Real-time visual tracking system modelling in MPSoC using platform based design

Article *in* Proceedings of SPIE - The International Society for Optical Engineering · February 2009

DOI: 10.1117/12.811332

5 authors, including:

Tomás Bautista
Universidad de Las Palmas de Gran Canaria
41 PUBLICATIONS **186** CITATIONS
SEE PROFILE

Antonio Nunez
Universidad de Las Palmas de Gran Canaria
141 PUBLICATIONS **438** CITATIONS
SEE PROFILE

Cayetano Guerra Artal
Universidad de Las Palmas de Gran Canaria
41 PUBLICATIONS **307** CITATIONS
SEE PROFILE

Mario Hernandez
Universidad Autónoma de Guadalajara Campus Tabasco
24 PUBLICATIONS **209** CITATIONS
SEE PROFILE

Real-time visual tracking system modelling in MPSoC using platform based design

Zai Jian Jia[*a], Tomás Bautista[a], Antonio Núñez[a], Cayetano Guerra[b] and Mario Hernández[b]
[a]Research Institute for Applied Microelectronics, University of Las Palmas de Gran Canaria
[b]Research Institute for Intelligent Systems, University of Las Palmas de Gran Canaria

ABSTRACT

In this paper, we present the modelling of a real-time tracking system on a Multi-Processor System on Chip (MPSoC). Our final goal is to build a more complex computer vision system (CVS) by integrating several applications in a modular way, which performs different kind of data processing issues but sharing a common platform, and this way, a solution for a set of applications using the same architecture is offered and not just for one application. In our current work, a visual tracking system with real-time behaviour (25 frames/sec) is used like a reference application, and also, guidelines for our future CVS applications development. Our algorithm written in C++ is based on correlation technique and the threshold dynamic update approach. After an initial computational complexity analysis, a task-graph was generated from this tracking algorithm. Concurrently with this functionality correctness analysis, a generic model of multi-processor platform was developed. Finally, the tracking system performance mapped onto the proposed architecture and shared resource usage were analyzed to determine the real architecture capacity, and also to find out possible bottlenecks in order to propose new solutions which allow more applications to be mapped on the platform template in the future.

Keywords: MPSoC, Real-time visual tracking, platform based design, multi-applications performance analysis

1. INTRODUCTION

Real-time computer vision systems (CVS) have gained an increasing importance in many fields, including applications such as robotics, multimedia, industrial inspection, medical imaging, and autonomous navigation. Many of these CVS have been developed with notable contributions, although these new ideas and powerful techniques are usually focused on solving specific and partial problems using *ad hoc* solutions. Moreover, only a few of them have considered the problem of extending the designed system to newer contexts than those initially considered. This fact makes that these solutions are hardly reusable for solving other vision problems than those originally considered.

In this way, our motivation pursues to build a complex CVS in an incremental and modular way using programmable devices which share a common but reconfigurable system architecture. This latter emphasizes that there must be not only individual programmable processing resources on a chip, but also the chip must be considered programmable as a whole. Using a predefined platform, this framework offers a solution not just for one application but for a family of applications which are integrated to form a complex system.

In this paper, our current results are presented. Initially, a visual tracking algorithm based on the correlation technique, part of a CVS still in development, is modelled and analyzed on the proposed architecture. The dynamic pattern or reference block update is the main difference with regard to other tracking algorithms based on the correlation method. This visual tracking system can track the movement of a target in real-time (25 frames/sec), and it is modelled using a platform-based design (PBD) approach. Compared with traditional HW/SW Co-design methodologies, platform-based design involves numerous advantages such as raising the design team productivity, maximizing design reuse, and reducing the risk and effort in SoCs design[1].

Although there are many approaches to real-time video analysis, the modelling of an embedded system which is capable to incorporate further functionality in the future together with this tracking mechanism is the key target of this work, and this way, a solution for a set of applications using the same architecture is offered and not just for one application. In order to measure the capacity of this architecture for future integration of other applications and also to detect and correct

[*]cjia@iuma.ulpgc.es; phone: +34 928 451146

Real-Time Image and Video Processing 2009, edited by Nasser Kehtarnavaz, Matthias F. Carlsohn, Proc. of SPIE-IS&T Electronic Imaging, SPIE Vol. 7244, 72440G · © 2009 SPIE-IS&T · CCC code: 0277-786X/09/$18 · doi: 10.1117/12.811332

inefficient shared resource usage, overall system performance is analyzed when additional traffic in it is taken into account.

The remainder of the paper is organized as follows. In the next section, related works are surveyed. In section 3, the design flow and simulation tool used in this research are presented. Our tracking algorithm is introduced in section 4, and different system implementations are analyzed in section 5 and section 6. Finally, conclusions and future works are drawn in section 7 and section 8, respectively.

