Xilinx System Generator Based HW Components for Rapid Prototyping of Computer Vision SW/HW Systems[*]

Ana Toledo[1], Cristina Vicente[2], Juan Suardíaz[1], and Sergio Cuenca[3]

[1] Departamento de Tecnología Electrónica, Universidad Politécnica de Cartagena, Spain
{ana.toledo,juan.suardiaz}@upct.es
[2] Departamento de Tecnologías de la Información y Comunicaciones
Universidad Politécnica de Cartagena, Spain
cristina.vicente@upct.es
[3] Departamento de Tecnología Informática y Computación, Universidad de Alicante, Spain
sergio@dtic.ua.es

Abstract. This paper shows how the Xilinx System Generator can be used to develop hardware-based computer vision algorithms from a system level approach without the necessity of in-depth knowing neither a hardware description language nor the particulars of the hardware platform. Also, it is demonstrated that Simulink can be employed as a co-design and co-simulation platform for rapid prototyping of Computer Vision HW/SW systems. To do this, a library of optimized image processing components based on XSG and Matlab has been developed and tested in hybrid schemes including HW and SW modules. As a part of the testing, results of the prototyping and co-simulation of a HW/SW Computer Vision System for the automated inspection of tangerine segments are presented.

Keywords: image processing applications, FPGAs, prototyping, co-simulation, Simulink.

1 Introduction

Nowadays, the key for implementing high-performance digital signal processing (DSP) systems, especially in digital communications, video and image processing applications, is the use of programmable logical devices, in particular Field Programmable Gate Arrays (FPGAs). However, for those applications in which high-level complex algorithms are involved, a complete HW implementation is unpractical. In these cases it is usual to employ a hybrid SW/HW implementation, in which the hardware (typically a FPGA) carries out the acceleration of specialized functions and a processor, usually a conventional CPU, accomplishes general purpose computing.

Traditionally, the HW/SW application prototyping is performed in different environments, using a high-level programming language for the SW, e.g. C, C++ or Mat-

[*] This work has been partially supported by the Spanish CITYT Project COSIVA (TIC 2000-1765-C03-02).

lab, and a Hardware Description Language (VHDL or Verilog) for the HW description. This makes the co-simulation difficult, and leads to two versions of the same code in different programming languages. Moreover, the rapid evolution of FPGAs makes the time employed in the development of hardware-based applications be a critical parameter. For these reasons, some efforts have been made to construct co-design environments allowing rapid prototyping, high-level modeling, co-simulation, and straightforward HW/SW code generation [1-3].

Currently, there exist two main tendencies in the system-level co-design environments: high-level languages [4] and dataflow-based visual environments [5, 6]. High level languages are efficient for specification modelling and algorithm verification, but they are not suitable for the implementation of high-performance dataflow systems as in the Computer Vision Systems (CVS). On the contrary, the visual dataflow-based environments are similar to the traditional schematic-based tools, which usually provide libraries composed of blocks with a high degree of functional abstraction that allow graphically constructing system models.

Simulink is an extension of the widely-used MATLAB environment that is specifically oriented for graphic prototyping and simulation of dynamical systems [7]. It also has a natural interface with MATLAB, so that its analysis and graphical representation tools can be used in the MATLAB workspace for post-processing and visualization. Like MATLAB, Simulink supports the extension of its functionalities by means of the add-in of application-specific libraries of components (*toolboxes*). Since the inclusion of toolboxes that allow the simulation and generation of hardware components, as the DSP Blockset toolbox or more recently the Xilinx System Generator (XSG) [5] and the Altera DSP Builder Altera [6] toolboxes, Simulink has become a powerful tool for HW/SW co-design [8, 9].

The Xilinx blockset contains high-level blocks that map intellectual property (IP) cores that have been handcrafted for efficient implementation in the target Xilinx FPGA. However, the XSG toolbox includes only some basic blocks that can be used as "bricks" for developing more complex structures. Based in these simple blocks, in this work the development of a visual processing library is presented. The library components have been optimized both in processing efficiency and in FPGA occupancy. Taking advantage of the XGS link with Matlab, the library blocks have been parameterized. This greatly eases the use of the library, as it can be used for a wide variety of applications without the need for the user of changing the code. Also, some Matlab-based software components have been implemented to allow co-simulation. They constitute the foundations of a complete HW/SW co-design and co-simulation Simulink-based scheme that will allow the rapid prototyping and implementation of hybrid CVS applications using Xilinx FPGAs.

