Stream Processing of Scientific Big Data on Heterogeneous Platforms with Image Analytics on Big Data

Ms. Shruti B Karki, Mr. Auntin Jose. T
PG Scholar, Associate professor
Department of Computer Science, RRCE, VTU, Bengaluru, INDIA
, Department of Computer Science, RRCE, VTU, Bengaluru, INDIA
shruthikarki@gmail.com, auntin123@yahoo.com

ABSTRACT-Image analytics with high performance is a challenge for the processing of big data, as the image data and the video data is a huge amount of big data. This paper will be presented with a case study for image analytics called the parallel connected component labeling(CCL). Generally this is the first step in image analytics. The high performance CCL implementation can be obtained on the heterogeneous platforms, if the suitable parts of the algorithm are processing on a fine grain parallel field programmable gate array (FPGA) along with the multi core processor. The implementation is suitable for the processing of big image and video data in motion which results in the reduction of the amount of memory that is required by the hardware architecture for different image sizes.

Keywords-- Component labeling, Data in motion, Feature extraction, Heterogeneous platform, Image analytic.

I. INTRODUCTION

One of the big data challenges are image analytic tools applicable to photos and surveillance videos. Statistics indicate that 2.5 quintillion bytes are generated every day. Big data is defined based on its main characteristics which are growing in three dimensions: volume, velocity and variety. In this concept, the unstructured data‘s volume is in the scale of petabytes, and creation of them will in the fraction of the second. Big data in motion is defined as continues data streams at high data transfer rate in the literature. Such kind of big data represent the data sets that cannot be analyzed with conventional algorithms and standard hardware platforms. The processing of the continuously increasing amount of data is done online and locally on the streamed data due to the typical memory capacity and bandwidth limitations, which determine the overall throughput. The new scale of volume, velocity and variety requires the redesigning a number of infrastructure components along with algorithms to support the large-volume, complex and growing data.

Combination of field-programmable gate arrays (FPGAs) and general-purpose computing on graphic processing units (GPGPUs) enable the processing of large scale problems in the field of genomics, graph analytics, social network analytics, bioinformatics etc. This was not possible before, see [5], [6]. For instance, high-performance hybrid core system proposed by Convey Computer which pairs Intel processors with a coprocessor of FPGAs is able to execute data-intensive problems much more effectively [7]. IBM also established Netezza- an advance high performance data analytical tool that has led to an exponential growth in the field of big data analytics. Netezza is based on the IBM blade architecture that uses FPGAs for filtering the input data before data is being processed.

Scientific data are of 4 kinds: raw data, structured data, published data and data linked to the publication [5]. The raw data is generated from observation and experiment of different phenomena. Biological, life sciences and climate generate huge amount of scientific data [5,6]. Usually Images and videos will have the highest amount of volume in scientific data which are analytically prepared to acquire additional value [8], [9]. The preparation is done by extracting different properties of the image such as objects and movements.

Segmentation is the first step in analytic image processing for many video-based applications which is followed by connected-component labeling [10]. Connected component labeling carries the task of labeling all connected image pixels in a binarized image to identify objects or to extract required features of a particular object. The throughput of the CCL can strongly influence the performance of the whole image processing system as it is one of the first complex processing steps in image processing applications. Because of this purpose, a parallel CCL algorithm having memory-efficient architecture is proposed which is suited for high performance image processing applications such as image analytics. The single pass CCL algorithm is memory efficient and therefore mainly suited for FPGAs. It is possible to achieve a high processing throughput by using this proposed architecture and algorithm for performing connected component labeling of streamed images without the necessity of buffering a full image, which will cause a performance limitation either due to limited FPGA-internal memory or due to limited FPGA-external memory bandwidth.

