# A NOVEL SYSTEM ARCHITECTURE FOR REAL-TIME LOW-LEVEL VISION

*A. Benedetti, P. Perona*

California Institute of Technology
Division of Engineering and Applied Science
Pasadena, CA 91125
{arrigo,perona}@vision.caltech.edu

**ABSTRACT**

A novel system architecture that exploits the spatial locality in memory access that is found in most low-level vision algorithms is presented. A real-time feature selection system is used to exemplify the underlying ideas, and an implementation based on commercially available Field Programmable Gate Arrays (FPGA's) and synchronous SRAM memory devices is proposed. The peak memory access rate of a system based on this architecture is estimated at 2.88 G-Bytes/s, which represents a four to five times improvement with respect to existing reconfigurable computers.

## 1. INTRODUCTION

It is well known that real-time processing of video streams is a most expensive task from a computational point of view, due to the high amount of information to be processed. At a resolution of 640 × 480 pixels and 30 frames/sec, for example, the bandwidth of a single monochrome NTSC video stream is 9.2 M-Bytes/sec. The bandwidth of a color video signal is three times as much. Even when simple operations on pixel neighborhoods need to be carried out on such a data stream, the high bandwidth requirements rule out the use of conventional processors. For this reason, general purpose or dedicated massively parallel supercomputers based on the Single Instruction Multiple Data (SIMD) paradigm have long been advocated as a cure to this problem [1]. Massively parallel systems, however, have failed so far to provide a cost effective and flexible solution to the development and widespread use of vision systems, due to the physical constraints preventing their use on the field and their million-dollar price tags. Application Specific Integrated Circuits (ASIC's) have been widely used to implement low-level vision systems. Although they offer good performance, ASIC's do not lend themselves to rapid prototyping of systems and their development has high non-recurring engineering costs. Field Programmable Gate Arrays (FPGA's) emerged as a new technology for the implementation of digital logic circuits during the mid 80's. The basic architecture of an FPGA consists of a large number of Configurable Logic Blocks (CLB's) and a programmable mesh of interconnections [2]. In the beginning FPGA's were mostly viewed as large Programmable Logic Devices, thus they were usually employed for the implementation of the "glue-logic" used to tie together complex VLSI chips like microprocessors and memories used to build general purpose computer systems. In the late 80's and early 90's it became clear that the ability to change electrically the logic functions of FPGA's at almost any point during operation could open an entirely new spectrum of applications. Accelerators built using arrays of reconfigurable devices proved to boost the speed of several applications by up to three orders of magnitude, comparing favorably with supercomputers [3]. Recently, we have designed and demonstrated a 2-D feature selection system implemented on a commercially available FPGA-based reconfigurable computer [4]. This system is composed of a camera, a video decoder, an array of 6 Xilinx FPGA's and an interface to a host PC. This system is, to the best of our knowledge, the only feature selection system developed using reconfigurable devices. During this process we have learned several lessons:

- The use of an array of FPGA's to accomplish a given task adds a level of complexity to the design process, due to the need of manually partitioning the system across several chips.
- Most low-level vision tasks can be accomplished by simple local operations performed across the image, which for the most part map nicely onto FPGA architectures.
- The majority of low-level vision algorithms process the image through a series of independent pipelined stages operating on local pixel neighborhoods of similar size (e.g. gradient computation, followed by non-linear operations).