The rest of this paper is organized as follows. The blocks of the visual processing library, jointly with a brief description of some of the underlying algorithms are explained in Section 2. The steps followed to design an HW/SW CVS using this library are briefly reviewed in Section 3. In Section 4, the use of this library to develop a complete study case is presented. Finally, conclusions and future research lines are included in Section 5.

2 The HW-SW Visual Processing Library

The designed CVS library consists of Simulink blocks. This library, whose purpose is the modelling and generation of HW-SW image processing and computer vision applications, behaves as any other Simulink library, and fully integrates with the Matlab/Simulink simulation environment. The design of a HW-SW system is thus performed by "dragging and dropping" the library blocks onto the Simulink editor, in which they are linked to construct the functional prototype.

2.1 HW Blocks

The HW components process the input pixels as they come in raster-scan order (from left to right and from top to bottom), so there is no necessity of having a whole image stored to begin the processing. This reduces the storage requirements and the amount of memory accesses, which generally constitute a bottleneck.

All the blocks have been homogeneously designed to assure interconnectivity. At each clock cycle, every block receives jointly with the input data (usually a pixel value) two control signals corresponding with the synchronization signals: line blank (LBL) and frame blank (FBL). Besides of the output data (generated at each cycle), the blocks include two additional output signals corresponding with the LBLo and FBLo control signals of the output stream.

The designed blocks are straightforwardly parameterized, so that they can be used for processing images with different sizes or formats without reprogramming. The arithmetic precision of the blocks in the data path is specified using Matlab expressions, making possible to minimize the hardware used, and avoiding the possibility of overflow. Therefore, changing parameters automatically gives an appropriately customized implementation.

Several image processing algorithms have been implemented using XSG.

- Colour space conversions.
- Image cropping.
- Brightness/Contrast shift and scale.
- Threshold and double threshold.
- Specific filters: Gaussian, Laplacian, Prewitt, Sobel, Mean, Sharpening Median…
- Generic convolutions 3x3 and 5x5.
- Morphological binary operations with generic 3x3 and 5x5 structuring elements: erosion and dilation.
- Logical and Arithmetic operations: and, or, nor, add, sub…
- Connected Component Labelling.
- Calculus of the zero and first-order moments.

Some of them are detailed in the following lines.

Specific convolutions: As there exist some very frequently used convolution kernels, as the Prewitt, Laplacian and the Sobel kernels for edge detection, the Mean and Gaussian filters for noise filtering or the Sharpening kernel for image enhancement, specific optimized code has been developed for them, to minimize resources while achieving high performance.

The input data stream arrives to the convolver in "row scan" format, which means that for every processing pixel it is necessary to wait until all the elements involved are available. Therefore, it is required a delay to store N-1 image lines (N being the row number of the convolution mask) that is implemented using N-1 FIFO memories.

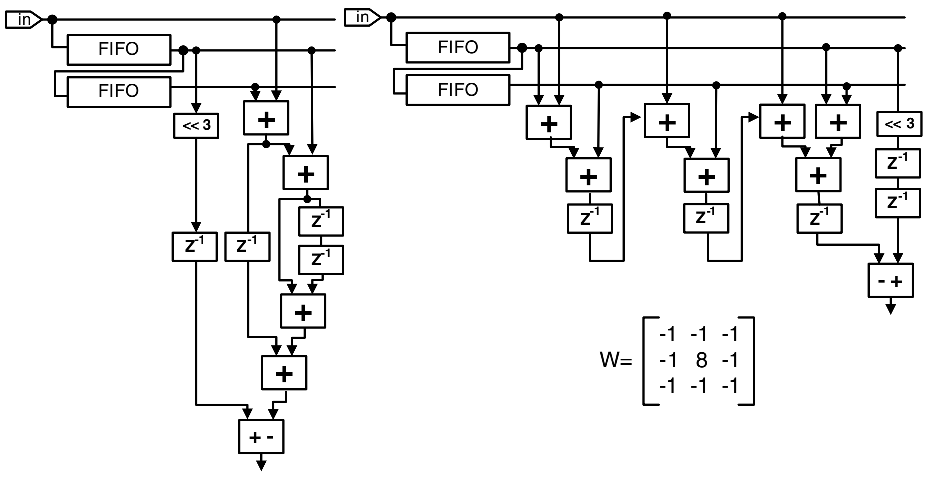


Fig. 1. Two implementations of a 3x3 convolution. Left: proposed scheme. Right: Lisa's scheme.