2. RELATED WORKS

A lot of work in the field of image processing for CVS has been developed. Wayne Wolf et al. have developed a Smart Camera System[1]. A difference compared with this study is the language used in their design. They initially used MatLab to develop their algorithms, and they later needed to port their MatLab implementation into C code.

Also research on reconfigurable computing has been done based on FPGAs due to their flexible configuration and their shorter design cycles compared to ASICs. Heidi Ziegler et al. present a system composed by several FPGAs, representing each one a single pipeline stage[2]. There are two main disadvantages in this work; on one hand, they need several storage elements to connect the different adjacent stages, so when the number of stages increases, also does the number of memories, what may be not efficient. On the other hand, they need to translate into behavioural VHDL the part of code to be mapped on FPGAs, and in C the rest of the code to run on the microprocessor. Therefore the designer has to make a big effort doing this if the system complexity grows.

Jason Schlessman et al. propose an FPGA implementation of an embedded vision system[3]. Their algorithm is based on optical flow and they used a HW/SW Co-design methodology in the traditional way. Sebastien C. Wong et al. also propose a tracking system initially implemented in MatLab and converted to C++ later, but unlike in the works mentioned above, a single FPGA was used[4].

All these works are *ad hoc* solutions for each treated problem, but they are dependent of a single application, and any modification in the original considerations could imply to redesign some system elements, and even in the worst case, the whole system.

3. METHODOLOGY

3.1 Design flow

Modular design used with an adequate and reconfigurable architecture could be a possible solution to deal with a multi-application SoC problem. Modular design considers a complex system like a combination of several applications which are built with smaller modules. This approach allows organizing and mapping these modules in different ways according to the system modifications or new restrictions. Obviously, in order to work in this way, the architecture should possess a certain level of programmability and reconfigurability.

This work is based on the platform-based design (PBD) concept, which can handle both approaches commented above. Advantages of PBD have been widely discussed in the literature[5,6]. Together with this approach, Keutzer et al. introduced the notion of orthogonalization of concerns[7]. This concept promotes the separation of functionality from architecture and the separation of communication from computation, being the Y-Chart scheme a diagram that properly reflects this idea. The Y-Chart helps simplifying the design space exploration process by modelling independently functionality and architecture, and later on combining them in a separated mapping phase. A design flow according to the above is applied in this research, and it is shown in Fig. 1.

3.2 Simulation tool

Exploring today's complex SoC designs at the RTL level is an intimidating aspect. Not only is RTL simulation speed too slow to allow adequate coverage of the large design space in modern SoC designs, but making small changes in the design can require considerable re-engineering effort due to the highly complex nature of these systems. To overcome these limitations, system designers have been forced to raise the level of abstraction of these models. Since these models are faster to develop and simulate than RTL models, designers can explore and verify the system and make design decisions early in the design flow, which improves the design quality and reduces design time.

In order to work on a higher abstraction level than RTL, new simulation tools are being developed. A SystemC-based system-level simulation environment, called CASSE, is used in this work. This tool provides a unified environment for the whole design flow shown in Fig. 1. An interesting feature of CASSE is that the user controls all the stages in the design flow by means of textual description files. These description files are parsed by the tool during elaboration time in order to create and properly configure the simulation of the desired system model. The result is an executable model that is executed using the SystemC kernel. Since changing the description files does not require recompiling the existing models, extensive parameters sweeps can be performed easily using scripts. Using CASSE, both results of functional and performance simulation can be obtained. The current tool version generates numerous results spanning busses load, memory access load, processor idle and computation time. Unfortunately, CASSE is a tool under development at the moment. Due to this reason, other important metrics in SoC design such as power consumption and area/cost are not yet included in the analysis presented in this paper. More information about the internal structure of the tool can be found[8,9,10].

