Intelligent Vehicles Symposium 2006, June 13-15, 2006, Tokyo, Japan

9-4

Mapping an obstacles detection, stereo vision-based, software application on a multi-processor system-on-chip

Alain Greiner
UPMC / LIP6
4, place Jussieu
75005 Paris
France
Email: alain.greiner@lip6.fr

Frederic Petrot
TIMA - SLS
46, avenue Félix Viallet
38031 Grenoble cedex
France
Email: frederic.petrot@imag.fr

Mathieu Carrier
UPMC / LIP6
4, place Jussieu
75005 Paris
France
Email: mathieu.carrier@lip6.fr

Mounir Benabdenbi
UPMC / LIP6
4, place Jussieu
75005 Paris
France
Email: mounir.benabdenbi@lip6.fr

Roselyne Chotin-Avot
UPMC / LIP6
4, place Jussieu
75005 Paris
France
Email: roselyne.avot@lip6.fr

Raphael Labayrade
LIVIC – INRETS / LCPC
14, route de la miniere
Batiment 824
78000 Versailles France
Email: raphael.labayrade@lcpc.fr

***Abstract*—In this paper, we present the implementation of a multi-threaded software application for pre-crash obstacle detection, using stereo vision, and the "V-disparity" algorithm, that requires intensive computation. This application runs on a generic, low cost, massively parallel, multi-processor system-on-chip (MP-SoC). This hardware architecture is suitable for automotive area with respect to performance, cost, and flexibility constraints. This hardware/software embedded application is able to process 40 stereoscopic pairs per second with 256 lines of 512 pixels images and a disparity range of 256. Our architecture is made of 8 clusters, 30 general-purpose 32-bit processors and 750 Kbytes embedded memory.**

I. INTRODUCTION

In Europe, 90% of the road accidents are caused by human error. 52% are due to a collision. For example, before a collision 39% of drivers do not brake at all. To cope with this fact, car manufacturers are looking for systems to prevent and avoid accidents. For instance, the Mercedes-Benz S-Class introduced in 2002 the PreSafe system, merging the passive and active safety. This system activates the seatbelt tensioners and automatically adjusts the windows and seats. This type of system combines sensors, electronics hardware and actuators. The sensor used is a radar sensor. Since 2002, other systems have been developed as the Pre-Collision System from Denso in 2003, also using a radar sensor. The PCS identifies inevitable obstacles a fraction of second prior to collision, tightens passenger seatbelts and activates a pre-collision brake to reduce vehicle speed. In 2003 Honda introduced the Collision Mitigation Brake System, based on on-board cameras and radar, which detects if inter-vehicular distance is insufficient or collision is unavoidable; in these cases, the system triggers audio and visual warnings, brakes the vehicle and tightens the seatbelts if necessary.

In academic laboratories, different platforms have been studied like the ARGO autonomous vehicle based on GOLD (Generic Obstacle and Lane Detection) developed at University of Parma (Italy) [1][2]. The DARPA challenge exhibits the research work of several US universities in obstacles detection, such as the California Institute of Technology or Stanford University which won this race. Traditionally, car manufacturers/automotive OEMs used ASIC (Application Specific Integrated Circuit). Today, ASICs are replaced by complete systems on chip, integrating both hardware and software components on a single chip for a given embedded application. This approach significantly reduces the fabrication cost, and improves system performances. Thus, SoC will begin to proliferate throughout automotive applications.

This paper presents a multi-processor SoC architecture, implementing an obstacle detection software application, in pre-crash situation. This generic hardware architecture supports intensive computation with several microprocessor cores running in parallel, and is easily reprogrammable. The same hardware can be used for other software applications if required.

In this paper, we focus on the results obtained by mapping a parallel, multi-threaded software application, implementing a stereo vision method, based on the "V-disparity" algorithm [11] on the architecture mentioned above. It achieves a robust and reliable obstacles detection, processing 40 stereoscopic pairs per second with a 256 disparity range. Each image has a resolution of 256 lines of 512 pixels.

The paper presents a review of the related work in section 2. Section 3 details the "V-disparity" algorithm which is used in our study. Section 4 introduces the precise application requirements. Sections 5 & 6 describe respectively the hardware and software architectures. Section 7 presents the hardware/software co-design process used to map the multi-threaded software application on the multi-processor architecture. Finally, section 8 presents the experimental results obtained by cycle-precise SystemC [23] simulation, using the SoCLib [21] platform.