II. RELATED WORK

Connected component labeling (CCL) is the first step in image analytics algorithms. Recently more sophisticated single pass algorithms have gained interest as an alternative to the classical two-pass CCL algorithm for reconfigurable computing. After two scans of the same image, the CCL algorithm will be completed and these have the sequential dependencies. This results in the requirement of high amount of storage and memory for storing full images. To store the full image and labels, a memory with the same size

as the original image is required. The labels are characterized by the place of the pixel within a particular image. If the pixel belongs to the background area of the image a special background label is used, otherwise its label is determined by the labels of adjacent pixels.

Baily et al. proposed FPGA based single pass architectures [12]. There are two drawbacks with the proposed architecture: Its worst case memory requirement is related to the height and width of the image being processed, and also due to the lack of parallelism in which one pixel is processed per clock cycle causing a performance bottleneck in stream processing. Ma et al. reduced the memory requirements [13] of the algorithm significantly described in [14] by reusing the labels. As of this result, the required worst case memory is proportional to the width of the processed image. Kumar et al. proposed a parallel architecture which enhances the single pass algorithm that is used in [15]. The main aim is to store the whole images in prior to processing. In order to have the performance speed-up, the image is divided into different slices, and different CCL units have to process them independently. After this, each of the line from the image slice is fetched sequentially at a time in a round robin manner, and each CCL unit sends a vector to a global FIFO memory which acts as a vector collector. The vector mainly indicates the regions of the processed image slice except the edges. In the next step, a coalescing unit (CU) determines whether two regions of adjacent slices are connected and belong to the same object by processing one or both border of the image slice.

It is necessary to connect the CCL units to the CU to perform the merging operation of neighboring image slices in the round-robin manner. To have a successful merging, each edge region must have a unique, distinguishable label in its image slice. To overcome the limitations of [13] a modified version was proposed [16] which reduces drastically the amount of memory needed for processing by using two kinds of labels, slice local label and slice global labels. In this paper, a CU was introduced that improve the real-time processing of CCL architecture by merging the results of image slices in a memory-efficient way.

III. VIDEO AND IMAGE ANALYTICS FRAMEWORK

Image analytics needs the processing of videos and images to transform pixel level information to object based information for the analysis of certain properties specific for the considered application domain. The domain includes science, industrial measurement techniques or life science applications. Videos are taken as sequence of images with a specific frame rate. The frame rate may reach up to several hundred or thousand frames per second or more for high-speed scientific applications, in such a way that the video data of a single image sensor are in the range of 1 to 10 Gigabytes per second or 0.1 to 1 Petabyte per day. To overcome with this big data in motion, a high performance reconfigurable computing framework is proposed in this section. The framework is composed of a high-speed input data stream and a heterogeneous platform based on high performance reconfigurable computing devices making use of field programmable gate arrays (FPGA) and multi-core CPUs to which the image processing functions are mapped jointly. As shown in Figure 1, the input data stream is connected via several high speed links to the FPGAs that are able to acquire and analyze images as well as videos with a very high frame rate in real-time. Therefore, the framework based on the heterogeneous platform is equipped with integrated image processing capabilities such as segmentation, component labeling and feature extraction.

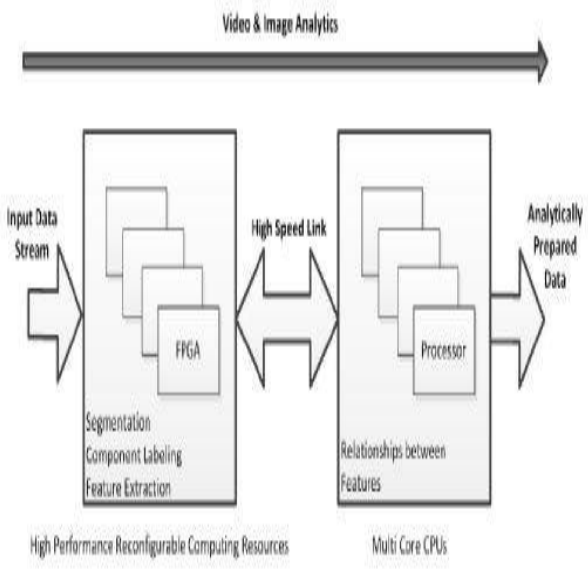

Figure 1. High performance reconfigurable computing framework for processing real-time image stream processing

The image processing steps, which enable a huge data reduction from GBytes of Pixel data to only KBytes of abstract object descriptors - so called feature vectors, are transferred to the multi-core CPU. This reduction allows the framework to output information on every object in every frame even in real-time for very high frame rates. The amount of data which has to be transferred from the FPGA is reduced by several orders of magnitude in this way. Image segmentation and feature extraction are the tasks for image processing architecture realized on the FPGA.