Lisa [10] offers a column-access architecture, which minimizes the amount of resources used in the FPGA by in-parallel processing the columns of pixels involved on each computation. Besides, by decomposing the mask weights in their binary representation, many multipliers can been replaced by shift-registers and adders. From the Lisa architecture, the convolutions have been optimized by decreasing the number of adders. To illustrate the followed procedure, in Fig. 1 the Lisa scheme for a simple filter is shown on the right, while the implemented one is shown on the left of the figure.

Another implemented algorithm that requires special mention is the connected component labelling. This algorithm is usually in the base of high-level image processing. Its input is a binary or grey-level image, while its output is a symbolic image, in which a label (usually a natural number) is attributed to each pixel in the image to symbolize that it belongs to an object represented by the label.

Since the shape of the object can be arbitrary, connected component labelling involves significant data computation and communication between the pixels in the image. To solve this problem, several sequential and parallel algorithms have been proposed [11]. In the library, the classical algorithm, which makes two forward raster scan passes through the image, has been implemented.

However, most times labelling is only required as a previous step for calculating properties of the objects as their masses, centres-of-mass or higher-order moments.

Taking this into account, special blocks have been constructed that perform these calculations without needing the second forward pass through the image, thus saving processing time and FPGA occupancy.

2.2 SW Blocks

Generic high-level image processing algorithms have been implemented by encapsulating in Simulink blocks some functions from the Image Processing Toolbox of MATLAB. In the construction of these wrappers, it has been taken into account that while the hardware processing is pixel-oriented, the software processing is frame-oriented. This causes synchronization problems, as the SW and HW will typically run on different frequencies. To tackle this, each SW block has been provided with an *enable* input that is marked as TRUE every time a frame is available from the HW.

For allowing the simulation of a whole system including acquisition, some blocks, which do not generate code in the final implementation, have been developed.

Camera blocks. Several camera blocks are provided, which model the behaviour of various common digitizers and non-interlaced digital cameras. Currently, there are available three kinds of blocks, giving YCrCb 4:2:2, Luminance and RGB output signals respectively. These blocks read one or more image files (they accept most of the common file formats for image storage) and provide, jointly with the corresponding pixel values, the appropriate LBL and FBL synchronization signals.

Viewer blocks. A number of viewer blocks have been built, to allow an easy inspection of the data flowing by the pipelines. At the moment, there are viewers for all the image types given by the camera blocks, plus several specific viewers that show the outputs of some blocks like the labelling block or the area and centre-of-mass blocks.

3 Design Flow

Using this library, to create a CVS application the user must only drag the corresponding blocks from the library and drop them into a Simulink empty model, then interconnect the blocks to form the application flow diagram. The constructed model can be simulated in Simulink, employing the stimuli and visualization blocks included in the library. This simulation allows verifying if the desired functionality has been achieved, and it is considerably faster than simulations performed by specific hardware simulators as ModelSim [12]. Due to the library HW blocks are made up of XSG simple blocks, for the HW the simulation results are identical to those that would be obtained in the real FPGA implementation.

After simulation, if the functional requirements are met, the user must decide on the target platform for the HW partition. Once the target platform has been selected, the hardware code (VHDL) can be automatically generated. This code incorporates optimized Xilinx LogiCORES, thus assuring that the implementation will be efficient.

The generated HW code is automatically encapsulated inside a VHDL project, in which the specific code for I/O interfacing related to the selected platform is included. This is required because XSG is a good setting in which to implement data paths, but is less well suited for sophisticated external interfaces that have strict timing requirements (for example, it can not work with several external clock sources). To tackle this, a HDL wrapper has been created that automatically generates the necessary code for the I/O interfaces, i.e. the video transfer from the digitizer (or from the camera), the transfers to/from external memories and the data transfer to/from SW blocks. As said before, this wrapper is specific for each HW target platform.

This enveloping VHDL project can then be automatically synthesized. As a result of the synthesis, the cost (measured in FPGA area occupancy), the maximum working frequency for the FPGA and the HW execution time are obtained.