II. Related Work

Since several years, different methods have been evaluated for obstacle detection in road context. Some studies are based on exteroceptive sensors like, laser lidar, laser radar, radar or vision sensors with different approaches: monocular vision, stereo vision, trinocular vision, wide field of view stereo. Last decade, many systems are stereo vision-based due to its robustness and accuracy with different types of obstacles. This paper focuses on a stereo vision-based approach.

For the last decade, hardware stereo vision-based systems have been developed. In 2004, Di Stefano et al.[3] studied a PC-based real-time stereo vision system and surveyed different hardware implementations.

A lot of systems are FPGA-based, due to the reprogramming possibility of FPGAs. Tyzx proposed an FPGA platform (on PCI board) [4] able to reach 200 fps on 512 by 480 resolution images, with 52 disparity range. This platform uses the Census transform. Han et al.[5] have recently developed another FPGA-based designed for robot navigation applications and especially for household robots where a short distance to the obstacle is very frequent. They reached 60 fps on 640 by 480 resolution images with a 128 disparity range. These approaches show good results, but these FPGA-based boards are very costly, and not suitable for automotive applications requiring ultra-low costs.

Other proposals are based on dedicated ASICs (Application Specific Integrated Circuit) [6][7]. PAPRICA[8] is a parallel image processing architecture employing a 16x16 square matrix of 1-bit PEs (Processing Elements) operating in a SIMD (Single-Instruction Multiple-Data) style, achieving processing of about 8 fps for lane detection. Jeong et al. presented a VLSI architecture for a highly parallel stereo matching algorithm [9]. The architecture is a systolic array with a strong parallelism, a neighborhood connectivity and a global clock. This architecture with 10 PEs requires 40 ms to compute a disparity map for 320 by 240 pixels images. Recently, Techmer et al. from Infineon proposed a vision platform for intelligent vehicles [10] able to achieve contour extraction in 520 µs on 320 by 240 pixels image. This complex platform mainly contains a DSP, a programmable SIMD processing array and memory banks. Of course, these SIMD architectures, using dedicated PEs arrays are very efficient, but they are not flexible.

For automotive applications, high speed, flexibility, and low cost are required. As will be described in section 4, our approach is based on MIMD (Multiple-Instructions, Multiple-Data) parallelism, and the selected hardware architecture is a single chip, shared-memory SMP (Symmetric Multi-Processor).

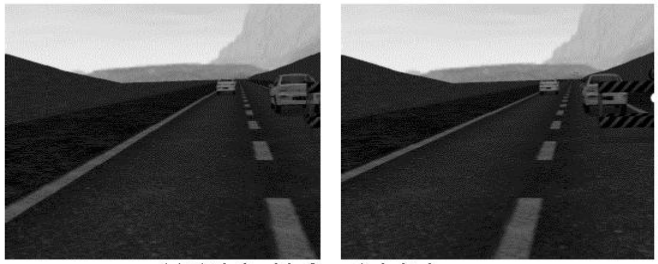
III. The Proposed Approach

The algorithm used in this study is based on SAD (Sum of Absolute Differences) and "V-disparity" approach. This algorithm works whatever the road surface is (due to the estimation of the road longitudinal profile) [11][12]. The detection is generic, even in the event case of partial occlusion, the system is robust with respect to noise or bad conditions (weather, night)[12]. Thus, this algorithm is suitable for obstacle detection in road context (moreover it has been recently used by Broggi et al. in [13], Matuszyk et al. in [14], Gandhi et al. in [15]).

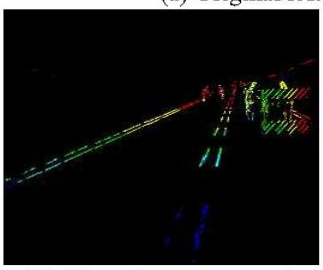
The study we expose performs stereo vision-based obstacle detection in pre-crash situation (the obstacle distance is about 6 meters). The application includes 5 main components:

- grabbing the right and left images,
- computing a disparity map,
- computing the "v-disparity" image,
- extracting global surfaces through Hough Transform,
- estimating obstacles position and free road surface.