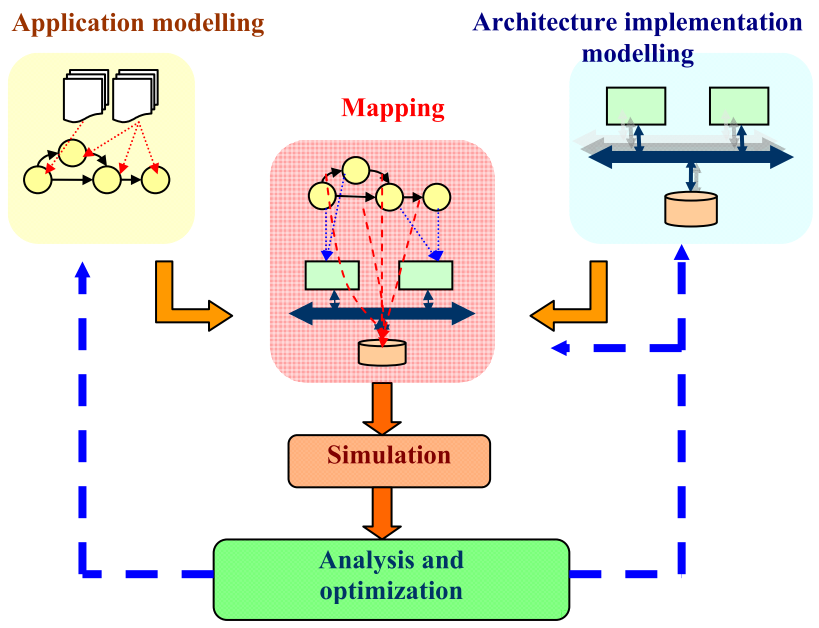
Application modelling
Architecture implementation modelling
Mapping
Simulation
Analysis and optimization

Fig. 1. Design flow used in this work.

4. REAL-TIME TRACKING ALGORITHM

CVS is usually composed of different types of applications, which may include algorithms for detection, recognition and tracking. Although our final objective is to obtain a CVS composed of these applications, at this moment, a visual tracking algorithm based on correlation or block matching is being used as main reference application, which works with the real-time requirement of 25 frames/sec. The task graph of the tracking algorithm is shown in Fig. 2. This application modelling allows establishing a preliminary experimental evaluation of the modular design approach, and also guidelines for our future CVS applications development.

Several methods have been proposed for tracking. Some authors employ optical flow[4], while others work into the frequency domain using the Fast Fourier Transform (FFT)[11]. Research has also been developed working with grouping several features extracted from one frame, and then trying to match them in the next frame[12]. These features can be anything that image processing techniques can extract quickly, such as line segments, points and corners. Finally, there are also researchers who studied the tracking algorithm based on a correlation computation[13].

Correlation or block matching is the technique used to develop the tracking algorithm in this work. Our algorithm implies an initial image format conversion. Thus, a small piece of the original image area with the target is extracted, and it is used like a pattern or reference image in the search. The next step is to perform the block matching process, i.e., to compare the pattern from the current frame within a search area to measure how well a candidate block matches the reference block. The matching criterion used is known as the sum-absolute difference (SAD). It requires the execution of simple operations (sum, subtraction and absolute value), and its expression is the following:

$$S_{ij} = \sum_{x=0}^{m-1} \sum_{y=0}^{n-1} \left| image_{i+x,j+y} - pattern_{x,y} \right| \tag{1}$$

Sij is called a distortion value and a smaller value implies a higher correlation.

After displacing the pattern for the whole search area, a matrix with all distortion values is obtained. If the minimum distortion is lower than a specific threshold (middle value between the two smallest distortion values), then the displacements (*i* and *j*) correspond to the position *(x,y)* of the target object in the current image, and the same pattern can be used to match the next coming image. In this other case, a new threshold and pattern need to be determined before working with a new image.

Unlike other traditional tracking algorithms based on the correlation method, the threshold dynamic update approach is the main difference of our algorithm. The pattern is updated only when there is a probability to lose the target object, which is indicated by the minimum distortion value with respect to the threshold. This issue avoids unnecessary matching operations and also reduces substantially the number of execution operations and accesses to the memory.

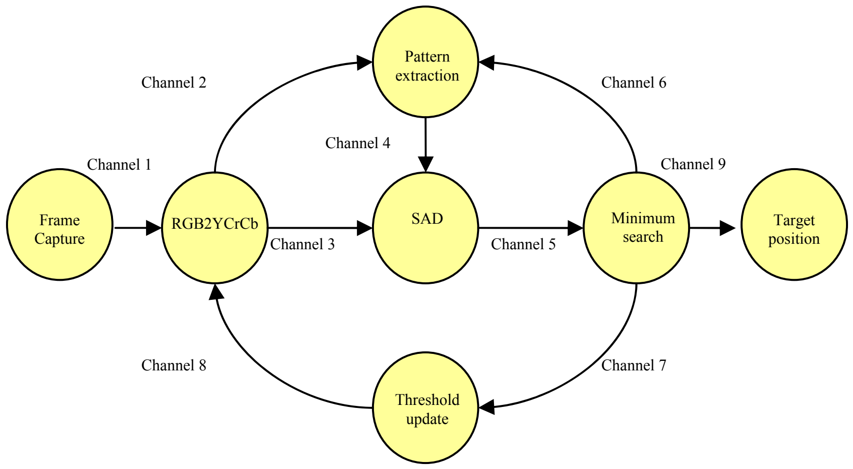

Fig. 2. Tasks graph of our tracking algorithm.

