HOMOGENEOUS MULTISTAGE ARCHITECTURE FOR REAL-TIME IMAGE PROCESSING

Hanen Chenini[1], Jean Pierre Dérutin[1], Romuald Aufrère[2], Roland Chapuis[1]

[1]Institut Pascal-UMR 6602 CNRS, Blaise Pascal University, 24 Av. Des Landais, Clermont-Ferrand, France
Hanen.Chenini@etudiant.univ-bpclermont.fr,{jean-pierre.derutin, chapuis}@univ-bpclermont.fr
[2]LIMOS-UMR 6158 CNRS, Blaise Pascal University, 24 Av. Des Landais, Clermont-Ferrand, France
aufrere@univ-bpclermont.fr

ABSTRACT

In this article, we present a new multistage architecture oriented to real-time complex processing applications. Given a set of rules, this proposed architecture allows the using of different communication links (point to point link, hardware router…) to connect unlimited number of parallel computing elements (software processors) to follow the increasing complexity of algorithms. In particular, this work brings out a parallel implementation of multi-hypothesis approach for road recognition application on the proposed Multiprocessor System-on-Chip (MP-SoC) architecture. This algorithm is usually the main part of the lane keeping applications. Experimental results using images of a real road scene are presented. Using a low cost FPGA-based System-on-Chip, our hardware architecture is able to detect and recognize the roadsides in a time limit of 60 mSec. Moreover, we demonstrate that our multistage architecture may be used to achieve good speed-up in solving automotive applications.

KEYWORDS

Multistage architecture, software processors, road recognition, MP-SoC architecture

1. INTRODUCTION

Intelligent sensor-based multiprocessor architectures has appeared in a number of fields especially for mobile robots monitoring and surveillance. Here, the sensors are embedded on board a moving vehicle and the algorithms are constrained to face the hard real-time constraints imposed by moving vehicle applications where speedup is a crucial factor. Furthermore, this approach consists mainly in moving the image processing to the sensor itself in order to provide sufficient processing power to run the algorithms near the camera. The ultimate aim is to combine smart sensors and algorithms to understand the vehicle environment. To integrate these applications on a mobile vehicle, it is necessary to execute the algorithms under real-time constraints using network architecture with high computation capability, low power consumption, flexibility, low-memory cost and small size to ensure the quality as well as the practicability.

Generally, to enhance performance especially in term of computation time, we exploit a Multiprocessor System on Chip (MP-SoC) architecture to embedded complex applications that require intensive parallel computations and manage a large amount of data. Unfortunately, as more processing elements are integrated in a single chip, the mapping of software tasks in a multi-processor system can be significantly more complicated since it affects the degree of parallelism among multiple processors and the utilization of the available hardware/software resources. In practice, as you increase the number of parallel processors involved in a particular task, you'll generally affect some performance criteria, such as area and execution time and also demonstrate how up to a certain number of processing units to reduce the amount of time before achieving a satisfying solution. The goal here is being able to run PC applications on System on Chip (SOC) platform and the parallel processing is the only way to scale performance while remaining within low-power environment. Moreover, it is important to use the minimum required resources to allow the entire system to be integrated on the same chip. This allows the reduction of the SoC complexity and cost.

To overcome these requirements, we have proposed new design methodology able to support parallelisation of complete image processing applications using Multiple Instruction Multiple Data architecture with distributed memory (MIMD-DM). In this way, to reduce the overall time to completion, the approach mentioned above is extended by the usage of new design flow [1] in order to increase the abstraction level of the specifications for both software and hardware description. The greatest advantage of our MPSOC approach is the short design time by using a graphical programming environment (called CubeGen). Using this framwork, the proposed MPSOC approach offers the opportunities to explore all the possibilities for designing multiprocessor architecture where processing nodes can operate in parallel and independently within the same chip. The using of such approach makes it possible to adjust the architecture, by refining of the material architecture where it is needed to efficiently meet the requirements of a given application.

In this work, we focused to real-time image processing algorithm (as object recognition, feature extraction, learning, tracking, 3D vision, etc.) in FPGA suitable for embedded vision systems to develop real-time algorithms able to assist the driving activity. In particular, the ability to recognize objects from data acquired by sensors is important for building intelligent systems. In fact, the proposed road recognition algorithm based lane model is able to recognize a wider range of lane structures such as straight, curve and parabolic models according to various driving environment. In addition, it is robust against shadows, noises, etc. due to the use of the parallel knowledge of road, vehicle and camera parameters. Hence, this approach relies on roadway sensors to obtain the lateral vehicle position in its lane as well as the steer angle and the road curvature. The lane detection problem is formulated by determining the set of localization parameters. For embedded the developed multi-hypothesis approach for road recognition, a multistage architecture is applied in this paper. In addition to its ability of road detection and tracking, our architecture is able to recognize the roadsides by taking into account the coherence of the two sides of the road.

The paper is organized as follows. Section 2 proposes a MIMD-DM multistage architecture, where several of pipeline stages are connected through synchronization links, to correctly utilize the processing elements (PEs) to work cooperatively. Section 3, we use a motivating example of image processing application multi-hypothesis recognition approach to illustrate the proposed MPSOC approach. Section 4 describes the parallel structure of the algorithm and the choices made while porting the application software to the embedded system. We show in section 5, the results of the implementation of the road recognition into Xilinx FPGA following the proposed design methodology. In section 6, we briefly present some works interested in implementing object recognition applications. Section 7 concludes the paper.

2. THE PROPOSED PARALLEL ARCHITECTURE

This section describes a generic MPSoC architecture oriented to real-time image processing. However, as mentioned before, and for fast and efficient parallelization of software application on MPSOC architecture, we suggest the use of identical processors. In addition, we develop homogenous multistage architectures which are well suited to the computational needs and real time performance of multi-task real time application requiring much parallelism.

2.1 Parallel Homogeneous Architecture

Traditionally, the most of MPSOC architectures are heterogeneous with several processors, complex memory hierarchy (shared and distributed), etc. Subsequently, for programming these architectures efficiently, hardware description languages (HDL), such as Verilog or VHDL, are required. Unfortunately, the users especially the programmers work with high-level programming languages (C, C++, etc…) and they don't know how to deal with hardware description languages. Hence, they have to focus on low-level system issues and they have to manage low level communication problems such as deadlocks and parallelism details which requires a technical background rarely found among non-expert users. Consequently, designing an interconnection architecture for MP-SoCs while simultaneously reducing power consumption and satisfying the performance constraints is a difficult problem.

