Design and Implementation of a Reconfigurable, Embedded Real-Time Face Detection System

V. Mariatos[1], K.D. Adaos[2], G.P. Alexiou[2]

[1]Diaplous Machine Vision
6 Ippodamou Str, Patras
26442 Greece

[2]Dept. of Computer Engineering and Informatics,
University of Patras,
26500 Greece

(Contact author: K.D. Adaos: adaos@ceid.upatras.gr)

Abstract

This paper presents the design and implementation of a real time face detection system on an embedded reconfigurable platform. Our approach to face detection is based on a skin-segmentation algorithm followed by feature extraction and face verification. Our implementation is done on DMV, a reconfigurable platform with novel features targeting real time computer vision applications. DMV is a system on chip based on the combination of a high performance 32-bit SPARC-compliant processor with data-flow processing blocks.

1. Introduction

Face detection and face localization are tasks that find application in many areas of contemporary systems. They are used for authentication purposes (together with face recognition [8]), in security and surveillance systems, in the management of image and video databases, in intelligent human-computer interfacing and so on.

Face detection is regarded as the most important task of face recognition. Efficient and accurate detection of the presence and location of a face in an image or sequence of images, simplifies the task of determining the identity of the person. The algorithms used for implementing face detection and other similar computer and machine vision tasks are demanding in terms of speed.

Most practical face detection and recognition systems must be capable of performing their functionality in real-time or near real-time manner. Hardware implementation is therefore desirable to achieve the required speed and obtain better results in terms of the mobility of the final system or reduced power consumption.

The concept of system-on-chip (SoC) that integrates one or more processors with custom hardware, is a popular approach to build systems with characteristics close to the all hardware implementation. At the same time, SoCs offer the capability to alter their functionality with software revisions. Configurability can be further enhanced by using Field programmable Gate Arrays (FPGAs). Even when FPGAs are not desirable for the final implementation of the system, they are quite often used in the first stages of the product development, to build a system prototype. This allows the developers to perform extensive architectural exploration and fine-tuning of their design.

The platform used for the implementation of our face detection system is DMV (acronym of Digital Machine Vision). DMV has been introduced by Diaplous [1], a Greek fabless semiconductor company, to address the needs of implementing demanding computer vision algorithms.

Section 2 describes the generic architecture of DMV and its current implementation. Section 3 describes the face detection procedure. Section 4 presents our implementation results. Finally we conclude the paper and give the future directions of our work.

2. The DMV Architecture

2.1. Generic DMV Architecture

The DMV architecture extends the concept of a classic system-on-chip with hardware blocks that can perform complex image manipulation tasks. A set of front-end hardware image processing blocks reduce the amount of data that need to be handled by software and stored in memory. To further relieve the processor, another set of hardware blocks implement higher-level image processing functions. The result is a compact architecture, that does not require the fastest available processor nor does it require huge amounts of temporary-storage memory. Figure 1 depicts the main parts of the architecture that is built around a conventional system-on-chip based on a shared bus.

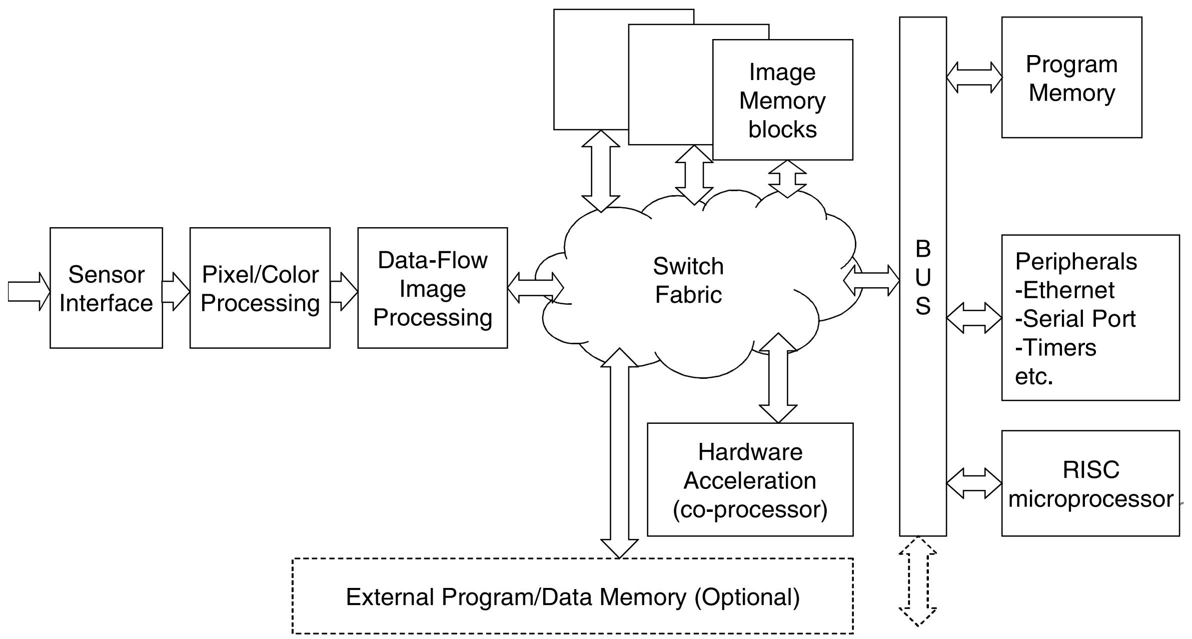
Image Memory blocks
Program Memory
Sensor Interface
Pixel/Color Processing
Data-Flow Image Processing
Switch Fabric
B U S
Peripherals -Ethernet -Serial Port -Timers etc.
Hardware Acceleration (co-processor)
RISC microprocessor
External Program/Data Memory (Optional)