The "V-disparity" method requires intensive computations. An example is shown in Fig. 1. The first step consists in grabbing a pair of stereoscopic images. These two images of the same scene, with two different angles of the scene (a), will allow to rebuild the 3D road scene. From the stereoscopic pair, a sparse disparity map is computed (b) and used to build the "V-disparity" map (c). Thanks to the Hough transform, the longitudinal profile of the road (d) is estimated allowing to extract objects above the road surface as potential obstacles.

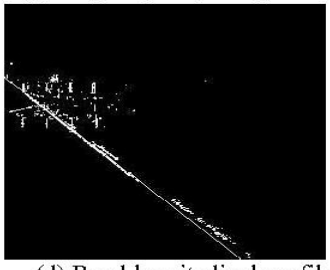


(a) Original left and right images

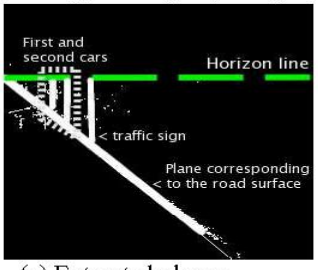


(b) Disparity map computed

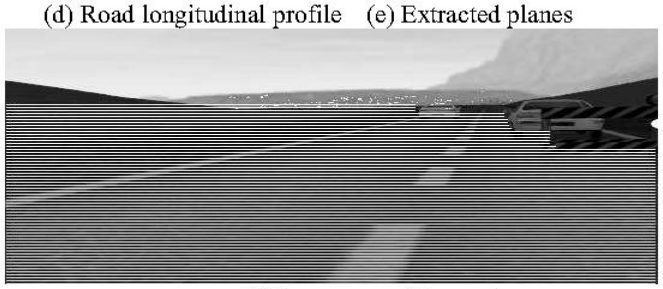
(c) "V-disparity" map



(d) Road longitudinal profile



(e) Extracted planes



(f) Free space of the road

Fig. 1. Steps of the "V-disparity" algorithm

Then road obstacles are detected accurately, each one corresponding to a distinct plan in the "V-disparity" image (e). The final step extracts the free road space (f), which is the part of the road without obstacle. This method does not require any extraction of lane-markings but exploits all the relevant information in the image (texture of the road, shadows, road edges and so on).

This algorithm has been implemented by the LIVIC laboratory as a sequential C program, using only integer numbers. It has been evaluated on a Intel Pentium IV 1.4 GHz running under Windows 2000. Images are grabbed using a Matrox Meteor II graphic board. Image size is 380 by 288 pixels and the disparity range is 150. With this system, the computation time is about 40 ms for a stereoscopic pair.

This computation time, on a single processor PC is too large for a pre-crash detection, and is too costly to introduce a PC in each car. Fortunately, this algorithm can be parallelized, as a multi-threaded application, using coarse-grain pipe-lining: several threads may work in parallel on different lines of a given images pair. Therefore, we decided to use a multi-processor system-on-chip as the hardware target.

Our approach differs from previous approaches based on mono-processor architectures, or SIMD architectures, such as [3][9][10]. The proposed hardware architecture is a clusterized SMP (Symmetric Multi-Processor) where several general-purpose processors share the same address space. Such generic architecture is well suited for task parallelism exploitation, and is generic enough to support other multi-threaded software applications.

The inter-task communications, the shared resources and the real-time constraints are managed by a POSIX compliant real-time embedded operating system developed at LIP6 [16] for such shared memory, MP-SoC architectures.

IV. Detailed Application Requirements and Physical Constraints

The images resolution is 256 lines of 512 pixels, 8-bit grey level, which means 128 Kbytes per image. The disparity range is 256. We have chosen a rather low resolution because, in pre-crash situation, the obstacle is so close that a higher resolution would not be useful. The wide disparity range allows detecting obstacles close to the vehicle. Images are grabbed by two synchronized cameras connected to the MP-SoC by two separated Firewire busses.

In pre-crash situation, timing constraints are strong: the time spent by a vehicle to go forward 6 meters, with a 36 meter/s velocity, is about 150 ms. This time is allocated before a collision to detect the obstacle presence and activate the safety systems (in and outdoor airbags, seat belts, emergency / assistance call and so on). However, safety systems activation requires 125 ms. Thus, the time allocated to the stereo vision algorithm to detect an obstacle is 25 ms.