In our research, we are motivated to develop a new design flow enable describing parallel hardware architecture at a much higher abstraction level than traditional hardware description languages. We focus on Multiple Instruction Multiple Data (MIMD) with distributed memory (DM) parallel architectures which permit diverse communication types (data, task and flow parallelism) [1]. Communication between nodes is realized thanks to the well-known message passing communication model (each node can send and receive message). We chose this architecture because of its ability to execute any parallel scheme efficiently. The interconnection network is static hypercube-based topology structure. The figure 1 provides an overview of a generic homogeneous MPSoC, composed of a several software processors (P_i) with private memory element (*M*) for application software and data storage, and communication device (simple FIFO point to point (FSL link) communication, or more complex communication such as DMA-router packet).

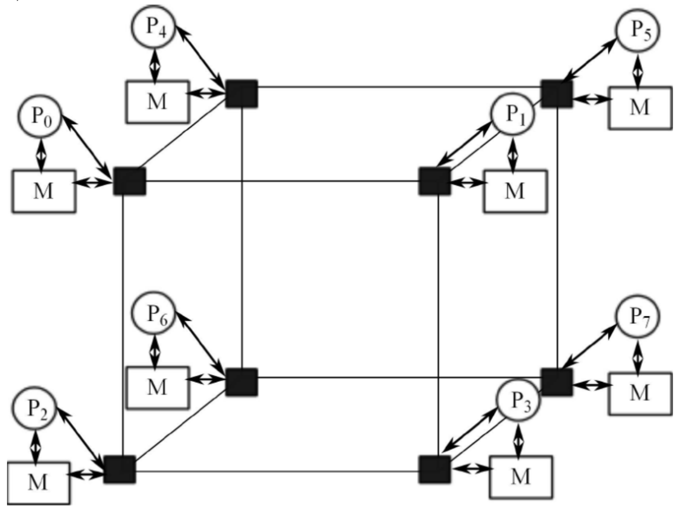

Figure 1. Homogeneous parallel processor architecture

All components can be chosen inside a library of available custom components or commercial IP. In fact, we intend to define an environment that enables automatic generation and configuration of the processor. The heart of this approach lies in the ability of the developed CubeGen environment to automatically generate a HDL description of network processors from a set of available IPs and configurations choices. This tool automates the creation of the configuration file dedicated to EDK or QUARTUS environment. This configuration file defines the processor network architectures by using different interconnected IP blocks (hypercube topology). The designer of an FPGA embedded MPSOC architecture has complete flexibility to select any combination of parameters (number of processors, MicroBlaze parameterization, memory size allocated to each processor, FIFO size, etc.) needed in their implementation in order to either save memory or meet performance requirements. And then automatically CubeGen framwork will create the processors network and the associated communication function. Furthermore, for generating parallel code to run on the different soft-cores, we developed a set of specific communication functions based on Parallel Skeleton concept, such as FARM for dynamic data sharing (or dynamic task sharing), SCM (Split Compute and Merge) for data or static task parallelism and PIPE for task-parallel parallelism.

Recently, the critical characteristic of image processing applications is often related to the increasing complexities in terms of software algorithm, embedded systems design and real-time performance constraints. However, in order to achieve potential performance of parallel implementation of complex image processing applications, new parallel architecture must be developed.

2.2 Multistage Architecture

Generally, applications can be divided into many sequential subtasks, the parallelism can be achieved by computing different stages simultaneously. For heterogeneous computing (requiring both task and data parallelism), the designer needs a multi-stage architecture which is structured as a set of parallel computing stages connected through synchronization links. In this context, the developed MPSOC approach present a solution of the previous problem with respect to some rules related to the type of communication link between processors in each stage, number of pipeline stages etc. This solution brings the necessity of reducing the communication latency between processors and changing the interconnection network according to the application.

Hence, this solution can be called an on-chip multistage architecture with minimum synchronization links between stages. Actually, it can be used for high performance real time applications which need parallel architectures integrating a large number of processors.

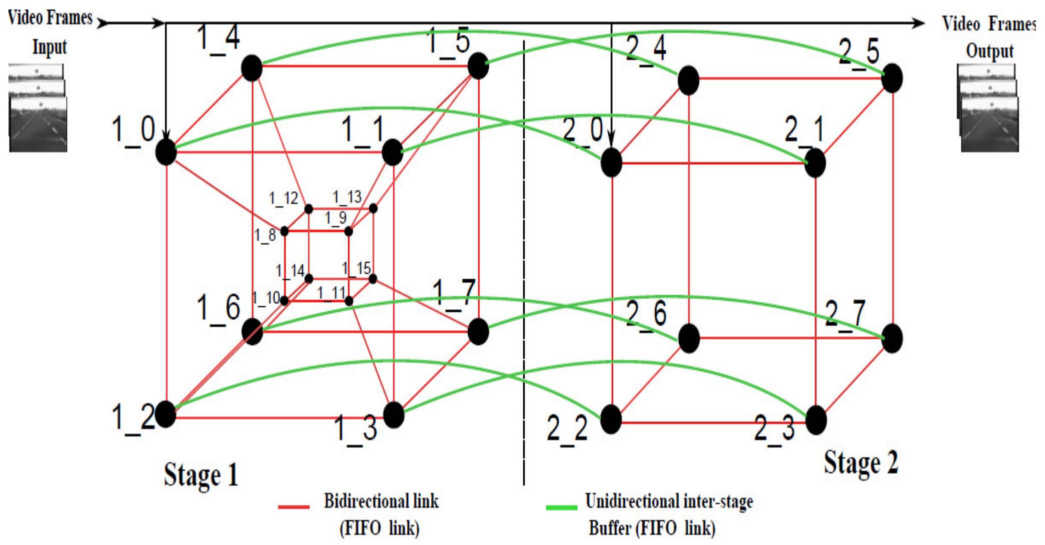

Figure 2. An example of pipeline architecture with two stages (16 processors based on FSL links, 8 processors based on FSL links)

If we consider K pipeline stages which work in parallel, each stage comprises $N_i = 2^{D_i}$ nodes $(i \in [1, K])$. Here, we define two different types of multistage architecture as illustrated in figure 2 and 3 where only the processor with 0 index in each stage will receive the input image. If the cube nodes of each stage are connected by a point-to-point communication channel, each processor P_{i-j} in the stage i will communicate with the processor P_{i+1-j} in the next stage via point-to-point Fast Simplex Link (FSL) which is unidirectional $(j \in [0, N_{i+1}])$ where N_{i+1} is the number of processors in the stage i+1. Figure 2 illustrates this point, a multistage architecture which contains two consecutive stages based on bidirectional point to point Fast Simplex Links (FSL), with 16 processors in the first and last stage and only 8 processors in the second stage. If we have two consecutive stages where the first is based on point to point links (FSL) and the second is based on hardware router (figure 3), only processor P_{i-0} in the stage i will communicate with the processor P_{i+1-0} in the stage i+1 via point-to-point Fast Simplex Link (FSL) which is unidirectional (only the processors with index 0 are connected).