Figure 1: The DMV Engine Architecture

The system processor is mainly used for control tasks and non time-critical tasks. All hardware resources are controlled by the main processor through the shared bus.

The DMV engine capabilities can be extended by including a second processor with its own local bus to communicate with the image processing blocks. This processor can also have its own interface to external high speed memory. This hierarchical bus organization with separate memory subsystems can further relieve the main processor from the data processing tasks and provide a memory organization that is optimal for real-time image processing algorithms. It must be noted that the DMV architecture is not limited by the selection of any specific processor or memory subsystem architecture.

2.2. Current DMV Engine Implementation

The generic concept of the DMV Engine has been implemented in a real hardware system by use of a Xilinx Spartan3 FPGA (figure 2). The processor used in this implementation is LEON2 available from Gaisler Research [2] under LGPL license (free for both research and commercial applications). LEON2 is a 32-bit SPARC V8 compliant processor, provided as a synthesizable VHDL model. This model is highly configurable, and particularly suitable for SOC designs. Its architecture includes a pipelined integer unit, hardware multiply, divide and MAC units, configurable cache subsystem, AHB and APB on-chip buses, memory controllers for external PROM, SRAM and SDRAM, on-chip low speed peripherals, interrupt controller and a 10/100 Ethernet MAC (based on the Opencores MAC core). Software development can be done with the GNU toolset. LEON2 is capable of executing a total of 0.85 dhrystone MIPS/MHz.

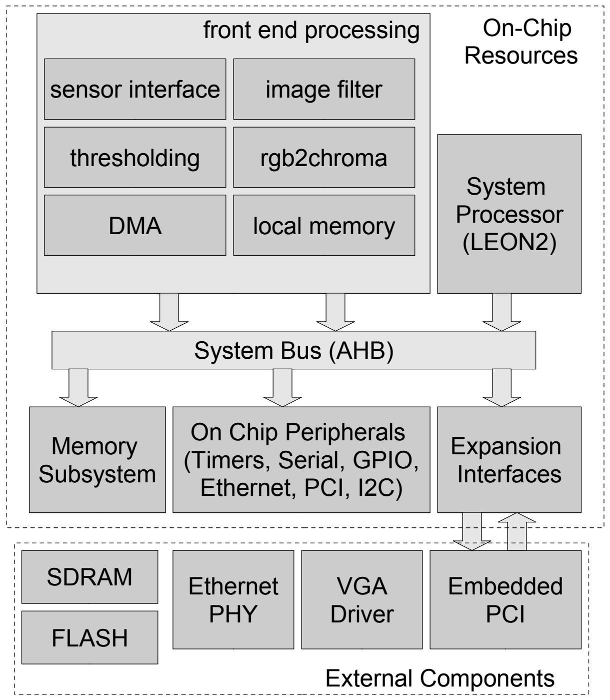

Figure 2: Current Implementation of DMV

This base system has been extended with an embedded master/target PCI-like interface to allow two or more similar systems to connect at the board level. Custom tools have been developed to allow control of the system from any workstation possessing a serial port. This debugging interface allows accessing the system AHB bus during real time operation without processor intervention.