As the target fabrication process for the MP-SoC is a 90 nm CMOS technology, it is reasonable to assume an internal clock speed of 300 MHz. So, the 25 ms duration corresponds to 7 500 000 cycles, which is the maximal number of cycles to process a stereoscopic pair. On the other hand, the parallel multi-thread application requires a rather large embedded memory (for both inter-thread communication channels and image storage).With this 90nm CMOS process, we allocated a total budget of about 1 Mbyte of embedded SRAM. Of course this storage capacity will be physically distributed between several memory banks in our multi-processor

architecture. Finally, we are targeting a chip area no larger than 60 mm2.

V. HARDWARE ARCHITECTURE

In this section, we describe the selected hardware architecture, which is a clusterized multi-processor system-on-chip. As described in Fig. 5, this generic architecture contains a variable number of clusters, respecting a 2D mesh topology.

A cluster is a synchronous sub-system containing a variable number of 32-bit general-purpose microprocessors (up to 4 processors per cluster). All processors are MIPS R3000 RISC microprocessors (including instruction and data caches). Each cluster contains also two local memory banks (SRAM), and several system peripherals (Timer, Locks engine, Interrupt controller, etc.). Fig. 2 is an example of cluster. Inside a cluster, a local "system bus" interconnects the processors to the local memory and peripherals. The communications between clusters are supported by the DSPIN integrated micro-network [17]. Each cluster accesses the network through a dedicated NIC (Network Interface Controller). This packet-switched network on chip, respects the VCI/OCP standard [18], providing the system designer a "flat" address space for the whole entire system: any processor in cluster i can directly address any memory or peripheral in cluster j. This network architecture is adapted to the GALS (Globally Asynchronous Locally Synchronous) paradigm, as each cluster can be clocked by a different system clock. In our case all clock signals have the same frequency, but different phases.

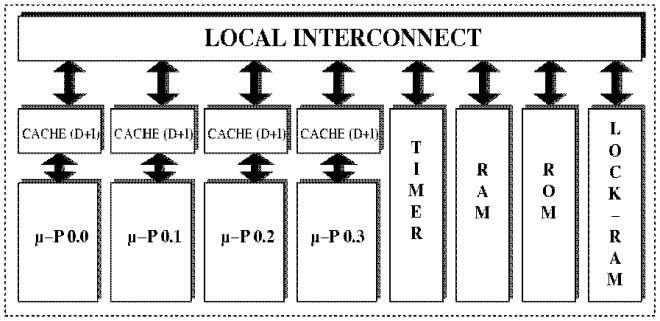


Fig. 2. Internal cluster architecture

This hardware architecture is very generic since it involves only general-purpose microprocessors. For this stereo vision application, we use 8 clusters, and each cluster contains 4 processors (entire hardware architecture is illustrated in Fig. 5). As described in the next section, the number of processors has been defined in order to fit the number of software threads. Two clusters are specialized, as they contain the hardware controllers to interface the two cameras, through the Firewire bus. Those I/O controllers have of course a DMA (Direct Memory Access) capability. The two external cameras are synchronized by the MP-SoC, thanks to an adjustable timer.

VI. SOFTWARE ARCHITECTURE

The initial software application was developed by the LIVIC laboratory as a C program targeting a mono-processor PC. The execution time on this mono-processor architecture was about 40 ms on smaller images (380 lines of 288 pixels) and a disparity range of 150. Our first task was to transform this sequential C program into a multi-thread application, in order to be efficiently executed on our multi-processor hardware architecture.

The MUTEK operating system supports multi-tasking (i.e. several threads running on the same processor, with time-slicing), as this method reduces silicon area and SoC cost. For performance reasons, we decided to have only one thread per processor in order to avoid all the context switching penalties.