To summarize, the communication between different pipeline stages is performed in three ways:

- Each processor in a given stage is connected to the corresponding processor in the next stage if we use FSL connections between the processors in each stage.
- Only one processor in a given stage is connected to the corresponding processor in the next stage if we use hardware router as communication link between the processors in at least one stage.
- Each processor in the first stage can send their output to the processors in the next stage but they cannot receive from these processors using unidirectional connections.

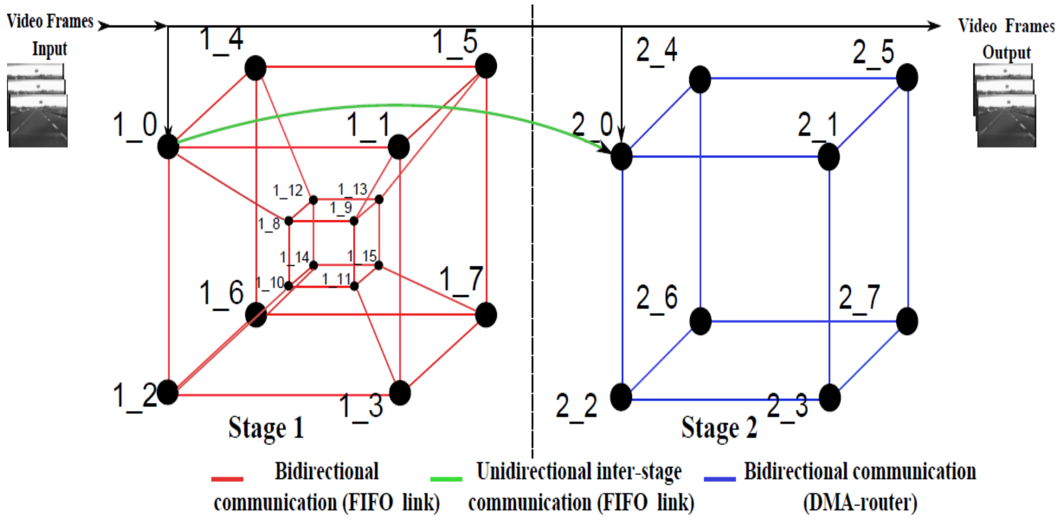
Video Frames Input
Video Frames Output
1_0
1_1
1_2
1_3
1_4
1_5
1_6
1_7
1_8
1_9
1_10
1_11
1_12
1_13
1_14
1_15
2_0
2_1
2_2
2_3
2_4
2_5
2_6
2_7
Stage 1
Stage 2
Bidirectional communication (FIFO link)
Unidirectional inter-stage communication (FIFO link)
Bidirectional communication (DMA-router)

Figure 3. An example of pipeline architecture with two stages (16 processors based on FSL links, 8 processors based on hardware router)

Actually, the performance of this kind of architecture depends mainly by the slowest pipeline stage and the speed gain may be potentially high. To have a balance between stages, multiple stages can be generated with different dimensions. To handle various simultaneous inputs, we have added video bus module which split the input image into several independent blocks. These blocks will be distributed over all available processors in each stage.

In fact, the video bus module is used to transfer image data from one frame generator module to one or more frame Grabber module. The digitized pixels are collected in an image buffer (FIFO buffer). The image buffer stores at least one complete frame and is used if the bandwidth of the bus is not small to transport the digitized video data stream without loss. The size of each received frame is 256x256 pixels and the time between two successive images is fixed by the user when he sends the frame via Ethernet port. Thereafter, it is possible to collect the digitized video data stream directly in the main memory (e.g. the input buffer) of each frame Grabber module associated to processing nodes. Thankfully these grabbers used sync signals to control the synchronization pulses between the input (incoming data from video bus) and the output buffer (outgoing data to processing nodes). The vertical swap (V-Swap) and the vertical sync (V-Sync) indicate the beginning of a new frame (figure 4). Our proposed hardware architecture provides an end-to-end solution; it receives video input from a PC via an Ethernet connection, process this input, and outputs the results to a display without distorting the video.