0-7803-5471-0/99/$10.00©1999 IEEE

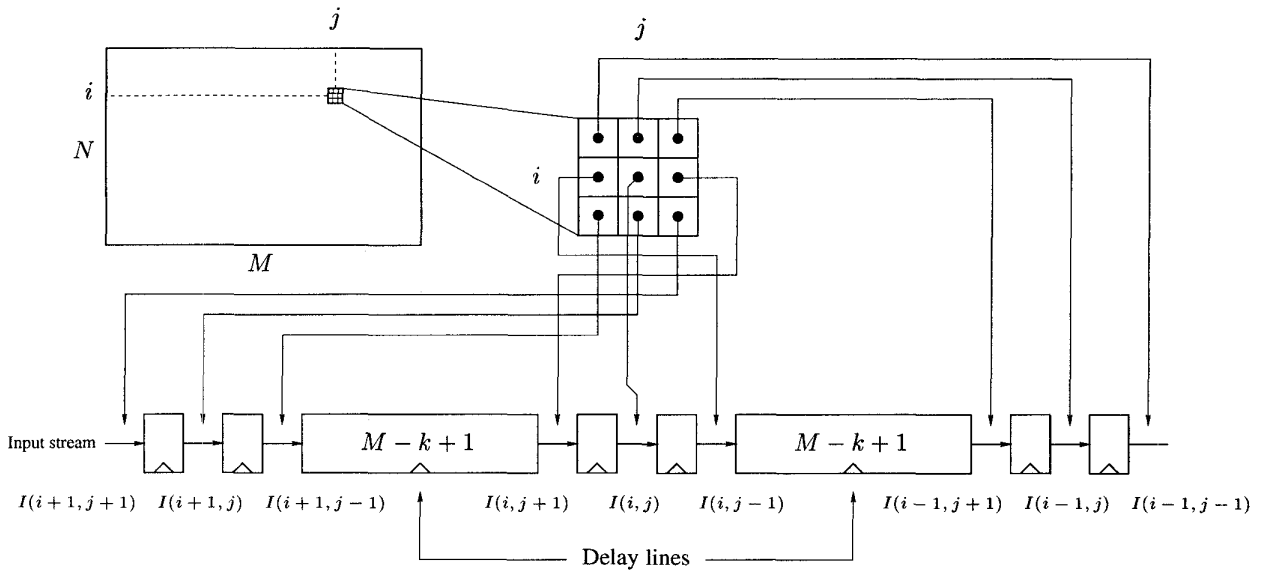


Figure 1: Formation of a 3 × 3 pixel neighborhood.

- Performance of real-time image processing systems is limited by the throughput of memory and I/O channels.

Based on these motivations, and the need felt by many practitioners in the computer vision community, we have designed a novel system level architecture tuned to real-time processing of video streams. This architecture exploits the locality of data access found in low-level vision algorithms and the recent availability of high pin count FPGA devices to partition in an optimal way memory and computation resources. The system that we envision is a PCI expansion board for a PC featuring a high density reconfigurable device, several synchronous SRAM memories and a digital interface for a high resolution progressive-scan video camera. A conservative estimate of the memory bandwidth that we will be able to achieve using off-the-shelf synchronous SRAM memory devices is 2.88 G-Bytes/s at a 60 MHz memory clock rate, which represents a four to five times improvement with respect to existing reconfigurable computers [5].

## 2. REQUIREMENTS OF REAL-TIME IMAGE PROCESSING SYSTEMS

Image processing tasks carried out by low-level vision systems require both memory and computation resources. Memory resources are needed to feed the data to be processed to computation resources in a steady flow, and vary according to the nature of the space where the operation is defined. Spatial operations take into account every pixel of the image and require the availability of the pixel values belonging to a neighborhood defined by some geometric shape. Suppose that a pixel stream is transmitted in raster scan order by a video decoder, and that at every clock cycle a new pixel is available. The simple structure presented in Fig. 1 will make the values of the pixels belonging to a 3 × 3 square window available to computing resources. This window will slide across the entire image, covering a different pixel neighborhood at every clock cycle. This structure is composed of several First In First Out (FIFO) memories and registers synchronized with the video decoder. For a $k \times k$ square neighborhood the length of the FIFO is $M - k + 1$, where $M$ is the width of the image and usually $k \ll M$. In most FPGA architectures registers are abundant, and their implementation does not require excessive area. FIFO memories, however, require an excessive amount of CLB's when implemented as long shift register chains. In the Xilinx XC4000 FPGA architecture, for instance, each CLB contains two flip flops. At NTSC resolution, forming a 3 × 3 neighborhood would require six 8 bit registers and two 8 bits wide and 638 stages deep FIFO's, for a total of 5128 CLB's. On the other hand, the configurable logic blocks found in the XC4000 architecture can be configured as 34 bit SRAM cells, thus bringing the the number of required CLB's down to 302. However, when we consider operations requiring the pixel values of several frames, like filtering a video signal in the time domain, even last generation FPGA devices are not able to provide enough memory resources. The mechanism for neighborhood generation presented in Fig. 1 is easily adapted to the scheme employing an external RAM memory, as exemplified in Fig. 2. The two delay