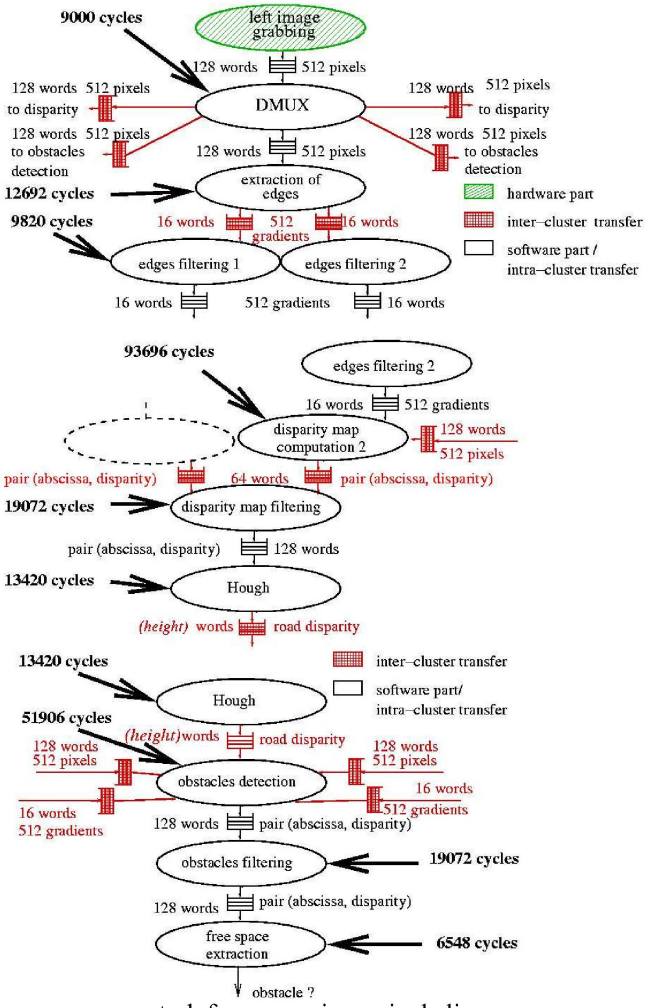
The multi-threaded application uses the inter-tasks communication infrastructure provided by the DISYDENT [19] hardware/software codesign environment. The entire software application is described as a task graph, where all tasks are running in parallel and communicate only through point to point communication channels, using blocking "read" & "write" primitives. Each communication channel is implemented as a shared memory buffer, working as a software FIFO (First-In First-Out), protected by a specific lock for exclusive access. Following the Kahn Process Networks semantic [20], the synchronization between tasks is entirely done by the exchanged data, as long as the real-time constraint is respected.

Both the channel communications and the resynchronization (in case of violation of the real-time constraint), are supported by the MUTEK real-time operating system.

For the largest part of the disparity map computation, the processing is done line per line. Therefore, we rely on coarse grain pipe-line parallelism: each task performs a part of the computation on a single line, and transfers the resulting data to the next task in the pipe-line. Of course, it has been necessary to equilibrate the computing time between tasks in order to have a balanced pipe-line. It must be noticed that the longitudinal profile of the road is extracted through the Hough transform, that requires the entire image and is not the last task of the application. Fig. 3 illustrates the tasks graph. It presents the tasks that belong to the critical chain, and the trade-off between execution time

and communication. This parallelization effort led us to 30 parallel tasks.

Fig. 3. Task graph of entire application, number of cycles per



task for processing a single line

VII. Architecture Modelling And Application Mapping

The hardware architecture has been modeled using the library [21]. SoCLib is a modeling and simulation platform for multi-processor system-on-chip, that can be used with the DISYDENT framework. It contains a set of parameterized, cycle-accurate, simulation models for reusable hardware components (IP cores), written in SystemC[23]. The SoCLib library includes microprocessor cores, memory controllers, bus controllers, and dedicated system peripherals. All the hardware parameters values (such as cache sizes) must be defined at design time. The SoCLib library allows the system designer to build a multi-cluster, multi-processor hardware architecture dedicated to a specific application. All IP cores respect the "Virtual Component Interface" standard [18], normalized by the VSIA consortium.

For this stereo vision application, each cluster contains 4 MIPS processors and 2 physical memory banks of 64 Kbytes. For all processors the cache size has been fixed to 4 Kbytes for both Data and Instruction caches (128 lines of 32 bytes). For the targeted 90 nm CMOS fabrication process, the cluster area is about 8 mm2.

