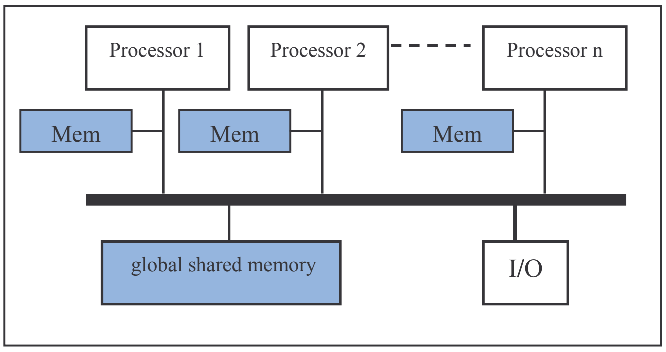

Figure 3. distributed shared memory system

Architectures with shared distributed memory are very much used within the framework of the multiprocessor systems on chip, but this type of architecture can influence the performances of the system especially when the number of communications between the processors is very large. We propose in this paper a new type of architecture which permits simultaneous access to the memory. This new topology is presented in the section 3.

III. DESIGN OF REAL TIME MULTIPROCESSOR ARCHITECTURE

The design of a new multiprocessor system is composed of two main steps. The first step is the design of the HW architecture. The second step is the set up of the software communication layer that manages the HW architecture.

by using the principal characteristic of the avalon bus "simultaneous multi master"

A. *Design of multiprocessor architecture:*

The new proposed architecture is an improvement of the distributed shared memory systems detailed in section 2.3. In this new topology, we propose to use for each processor (noted Px), two memories (Cf. fig 4). The first one is specific for Px. The second memory is shared with all the other processors in writing operation. All the messages which are intended to Px are stored in this memory. Processor x is the only processor that can read its contents. Px accesses to the shared memory must be managed by the programmer using semaphores or bolts.

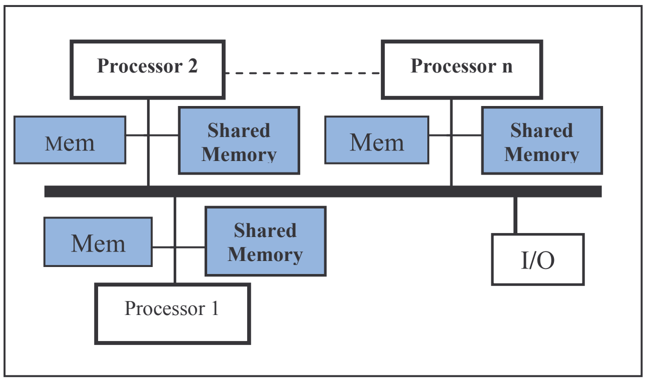

Figure 4. distributed memory, shared distributed memory systems

We use the kit EXCALIBUR of ALTERA to implement the new architecture. This kit is composed of :

- the NIOS-II processor,
- a development board containing a STRATIX-II FPGA,
- the Quartus-II environment of development.

We adopt as RTOS the MicroC_OS-II [10]

The NIOS II use the Avalon bus. The main characteristic of this bus is that it is simultaneous multi Masters. This characteristic facilitates the set up of the new architecture. The Masters can reach simultaneously their slaves and if necessary two Masters can exchange data through a shared memory. Generally the access to this memory is managed by the used multiprocessor operating system [11].

To facilitate the design of multiprocessors architectures, ALTERA recently introduce a new component. It is an RTOS module implemented in hardware called "mutex" to ensure the management of the access to the memories shared between the various processors [12]. The mutex provides an operation "test-and-set" containing material, permitting the software, in a multiprocessor environment, to determine the processor that possesses the access to a shared resource. The mutex is used in the conjunction with the shared memory to implement the coordination devices of inter complementary processors.

This proposed new architecture is implemented using ALTERA kit as shown in figure 5. The architecture is composed of a group of sub-systems which can communicate together through divided memories from which the access is protected by mutex Hardware.