Segmentation is the process of separating the objects from the background when certain threshold is applied. By making use of this method all pixels having an intensity value over a certain threshold are considered as an object and are converted to black. And all pixels below this threshold are converted to white and considered as background. The binary image is generated from the original image accordingly. The major importance in this case is that the threshold value for separating the objects from the background. The method proposed in [17] is used for segmentation by generating a histogram of the captured grayscale image of a scientific application in the field of spray process, as seen in Figure 2. The threshold value is calculated by using the arithmetic mean value of the two

peaks detected in the histogram, which represents the objects and the background.

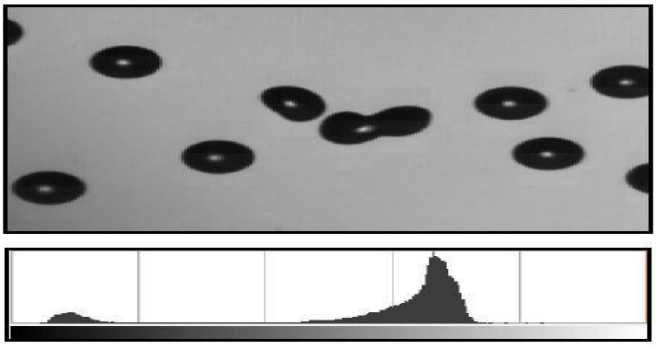

Figure 2. Grayscale raw image taken by image sensor and its histogram.

The next step is feature extraction based on connected component labeling. The main challenge is processing high image frame rate in real-time which requires high bandwidth in the range of 10 to 100 Gbit/s. To overcome this problem, a highly optimized and sophisticated architecture is used. Because of the limited resources in the embedded systems, algorithms which are especially dedicated to FPGA architectures have been proposed [12,13, 14, 16].

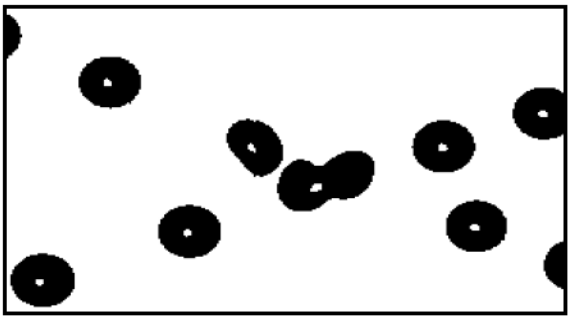

Figure 3. Segmentation of grayscale image with calculated threshold value.

Figure 4 shows the architecture of sequential processing [13] which consists of several components that can be described as follows: the neighborhood context block provides four registers A, B, C and D consisting of four previously processed pixels connected to the present pixel. A, B and C contain the label of the previous row and D contains the previous pixel label. For caching the new labels The row buffer block is used, as they are not saved in a temporary image. The key difference from [11] to [13] is that the labels are reused in every row, and cause the reduction in the need for memory.

As a result of mergers on the previous row, the merger table decides the equality. The translation table modifies a label allocated to the pervious row to the current new label. The label for the current pixel is selected in the label selection block based on the labels of its neighborhood.

The data merging unit records the features of each region by monitoring the labels of the pixels in the neighborhood context block. Each region will have one entry in the data merging unit indexed by the region's label. Whenever a region is updated, even its entry in the data merging unit is also updated.

