Invited Paper

High Performance Customizable Architecture for Machine Vision Applications

Nelson Acosta and Lucas Leiva[*]
INCA/INTIA, Facultad de Ciencias Exactas
Universidad Nacional del Centro de la Prov. De Bs. As, Tandil, 7000, Argentina

ABSTRACT

Vision based applications are present anywhere. A special market is industry, allowing to improve product quality and to reduce manufacturing costs. The vision systems applied to industries are known as machine vision systems. These systems must meet time constraints to operate in real time. Generally the production lines are more and more fasters, and the time to process and bring a response is minimal. For this reasons, dedicated architectures are emplaced. In this work a review of several commercial systems is presented, as well a proposed architecture is depicted. The architecture is concern as a customizable platform, avoiding having knowledge in hardware description languages. It is based on massive parallelism to achieve the maximum processing performance. Several optimizations at different levels are applied to increase the final system speedup. Also, time and area metrics are reported, showing that the architecture is well suitable for real time video processing in industrial applications.

Keywords: Video processing, Machine Vision, FPGA,

1. INTRODUCTION

The host PC cannot process images from the new higher-resolution cameras with faster frame rates. The industrial high-performance applications require higher image size, over 10MPixels with more than 10-bit deep; while the frame rate required can be higher than 500 frame per second.

All machine vision systems require some characteristics from the running platform: real-time computing power, high quantity of input/output data pin, determinism to fix the real-time process, high throughput, and low latency.

There are almost three ways to achieve the proposed image processing goals. The first approach is produced by the biggest commercial processor companies, like Intel, Hitachi, Philips, or Hewlett-Packard; where they produce processor highly adapted to develop a machine vision system. This topic is represented by small representative processors.

The Trimedia processor (Phillips) [1] can process 6.5 billon operation per second, can support until 64MB RAM, include 27 processing elements pipelined to exploit the VLIW architecture from the C programming style.

The SH-5 (Hitachi) [2] processor is oriented to develop grid configurations to high performance graphic operations at a high frequency. It has three operands instructions, 64 registers of 64 bits, running at 400 MHz executing 714 MIPS.

The Itanium2 was developed by Intel and Hewlett-Packard [3]. The main characteristics are: 1GHz running clock, 128 registers, two floating point units, 6 integer units, 6 multimedia units, and 4 load-store cache pipelines. The architecture allows the compiler to define the instruction-level parallelism; independent of the number of instruction groups a particular processor is capable to compute. For example the compiler could emit 128 instructions that could be executed in parallel, where the processor would execute them in groups of 6 instructions.

These approaches are oriented to a big market portion because the inversion to produce the processor only can be afforded by the biggest companies in the world. There are countless of these kind of processors running in hundred of cards in machine vision or smart cameras applications. These applications designs are restricted to program the processor without customizing the processor or the card architecture; normally it's very complicated to grown up to another camera resolution, or a higher FPS, or a more complex algorithm, because the architecture cannot let it.

The second way is based in the technological progress of CMOS scaling circuits, leaving to ingrate image capture with processing logic on a chip. These devise are known as FPSP (focal-plane sensor-processor[4][5]. There are two major components in all FPSP: the photo-sensor array and the processors. Inside the capabilities, implements image scalar operations (min, max, mean, global OR, number of black pixels on a binary image, histogram,etc.), image row/ column operations (profile, shadow), neighborhood processing (kernel filters) and global processing (FFT, wavelet transformation, Hough transform). Several vision chips were developed, as Q-Eye, powering AnaFocus' Eye-RIS system [6], SCAMP-3 [7], MIPA4k [8], ASPA [9][10], VISCUBE [11].

This is an emerging technology in evolution that can be used to solve particular problems, and still not provide support to complete system implementation.

The third way to reach the requirements of the machine vision system is through a custom architecture processor for each application. These processors can be VLSI or FPGA based system. The VLSI power only can be afforded by great companies, and it is the same as explained in the first point. So, in this paper the reconfigurable architecture is analyzed. The approaches in this area are numerous, the following discussion is not

[*] CONICET

exhaustive and is limited to a small representative well documented architectures.

In real-time video processing systems, a set of operations are repetitively performed on every image frame in a video stream. These operations are usually computationally intensive and, depending on the video resolution, can be also very data-transfer dominated. These operations, which often require data from several consecutive frames and many rows of data within each frame, must be performed accurately and under real-time constraints as the results greatly affect the accuracy of application.

The use of reconfigurable system platform for image recognition is well defined in [12] where the main application is the surveillance. A face recognition application on a Spartan3 with 1 GB DDR SDRAM, by using a 5 Mpixel at 14 FPS camera, is implemented.

A SIMFD with small memory in each processing element and a I/O array to reduce data transport is used as portable supercomputer for video processing in [13]. A very detail analysis is included comparing the architecture with DSP.