5. TRACKING SYSTEM MODELLING

5.1 A single processor solution

As it has been stated above, a tracking algorithm was developed in C++ and no kind of software optimization was applied. This system receives images data from a video source, which was captured in uncompressed RGB format with a resolution of 320×240 pixels. The search area in the image is of 160×96 pixels, being 24×24 pixels the pattern size. Later, this code was compiled on ADS (ARM Developer Suite) Version 1.2[11] for a reference processor, ARM922T at 200 MHz, and the assembler code and the processor execution time for a frame is examined in order to identify the percentages of time spent in the different sections of the source program. This information is shown on Table I. This information shows that the system performance obtained in this processor is too far to achieve real-time behaviour.

In order to improve the system performance, the computation may be partitioned among multiple processing elements (PE). For this, the modular design concept is applied and thus, the algorithm is decomposed into a tasks graph where parallel tasks communicate with each other by means of unidirectional channels. Tasks are connected to channels via ports, and they communicate and synchronize with each other by calling Task Transaction Level (TTL) interface primitives on their ports[12]. The most computationally costly tasks are listed in Table I. Moreover, the execution time obtained after compiling with the ADS compiler is annotated manually in each task, which allows analysis with benchmarks and design space exploration.

This algorithm decomposition stage into tasks is very important due to several reasons. First, from thus we can determine the order of the execution of tasks, their degree of dependence and the exact data that must be communicated between different tasks. This helps us in the latter performance communication analysis and to determine the exact communication data granularity. Second, the number of communication points or channels on the system can be determined, and the sending points for producers and the receiving points for the consumers in each task entity identified. Finally, during the transformation at the original C++ code into a tasks graph, only read and write primitives for communication have to be added, remaining the rest of code unchanged. This reduces substantially time and effort in the design process.

TABLE I. Main algorithm tasks characteristics in template architecture

PE assigned	Task name	Cycles per frame for a reference processor (ARM922T at 200 MHz)	Number of channel used per task	Access memory size (Bytes)
PE 4	Rgbtoycrcb	3381360	4	15360
PE 1	Pattern	11133	3	592
PE 2	SAD	141462777	3	9792
PE 2	Minimum search	632939	4	9808
PE 1	Others	852079	4	15968

5.2 Multiprocessors solution analysis

Concurrently with the functionality correctness analysis, an implementation of an application-independent architecture can be developed. A given algorithm can be implemented using different architecture solutions, but their performance and complexity could differ markedly. Many aspects must be taken into account in this step. On the one hand, it is interesting to analyze coprocessors requirements to determine the minimum number of processing elements needed, and which tasks are assigned to each processing element. When the number of required processing elements increases, also does the synchronization load on the busses and therefore the transport delays as well. On other hand, it is important to determine whether data will be communicated via shared memory or via direct links between processing elements. These alternatives poses different inter-task data interchange through the network and what kind of information is allocated inside the processing elements. In any case, processing elements and memories number must meet the architecture implementation constraints.

The architecture implementation template proposed in this work is depicted in Fig. 3, and this decision is due to several reasons. On the one hand, this platform allows a variety of PEs, which can be configured like programmable RISC processors, video I/O processors, and specific coprocessors working like hardware accelerators. On the other hand, once the correctness is verified, this platform can be used, and even expanded, to other image applications due to the system flexibility and scalability by using a standard system bus. This reduces future design decisions and shortens the design time. Finally, this architecture could seem over-designed at first. However, this is necessary to not become restricted to a solution for a single problem but to allow responding to a set of situations or a set of applications.

In the case of this work, this template is built as a modular composition of configurable predefined elements provided by the CASSE libraries. For instance, a RISC ARM-9 processor can be obtained configuring the parameters of a processing element (PE), and also a network element (NE) can be configured as an AMBA bus with an arbiter. Bridges can be also incorporated for inter-buses communication in order to adapt different clock domains. These architecture parameters are configured with values which are shown in Fig. 3.

In our initial multiprocessors solution, only three PEs (PE 1, PE 2 and PE 4) were used. From the profiling information shown in Table I, it can be seen that the SAD task is computationally very costly. Moreover, this latter is tightly coupled with the Minimum search task, so both tasks will be assigned to the same hardware accelerator. In this system, a video I/O processor receives incoming frames and it can also execute more efficiently the format conversion task than a generic programmable RISC processor, so this task is mapped, for instance, to PE 4. Finally, once all the channels are mapped on the shared memory and the rest of tasks in the graph are assigned as indicated in the first column of Table I, an executable model is delivered by CASSE for performance simulation. In this model, both functionality and processing timing annotations are taken into account in the architecture model.