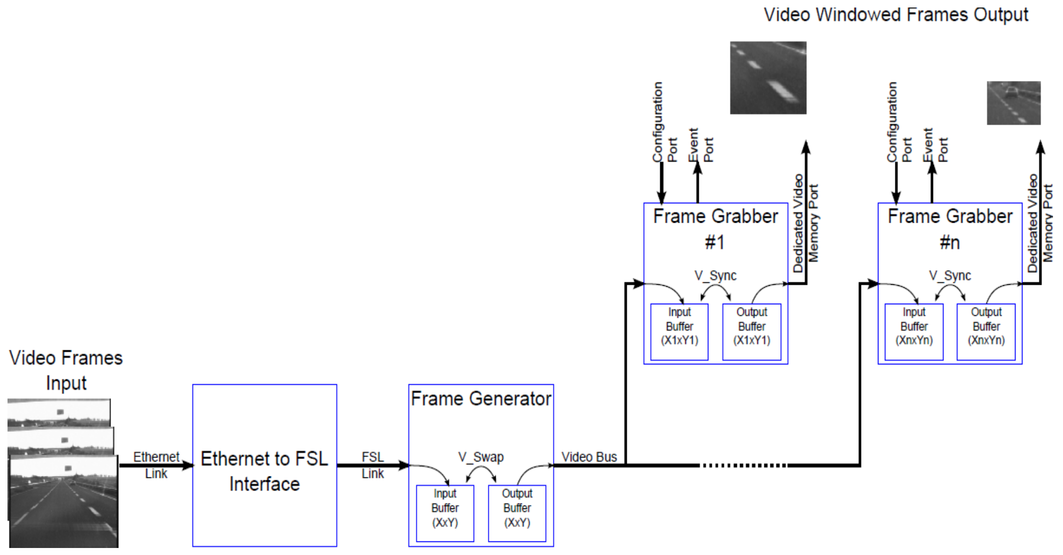

Figure 4. IO architecture based frame Grabbers

3. The Multi-Hypothesis Approach for road recognition

Since the topic of our research has a strong focus on automotive or traffic applications, we are interested particularly in detecting and tracking the roadsides through a sequence of video. The proposed algorithm is inspired by the road recognition approach developed by [2]. This application is usually the main part of the lane keeping systems which are designed to maintain the vehicle in its lane. Ideally, road edges are generally characterized by a high-contrast region with low curvature and constant width. With real road images, a determinist description of the road geometry can be a challenge since the road may vary in the form of various road conditions and quality of road markings. Actually, detection strategy proposed by [3] presents a conceptually simple approach for edge detection, followed by edge grouping and pruning. However, as a single hypothesis is used the recognition process will still have problems to follow the lane road through video sequence due to false detections.

To keep track of the roadsides, a multi-hypothesis approach of road recognition can be employed by using various classifications of zones of interest (representative of the road region in image). In fact, this method allows eliminating all the false detections of road edges. The algorithm of sequential recognition is obtained on the basis of the recursive recognition approach presented in [3] applied to accomplishing the different hypotheses in order to identify an optimal hypothesis before the algorithm converges. In addition, the recognition of the roadsides is performed by two main steps: learning, and multiple-hypothesis recognition. The main properties of the proposed road recognition are:

- The road model initialization or learning step is presented for providing a good initial position for the lane model in the image. This algorithm is robust to noises, shadows, and illumination variations in the captured road images, and is also applicable to both the marked and the unmarked roads. In practice, the training information will be estimated directly from vehicle localization parameters and camera's parameters.
- The lane recognition problem is formulated by using a set of hypotheses to determine the parameters of road model. Using the detected lines and the image gradient, lane refinement is performed using IIR filter and Least Median of Squares to give more exact lane position by only a local search.
- The road model is updated in time each frame as well as updated from the measurements via a Kalman filter.

2.3 Model of Road Edges

In this work, the road model used is similar in form to that used in [2]. The road examined in this study is divided into 2P interest regions (figure 5). Under the assumption that the road lane can be decomposed into a set of lines segments, the roadside has an associated model represented by a vector X_d and a covariance matrix C_{X_d} which consists simply of a set of 2P+2 images parameters:

$$X_d = \left(u_{1l}, \dots, u_{(p+1)l}, u_{1r}, \dots, u_{(p+1)r}\right) \quad , \quad C_{X_d} = \begin{pmatrix} \sigma^2_{u_{1l}} & & & & & \\ . & . & . & . & . & . \\ & & \sigma^2_{u_{(p+1)l}} & & & \\ & & & \sigma^2_{u_{1r}} & & \\ . & . & . & . & . & . \\ & & . & & & \sigma^2_{u_{(p+1)r}} \end{pmatrix}$$

Where u_{1l} and u_{1r} represent the horizontal image coordinates of the left and right road side respectively for different image rows v_j ($j \in [1, P]$). The first variant makes use of windows of fixed position (i.e. the coordinates v_i remain fixed in the same locations for all frames). In fact, the $\sigma^2_{u_{il}}$ and $\sigma^2_{u_{ir}}$ defines the possible variations of u_{il} and u_{ir} respectively in real road image for ($i \in [1, P+1]$). The initial value of the model $\left(X_d, C_{X_d}\right)$ is obtained by assuming all parameters dispersion of a typical road during learning phase.

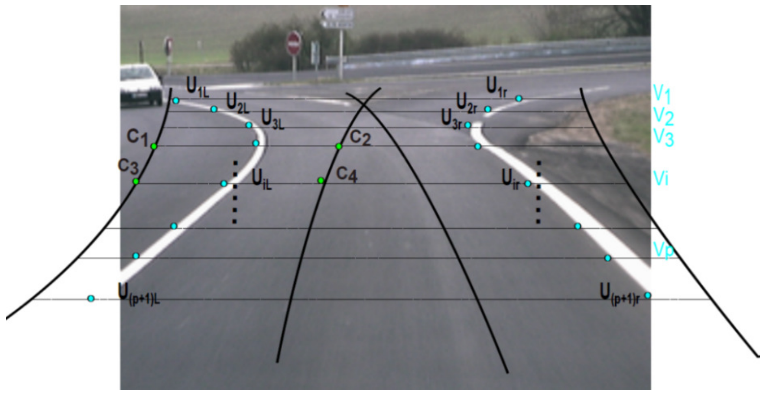

Figure 5. The roadside model deduced from the localization parameters

2.4 Learning Phase

The algorithm has an offline learning phase, which is independent of the road image data. The learning part is processed using vehicle environment (i.e. localization parameters, camera's parameters) as shown in figure 6. This phase is necessary to seek only realistic road configurations in the image. Furthermore, the goal of this step is to provide an initial value of the model in order to limit to search region of roadsides in the image. This model is the relationship between u_i coordinates of roadsides in the image and v_i coordinates, vehicle localization parameters (x_0, ω), camera inclination parameter α and the road geometry (C_l, L):

$$(u_{il}, u_{ir})^t = G(v_i, x_0, C_l, \omega, L, \alpha)$$

The model X_d is obtained starting from average values of the essential parameters $(v_i, x_0, C_l, \omega, L, \alpha)$, established from initial knowledge on the vehicle, the road, the camera, and various ordinates v_i in the image. Variations of these parameters according to a normal law in an interval of dispersion wished, will led to a set of realistic roads configurations in the image. From these various configurations, it is easy to extract the covariance matrix C_{X_d} by using the Jacobian J_d of the function G[3].