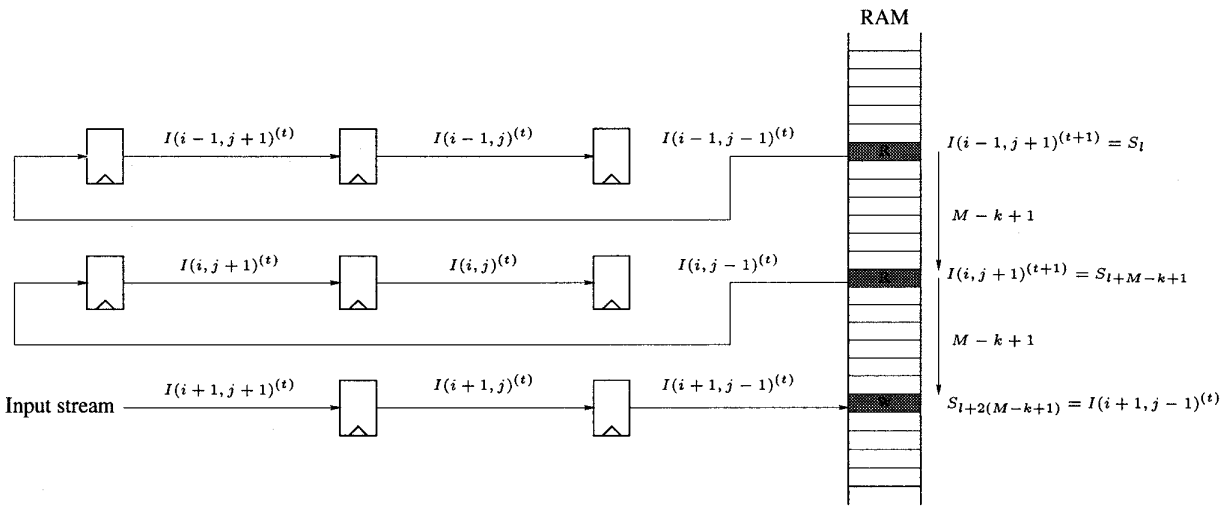


Figure 2: Building $3 \times 3$ pixel neighborhoods by external SRAM memory and internal CLB memory.

lines are here implemented by writing to the external RAM the pixel value entering the first FIFO memory and reading the values corresponding to the output of the FIFO's. The read addresses are obtained by decrementing the write address by $M - k + 1$, and after each pixel clock cycle they are incremented according to

$$\begin{aligned} l_W &= (l_W + 1) \mod 2^q, \\ l_{R_1} &= (l_{R_1} + 1) \mod 2^q, \\ l_{R_2} &= (l_{R_2} + 1) \mod 2^q, \\ &\vdots \\ l_{R_{k-1}} &= (l_{R_{k-1}} + 1) \mod 2^q, \end{aligned}$$

where $q$ is the number of address lines of the memory device. Obviously, $2^q \geq (k-1)(M-k+1)$ must hold. According to this scheme, for every pixel clock cycle one memory write and $k-1$ memory read cycles are issued. Typical values for the pixel clock frequency are in the $12 \div 40$ MHz range, while off-the-shelf synchronous SRAM's are usually clocked at 100 MHz. This means that, according to image resolution, two or three cascaded delay lines will usually fit into a single external memory device.

## 3. A RECONFIGURABLE ARCHITECTURE FOR LOW-LEVEL VISION

The data flow of many image processing systems can be decomposed as a sequence of operations on sets of data whose organization resembles that of the initial image. The first stage of the feature selection system presented in [4], for example, computes the image gradient components $I_x$ and $I_y$. The next operation is the calculation of $(I_x)^2$, $(I_y)^2$ and $I_x \cdot I_y$, which are defined for every pixel in the image. Then $a$, $b$ and $c$, defined by $a = \sum_{l=1}^{k^2} \left(I_x^l\right)^2$, $b = \sum_{l=1}^{k^2} I_x^l \cdot I_y^l$, $c = \sum_{l=1}^{k^2} \left(I_y^l\right)^2$, where the sum is extended over the pixels of a $3 \times 3$ neighborhood, are computed in parallel by three chains of adders interleaved with pixel and line delay elements in order to build a $3 \times 3$ mask in the $(I_x)^2$, $I_x \times I_y$ and $(I_y)^2$ planes. The rest of the system calculates the value of $P(\lambda_t) = (a - \lambda_t)(c - \lambda_t) - b^2$ by time-multiplexing a signed multiplier and performs the test expressed by $P(\lambda_t) > 0$ and $a > \lambda_t$. If the current $3 \times 3$ window passes the test, a red pixel is sent to the video encoder, meaning that that the window contains a "good" feature, otherwise the pixel value from the input stream is transmitted to the video encoder unchanged. Memory resources are necessary to build the pixel neighborhood, whose content is shifted across the "image" associated with the input stream. For the sake of clarity, we will consider a $k \times k$ pixels square neighborhood, and will later relax this assumption. At every clock cycle the current values associated with the neighborhood feed a pipelined function block, computing some (arithmetic) function of the input data. The only constraint imposed on this block is that, after an initial latency of one or more clock cycles, it must generate an output data stream synchronous with the input data stream. The total latency introduced by this stage is thus given by the sum of the latency of the pipelined function block and the number of cycles needed to fill the delay lines so that the central pixel of the neighborhood corresponds to the first pixel of the input stream. Due to these latency periods, the output stream will be delayed with respect to the input stream. Some processing stages, like those computing $(I_x)^2$, $I_x \times I_y$ and $(I_y)^2$, do not need memory resources since they compute numbers that are associ-