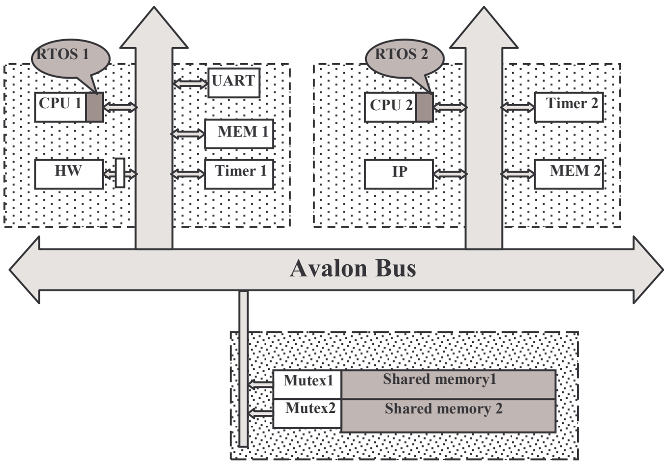

Figure 5. proposed topology of multiprocessor architecture

B. *Generic interprocessor communication layer*

Within the framework of a multiprocessor system, it is necessary that the processors communicate one with the other. This communication is achieved by sending messages (Message Passing) [13].

Generally, the communications can be synchronous or asynchronous, blocking or not blocking. The synchronous communications are slower and are generally blocking, i.e. the two processors engaged in the communication must wait for the end of the communication to continue.

On the other hand the asynchronous communications are most of the time not blocking. In fact, when a processor wants to send a message to another, it sends the message, and it can immediately take again its execution without worrying about the other processor when it receives the message. It is the main advantage of the asynchronous communications.

In our case we consider a shared memory whose access is protected by the mutex hardware and a message line (Queuing Message) through which the processes of the nodes send and receive messages. Thus, the layer of developed communication is a whole of routines making it possible to send and receive messages in various ways by combining the following parameters:

- Send/Receive blocking and not blocking;
- Send/Receive of single or composed messages;
- Send of messages synchronous or asynchronous.

Hence, we could implement a layer of interprocessor communication. The whole communication routines are implemented according to the following parameters': communication mode (blocking or not blocking), the communication type (sending or reception) and the message type (single or composed). More details about this communication layer will be given in future publications.

IV. Experimental results:

In this section, we present preliminary experiments using our new multiprocessors design. The aim of these experiments is to study the exploration of architecture design for the 3D graphic pipeline application (without texture mapping). (Cf. fig6) [14].

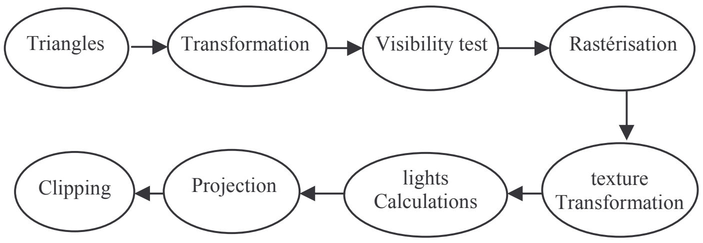

Figure 6. 3D graphic pipeline

We conceive a set of architectures for this application using one and two processors respectively equipped with specialized HW accelerators and coprocessors. For monoprocessor architectures, we carried out the implementation in hardware of some module of this application in the form of accelerators and coprocessors (implemented as custom instructions). For this, a study of this application was made to release the SW tasks which are computationally intensive and which are repeated several times. Those critical taks were implemented in hardware as accelerators and interfaced with the bus Avalon through the PIO[5] technique. We implement complex

[5] Paralel Input Output

mathematic operations (such as multiplication, division) as coprocessors to accelerate their execution locally.

The synthesis application of 3D image was decomposed into 11 tasks (corresponding to the 8 stages shown in figure 6) which cooperate between each other to fulfill the total function of the system. A manual stage of partitioning is performed to assign to each processor the tasks to be carried out and the data to be exchanged with the other processors.

The Table1 presents the results of the execution of the 3D synthesis images application on various architectural targets.