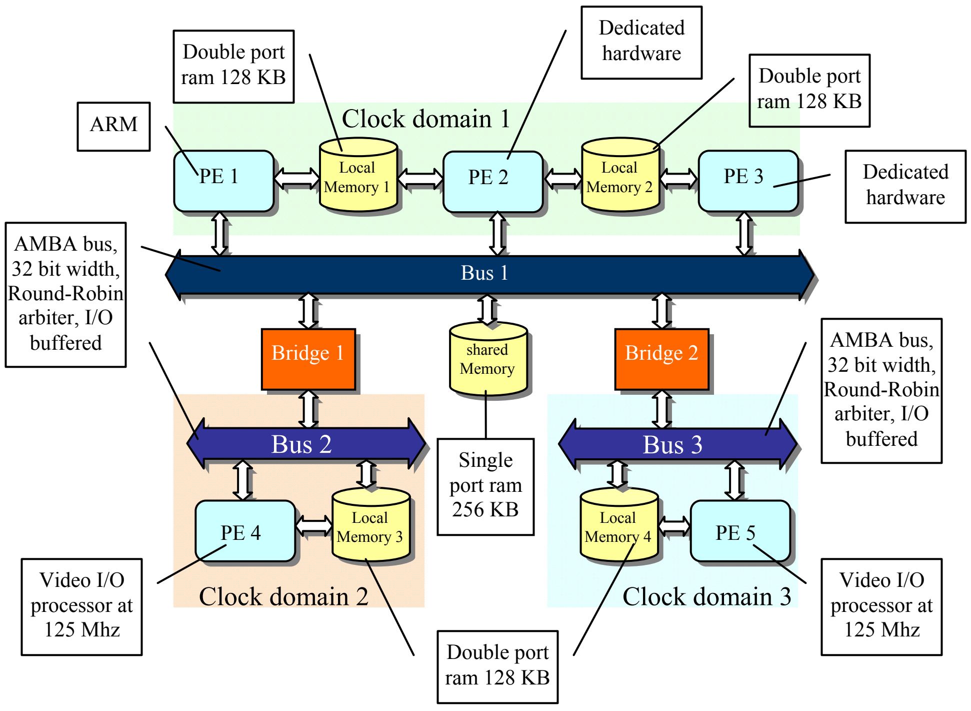

Fig. 3. Architecture template proposed.

Design space was explored and analyzed keeping fixed the bus, memory, video I/O processor and system architecture configurations while dedicated hardware parameters were altered. Fig. 4 shows the performance of the tracking system under different design points. This curve in the x-y plane is an isoperformance surface, which represents the time required for one frame processing with different combinations of dedicated hardware frequencies (from 200 to 800 MHz) and accelerations (from $\times 1$ to $\times 10$). Certainly, the number of possible solutions is very large, but given this system specification, not all implementations that fall into this range are acceptable. The information in Fig. 4 indicates that, for instance, working at 400 MHz with a $\times 10$ hardware acceleration can achieve real-time system specification, while at 600 MHz and with a $\times 5$ acceleration, just 17 frames per second can be processed.

The criterion of minimum system bus load is used to determinate the implementation solution, because this allows a bigger resource time sharing, and then more than one application running at the same time with the tracking system can be mapped in the same architecture proposed. Since the total transported data is fixed, then the minimum synchronization load solution must be chosen in order to enhance the shared resource access efficiency. According to the above, a solution at 400 MHz with $\times 10$ acceleration should be initially chosen.

In this point, it is important to highlight that the total bus load is composed, on the one hand, of the data from read and write operations, and, on the other hand, of all PEs synchronization information in order to access the shared memory via the main system bus, which becomes a bottleneck. If the processing time of the different processing elements is not well balanced, then processes have to wait for the availability of shared resources to access them. When a process wants to read an input from a channel, it first tests the channel for the presence of data. If a data item is available, it is consumed and processed; if no data is present, the process may decide to take some other action or continue testing the channel.

This latter is a common cause of oversynchronization, which produces excessive unnecessary traffic in the system and even could affect the system performance. Fig. 5 shows different synchronization loads produced by several ports per each PE in this architecture. If all PEs are balanced perfectly, both in computation delay and in communication delay, minimum synchronization load is obtained. However, this "ideally balanced" system is not always allowed due to communication delays or transmission effects of the network elements, although the tasks are balanced computationally. Finally, the synchronization loads of the initial solution are also shown in Fig. 5.