Regarding the software side, the first step was to validate the complete, multi-threaded software application. This multi-threaded C program has been compiled for a LINUX PC, and linked with the POSIX library. As the inter-threads channel communication primitives provided by the DISYDENT environment have been developed on top of a POSIX compliant operating system, it is possible to execute directly the software application on the LINUX PC. In this fully software approach, the two I/O controllers interfacing the cameras are replaced by two specific software threads directly reading the images on the LINUX PC disk.

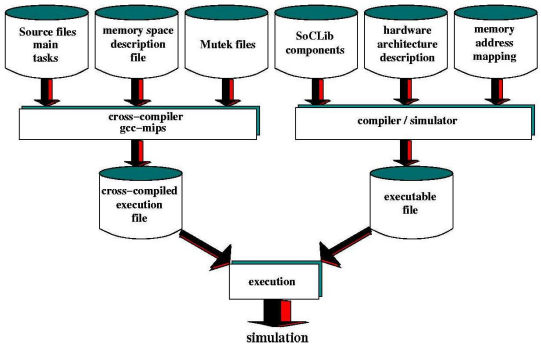


Fig. 4. Compilation/simulation chain

The next step was to map the multi-threaded software application on the multi-processor SoC (of course, the two threads emulating the two I/O controllers must be removed, as those I/O controllers are now part of the simulated architecture). In a clusterized architecture, the placement of software tasks on the hardware processors, as well as the placement of the software communication buffers on the hardware memory banks is very important: a local transaction (such as a read memory access inside a cluster), uses only the local interconnect and requires about 10 cycles. A global transaction (such as a read memory access to a memory located in another cluster) uses the local interconnect in the source cluster, goes through the DSPIN micronetwork, reaches the local interconnect in the destination cluster, and back... This requires about 50 cycles.