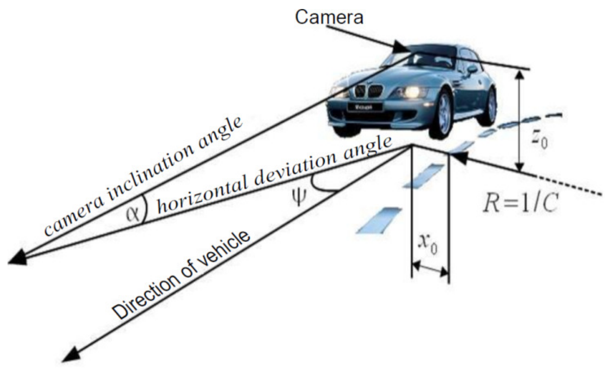

Figure 6. Localization parameters of vehicle in the image

2.5 Recognition Phase

The aim of the recognition step is to generate multiple hypotheses-candidate positions- for roadsides. During this phase, we would evaluate all possible detections of the road edges to ensure that we find the best position of the roadsides in real road images. Precisely, the road area is divided into 2P sub-regions that are representative of the entire roadsides. The interest zones are then defined by a trapezoid forms $C_1(u_i - \sigma_i, v_i)$, $C_2(u_i + \sigma_i, v_i)$, $C_3(u_{i+1} - \sigma_i, v_{i+1})$, $C_4(u_{i+1} + \sigma_i, v_{i+1})$ where σ_i is the variance of u_i deduced from C_{X_d} for $(i \in [1, P+1])$ (figure 5). In practice, we have used 8 interest zones representative of each road side.

Our approach relies on the use of multiple hypothesis recognition, corresponding to different ways of regrouping the different zones of interest (the example shown in figure 7 is taken only 5 interest zones) in the image road. More precisely, a hypothesis refers to one possible way of grouping the zones of interest. The assumption that the number of windows is known a priori or is constant across frames. The challenge, then, is to determine which hypothesis is likely to be correct and to accurately determine the roadside position for those regions.

Starting from the matrix covariance C_{X_1}representing the authorized variations around the localization parameters, the different interest zones are ranked according to its size. Then, for the first hypothesis, we start by treat the first interest zone. In order to achieve the best estimation of the roadsides, an iterative procedure is used. In this zone, segment is detected for an analysis depth $n = 0$ and the model is then updated to obtain new road model $(X_{dm}, C_{X_{dm}})$. After that, the recognition algorithm [3] is an iterative detection process: a new optimal zone is located and the algorithm works now at a depth $n + 1\ (0 \leq n < P)$. After, all the zones will be tested, the algorithm considers that the roadsides are found in the image. The same principles will be applying for the others hypotheses (starting by different interest zones). The search process (choice of the interest zone, detection and update) is recursively re-iterated for each hypothesis with different combinations of the interest zones. Depending on this recursive recognition of each hypothesis, we assign them a score. Finally, the recognition scores of the entire hypothesis are collected and the road model with maximum score is selected to be the final estimation for the roadside. As a result, we obtain the optimal value of X_{dop} and $C_{X_{dop}}$ representing the roadsides in the image. In the following section, we present our parallel architecture which brings solution for embedded the proposed recognition application.

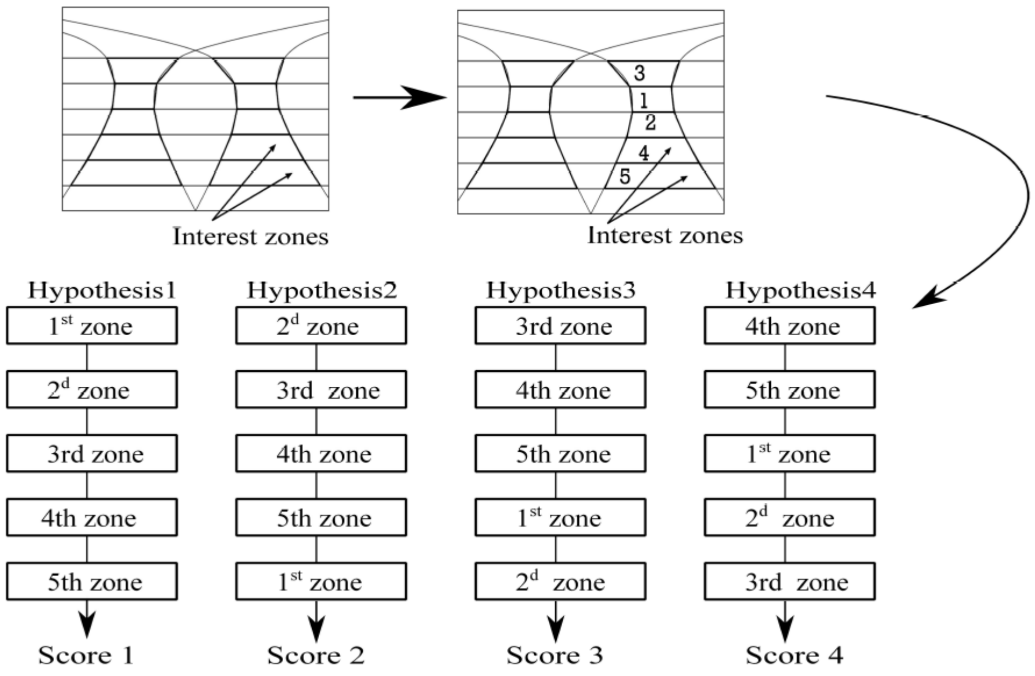

Figure 7. Multiple-hypothesis Recognition approach

4. PARALLEL IMPLEMENTATION OF RECOGNITION ALGORITHM

In this section, we describe the parallel implementations of the proposed recognition algorithm on the proposed multistage architecture based on MIMD-DM architecture. This architecture can cope with the majority of problems which crop up in real-time recognition application, mainly when dealing with more than one side simultaneously. For the quality of road line detection, the geometrical knowledge of the roadsides to follow makes recognition easier. Once the network architecture has been trained using these initial localization data, it is used to recognize roads in all the next frames.

Since the communication needs for this application are important and computation needs are much heavier, we have chosen the simplest hardware communication solution (using FSL point to point links). Thus, the point-to-point based network is generated (Figure 2) since it satisfies the application communication needs for relatively low implementation costs. Using the video bus, all the available processors in each stage can receive the input image. In addition, this multistage architecture represents staged computation where parallelism can be achieved by computing different stages simultaneously to produce different outputs (the model of the roadsides). In practice, we need a hardware network with two pipeline stages to parallelize the proposed recognition algorithm. Using the synchronization links between the two pipeline stages, the precise positions of the right and the left side are given by the first and the second stage respectively.

To implement the recognition algorithm on each pipeline stage, the software application can be referring to one or more independent tasks running concurrently. Consequently, the parallel execution of multi-hypothesis process is simply the concurrent execution of independent tasks (or hypothesis) in the different available processors in a given stage. The adopted approach of parallelization is based on SCM scheme (static task parallel) where each processor computes one candidate model of roadsides. We illustrate this point with figure 8. In this figure, each hypothesis is calculated by different processors in each stage and the results obtained from these processors are combined and the final result is obtained.