ated with individual pixels. Most stages, however, process pixel neighborhoods, thus a modular and efficient scheme for their generation is of the utmost importance for real-time video processing. In the architecture that we propose, the memory resources used to build pixel neighborhoods are provided by external synchronous SRAM memory devices, addressed according to the scheme presented in Fig. 2. The use of external memory devices has several important impacts on the design of the system. The most critical section of the system in terms of timing requirements is the FPGA to memory interface, which is clocked at up to 100 MHz, the maximum system clock frequency supported by most current generation FPGA's. The rest of FPGA logic can run at the slower pixel clock rate, usually in the $12 \div 40$ MHz range. In addition, the FPGA to memory interface can be easily generated from a high level specification of the algorithm that is being mapped. There is an additional key observation that can be exploited to further increase the memory bandwidth of a system based on this architecture. As shown in Fig. 2, the SRAM addresses are generated according to a fixed pattern, and their offset is $M - k + 1$. Different neighborhood sizes, denoted by $k_m$, may be used at the different $P$ stages of the algorithm by taking

$$k = \max_{m=1,\ldots,P} k_m$$

and adjusting the length of the FIFO's used in each processing stage by inserting $k - k_m$ additional registers inside the FPGA. Using this strategy, the address increment is fixed indeed, and this property can be exploited to increase the memory bandwidth of the system as follows. First, we observe that memory devices are addressed according to a fixed and repeating pattern:

1. FPGA writes data to memory location $l_{W1}$,
2. FPGA reads from memory location $l_{R1} = l_{W1} - (M - k + 1)$,
3. FPGA reads data from memory location $l_{R2} = l_{R2} - (M - k + 1)$,
4. ...,
5. FPGA reads data from memory location $l_{R_{k-1}} = = l_{R_{k-2}} - (M - k + 1)$,
6. Increment pointers to read and write locations,
7. Go to 1.

This property allows us to share the $q$ address lines driving the memory devices. Let us put our attention to a high density and high pin count re-programmable device recently developed by Xilinx, the XC40125XV FPGA. The total number of I/O pins available to the user of this device is 448. If we dedicate 32 of these pins to communication with the digital camera and video monitor and 32 pins to communication with the PCI bus interface chip, the remaining 384 are available for interfacing with external memory chips. Up to 12 128K $\times$ 32 bit memory devices can be connected to the main FPGA. The number of FIFO memories that we will be able to fit in a single memory device depends on the widths of the data paths and on the constraint given by the fact that the delay lines implemented in the same device are necessarily cascaded. An estimate of the memory bandwidth that we will be able to achieve using this architecture, accessing the memory at a conservative 60 MHz clock rate, is thus 2.88 G-Bytes/s. This rate, represents a four to five times improvement with respect to existing reconfigurable computers.

## 4. CONCLUSIONS

We have presented a novel reconfigurable architecture dedicated to fast prototyping of real-time low-level vision systems. An observation related to the mechanics of pixel neighborhood generation permits to increase almost by a factor of two the bandwidth of the communication channel between computation and memory resources. By exploiting this idea, an improvement of four to five times with respect to existing reconfigurable computers is achieved. We foresee the application of this architecture in general real-time signal-processing tasks, control systems for autonomous vehicle guidance, vision-based human-machine interfaces as well as in other applications not related to computer vision.

## 5. REFERENCES


[1] C.-L. Wang, P. B. Bhat, and V. K. Prasanna. High-Performace Computing for Vision. *Proceedings of the IEEE*, 84(7):931–946, Jul. 1996.

[2] S. D. Brown, R. J. Francis, J. Rose, and Z. G. Vranesic. *Field-Programmable Gate Arrays*. Kluwer Academic, New York, 1992.

[3] D. Buell (editor). *Splash 2: "FPGA's in a Custom Computing Machine"*. IEEE Computer Society Press, 1996.

[4] A. Benedetti and P. Perona. Real-time 2-D Feature Detection on a Reconfigurable Computer. In *Proceedings of the 1998 IEEE Conference on Computer Vision and Pattern Recognition (CVPR'98)*, pages 586–593, Santa Barbara (CA), Jun. 1998.

[5] J. Woodfill and B. Von Herzen. Real-Time Stereo Vision on the PARTS Reconfigurable Computer. In *Proceedings of the IEEE Symposium on FPGA's for Custom Computing Machines*, pages 201–210, Napa (USA), Apr. 1997.