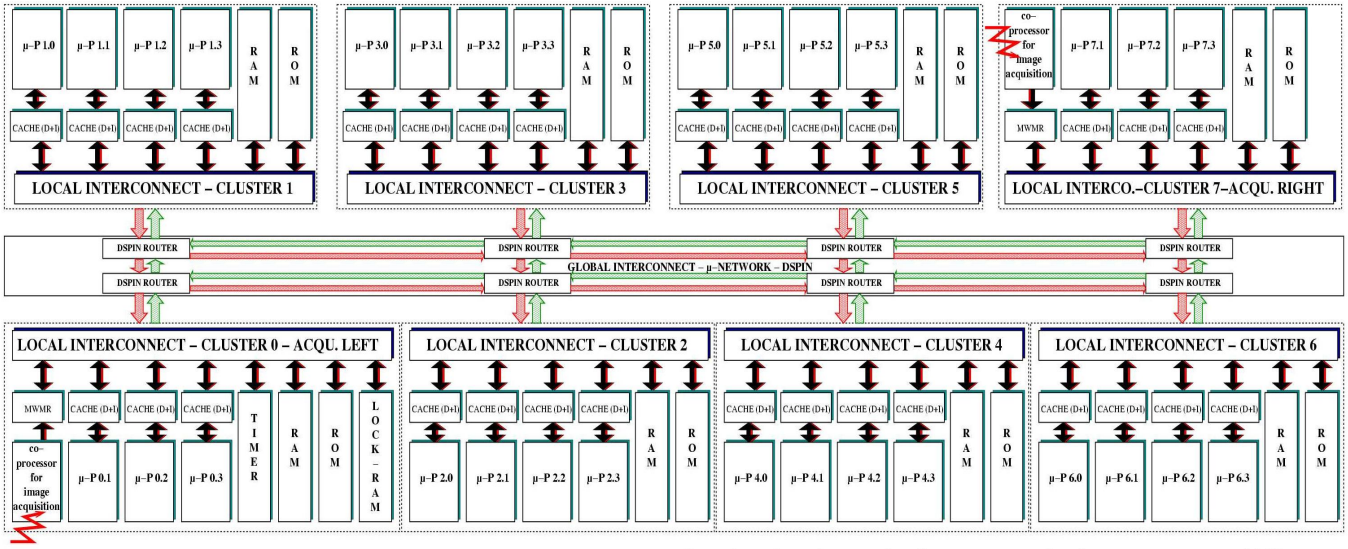


Fig. 5. 8 clusters and DSPIN interconnect architecture

Therefore, all threads are statically placed on the processors, and the communication channels are explicitly placed in the proper physical memory bank: as long as possible, two communicating threads i and j are placed on two processors in the same cluster, and the communication channel between i and j is mapped on a memory bank located in this cluster. The Mutek operating system allows the system designer to explicitly specify these placement constraints in the main program. Threads and communication channels are created during the initialization phase. The complete multi-threaded software application has been compiled, using the GCC cross-compiler for the MIPS R3000 processor. The complete application requires about 750 Kbytes of embedded memory, for both the data (software FIFOs) and the code. The resulting binary code is loaded into the embedded memory by the simulator, and the hardware/software, cycle precise simulation can proceed.

VIII. Performances Evaluation

As described in section 7, all hardware components are modeled using the SystemC[23] language and are fully compatible with the SystemC standard simulation engine. But, in order to improve the simulation speed, we used the SystemCass [22] simulation engine, that is part of the DISYDENT environment. SystemCass uses static scheduling, well suited to the SoCLib models, and provides the system designer a simulation speedup of one order of magnitude, versus the standard SystemC simulation engine. For this 30 processors architecture, the simulation speed was 4100 simulated cycles per second, on a 3.2 GHz Pentium 4.

In the hardware/software co-design process, different architectures have been evaluated for various number of processors, and the performances are ranging between 12 million cycles, and 6 million cycles for processing a stereoscopic pair of image. The detailed results for the final architecture, composed of 30 processors, 2 I/O controllers for images acquisition and 8 clusters, are given below, for different types of images:

- The initialization time is about 680 000 cycles. This time includes real-time OS, communication channels and threads initialization,
- The processing time is 5 490 000 cycles for a synthetic stereoscopic pair without obstacle.
- The processing time is 6 460 000 cycles for a synthetic stereoscopic pair with several obstacles.
- The processing time is 7 860 000 cycles with city situation and 7 020 000 cycles on highway.
- The processing time is 6 490 000 cycles with lighting condition as evening on a classic road.

The average processing time is about 6 800 000 cycles, i.e. 22,6 ms per stereoscopic pair at a clock speed of 300 MHz, which is compatible with the 25ms deadline. In case of violation of the 25 ms deadline for a given stereoscopic pair, the corresponding computation is cancelled, and the multi-threaded software application is gracefully re-initialized for the next stereoscopic pair by the operating system.

IX. Conclusion

We demonstrated in this paper that it is possible to implement an obstacle detection application on a multi-processor SoC architecture. The software application is based on stereo vision, and the "V-disparity" algorithm. The hardware architecture contains 8 clusters, 30 general-purpose 32-bit processors, and 750 Kbytes of embedded memory. The stereo vision algorithm, has

been parallelized using 30 threads, to exhibit a coarse-grain task pipe-line. We used the DISYDENT environment to model the multi-threaded software application as a KPN (Kahn Process Network), and the MUTEK real Time embedded operating system to map this software application on the multi-processor architecture. This architecture running at 300 MHz can process a stereoscopic pair of images (256 x 512 pixels) in less than 25 ms, supporting the soft real-time constraint of 40 images/s processing. The resulting MP-SoC has not been implemented on silicon, but the detailed architecture modeling allows to estimate a silicon area of about 60 mm2 in a 90 nm CMOS fabrication process. Such massively parallel architecture is very promising for automotive applications requiring high computing power. The proposed, shared memory, clusterized architecture is actually a generic SMP (Symmetric Multi- Processor) architecture that can be used for any parallel, multi-threaded application. The two Firewire I/O controllers used to interface the cameras are currently the only specific hardware components. Those I/O controllers could be easily implemented on embedded programmable hardware (such as Embedded FPGA), in order to have a fully generic architecture. The DISYDENT hardware/software co-design environment was very helpful in the process of design space exploration.