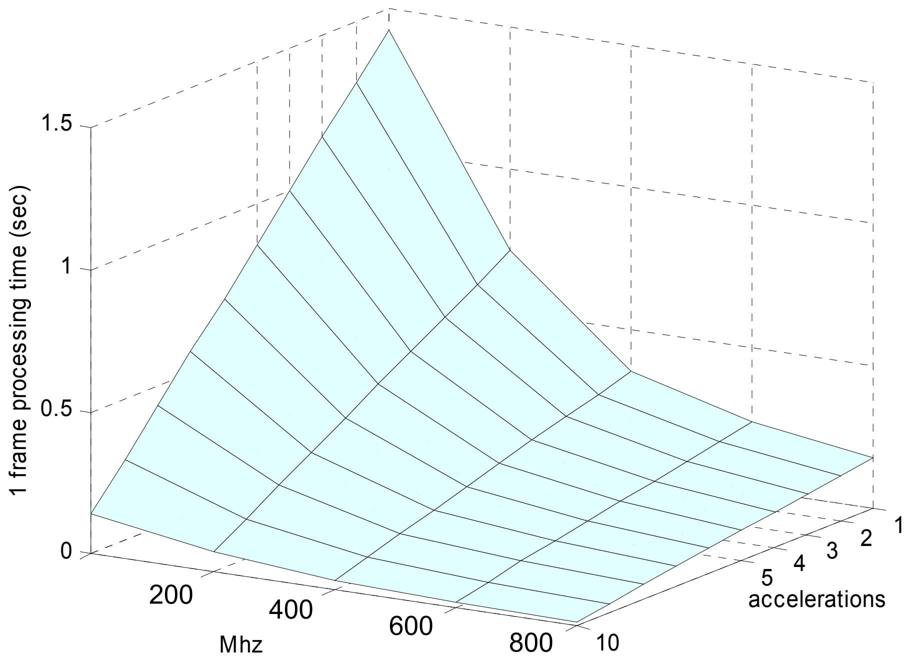

Fig. 4. Performance achievable with no optimized algorithm in architecture proposed.

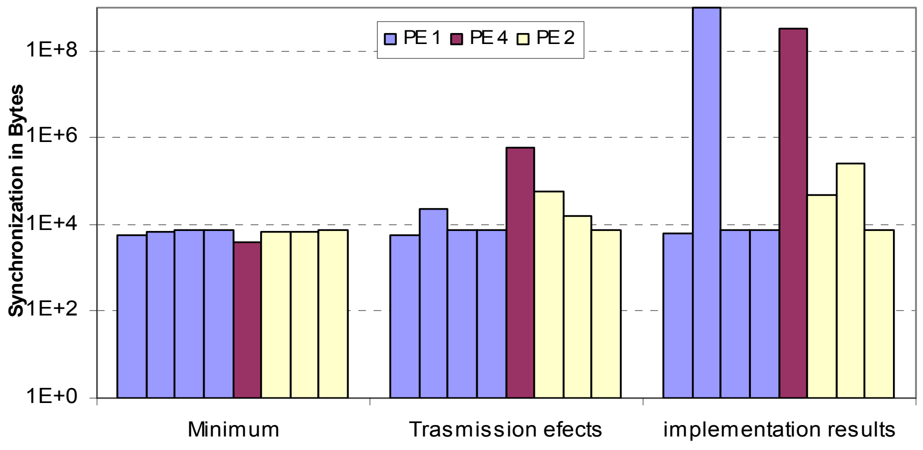

Fig. 5. Synchronization bytes for each port of PE to process 125 frames.

5.3 Optimized solutions analysis

Fig. 5 shows that the initial implementation solution has a clear symptom of oversynchronization in spite of being the solution with the lowest bus load among all possible candidates. This effect is relevant especially in port 2 of PE 1 and

PE 4. This information shows that these ports are blocked too often waiting for data in their channels, and such high load is produced while polling the channels status.

The first optimized solution implies to map the most oversynchronized channel of PE 1 in a local memory. However, the bus load in this solution can be decreased using a local memory for PE 4 synchronization operations, because the communication from a Video I/O processor to main shared memory is too costly in terms of latency and power consumption due to the large path of busses and bridges that data has to cross over. This way, both PE 1 and PE 4 can access the main bus and the shared memory to allocate data, while they access their local memory for synchronization operations. Fig. 6 shows the bus load results due to synchronization for these three options.

