An Advanced Embedded Architecture for Connected Component Analysis in Industrial Applications

Menbere Tekleyohannes, Mohammadsadegh Sadri, Christian Weis, Norbert Wehn
University of Kaiserslautern, Germany,
{tekley, sadri, weis, wehn}@eit.uni-kl.de

Martin Klein, Michael Siegrist
Wipotec Wiege- und Positioniersysteme GmbH
Kaiserslautern, Germany
{martin.klein, michael.siegrist}@wipotec.com

***Abstract*—In recent years, connected component analysis (CCA) has become one of the vital image/video processing algorithms due to its wide-range applicability in the field of computer vision. Numerous applications such as pattern recognition, object detection and image segmentation involve connected component analysis. In the context of camera-based inspection systems, CCA plays an important role for quality assurance. State-of-the-art hardware architectures offer high performance implementations of CCA using field programmable gate arrays (FPGAs). However, due to their high memory-demand, most of these implementations inhibit a large resource utilization. In this paper, we propose a hybrid software-hardware architecture of CCA for an industrial application using Xilinx Zynq-7000 All Programmable System on Chip (SoC). By offloading the most resource consuming part of the algorithm to the embedded CPU, we achieved high performance, while reducing the required resources on the FPGA. Our proposed architecture saves more than 30% of on-chip memory (Block RAMs) compared to state-of-the-art hardware architectures without affecting the throughput. Furthermore, due to the embedded CPU, our system provides a versatile and highly flexible feature extraction at run-time without the necessity to reconfigure the FPGA.**

I. Introduction

Due to the fourth industrial revolution, the limits of modern production machineries are being challenged. This trend forces technology-based innovations into self-optimizing, self-configuring, flexible and intelligent systems. As the machines get smarter, they perform most of the complex tasks independently. With autonomous and adaptable features, machines can also easily integrate production changes in field. In order to fulfill all these traits, the intelligence (smart-system) must be integrated or embedded directly on the machine.

The CCA is one of the image processing algorithms essential for final decision making used in inspection systems. CCA operates on an image to obtain a variety of information from different connected components. This may include area, bounding box, maximum/minimum intensity, center of gravity, component count and many others.

Our inspection system comprises a chained image processing operations using a fast X-ray camera. The current system is implemented in Core i7, which supports a variety of applications with different parameters. Challenges arise with the requirement of faster product lines with high flexibility and low power consumption. Additionally, the machine has to adapt to changes after production. To meet these four main requirements (flexibility, throughput, power and adaptability), choosing the most convenient platform is essential. Our initial platform analysis exhibits FPGA-based solution to be preferable, due to the lower power consumption. However, a CPU provides more flexibility. Due to its software-hardware programmability, we selected the Xilinx Zynq-7000 device as our target platform. With a proper hardware-software partitioning, we exploit the capability of both the programmable fabric and the embedded CPU to provide a flexible yet high-throughput system. The proposed architecture is part of this image processing system. Our experiments show, this architecture fulfills all the requirements including the field-customization property, where the system is able to adapt to changes post machine deployment.

II. Related Work

The classic two-pass connected component analysis (CCA) algorithm, requires two raster-scan passes through the image. In [1], this two-pass CCA is implemented on an FPGA, resulting a performance of two clock cycles per pixel. Wu et al. [2] proposed an optimized two-pass CCA with an array-based union-find algorithm. However, this design exhibits a high resource utilization. To avoid the need for buffering the image in the internal memory of the FPGA, a single pass CCA was introduced by Bailey and Johnston et al. [3]. They proposed to extract the features of interest for each connected component while performing the connected component labeling. The FPGA implementation of this algorithm resulted 1.2 clock cycles per pixel. Many variations of single-pass CCA architectures are also proposed in [4], [5], [6], [7] in an attempt to further optimize and save additional hardware resources.

To overcome the inefficient resource utilization and the low flexibility, we implemented a modified single-pass CCA. As a result, our proposed architecture with its low memory resource allocation, the feature-rich extraction and adaptability overcomes limits of the prior works.

III. Proposed Architecture

To keep the CCA module adaptive for any future requirements, the features of interest are not limited to certain operations. However, some of the required features are already known prior to the design of the CCA block. To balance both of these criteria, a single-pass CCA is applied to extract specific features, while setting up the CCA for future general use by storing the initially labeled image (initial-mask) in a dedicated storage unit (DRAM, BRAM or on-chip memory (OCM)).

For the implementation, DRAM is used to store the initial-mask and two dual port block RAM buffers are used to temporarily store label-connection table and data table. The main storages of the two tables are indexed by the labels they represent. Label-connection table holds the index of its connected label, while data table stores features.

There are three stages of CCA pipeline. These are explained below.

TABLE I: Comparison with the other hardware architectures.

Implementation	Technology [1]	Connectivity	Method	Extracted Feature	Image Size *[pixel]*	LUTs	FFs	BRAMs [bit]	f_{max}	Throughput[2] [MPixel/s]	Throughput[2] [cycle/pixel]
Ma et al. [4]	Virtex II	8	Single-pass	A, CC	640*480	1757	600	72k	40.64	32.5	1.25
Zhao et al. [5]	Virtex II	4	Single-pass	A, CG	256*256	4587	3154	234k	95.7	95.7	1
Appiah et al. [6]	Virtex 4	8	Two-pass	N/A	640*480	649	641	1142k	49.73	24.86	2
Klaiber et al. [7]	Kintex 7	8	Single-pass	BB	256*256	493	296	108k	185.59	148.47	1.25
This Work	Zynq 7020	8	Single-pass [3]	A, BB, CC, MI, S	256*256 640*480	414 452	546 608	72k 90k	124.92 124.22	123.56 122.87	1.011

A=Area, BB=Bounding Box, CG=Center of Gravity, CC=Component Count, MI=Maximum Intensity and S=Sum (counts the sum of gray values per region).