REFERENCES

[1] M. Bertozzi and A. Broggi, "GOLD: a Parallel Real-Time Stereo Vision System for Generic Obstacle and Lane Detection," IEEE Transactions on Image Processing 7, pp. 62-81, January 1998

[2] M. Bertozzi, A. Broggi, G. Conte, and A. Fascioli, "The Experience of the ARGO Autonomous Vehicle", In Procs. SPIE □ Enhanced and Synthetic Vision 1998, April 1998

[3] L. Di Stefano, M. Marchionni, S. Mattoccia, "A PC-based Real-Time Stereo Vision System", Machine Graphics & Vision, Vol. 13, No 3, pp. 197-220, January 2004

[4] J. Woodll, G. Gordon, R. Buck, "Tyzx DeepSea High Speed Stereo Vision System", in Proceedings of the IEEE Computer Society Workshop on Real Time 3-D Sensors and Their Use, Conference on ComputerVision and Pattern Recognition, June 2004.

[5] D. Han, D.-H. Hwang, "A Novel Stereo Matching Method for Wide Disparity Range Detection", ICIAR 2005

[6] G. van der Wal, M. Hansen, and M. Piacentino, "The Acadia VisionProcessor" Proc. of the IEEE Intl. Workshop on Computer Architecture for Machine Perception, September 2000

[7] S. Chiricescu, S. Chai, K. Moat, B. Lucas, P. May, J. Norm, R. Essick, M. Schuette, "RSVP II: a next generation automotive vector processor" in Intelligent Vehicles Symposium, 2005. Proceedings. IEEE, June 2005

[8] A. Broggi, G. Conte, F. Gregoretti, C. Sansoe, R. Passerone and L.M. Reyneri, "Design and Implementation of the PAPRICA Parallel Architecture" in The Journal of VLSI Signal Processing, 1998

[9] H. Jeong, Y. Oh, "A parallel real time implementation of stereo matching" in "Parallel and Distributed Processing Symposium., Proceedings 15th International", April 2001

[10] A. Techmer, N. Bruls, U. Hachmann, J. Harnisch, U. Ramacher, W. Raab, "A 100 GOPS vision platform for intelligent vehicles", Intelligent Vehicles Symposium, 2003. Proceedings. IEEE, 9-11 June 2003

[11] R. Labayrade, D. Aubert, and J.-P. Tarel, "Real time obstacle detection on non flat road geometry through V-disparity representation", in IEEE Intelligent Vehicles Symposium, Versailles, pages 646-651, June 2002

[12] R. Labayrade and D. Aubert, "Onboard Road Obstacles Detection in Night Condition Using Binocular CCD Cameras", ESV 2003 Proceedings, Nagoya, Japan, 19-22 May 2003

[13] A. Broggi, C. Caraf, R. I. Fedriga and P. Grisleri, "Obstacle Detection with Stereo Vision for Off-Road Vehicle Navigation", in Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (Cvpr□05), Vol. 03, June 2005

[14] L. Matuszyk, A. □elinsky, L. Nilsson, M. Rilbe, "Stereo panoramic vision for monitoring vehicle blind-spots", IEEE Intelligent Vehicles Symposium, June 2004

[15] T. Gandhi, M. Trivedi, "Vehicle Mounted Wide FOV Stereo For Traffic And Pedestrian Detection", in Proceedings of IEEE International Conference on Image Processing, September 2005

[16] F. P□rot, P. Gomez, "Lightweight Implementation of the POSIX Threads API for an On-Chip MIPS Multiprocessor with VCI Interconnect ", Design Automation and Test in Europe Conference (DATE□2003) Embedded Software Forum, March 2003

[17] H. Charlery, A. Greiner, E. Encrenaz, L. Mortiez, A. Andriahantenaina, "Using VCI in a on-chip system around SPIN network", Proceedings of the 11th International Conference Mixed Design of Integrated Circuits and Systems, June 2004

[18] Virtual Socket Interface Alliance, "Virtual Component Interface Standard (OCB v.2.2.0)", http://www.vsia.com (document access may be limited to members only), August 2000

[19] I. Aug□ F. P□rot, F. Donnet, P. Gomez, "Platform-based design from parallel C specifications", Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions, December 2005

[20] G. Kahn, "The semantics of a simple language for parallel programming " in Proc. Information Processing □4, Aug. 1974, pp. 471-475

[21] SoCLib, http://soclib.lip6.fr

[22] R. Buchmann, F. P□rot, A. Greiner, "Fast cycle accurate simulator to simulate event-driven behavior", Electrical, Electronic and Computer Engineering, 2004. ICEEC □4. 2004 International Conference, September 2004

[23] SystemC, http://www.systemc.org