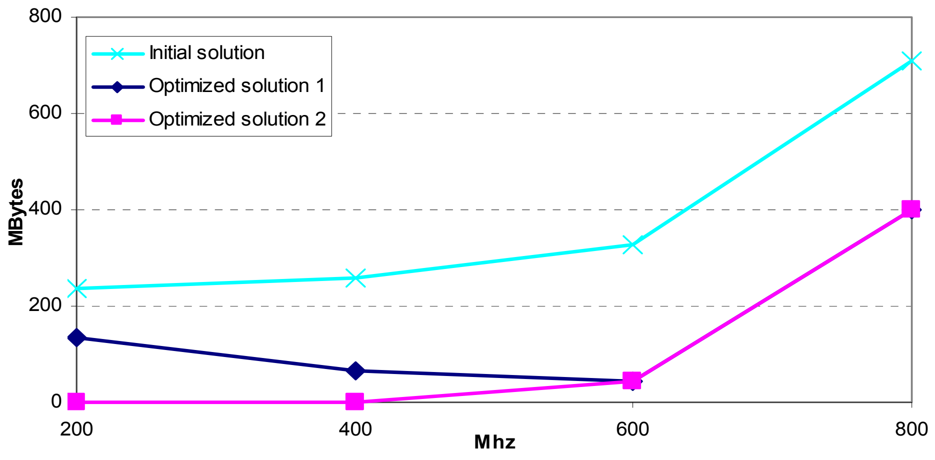

Fig. 6. Solutions based on total synchronization load (data for 25 frames processing).

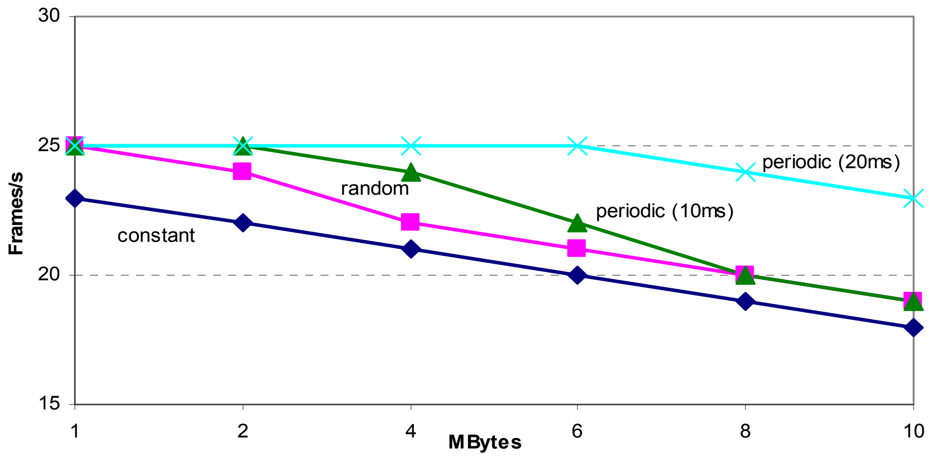

Fig. 7. Tracking system performance with other applications running on the platform.

6. MULTI-APPLICATION ANALYSIS

Following our purpose of building a complete CVS, another specific objective of our analysis is to determine the possibility of mapping other applications in this platform and still complying with the real-time tracking system requirements. The typical characteristics of image-processing, audio-processing, and signal-processing applications, for which regular streams of data are processed at a constant rate or periodically, were taken into account to develop these new applications. Four different types of traffic generated by additional applications in the system were analyzed in an independent way. First, constant traffic, where the new application shares the same shared memory with tracking application. Second, periodic traffic is taken into account, where the new application accesses the main memory at a specific rate, in this case with periods of 10 ms and 20 ms. Finally, random traffic also is included in this analysis. This analysis allows to the designer to know how the target system performance is with regard to additional load produced by another application in the system. The results of this analysis are shown in Fig. 7.

For this study, tasks of the new application are mapped onto PE 3 and PE 5. Moreover, the channels of the new tasks graph are also mapped onto the shared memory. Some experiments were performed varying the data size processed on the new application. Following with the last example, the tracking system performance running at 400 MHz and with a ×10 acceleration is shown in Fig. 7. The results indicate that the tracking system can work at 25 frames/sec when an additional application produces a constant traffic load with data of size lower than 100 Kbytes, or another one which produces traffic load with period of 10 ms, being its data size lower than 2 Mbytes.

7. CONCLUSIONS

The increasing complexity of systems and their applications usually demands for solutions like the used in this paper, with one or several microprocessors. The software of the complete application should be appropriately distributed among them, and mechanisms should be also provided in the system so the different tasks can be executed in the correct order. Therefore, a strategy to appropriately study the different alternatives and their impact in the performance is certainly of interest for the designers of this kind of systems.

The main objective of this work is the research of a system with programmable heterogeneous processors targeted for real-time computer vision applications at a very high abstraction level, being the main contributions the measurement of the performance of the basic tracking system tasks on the reference architecture, and the analysis of the behaviour of these tasks on the platform. This allows us to identify the capacity that can be actually supported by the platform. Moreover, we also analyzed the platform to locate the main bottlenecks when more than one application is mapped on it, in order to propose solutions to improve system efficiency.