We have customized the DMV engine architecture for the image processing tasks required for our face detection system by designing a set of operations that can be applied to the image captured by a high resolution image sensor. The set of tools provided by Gaisler and Diaplous proved sufficient for the rapid implementation of the face detection algorithms.

3. Face Detection

For face detection, we follow a three stage procedure (figure 3). The first stage performs skin segmentation on the image acquired by the image sensor. It is based on the skin color model proposed in [3]. This color model has been devised to overcome the sensitivity to illumination conditions, a problem that is common to color-based skin detection approaches. This model uses a modified GLHS space [4]. It converts the Red, Green, Blue (RGB) space to lightness, hue and saturation components. In [3], the original GLHS equations of [4] for the Lightness and Saturation have been modified in order to make them independent of the specific illumination conditions. In summary, the computations performed by this model are:

(1) Lightness: $l(c) = \max(c) + \min(c) - 2\mathrm{mid}(c)$

(2) Hue: $h(c) = (k(c) + f(c)) \times 60$

(3) Saturation : $s(c) = (R+G+B-3\min(c))/3$

R,G,B are the red, green and blue values of each image pixel in the range 0 to 255

$\min(c) = \min(R,G,B)/255$, $\mathrm{mid}(c) = \mathrm{mid}(R,G,B)/255$,

$\max(c) = \max(R,G,B)/255$

$$k(c) = \begin{array}{ll} 0 \text{ if } R > G \geq B, & 1 \text{ if } G \geq R > B \\ 2 \text{ if } G > B \geq R, & 3 \text{ if } B \geq G > R \\ 4 \text{ if } B > R \geq G, & 5 \text{ if } R \geq B > G \end{array}$$

$$f(c) = \begin{array}{l} (\mathrm{mid}(c)-\min(c))/(\max(c)-\min(c)) \text{ if } k(c) \text{ is even} \\ (\max(c)-\mathrm{mid}(c))/(\max(c)-\min(c)) \text{ if } k(c) \text{ is odd} \end{array}$$

Based on the above equations, for a pixel to be classified as being a skin pixel, the following conditions should be satisfied:

(4) $0.065 \leq s(c) \leq 0.25$

(5) $-0.15 \leq l(c) \leq 0.27$

(6) $0.005 \leq h(c) \leq 0.12$

(7) $R \geq 90$

Equations (1) to (7) are implemented in the front-end stage of our hardware implementation. They are applied to image pixels directly after the sensor interface. The output of this stage is a binary image (one bit is used to determine if a pixel is or is not a skin pixel). This output is used as a mask to remove the pixels of the original image that do not belong to skin. The image obtained after this masking is the input to the eye detection phase.

Eye detection uses an eye template as reference. This template consists of a rectangle of dark pixels surrounded by a zone of light ones (the skin surrounding the eye). This template is searched in all image positions. All positions that match this template are recorded.

In the third stage we perform face location and verification. Every pair of eyes detected in the second stage determines an image region that potentially belongs to a face. Before marking this region as an actual face region, a verification procedure is applied. In an actual face region, we expect that certain areas, relative to the detected eyes, are occupied by skin. By using the skin information of the first stage, we check whether skin is present in these areas. If these verification criteria are met, we mark the region as a face region and proceed to check other pairs of eyes.

The second and third stage of the face detection procedure are implemented by software executing in the main system processor.

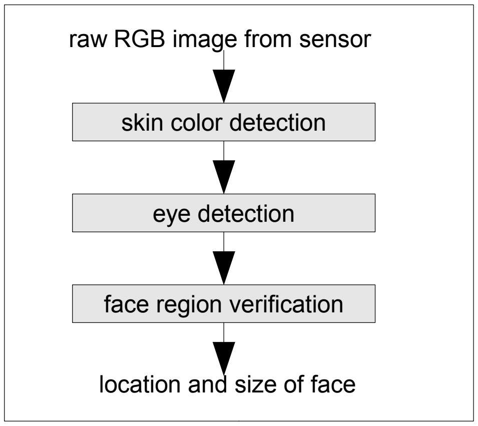

Figure 3: Face Detection Procedure

4. Implementation Results

The implementation of our system has been done with Xilinx FPGAs in a board that hosts one xc3s1000 device of the SPARTAN3 Family. Synthesis and implementation was done with Xilinx WebPack version 8.1i. The speed target of the implementation procedure was 66 MHz in order to match the nominal frequency of the SDRAM. This speed target was easily obtained with the medium optimization level of the Xilinx implementation tools. Better results can be expected by using a part with higher speed grade or changing the constraints given to the synthesis tool.