The XRI-1200 [14] card developed by DALSA can process 1 Mpixel 12-bit deep image at 30 FPS to obtain X-Ray analysis at real-time.

Other use of the machine vision is to construct a virtual 3D environment model [14], for example to guide a synthetic person on this world. The data fusion of the 3 camera must compensate the partial observation of each individual VGA camera working at 30 FPS.

The pipeline applied to the image processing [16] is analyzed to show that the pipeline model can significantly improve the speed of the large image processing.

A tool for automatic generation for FPGA real-time video processing systems is presented in [17]. The generator creates the memory and control functionality for a functional spatio-temporal video processing system over a FPGA. The main architecture is defined using VHDL to be automatically synthesized.

The FPGA enable system designers to develop applications with a large amount of parallelism [18]; this characteristic allows a cheap architecture for high-performance vision computing. The vision algorithms are implemented to analyze the performance with FPGA, DSP, Intel Core 2 Duo GPP and the FPGA on-chips Microblaze and powerPC.

The smart cameras [19] are presented with all its history evolutions from the Xerox first system to the future directions of the area. They were the first embedded systems to process real-time video images. The smart camera can be developed thanks to the great advances in embedded vision systems that are showed in [20].

The overview of real-time image or video processing algorithm [21] from a research environment is used in an actual real-time implementation on a resource constrained hardware platform. These strategies consist of algorithm simplifications, hardware architectures, and software methods.

The interdisciplinary efforts are analyzed for the successful development of machine vision applications [22]. The light, lenses, camera calibration, camera, interface, hardware platform, algorithms, and image analysis are the main topics involved in the machine vision inspection.

A review of mathematical principles and key issues in image processing [23] are detailed depicted, such as the description and characterization of images, edge detection, feature extraction, segmentation, texture, and shape; while a discussion topics are image matching, statistical pattern recognition, syntactic pattern recognition, clustering, diffusion, adaptive contours, parametric transforms, and consistent labeling.

A complementary review is presented in [24] by analyzing amounts of material on mathematical morphology, 3-D vision, invariance, motion analysis, artificial neural networks, texture analysis, X-ray inspection, foreign object detection, and robust statistics.

The end users prospective of machine vision technology are depicted in [25]. It is a powerful introductory material not for systems designers. The main topics are principles in lighting, optics, cameras, underlying image processing and three-dimensional and color machine vision techniques.

Application domains of real-time systems include machine vision, object recognition and tracking, visual enhancement and surveillance; these applications are analyzed in a method and a tool to enable efficient memory synthesis for real-time video processing systems on FPGA [26]. The central objective of this method is the optimized use of embedded memories in the process of buffering data on-chip for an RVTPS operation. The developed software tool is an environment for generating HDL codes implementing the memory subcomponents.

The high speed video architectures use the inherent data parallelism in applications by using deeply pipelined functional units, increasing the number of processing elements. The SIMD architectures are the most suited computational model for video processing because they can efficiently exploit massive data parallelism with minimal data movement. This architecture also has a programming model with minimum work.

The architectural solution proposed in this paper implement hardware-based acceleration algorithms for machine vision systems. The FPGA clock runs often a lower speed than standard microprocessors; but they can run on parallel units to get a higher throughput than microprocessor with higher clock speeds.

There are two parallelization techniques to increase the algorithm speed: sending different data sets to multiples processing units, or mapping operations onto a pipeline. So, parallel computing can be done by time-parallel or space-parallel.

The operational structures of those systems consist of on-chip processors or custom vision coprocessors implemented by using high parallel processing units with efficient memory and bus architectures.

This paper shows the parallel and pipeline techniques applied to a pattern recognition platform FPGA-based. These acceleration techniques can be used by an expensive custom design or by an automatic generation tool. The architecture generator produces the HDL code to be automatically synthesized, for example on FPGA.

The structure of this paper is as the following: section 2 cover the proposed architecture for high video computing, exploiting the subcomponents involved. Section 3 summarizes the metrics obtained. Section 5 presents the conclusions and further works.

2. ARCHITECTURE DESCRIPTION

A video processing system is generally formed by a set of stages interacting in a particular mode to perform a task. Those steps are image enhancement, image segmentation, and feature detection/ measurement. Some systems, as machine vision and surveillance, involve a stage which allows taking decisions. It can be implemented as a

classificator or a measure analysis technique. In this work a RBF Neural Network is used to give the intelligence to the system.
The architecture is intended to implement a generic soft-core, with the corresponding flexibility to support several application areas. The system is composed by a set of modules (or stages) executing in parallel, implementing a full image pipelining. A general description is presented in the figure 1.