The results obtained for different configuration alternatives have then been organized into figures from which the designer can draw conclusions about the best alternatives to cope with the real-time requirements with the elements available and with an affordable cost. With this strategy, decisions at system level can be settled in order to achieve with guarantee the further steps in the design flow down to final implementation.

8. FUTURE WORKS

Our future work will focus on studying the influence of different parameters of architectural elements in the system performance. We are also working on improving our mapping algorithm for the consolidation of our methodology based on platform design for multiprocessor and multi-application.

ACKNOWLEDGEMENT

This research is funded by the Spanish Ministry for Science and Technology, grant AP2006-02986, and Spanish Ministry for Science and Innovation, grant TEC2006-13599-C02.

REFERENCES

[1] Alberto Ferrari, Alberto Sangiovanni-Vincentelli, “System Design: Traditional Concepts and New Paradigms”, IEEE International Conference on Computer Design, ICCD '99, 2-12 (1999).

[2] Wayne Wolf, Burak Ozer, Tiehan lu, “Smart Cameras as Embedded Systems”, IEEE Computer 35(9), 48-53 (2002).

[3] Heidi Ziegler, Byoungro So, Mary Hall, Pedro C. Diniz, “Coarse-Grain Pipelining on Multiple FPGA Architectures”, Proceedings of the 10th Annual IEEE Symposium on Field Programmable Custom Computing Machines (FCCM’02), 77-86 (2002).

[4] Jason Schlessman, Cheng-Yao Chen, Burak Ozer, Kenji Fujino, Dazurou Itoh, Wayne Wolf, "Hardware/Software Co-Design of an FPGA-based Embedded Tracking", IEEE System Conference on Computer Vision and Pattern Recognition Workshop (2006).

[5] Sebastien C. Wong, Mark Jasiunas, David Kearney, "Towards a reconfigurable tracking system", IEEE International Conference on Field Programmable Logic and Applications, 456-462 (2005).

[6] Alberto Sangiovanni-Vincentelli, Grant Martin, "Platform-Based Design and Software Design Methodology for Embedded Systems", IEEE Computer Design & Test of Computers, 18(6), 23-33 (2001).

[7] Kurt Keutzer, Sharad Malik, Richard Newton, Jan M. Racaey, A. Sangiovanni-Vincentelli, "System-Level Design: Ortogonalization of Concerns and Platform-Based Design", IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 19(12), 1523-1543 (2000).

[8] Víctor Reyes, Tomás Bautista, Gustavo Marrero, Pedro P. Carballo, Wido Kruijtzer, "CASSE: A System-Level Modeling and Design-Space Exploration Tool for Multiprocessor Systems-on-Chip", Proceedings of the EUROMICRO Systems on Digital System Design (DSD'04), 476- 483 (2004).

[9] Víctor Reyes, Tomás Bautista, Gustavo Marrero, Antonio Núñez, Wido Kruijtzer, "A Multicast Inter-Task Communication Protocol for Embedded Multiprocessor Systems", Proceedings of the 3rd IEEE/ACM/IFIP international conference on Hardware/software codesign and system synthesis, 267-272 (2005).

[10] Victor Reyes, Wido Kruijtzer, Tomás Bautista, Ghiath Alkadi, Antonio Núñez, "A Unified System-Level Modeling and Simulation Environment for MPSoC design: MPEG-4 Decoder Case Study", Proceedings of the conference on Design, automation and test in Europe, 474-479 (2006).

[11] Mark W. Eklund, Gopalan Ravichandran, Mohan M. Trivedi, Suresh B. Marapane, "Real-Time Visual Tracking Using Correlation Techniques", Proceedings of the Second IEEE Workshop on Applications of Computer Vision, 256-263 (1994).

[12] Raphaël Canals, Anthony Roussel, Jean-Luc Famechon, Sylvie Treuillet, "A Biprocessor-Oriented Vision-Based Target Tracking System", IEEE Transactions on Industrial Electronics, 49(2), 500-506 (2002).

[13] Naoyuki Sawasaki, Toshihiko Morita, Takashi Uchiyama, "Design and Implementation of High-Speed Visual Tracking System for Real-Time Motion Analysis", Proceedings of the 13th International Conference on Pattern Recognition, 3, 478-483 (1996).

[14] ARM Developer Suite, Version 1.2, www.arm.com

[15] P. van der Wolf, E. de Kock, T. Hendrikson, W. Kruijtzer, G. Essink, "Design and Programming of Embedded Multiprocessors: An Interface-Centric Approach", In Proceedings of CODES+ISSS'04 (2004).