For executing the given application on the proposed parallel architecture, we use our skeleton library composed of specific communication functions in particular "Split", "Merge", "Pipe" functions and two others functions called "Request" and "Acknowledge" allowing to execute a "synchronization barrier". With these communication functions, one processor is chosen as the master and other processors serve as slaves.

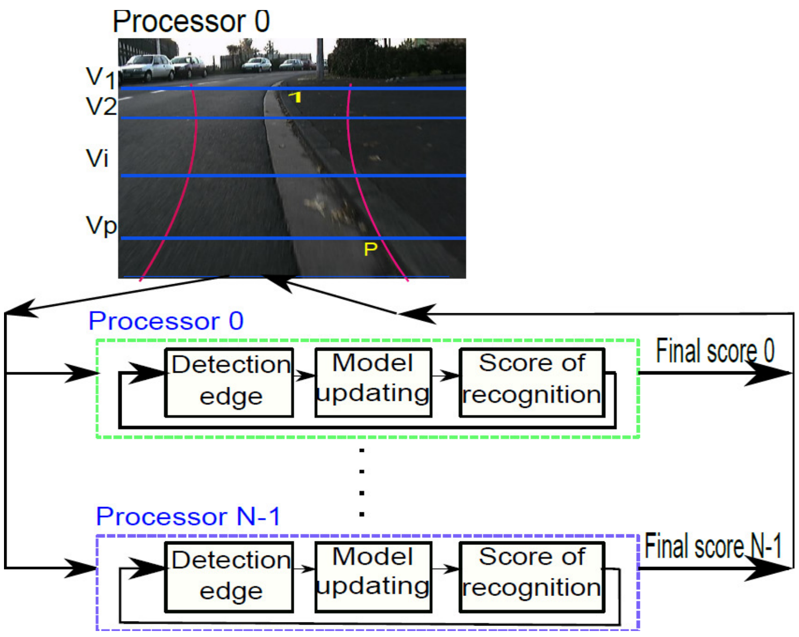

Figure 8. Parallel implementation scheme based SCM skeleton

Without considering the architectural problems, we can convert the sequential algorithm into a parallel C code. Consequently, the parallel implementation scheme in each stage is based mainly on task parallelism between the N_i available processors $(i \in [1,2])$. The number of zones to be tested is P. After receiving new image, the processor P_{1-0} (or P_{2-0}) starts generating the initial roadside model (in the first image) or the update model (in all the next images) which includes the vector X_d and the covariance matrix $C_{X_d}$and sends them to each processor P_{1-j} for $j \in [1, N_1]$ (or P_{2-j} for $j \in [1, N_2]$). On the other side, each processor simply waits for the model of the roadside and executes the recognition process (the search process is recursively re-iterated for all the zones of interest), including the two measurements detection and updating. Then, they return their results to the processor P_{1-0}(or P_{2-0}) which merges them to get the final result (i.e. the different processors supplied by the road image run a single hypothesis). Finally, the processor P_{1-0} (or P_{2-0}) compares the outputs of the different search hypotheses looking exhaustively for the best solution. The roadside model with maximum score of recognition is selected to be the final estimation for the roadsides in the current image.

5. EXPERIMENTAL RESULTS

To evaluate the benefits of the developed MPSOC architecture, we report some of our results concerning resource costs and area occupations for different FPGA implementations (by varying the network sizes in each pipeline stage). Several road sequences that include more than 1200 road images have been tested for lane recognition. Concerning the complexity of the multi-hypotheses recognition stage, it depends mainly on the number of interest zones and the number of hypotheses. The most time consuming step is the detection step, due to the large data consumption and the several operations (subtraction, addition, multiplication and comparison) required between all pixels on each interest zone.

5.1 Sequential Implementation

This road detection and tracking algorithm has been tested on real road images. These road images include curve and straight road, with or without shadows and lane marks. In our experiments under sequential mode, the number of search windows (zones of interest) is set to 8 in order to be better concentrated around the true state.

With regard to the microprocessor, we use a 32-bit RISC soft core processor, MicroBlaze, which is developed as soft IP by Xilinx Company, running at 200 MHz with *64k* data and instruction memory (BRAM). It can be easily implemented in FPGA-based system. For the sequential implementation on FPGA platform, the proposed hardware architecture has been implemented on a Xilinx ML605 FPGA development board. In a video sequence, this implementation has an output latency of approximately *264ms* in the first image using four hypotheses. The search zone deduced from the X_d matrix is limited in size for all the next frames.

Figure 9 shows the processing flow for road recognition applied to a single image (Example recognition of the right side). The model update/detection steps will be repeated until all the interest zones in the right side of the road will be processed. Our results show that it is possible to achieve real-time tracking even operating at relatively low clock frequencies. In addition, one can often expect and frequently achieve an improvement in performance by using far more hypothesis to recognize the roadsides according to various driving environment.