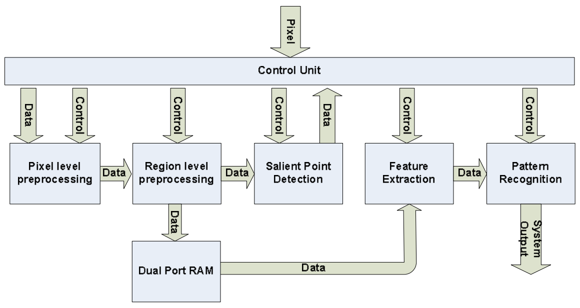

Fig. 1. System description

The image information is transferred to the control unit, enabling the stages when a valid data arrives. The pixel value is transmitted to pixel preprocessing level stage, transforming the value according to a function established at system definition. The generated information is passed to region preprocessing level stage. This stage performs an image transform analyzing the region surrounding a pixel. These steps are commonly called image enhancement, and could be used to perform a contrast expansion, border enhancement or noise reduction.
The image segmentation stage could be a system bottleneck, because these techniques have a great computational resources demand. To avoid this problem, an algorithm to reduce resource was implemented, which allows detecting salient points in images. In this way, the algorithm detects that regions containing usable information to be analyzed for the next stages. The results determined by this stage are communicated to the controller, deciding if the next stages must be activated or not. The information is not passed to the next stage because this stage does not bring image information. The output is a flag indicating if the region is analyzable or not.
The feature extraction stage consumes the data present in a double port memory, which contains the enhanced image. This stage performs a feature vector computation determining the pattern main features, transferred to a pattern recognition stage. The classification module is responsible to assign a category to the input feature vector, if the input is similar to a learning pattern.
Each one of the mentioned stages is depicted in the next subsections, dealing both functional descriptions as architectural implementation details.

Pixel preprocessing preprocessing
Given an input function this stage transforms each one of the input pixel into a new value. The set of transformed values represents an enhancement image. The operator allows applying any injective function as logarithm expansion, exponential, threshold and further (Fig. 2).

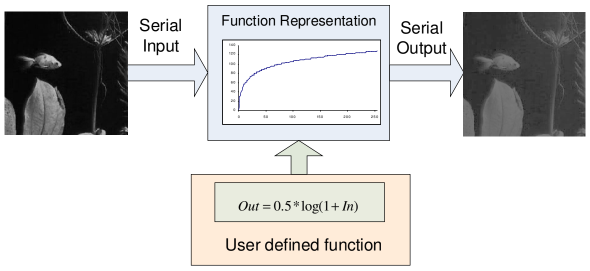

Fig. 2. Pixel preprocessing stage

The pixel preprocessing module is implemented in a ROM, containing the respective transformed value for each input. The values are generated at system definition (Fig. 3).
In this way, the input data is used to address the ROM. The output value is registered to increase the stability on the output. This implementation avoids the computation effort, demanding only one memory element.

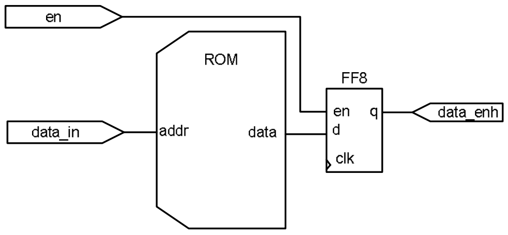

Fig. 3. Pixel preprocessing architecture

This architecture allows performing operations in grey level images (8 bits pixel depth). The ROM must be the capable to address 256 values, and can be synthesized in BRam or distributed in LUTs. Thus, a relaxation constraint is meted in order to adapt the system to existing resources. If the first option is taken, the implementation reduces the area required to store the complete values set, associating the stage to a single dedicated memory block. On the other hand, when these blocks are sparse in the platform, the function could be mapped to logic distributed in LUTs.

Region preprocessing level
This operator allows image enhancement through the analysis of image regions. These kinds of operators usually are implemented through a transformation matrix (kernel) which is applied iteratively on the entire image. At each step, this operation provides an intensity value corresponding to one pixel of the improved image. The transformation allows implementing border enhancement, noise reduction, sharpen, blur and other useful image processing filters.
During the specification stage, the kernel coefficients are defined. These values are applied during the system execution to each pixel of the input image (Fig. 4). The component input is a serial input, given a data result in each pixel cycle.

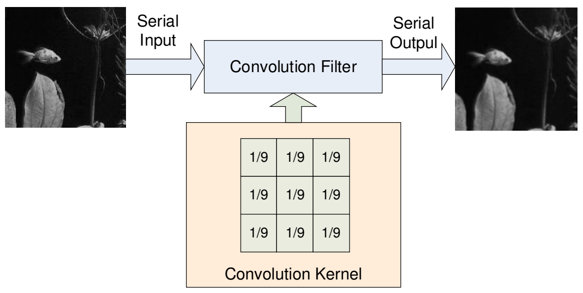

Fig. 4. Region preprocessing level enhancement.