[1] Virtex II and 4 are configurable as 4-input LUTs, but Kintex 7 and Zynq 7020 utilize 6-input LUTs — [2] Worst case throughput — [3] Single raster-scan and also outputs initial-label

A. *Initial Labeling and Table Creation*

In the first stage, the input image is streamed into the FPGA. Each pixel is then compared to its 4 pre-processed neighbors and the pixel is given an appropriate label. At the same time, the label-connection table and data table are updated. The initial-mask of each pixel is then streamed out to the dedicated storage, while the created label-connection and data tables are further processed in the next stage.

B. *Table Unification*

In the second stage, the tables are unified independently. To achieve this, the unification algorithm sweeps through the label-connection table. The data table is updated whenever the label-connection table is modified. Two conditions exist for unification. For any label-connection table entry, if the entry is 0, then it means the label is a root label (root node), hence no table modification is necessary. Otherwise, if the entry is a non-zero index, the indexed label is followed until a root label found. When the root node is found, the entry of the initial label is modified to this root node in the label-connection table. Additionally, the data table entry of the root label is updated to include the data entry of the current label.

C. *Generation of Output Results*

There are three different means of output generation: default, unified-mask and custom generation. In our implementation, the data table includes five features: area, maximum intensity, bounding box, component count and sum. To extract these features from the processed image, the ***default*** method is used. If the input query requests a unified labeled-mask (***unified-mask***), the result is generated using the initial-mask and the two processing tables. For all other (new) feature extraction requests, i.e., for field-customization, a ***custom*** output generation is applied. With the later option, it is possible to compute new operations, by easily integrating an external input algorithm into the device.

IV. Hardware-Software Partitioning

To exploit the programmability of ARM-based processors and hardware acceleration of the fabric, we chose our target to be Zynq-7000 All Programmable SoC from Xilinx Inc. Several throughput-memory requirements were examined to choose the best software-hardware partitioning method. As a result, the first stage of CCA pipeline (initial labeling) is processed by the Programmable Logic (PL) side of Zynq. The table unification stage is operated by the ARM cores with direct access to the cache. The CPUs are also used for query request and output generation. Additionally, they provide means for post machine deployment flexibility where any feature extraction can be easily integrated.

V. Experimental Setup and Results

To analyze the performance and memory requirement, the proposed architecture is implemented in the ZedBoard ARM/FPGA SoC Development Board which contains Xilinx Zynq-7000 AP SoC XC7Z020-CLG484. The CPU executes bare-metal application and processes unification and output generation algorithms. The implementation is based on a gray-scale 16-bit pixel input image. To process a complete image frame, it takes one clock cycle per pixel for the first stage of CCA computation in the PL. Since the rest are pipelined, it gives a worst case throughput of 1.011 cycles per pixel. Table I shows the comparison of proposed architecture with the state-of-the-art implementations. For a fair comparison, a binary image is used with worst case number of connected components. Furthermore, the proposed architecture simultaneously extracts five features which impacts the resource consumption and throughput directly. However, even with such conditions the proposed architecture saves more resources compared to the most memory-efficient FPGA implementation [7].

VI. Conclusion

Attributing to Industry 4.0, the demand for highly flexible inspection systems is increasing. The connected component analysis design, as part of an industrial project, has a requirement of highly flexible, field-customizable system with high throughput and low power. The proposed hybrid software-hardware architecture is implemented on Zynq-7000 All Programmable System on Chip (SoC) and fulfills all the requirements. Moreover, compared to the state-of-the-art, it saves more than 30% memory space although it is processing more feature extractions.

References

[1] M. Jablonski and M. Gorgon, "Handel-c implementation of classical component labelling algorithm," in *Digital System Design, 2004. DSD 2004. Euromicro Symposium on*. IEEE, 2004, pp. 387–393.

[2] K. Wu, E. Otoo, and K. Suzuki, "Optimizing two-pass connected-component labeling algorithms," *Pattern Analysis and Applications*, vol. 12, no. 2, pp. 117–135, 2009.

[3] D. G. Bailey and C. T. Johnston, "Single pass connected components analysis," in *Proceedings of image and vision computing New Zealand*, 2007, pp. 282–287.

[4] N. Ma, D. G. Bailey, and C. T. Johnston, "Optimised single pass connected components analysis," in *ICECE Technology, 2008. FPT 2008. International Conference on*. IEEE, 2008, pp. 185–192.

[5] F. Zhao, H. zhang Lu, and Z. yong Zhang, "Real-time single-pass connected components analysis algorithm," *EURASIP Journal on Image and Video Processing*, vol. 2013, no. 1, p. 1, 2013.

[6] K. Appiah, A. Hunter, P. Dickinson, and J. Owens, "A run-length based connected component algorithm for fpga implementation," in *ICECE Technology, 2008. FPT 2008. International Conference on*. IEEE, 2008, pp. 177–184.

[7] M. J. Klaiber, D. G. Bailey, Y. O. Baroud, and S. Simon, "A resource-efficient hardware architecture for connected components analysis," 2015.