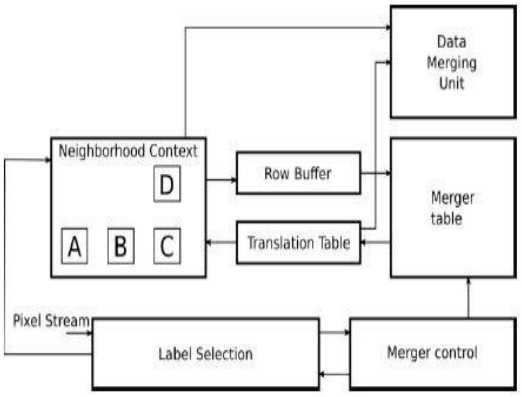

Figure 4. Connected Component LabelingArchitecture proposed in [13]

A bounding box is defined by A and B, two coordinates. Coordinate A indicates the upper left corner and coordinate B indicates the lower right corner. To extract the bounding box for each object the structure and the merging process within the data merging unit is proposed [4,6].

To provide bounding box extraction, the data merging unit has to be changed as shown in Figure 5. The modified block has two inputs a and b. One for providing the currently processed pixel‘s coordinates and the other one for giving information on the neighbor pixels label. The input data is read from the corresponding data table. To find the bounding box for two entries of the data table the following equations are used.

$$x_{1c} = x_{1a}, x_{1b}) \quad (1)$$

$$y_{1c} = y_{1a}, y_{1b}) \quad (2)$$

$$x_{2c} = \max(x_{2a}, y_{2b}) \quad (3)$$

$$y_{2c} = \max\ y_{2a}, y_{2b}) \quad (4)$$

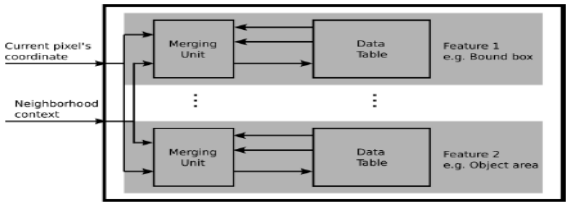

Figure 5. Data Merging Unit for extracting several features of the image objects simultaneously.

A parallelization approach is proposed in order to handle high data throughput. Therefore, the image is divided into different slices, and each slice is processed separately in parallel. In each slices the objects are identified and clubbed by a central unit called the coalescing unit. As a result different pixels are processed simultaneously and speedup based on the number of image slices can be obtained.

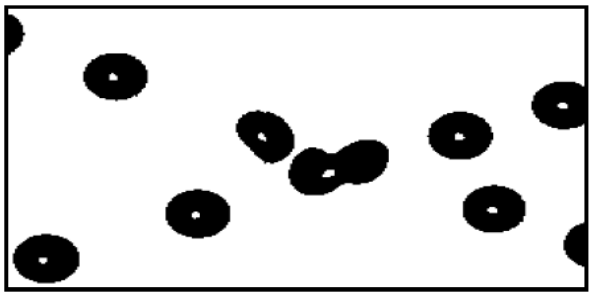

Figure 6. Binarized image after thresholding.

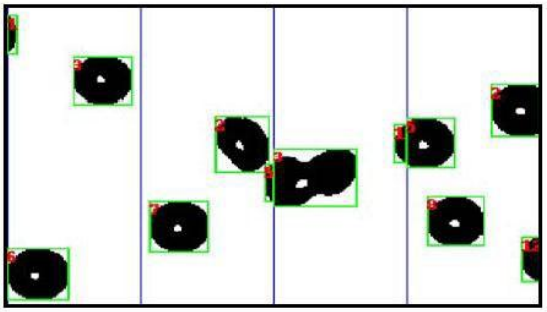

Figure 7. Extracted object features for all sub-images.