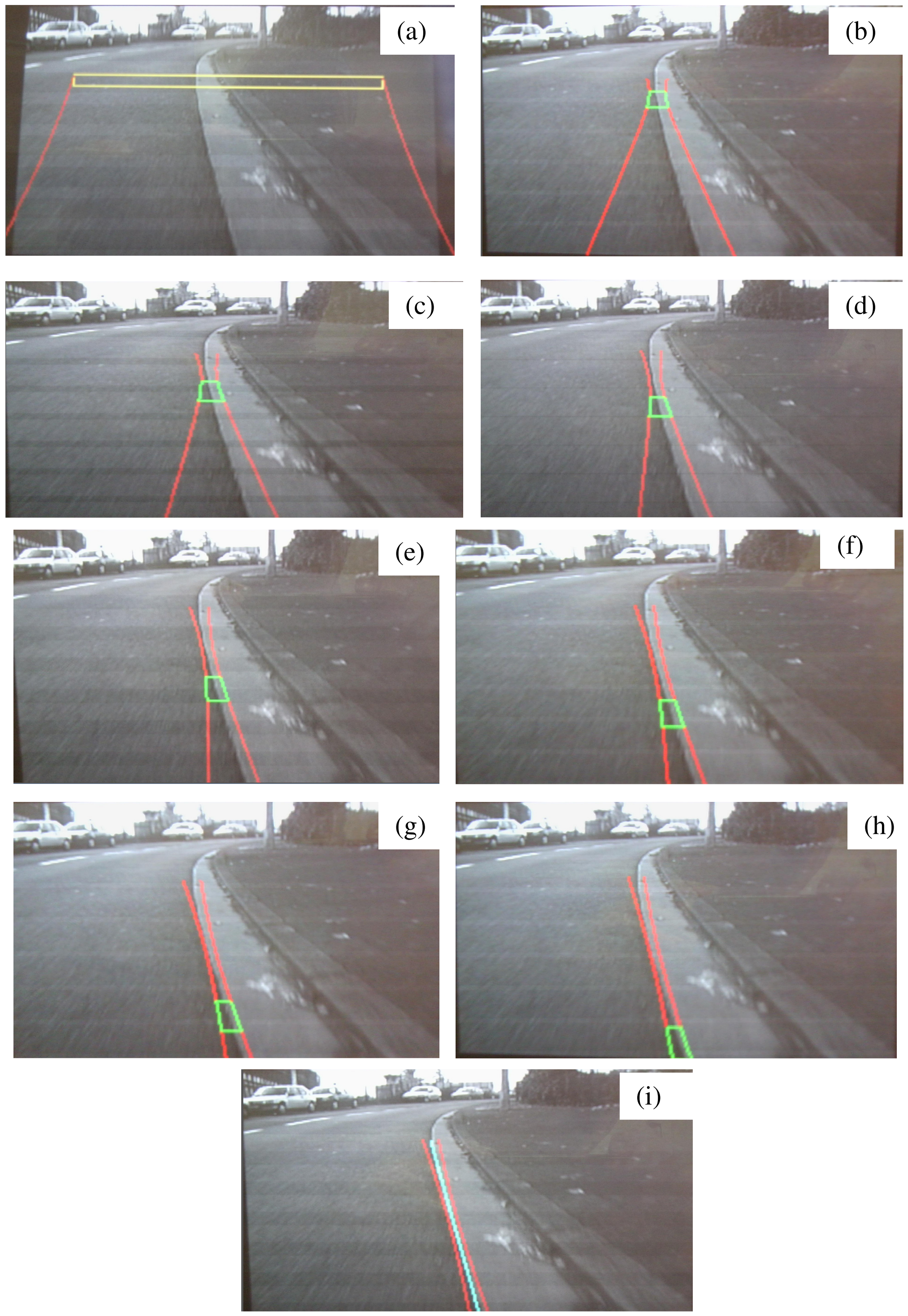

Figure 9. Processing flow for road recognition: (a) learning phase and the first interest zone, (b, c, d, e, f, g, h) the next Interest zones, (i) the recognition result

5.2 Parallel Implementation

We have chosen the multistage architecture which uses point to point FSL links (the architecture described above) since it satisfies the application communication needs for relatively low implementation costs. To first get a sense of the presented architecture implementations in terms of FPGA resource usage and clock frequency, we present a number of experiments in which we show the FPGA synthesis results and the execution times of the proposed recognition algorithm. Figure 10 shows the space and resources used by the proposed MP-SoC architecture (the pipeline network architecture), in function of number of processors used in each stage. The size of memory for data and program (BRAM) was set to 64KB. Since the processors connected through a hypercube network, each node is associated with Frame Grabber which contains two buffers (Input/output buffer) and supplied with a 64k local memory. This may explain the results of resource utilization in terms of area occupation. Due to the large size of the contest networks and the limited resources on the XilinxVirtex-6 LX760T FPGA, we were able to implement from 1 to 8 processors in each stage.

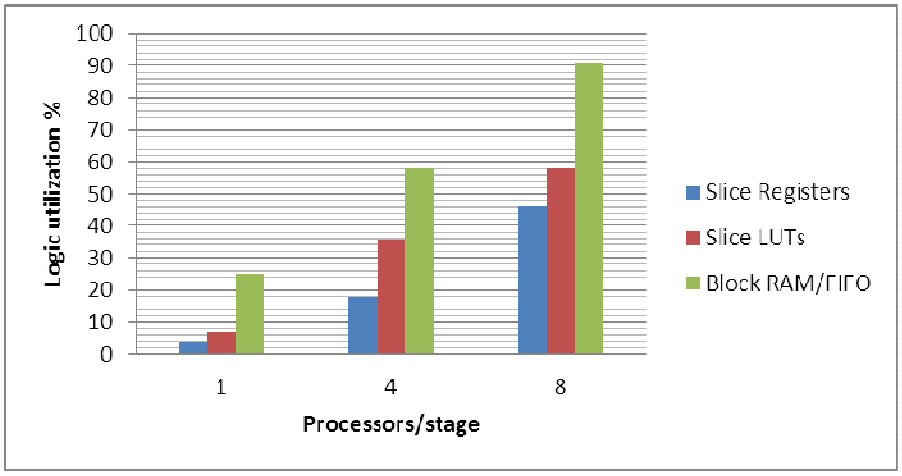

Figure 10. FPGA resource utilization of target device (Xilinx xc6v2x240tff 1156-1)

The time to execute the road lane detection and tracking depends greatly on the number of hypotheses which will be distributed over different computing cores. The parallel application uses task parallelism, tasks (hypotheses) being executed by processing cores as the number of processing cores become available. As a result, processing performance is limited by the maximum number of tasks that can be allocated during iteration.

Example speedup graph for an image resolution of 256x192 is shown in figure 11 for the cases 1, 4 and 8 processors in each of the two pipeline stage (measured with 4 hypotheses). Some differences in processing time between experimental findings and theoretical assumptions are due mainly to communication effects. The ideal results represents a parallel implementation in which all processors operate 100% efficiency, communication overheads could be ignored between processors and assumes static task execution times. In reality, efficiency will be less than 100% due the communication cost and the residual sequential overheads of each pipeline stage are taken into account by applying Amdahl's law. For this parallel implementation and from up to 8 processors per stage, communication cost (which is a sequential part) becomes significant. In practice, the speedup of a program using multiple processors in parallel computing is limited by the time needed for the sequential fraction of the program. However, the remaining sequential algorithm part in the parallel algorithm (zones of interest generation and evolution step) represents a minor part of the processing time in the sequential implementation. Hence, the performance of the proposed algorithms is rarely close to the theoretical speedup.