TABLE I. EXPERIMENTS RESULTS OF ON A SET OF ARCHITECTURES RUNNING 3D IMAGE SYNTHESIS

	1 cpu	1cpu+4co Proc	1cpu+acc	1cpu+4coproc +acc	2 cpu	2cpu + 4coproc
Total Aluts	3505	4676	4254	5425	9020	9153
Utilisation ratio of the ALUTS	7%	9%	8%	10%	18%	18%
Total memory bits	571136	571136	571136	571136	657920	657920
Utilisation ratio of memory	22%	22%	22%	22%	25%	25%
Execution time (tic)	1071095628	720223958	707872710	490876456	717162554	570376431
Profit % 1 CPU	**0,00%**	**32,75%**	**33,91%**	**54,17%**	**33,04%**	**46,74%**

We notice, according to the table above, that by using two processors, we will have the same profit (in execution time) obtained as with one processor equipped with accelerators or coprocessors, but, conversely, we will find that we lost, in terms of circuit surface, considering the increase in the number of resources used.

This result can be explained with the sequential nature of the target application. Indeed, to get the best results form multiprocessor architectures, the application must have a spatial parallelism. The total acceleration of this sequential application depends on the acceleration of its critical stages using HW accelerators and coprocessors. This shows that the nature of the target application can deeply influence the performance of the system even equipped with high performance architecture. We are planning to use a high level codesign tool "Design Trotter" [15] to select the more suitable applications for multiprocessing applications.

V. CONCLUSION

The work undertaken enabled us to examine in advance the constraints and the problems brought about by the prototyping of the multiprocessors real time systems on reconfigurable architectures using a monoprocessor RTOS. Our aim is the set up of a multiprocessor platform and the proposal of a generic layer of inter-processor communication which allows the adaptation of the operating systems single processor for multiprocessor architectures. In this paper we focus on the first aim. The second aim is the object for our future publication. We set up a new multiprocessor platform and we validate it through the 3D image processing application. As a future work, we plan to analyze the benefit of the new architecture over the existing architectures using more suitable application for multiprocessor's architecture.

REFERENCES

[1] Bambha, N. Kianzad, V., Khandelia, and Bhattacharrya, "Intermediate Representations for Design Automation of Multiprocessor DSP Systems". In Design Automation for Embedded Systems, vol. 7, 307-323, Kluwer Academic Publishers, 2002.

[2] L.Wang and N. Manjikian. "A performance study of chip multiprocessors with integrated dram". In Proc. 2003 Symp. on Perf. Eval. of Computer and Telecommunications Systems, Montreal, Quebec, July 2003.

[3] N. Manjikian. "Multiprocessor enhancements of the SimpleScalar tool set". ACM Computer architecture News, 29(1):8–15, March 2001.

[4] Le Moigne, R. Pasquier, O. Calvez, J.-P. "A generic RTOS model for real-time systems simulation with systemC", Design, Automation and Test in Europe Conference and Exhibition, Feb. 2004.

[5] D. Shin and J. Kim. "Power-Aware Scheduling of Conditional Task Graphs in Real-Time Multiprocessor Systems. In Proc". International Symposium on Low Power Electronics and Design (ISLPED), August 2003.

[6] MDARTS: "A Multiprocessor Database Architecture for Hard Real-Time Systems" IEEE transactions on knowledge and data engineering, VOL. 12, NO. 4, JULY/AUGUST 2000

[7] "Conception d'un système à haute performance, le calcul parallèle" , CETMEF 2004.

[8] Amer BAGHDADI: "Exploration et conception systématique d'architectures multiprocesseurs monopuces dédiées à des applications spécifiques"thèse PhD, TIMA France.

[9] W. Daniel Hillia and Lewis W.Tucker " The CM-5 Connection Machine : A Scalable Supercomputer". Communication of the ACM, November 1993, Vol. 36, No. 11.

[10] J.J. Labrosse, "Micro C/OS-II, the Real-Time Kernel", Second Edition.

[11] http:// www.altera.com

[12] K. loukil, Y. aoudni, G. Gogniat, M.abid, J.L. philippe, "Estimation du tesmps d'exécution des systèmes sur puce temps réel". GEI 2007

[13] G. Krawezik "Présentation rapide de MPI : Message Passing Interface ", LRI – Université de Paris Sud, EADS CCR – Blagnac, 28 Octobre 2003.

[14] K. Loukil et H. ben chikha : "Conception d'accélérateurs pour le traitement d'images 3D ". rapport Juin 2003

[15] Y. le moulec, J.P. Diguet, N. B. Amor, T. Gourdeaux and J. L. Philippe, "Algorithmic-level Specification and Characterization of Embedded Multimedia Applications with Design Trotter", Journal of VLSI Signal Processing 42, 185–208, 2006