The architecture of this component is formed by storage and an operational subcomponent. The first of those stores incoming intensity values in a pipeline to be processed, using two FIFOs and a set of 6 registers (Fig. 5). The FIFOs store temporarily intermediate values acting as a sliding window that applies the filter to all regions of the image. The size of each FIFO is set by w-2, where w is the value corresponding to image width. Only 6 registers are required because the FIFOs outputs and the input value form the data needed to apply the operation.

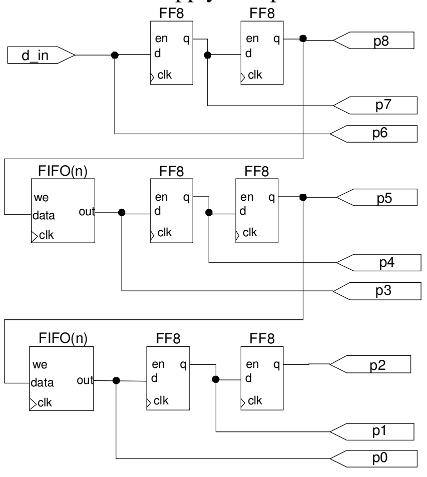

Fig. 5. Region preprocessing level storage module

The operational logic module is described at a high level of abstraction, allowing the synthesis tool to find the best implementation for operations. While the multiplication operation has a high cost, the component description is made in order to be as optimal as possible, allowing instantiate the hardware DSP cores provided in some families of Xilinx FPGAs (Virtex). For this case nine DSP cores are used, which performs all the products between the region and the coefficient matrix in parallel. The results of the products are added across the connection of these modules in cascade, but the design of these cores are optimized for sums of products, and is unnecessary to incorporate an architecture such as only increase the complexity by using additional adders (Fig. 6). The critical path for implementing this operation on a Xilinx Virtex4 FPGA has a delay time of 5.575 ns .

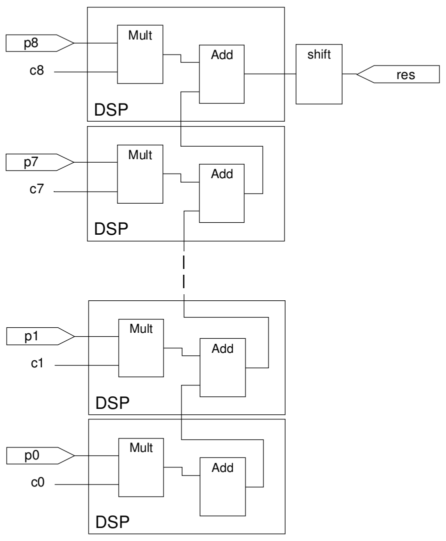

Fig. 6. Region preprocessing level operational module

To avoid the computation complexity, all the operators are reduced to fixed-point arithmetic. This type of operation ignores exponent calculations and performs the necessary arithmetic operations followed by a shift to obtain the required result. In this way, the set of operations required are implemented according to:

- Pre-computed coefficients are defined as constants in the code that describes the system architecture. Being c_i the real value that corresponds to the i^{th}-position of the matrix, the constant value for the coefficient in fixed point is $c\_fp_i = c_i \times 2^{10}$. The fractional part obtained from this operation is discarded, and the sign must be taken into account, since the coefficients can be negative.

- The input data is normalized with respect to coefficients shifting left 10 bits, and the multiplication is performed between the two values.

- The multiplications are summed, and the result is shifted right 10 bits, to reconstruct the data subjected to normalization.

A value 2^{10} is used to maximize the calculus precision; the multiplier core contained in the device hardware has 18 bit entries. The implementation maximizes accuracy with minimal logic.

Candidate region detection

To avoid the computational costs involved in image segmentation, a salient point detection technique is used. This technique allows detecting areas with objects presence to be analyzed later. The algorithm is based on the FAST salient point detector optimization [27].

This technique determines a salient point analyzing the pixels corresponding to the main axes to the central pixel. If at least three surrounding pixels are darker o brighter, the central pixel is a salient point.

The results obtained by applying the technique are evaluated in regions. In this way a region is candidate if contains at least *n* salient points.

The implementation of this stage consists of storage and a processing modules, as well as region level image enhancement stage. The storage subcomponent is formed by two FIFOs and three registers (FF8) implementing a sliding window. In this case the number of registers is reduced because the data to be processed is lower (five per region). The size of the first FIFOs is *w'* and the second's size is *w'-1*, where w' is the half of the image width. The storage capacity of these structures is reflected in the figures, where $n = w'$. The outputs of the FIFOs represent the right and the upper pixel (Fig. 7).