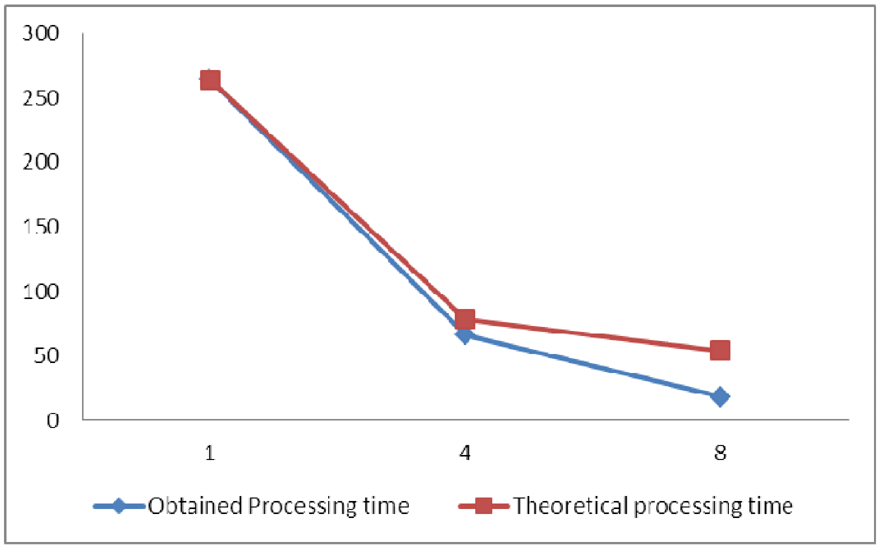

Figure 11. FPGA processing time results

The latency measured of the parallel implementation decreases as more processors are added. The latency of the parallel implementation with 8 processors in each stage is approximately 54ms whereas it is 264ms for the original sequential application. The recognition function requires 41ms and 13ms respectively for the detection and least median of squares methods in the first frame. Then, their execution times are reduced across all the next frames. As the algorithm is simple and efficient, in real practice, we can achieve a speed of at least 18 frames per second. The percentage of correct lane tracking is over 95%, depended on the real road conditions and the number of the hypotheses used. As a conclusion, the presented architecture achieves the best performance and its throughput scales very well as the number of cores increases. The benefit of this parallel architecture becomes much clearer in the terms of latency and performance.

6. STATE OF THE ART

Previous works have attempted to present a parallel architecture with hardware optimizations oriented to object recognition applications using an FPGA platform. In [4], the authors designed a new hybrid programmable FPGA-based architecture able to recognize the obstacles in the way of the host vehicle. Taking into account the real time processing and the limited resource issues, this architecture contains both the hardware and software components. Recently, many approaches on the hardware implementation of human action recognition have drawn attention [5]. As part of that effort, the authors have developed a reconfigurable FPGA based video processing architecture which computes in parallel to solve tasks with a high computational cost, as in real time image processing.

Other implementation strategy [6] was done using a high level design methodology for FPGA-based multiprocessor architecture. This design methodology enables the architecture development and the application mapping onto an FPGA-based runtime reconfigurable multiprocessor accelerator. Therefore, other interesting architectures for parallelism are carried out by [7, 8] to implement various automotive applications. These two architectures provide support for a specific set of real time data intensive applications. The Auto Vision architecture is a new dynamically reconfigurable MP-SoC architecture for future video-based driver assistance systems, using run-time reconfigurable hardware accelerator engines for video-specific pixel processing. The SIMD Processor Image Recognition called IMAPCAR is another example of full programmable processor that is dedicated to automotive security applications. The ImapCar architecture adopts a SIMD (Single Instruction Multiple Data) architecture of 128 processing elements and a 4-way VLIW (Very Long Instruction Word) control processor.

7. CONCLUSION

In this paper, we present an improved MP-SOC approach to design, test, evaluate and generate parallel homogeneous architectures that satisfies the severe requirements of real-time image processing applications. The used of case studies allows us to design a dedicated parallel computing system where we can see the found trade-off between performance and cost as compared to other MP-SoC design approaches.

Experimental results show that the proposed architecture speeds up significantly the execution time and has shown a good performance using images of a real road scene. In this work, the main idea was to investigate the proposed FPGA tools (both hardware and software tools for rapid development for network architecture) against powerful and complex tasks in autonomous vehicles and robotics.

Similarly, our current work addresses the parallel implementation of Artificial Neural Networks (NNs) applications using our multistage architecture. Since NNs is a series of sequential layers, we have chosen to use pipeline architecture composed by multiple stages communicating with each other through synchronization links. In addition, we have developed parallel programming model for distributed computing that supports feed forward NNs algorithms requirements.

REFERENCES

[1] H. Chenini, J. P. Dérutin, and T. Chateau, "Fast prototyping of embedded image processing application on homogenous system - a parallel particle filter tracking method on homogeneous network of communicating processors (hncp)," *in VISAPP (2),* 2012, pp. 122–133.

[2] A. Romuald, C. Roland, C. Frederic, "A model-driven approach for real time road recognition", *Mach. Vis. Appl. 13* (2) 95–107 (2001).

[3] A. Romuald, C. Roland, C. Frederic, "Real time vision based road lane detection and tracking", in: *Proceedings of the IAPR Conference on Machine Vision Application*, 2000, pp. 75–78.

[4] H. Liu, S. Niar, Y. El-Hillali, A. Rivenq, "Embedded architecture with hardware accelerator for target recognition in driver assistance system", *SIGARCH Comput. Archit. News* 39 (2011) 56–59.

[5] M. Hongying, F. Michael, P. Nick, B. Chris, "Real-time human action recognition on an embedded, reconfigurable video processing architecture", *J. Real-Time Image Processing* 3 (2008) 163–176.

[6] Y. Chung, S. Choi, V. K. Prasanna, "Parallel object recognition on an fpga based configurable computing platform", in: *Proceedings of the 1997 Computer Architectures for Machine Perception (CAMP '97)*, Washington, DC, USA, pp. 143–.

[7] C. Christopher, S. Walter, H. Andreas, "Autovision - a run-time reconfigurable mpsoc architecture for future driver assistance systems", *it - Information Technology* 49(3):181- (2007).

[8] S. Kyo, S. Okazaki, "Imapcar: A 100 gops in-vehicle vision processor based on 128 ring connected four-way vliw processing elements", *J. Signal Process. Syst.* 62 (2011) 5–16.