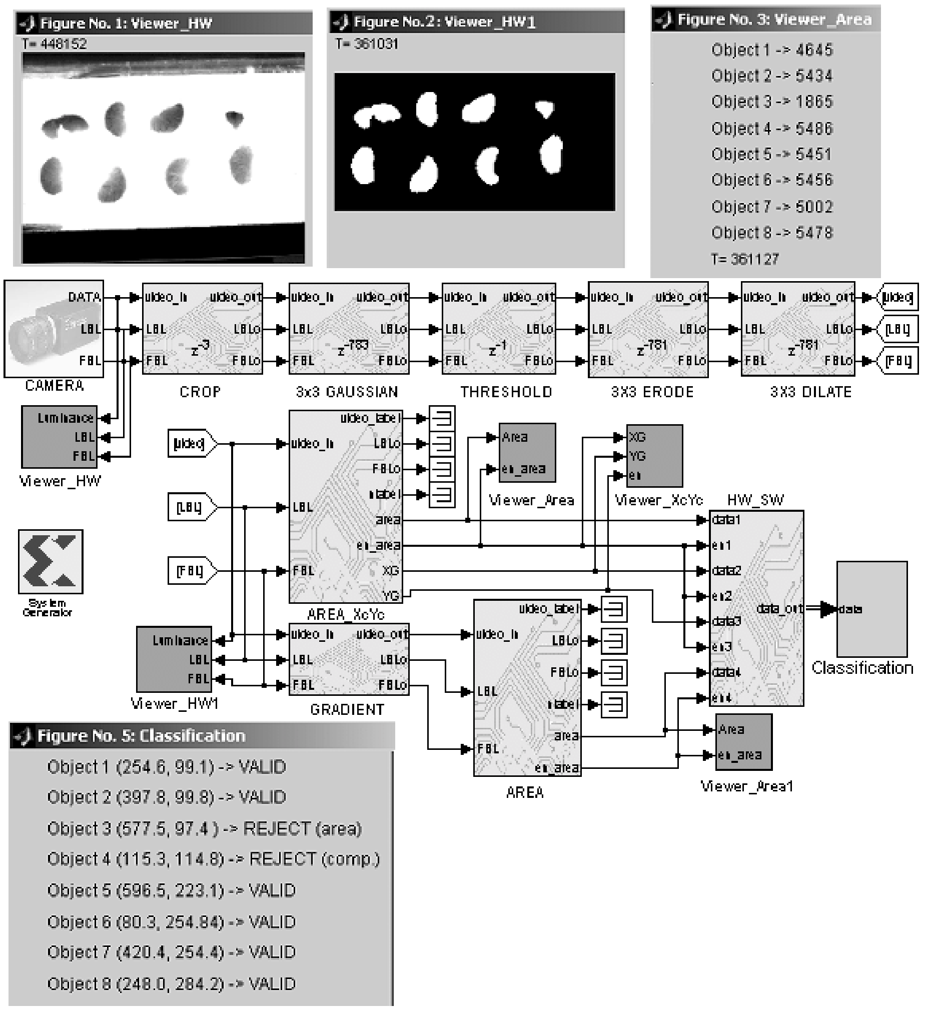
Finally, if the temporal requirements are fulfilled, the FPGA on the HW platform is configured with the bitstream file (automatically generated during the synthesis), by means of the appropriate tool provided by the manufacturer.

4 A Practical Study Case

In order to test the capabilities of the library, a CVS for tangerine segment inspection has been constructed from scratch. The objective of the CVS is to reject the tangerine segments that appear split in pieces, or that are simply too small to be canned. This must be done in real-time, because the inspection system is working on line in the final step of the canning line. This canning line is composed of a conveyor belt, which transports the tangerine segments under a camera and through several air-jet ejectors to the canning mechanism.

The scheme of the HW/SW proposed CVS is shown in Fig 2. The algorithm works as follows: as the pixel information is submitted from the camera, a cropping is performed to discard pixels from areas outside the conveyor belt. On the passing pixels, a 3x3 Gaussian filter (with standard deviation = 0.9) is applied to remove noise. Thus, a thresholding is carried out to separate the tangerine segments (dark) of the conveyor belt (bright). A binary opening (erosion + dilatation) is then performed with a 3x3 solid structural element, to remove binary noise in the form of small blobs. After this, the pipeline forks. In one of the lines, the binary image is processed to obtain the area and center-of-mass of the blobs. This information is fed into a HW/SW interface block which pumps the data into the SW blocks. Simultaneously, in the other line, edge detection is carried out, so that a binary image containing the borders of the blobs is obtained. This image is processed by a block that gives the area of these borders. In this way, a measure of the perimeter of the blobs is obtained and, though the HW/SW interface block, it is fed to the SW processing blocks.