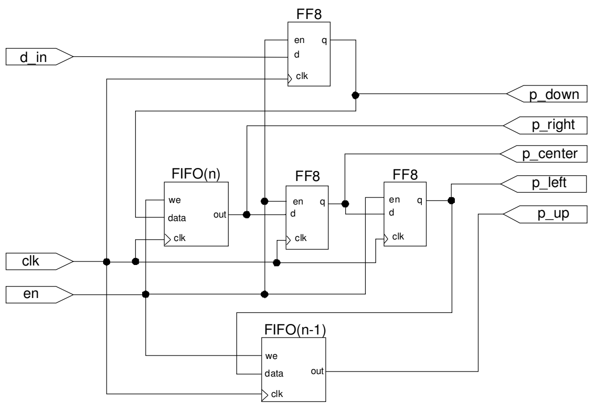

Fig. 7. Salient point detection storage module

The processing logic comprises one adder and one subtractor responsible to perform these operations between the central point and a threshold value (Fig. 8). The implementation is optimized using two LUTs to achieve the result of both operations. The threshold is established during system definition. The operations are defined in macro level, relying to synthesis tool to meet the best emplacement. This implementation can get better results allowing dynamics thresholds in later versions.

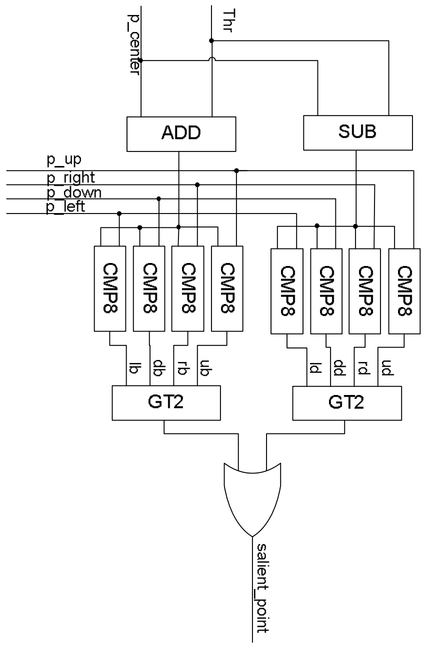

Fig. 8. Salient point detection processing module

Feature extraction

The feature extraction stage is the main phase of image processing systems. At this stage all relevant information is collected to perform the pattern recognition/classification. Moreover, this stage is essential to achieve proper system operation and is totally dependent to the application area to be developed.

A region of interest (ROI) is projected to get some feature or set of features, such as shape, texture, color, etc. A ROI is defined as the area within an image that can hold a pattern.

While the operation imposes further delay of the system (operating at region level), the candidate region detection module avoids to analyze all the regions.

This stage may be viewed simply as a set of transformation functions applied iteratively on each ROI. The range of functions available can be very extensive. Thus, the flexibility is a key factor in the development stage. For this reason a compiler is proposed, capable to transform a high level user code to a hardware optimized description (Fig. 9). The optimizations are performed by compiling a intermediate code capable of group operations minimizing both the area and the processing time. A detailed description of the compiler was presented in [28].

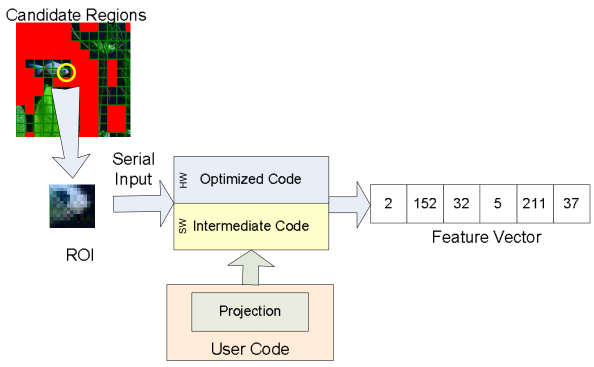

Fig. 9. Feature extraction stage

The architecture of this stage is formed by a controller and a processing logic. The controller architecture is defined as a FSM (Fig. 10.), and performs the data requests to memory according to the ROI to be analyzed. The readed data is delivered to the processing logic to compute the feature vector.

The controller remains in idle (*IDLE*) state, until the an enable signal activation. When this event occurs, the controller change his state to execution (*EXEC*), reading the ROI.

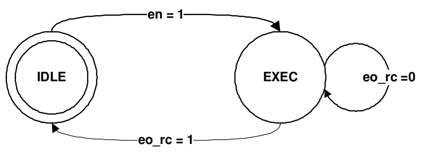

Fig. 10. Feature extraction main FSM.