Table 1: FPGA Implementation Results

Speed	66 MHz
Block Rams	21
Flip-Flops	3809
4-input LUTs	10339

We also performed synthesis for a 0.18μ Standard Cell Library (UMCL18U250) provided by Europractice, by using Synopsys' Design Compiler version 2006.06-SP3. All experiments used the most pessimistic wire-load model.

To investigate the potential of our implementation we followed two synthesis strategies. Strategy 1 (area optimized design) performed synthesis in two stages. Initially, we synthesized the design with area only constraints. The result which was optimal in terms of gate count was then constrained for a maximum clock period of 10 ns. Synthesis was reinvoked with input the result of the first stage and improved timing obtaining a delay close to 10 ns. Strategy 2 targeted the speed optimized implementation with a single synthesis pass. Design Compiler was instructed to target a 6 ns delay.

Further frequency increase can be obtained by using a more optimistic wire-load model that matches the gate count of the design. In any case, the results of table 2 are indicative of the efficiency of the design.

Table 2: ASIC Implementation Results

	Strategy 1	Strategy 2
Total Area (equiv. Kilo -gates)	62.2	65.9
Speed (MHz)	98	166

Both FPGA and ASIC implementations required a total of 42 KBytes of on-chip RAM.

To measure the performance of our face detection system, we experimented by defining operation in a region of 512 by 512 pixels. We also restricted the search space of the third stage of the face detection algorithm to use pairs of detected eyes that reside close to the horizontal axis in order to avoid the expensive in terms of computational time operation of rotation. We are working on implementing specific hardware blocks for rotation and remove this task from the system processor. Another limitation inherent to our search procedure is that we cannot provide detection of faces when one of the eyes is not visible. We investigate promising techniques described and referenced in the literature [6] [7] to overcome this obstacle. With the frequency of 66 MHz that was used in the FPGA implementation we have obtained a processing rate of 15 frames per second.

5. Conclusions – Future Work

We have presented the design and implementation of a real-time system for face detection in images captured from an FPGA board that implements DMV, an architecture of a system-on-chip with data-flow processing enhancements. The set of support HW and SW tools developed and used, provide a flexible environment of rapid implementation and evaluation of image and computer vision algorithms. Results obtained from the implementation in FPGAs, as well as from synthesizing the design for a standard cell library indicate that our approach is suitable for both ASIC and FPGA based implementations of computer vision tasks.

We are currently working in enhancing the performance of our implementation by designing hardware acceleration modules to be placed in the front-end section of the DMV engine. We are also working on eliminating the limitations regarding the processing of only frontal images with horizontal eye-pair position.

References

[1] Diaplous Home Page, www.diaplous.com

[2] Gaisler Research Home Page, www.gaisler.com

[3] W. Zheng, Z. Lu and X. Xu, "A novel skin clustering Method for Face Detection", *Proc. of 1st Int. Conf. on Innovative Computing, Information and Control*, Beijing, China, August 2006, pp. 166-169

[4] H. Levkowitz, G. Herman, "GLHS: a generalized lightness, hue, and saturation color model", *CVGIP: Graphical Models and Image Processing*, Vol 55 , No 4, 1993, pp. 271 – 285

[5] G. Shakhnarovitz, P. Viola, B. Moghaddam "A Unified Learning Framework for Real Time Face Detection & Classification", *Mitshubishi Electric Research Laboaratories TR2002-23*, May 2002

[6] A. Pnevmatikakis and L. Polymenakos, "An Automatic Face Detection and Recognition System for Video Streams" *2nd Joint Workshop on Multi-Modal Interaction and Related Machine Learning Algorithms*, Edinburgh, UK, July 2005.

[7] R. Hota, V. Venkoparao and S. Bedros, "Face Detection by using Skin Color Model based on One Class Classifier", *Proc. 9th Int. Conf. on Information Technology*, 2006, pp. 15-16

[8] W. Zhao, R. Chellappa, A. Rosenfeld, P.J. Phillips, "Face Recognition: A Literature Survey", *ACM Computing Surveys,* 2003, pp. 399-458