The HW part of the algorithm was implemented in a Nallatech Ballynuey 3 card [13]. This card is a general-purpose PCI board, specially designed for prototype development, and it is based on a Virtex 2V3000 FPGA. It incorporates two external ZBT SSRAM memories (8 Mb) and four DIME slots of expansion. In one of them, a Nallatech Ballyvision module was connected, to allow input from an external PAL camera.



```
Figure No. 3: Viewer_Area
Object 1 -> 4645
Object 2 -> 5434
Object 3 -> 1865
Object 4 -> 5486
Object 5 -> 5451
Object 6 -> 5456
Object 7 -> 5002
Object 8 -> 5478
T= 361127
```

```
Figure No. 5: Classification
Object 1 (254.6, 99.1) -> VALID
Object 2 (397.8, 99.8) -> VALID
Object 3 (577.5, 97.4 ) -> REJECT (area)
Object 4 (115.3, 114.8) -> REJECT (comp.)
Object 5 (596.5, 223.1) -> VALID
Object 6 (80.3, 254.84) -> VALID
Object 7 (420.4, 254.4) -> VALID
Object 8 (248.0, 284.2) -> VALID
```

Fig. 2. Simulink co-simulation screenshot. SW blocks are shown in a plain-color while patterned ones denote HW components. Images resulting from some SW/HW processing steps are shown together with the corresponding temporization cycles.

As a result of the synthesis on the Virtex 2V3000 FPGA, the proposed hardware architecture is able of processing 778x576 images with a cycle time of 11.62 ns (82 MHz). This implies that more than 190 frames per second can be processed, thus allowing a high pace to the conveyor belt.

5 Conclusions

In this work, a CVS library has been developed, which allows using the Matlab/Simulink environment for prototyping, co-simulating and automatic HW code generation of HW/SW computer vision systems. The hardware blocks, based on the XSG tool, have been parameterized and optimized. Also, the HW code generation has been fully automated, including wrapping mechanisms that extend the original capabilities of the XSG. The result is a library of blocks fully integrated in Matlab/Simulink that greatly eases the functional prototyping, verification and final implementation of HW/SW computer vision systems without the necessity of mastering neither a hardware description language nor the intricacies of the hardware platform.

To test the CVS library, a HW/SW computer vision system for the automated inspection of tangerine segments has been constructed form scratch, with excellent results.

References

1. Arnout G., C for System Level Design, Proceedings of Design, Automation and Test in Europe Conference and Exhibition, March 1999.
2. Panda P.R., SystemC - a modeling platform supporting multiple design abstractions, Proceedings of the 14th ISSS, 2001 pp. 75 -80
3. Hwang J, Milne B, Shirazi N, Stroomer J., System Level Tools for DSP in FPGAs. FPL 2001, Lecture Notes in Computer Science, pp 534-543.
4. SystemC. Available: http://www.systemc.org
5. System Generator: Reference guide, http://www.xilinx.com/
6. DSP builder. Available: http://www.altera.com/
7. The Math Works Inc., http://www.mathworks.com
8. Líčko M., Schier J., Tichý M., Kühl M.: MATLAB/Simulink Based Methodology for Rapid-FPGA-Prototyping. FPL 2003.Vol. 2778, pp 984-987.
9. Denning D., Harold N., Devlin M., Irvine J.: Using System Generator to Design a Reconfigurable Video Encryption System. In: P.Y.K. Cheung et al. (Eds.): FPL 2003. Lecture Notes in Computer Science, Vol. 2778, pp 980-983.
10. Lisa F, Cuadrado F, Rexachs D, Carrabina J: A reconfigurable coprocessor for a PCI-based real time computer vision system. FPL 1997. pp 392-399.
11. Wang K., Chia T., Chen Z., Lou D.: Parallel Execution of a Connected Component Labeling Operation on a Linear Array Architecture. Journal of Information Science and Engineering 19, pp 353-370 (2003).
12. ModelSim. Available: http://www.model.com
13. Nallatech. http:// www.nallatech.com