The ROI read is implemented using two counters to address relative positions of pixels in memory. One of these calculates the relative displacement required per row (*row_cnt*) and another calculates the relative displacement in columns (*col_cnt*). These values together with the base address bring the desired memory address.

$$addr = col\_cnt + row\_cnt + base\_addr$$

The architecture considers that reading is always active, avoiding to incorporate a new state inside the FSM to read the first ROI pixel. A continuous memory read is accepted, because the memory is implemented as a dual-port memory, allowing to write data for previous stages, and ROI component readings simultaneously. A continuous data reading not affect the component, because the data only will be used when the architecture require.
The data processing is carried out by a SIMD architecture, where the input data is delivered to a number of processors. The processing elements, computes the FV components values, and can be read asynchronously.
To avoid overwriting data, a double buffering technique is applied. This configuration enables to perform a read operation on the last valid and complete FV. When a vector computation finishes, the signal *fv_sel* is changed to indicate to this vector as the last valid. Thus, the read operation will be performed on this vector, and computations will be performed on the other FV. This logic not affects to the auxiliary registers set, since they only contain partial results (Fig. 11.).

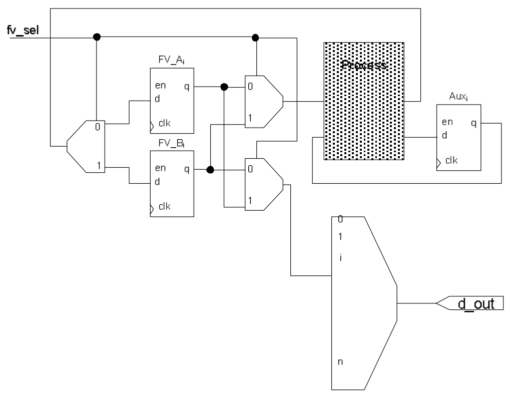

Fig. 11. FV storage and computation architecture

Pattern recognition
The pattern recognition stage is based on ZISC78[29] and CM1K[30] devices architecture. This classifier is defined as a Radial Basis Function Neural Network, allowing associating a category to a pattern. This neural network is trained off-chip. Its means that the training set is used by a training tool capable to generate a hardware description of the RBFNN.
The architecture is composed by a controller and a set of neurons. The neurons are interconnected via a neuron communication bus (Fig. 12).
The controller is implemented by a state machine. This component coordinates the operations of the neurons to perform the classification. It is also responsible for controlling feature vector readings and reports the status of the component, indicating when it has achieved the outcome of a classification (rdy).

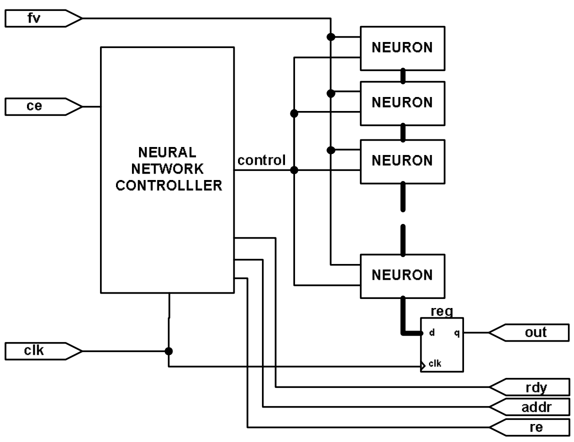

Fig. 12. Pattern recognition architecture

The neural intercommunication bus imposes a critical path to the system. This problem is directly associated with the neurons number in the network. To avoid system bottlenecks, intermediate registers are placed uniformly distributed on the inter-neuron communication bus. This segmentation allows increasing the maximum operational frequency.
A detailed description of this architecture can be founded in [31]. This work presents a comparison between the architecture presented and commercial devices.

3. RESULTS

There are a lot of variables acting in system definition phase, including the frame size. Thus, the freedom degree is high. The system can operate if the stages frequencies are lower than pixel arriving frequency. In this way, the system can be seen as a fully pipelined architecture governed by the pixel clock.
Because this system is intended to be use in industries, the response time must be the shorter possible. Also, as this is image pipeline architecture, the best way to assess performance is through latency. This factor provides an understanding of the time to give a response.
Given the type of architecture that arises, the latency for each stage individually is known. Thus, the entire system latency is given by the accumulation of partial latencies. For the stage of improvement pixel level, this factor (L_{PP}) is associated with a ROM reading and storing this data in a register, so that the latency in this case is 1 pixel period (P_{Pixel})

$$L_{PP} = 1\ P_{Pixel}$$

However, the region-level improvement requires that two rows of the image are temporarily stored and three extra pixels to fill the data set needed to implement the first filter. So, the latency for this step (L_{RP}) is given by:

$$L_{RP} = (2w+3)\ P_{Pixel}$$

Where w is the image width.
The salient point detection, only stores the odd columns and odd rows of the image, needing 4 rows of the image and three additional pixels available to operate. The latency of this stage (L_{SPD}) is:

$$L_{SPD} = (4\,w + 3)\,P_{Pixel}$$

The results of salient point detection are used by the candidate regions detector, while it may provide an answer at an earlier time in the worst case required to verify that the number of salient points of an entire region to be analyzed. The worst case is comprised to all rows of the image that comprise the candidate region and the width of one candidate region (*ROI_w*). The latency (*L_{ROCD}*), for a ROI size of *$ROI_H \times ROI_w$* is given by:

$$L_{ROCD} = (((ROI_H\text{-}1)*w) + ROI_W\,)P_{Pixel}$$

The operational frequency of these stages does not correspond to the operating cycle of the system. It is related to the cycle of transmission of one pixel (pixel clock). For precise latency calculation is necessary to consider also the space defined between two rows determined by the horizontal sync signal.
Feature extraction is done by reading each point of the ROI, so latency (*L_{FE}*) is given by this value plus an additional cycle to update the feature vector. The period considerate in this case is the related to system clock (*P_{system}*).

$$L_{FE} = ((ROI_w * ROI_H) + 1)\,P_{system}$$

Finally the recognition latency (L_{PR}) is given by reading the feature vector (FV_{Size}) plus the time required for the response, and 2 additional cycles of control. The time to generate the output is equal to the number of registers in the neuron communication bus (NR). In this way, the latency is:

$$L_{PR} = (FV_{Size} + NR + 2)\;P_{system}$$

Thus, the overall latency of the system (L_{System}) is comprised of the partial sum of the latencies of each stage.

$$L_{System} = L_{PP} + L_{RP} + L_{SPD} + L_{ROCD} + L_{FE} + L_{PR}$$

On the other hand, the double port memory must be capable to allocate $w*2ROI_H$ to optimize system performance. The memory is emplaced in physical block memories presents in the FPGA.

4. CONCLUSIONS AND FURTHER WORK

This work presents a flexible architecture capable to be adapted to several application areas. All the stages of a machine vision system are covered, adjusting to the system needs.
The use of FPGA as platform provides a great level of parallelism to implement video processing application. On the other hand, the image pipelining proposed allows optimizing system performance. This architecture is well suitable for special video sensors as line scan cameras.
A high level of spatial and temporal parallelism is exploited in the design. The architecture is a high performance platform solution, capable to be used by vision system developers without experience in hardware description languages.
A detailed analysis is presented also, showing that the proposed solution can achieve the constraints imposed in industrial application. So, with all custom application parameters defined, an accurate processing time, frame rate and global system latency can be estimated at pre-implementation time. This feature lets evaluate system constraints at earlier design stages.

This architecture cannot allow changing configurations parameters in operation time. But these features are contemplated in new versions. Actually, a research is doing to use this architecture in floor tiles inspection.
A machine vision generator tool is currently building, capable to generate customizable architectures. The tool allows defining a complete system to be generated, bringing the possibility to validate it in software.

5. REFERENCES

[1] Philips, "*Programmable Media Processor – TriMedia TM-1300*". Internal Report. Processor datasheet. 2001. Pp: 1-9.

[2] J. Brambor: "*Implementation notes of binary dilation and erosion on 64-bit SH5 processor*". Centre de Morphologie Mathematique, Ecole National Superieur des Mines de Paris, France. October 2002. Pp: 1-17.

[3] HP: "*Inside the Intel Itanium 2 Processor*". A Hewlett-Packard Technical White Paper. 2002. Pp: 1-44.

[4] Orly Yadid-Pecht, Ralph Etienne-Cumming, *CMOS Imagers: From Phototransduction to Image Processing*, Springer, 2004

[5] A. Zarandy, Focal-Plane Sensor-Processor Chips, ISBN 9781441980076, Springer, 2011.

[6] A. Rodríguez-Vázquez, R. Domínguez-Castro, F. Jiménez-Garrido, S. Morillas, A. García, C. Utrera, M. Dolores Pardo, J. Listan, R. Romay, "A CMOS Vision System On-Chip with Multi-Core, Cellular Sensory-Processing Front-End", In Cellular Nanoscale Sensory Wave Computing, C. Baatar, W. Porod, T. Roska, ISBN: 978–1–4419–1010–3, 2009

[7] P.Dudek, D.R.W.Barr, A.Lopich and S.J. Carey, "Demonstration of real-time image processing on the SCAMP-3 vision system", IEEE International Workshop on Cellular Neural Networks and their Applications, CNNA 2006, pp.13-13, Istanbul, August 2006

[8] J. Poikonen, M. Laiho, and A. Paasio, MIPA4k: A 64×64 cell mixed-mode image processor array, in IEEE International Symposium on Circuits and Systems Taiwan, 2009, pp. 1927–1930

[9] A.Lopich and P.Dudek, "ASPA: Focal Plane Digital Processor Array with Asynchronous Processing Capabilities", IEEE International Symposium on Circuits and Systems, ISCAS 2008, pp 1592-1596, May 2008

[10] A.Lopich and P.Dudek, "Implementation of an Asynchronous Cellular Logic Network as a Co-Processor for a General-Purpose Massively Parallel Array", European Conference on Circuit Theory and Design, ECCTD 2007, pp.84-87, Seville, Spain, August 2007

[11] P. Földesy, R. Carmona-Galan, A´ . Zarándy, C. Rekeczky, A. Rodríguez-Vázquez, T. Roska, 3D multi-layer vision architecture for surveillance and

reconnaissance applications, ECCTD-2009, Antalya, Turkey

[12] A. W. Azman, A. Bigdeli, Y. M. Mustafah, and B. C. Lovell: "*Optimizing resources on an FPGA-based smart camera architecture*". Digital image computing techniques and applications. 2007. Pp. 600-606.

[13] A. gentile and D. Scott Wills: "*Portable video supercomputing*". IEEE Transactions on Computers, Vol 53, Nro 8, August 2004. Pp: 960-973.

[14] Dalsa, "XRI-1200: PC based Digital Image Processor for X-ray Imaging", Datasheet, www.teledynedalsa.com, 2007.

[15] C. Wu, H. Aghajan, and R. Kleihorst: "*Mapping vision algorithms on SIMD architecture smart cameras*". ICDSC 07, 2007. Pp: 27-34.

[16] Z. Xiao and B. Zhang, "Parallel image processing based on pipeline", in Proc. Geoinformatics, 2010, pp.1-4.

[17] H. Norell, N. Lawall and M. O'Nils: "*Automatica generation of spatial and temporal memory architectures for embedded video processing Systems*". EURASIP Journal on Embedded Systems, Volumen 2007, Article ID 75368. 2007. DOI: 10.1155/2007/75368. Pp: 1-11.

[18] Mahendra G. Samarawickrama: "*Performance evaluation of vision algorithms on FPGA*". ISBN: 1-59942-373-1. 2010. Pp: 1-25.

[19] Ahmed Nabil Belbachir: "*Smart Cameras*". Springer. ISBN: 978-1-4419-0952-7. DOI: 10.1007/978-1-4419-0953-4. 2009. Pp: 1-394.

[20] N. Kehtarnavaz and M. Gamadia: "*Real-Time image and video processing: From research to reality*". Springer. DOI: DOI 10.2200 / S00021 ED1 V01Y 2006 04IVM 005. A publication in the Morgan and Claypool Publishers 2006. Pp: 1-108.

[21] B. Kisacanin, S. Bhattacharyya and S. Chai: "*Embedded Computer Vision: Advances in Pattern Recognition*". ISBN 978-1-84800-303-3. DOI 10.1007/978-1-84800-304-0. Springer-Verlag London. 2009. Pp: 1-300.

[22] Alexander Hornberg: "*Handbook of Machine Vision*". ISBN-13: 978-3-527-40584-8. Wiley-VCH 2006. Pp: 1-823.

[23] W. E. Snyder and Hairong Qi: "*Machine Vision*". ISBN: 978-0-521-83046-1. Cambridge Press. 2007. Pp: 27-34.

[24] E. R. Davies: "*Machine Vision. Theory, Algorithms and Practices*". ISBN: 8131201775, Elsevier Press. Oxford University Press. 2003. Pp: 1-938.

[25] Kello Suech: "*Understanding and Applying Machine Vision*". ISBN: 0-8247-8929-6. by Marcel Dekker, Inc. 2000. Pp: 1-336.

[26] Najeem Lawal: "*Memory Synthesis for FPGA Implementation of Real-Time Video Processing Systems*". Mid Sweden University Doctoral Thesis. 2009. ISBN 978-91-86073-26-8.

[27] L. Leiva, N. Acosta, "Detección Rápida de Puntos Salientes en Imágenes", XV Workshop Iberchip, 25 a 27 de marzo 2009, Buenos Aires, Argentina.

[28] L. Leiva, N. Acosta,"MISD Compiler for Feature Vector Computation in Serial Input Images", ARPN Journal of Systems and Software. vol. 1, no. 3, pp: 108-116, June 2011.

[29] Silicon Recognition, "ZISC: Zero Instruction Set Computer", Version 4.2, Silicon Recognition, Inc., 2002

[30] Cognimem, CogniMem_1K: Neural network chip for high performance pattern recognition, datasheet, Version 1.2.1, www.recognetics.com, 2008.

[31] L. Leiva, N. Acosta, "Hardware Radial Basis Function Neural Network Automatic Generation", JCS&T: Journal of Computer Science & Technology. vol. 11, no. 1,pp: 15-20, April 2011.