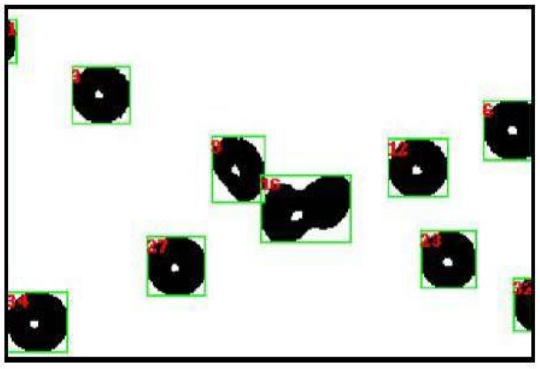

Figure 8. Merged object features for input image.

Figure 6 to Figure 8 show the processing steps for feature extraction. Binary image after the segmentation is shown in Figure 6. In the next step, the image is divided into many image slices for processing in CCL processing units that work in parallel. The bounding box is extracted by each CCL proceeding unit, as it is shown in Figure 7. The bounding boxes touching the slice borders will be merged together to have the correct result which is the last step. This step is depicted in Figure 8.

IV. EXPERIMENTAL RESULTS

The experimental data to examine the potential of heterogeneous systems including reconfigurable logic devices and general purpose processors (GPP) in the context of an image processing system for big data analytics is given. For this examination the approaches from [14], [11] and [16] are evaluated on both reconfigurable logic devices and in software on a general purpose processor. The approach in [16] gains a speedup from dividing the image in different vertical slices using the architecture from [14] as slice processors and merging the results using a coalescing unit. The results for the implementation on a single FPGA where parallelism is inherently used and in software for different image sizes on a single core are given in Table 1.

For the implementation of an array of slice processing units (SPU) having up to 100 SPUS on a single FPGA was realized, in which all are processing an individual image parallely. For the software implementation the classical two-pass algorithm [11] is used to process one image on a single core of a general purpose processor (GPP).

TABLE I. CCL-BENCHMARK CPU VS. FPGA

		Hardware Platform				
Image size		CPU		FPGA		Result
Width	Height	Gigapixels/s	Cores #	Gigapixels/s	Area %	Speedu
128	128	0,063011	1	15,3	8	242
256	256	0,06277	1	11,3	43	182
512	512	0,06286	1	12,3	81	198
1024	512	0,06189	1	9,1	93	146
2048	1024	0,06062	1	6,9	83	114

When comparing the throughput of the software implementation on a GPP and the dedicated architecture on an FPGA, the bandwidth of the FPGA architecture for CCL is one to two orders of magnitude higher. The hardware architecture is highly optimized to the FPGA structure, while for the presented software solution further research has to be done to make a proper comparison, but no decrease in speedup by less than one order of magnitude can be expected. This allows the FPGA architecture to process an image stream consisting of several different image slices simultaneously in real-time. For the case of processing a single image, only one slice processing unit of the SPU array can be used. Still the FPGA architecture accelerates the processing by approximately a factor of 2 compared to a single GPP core. By using the parallelization scheme from [16], several slice processing units can be used to process a single image in parallel enabling a higher throughput. Figure 9 through Figure 11 show the results for this approach where the coalescing unit is realized for FPGAs. Depending on the image size up to 4.5 GPixels/s can be processed. To study the performance of the coalescing process on a GPP, a prototype implementation of a software version of the coalescing unit was realized. The result for the throughput given in Table 2 can be achieved for the case that the feature vectors provided by the slice processing units are already

stored in the systems RAM and can be accessed at full memory bandwidth.

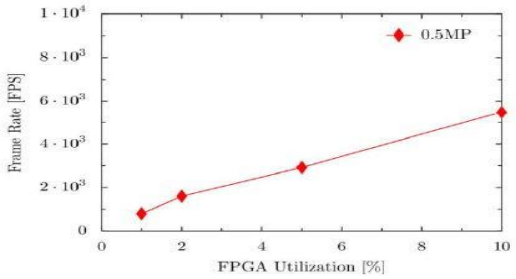

Figure 9. Achievable frame rate for an image with 0.5 Megapixels.

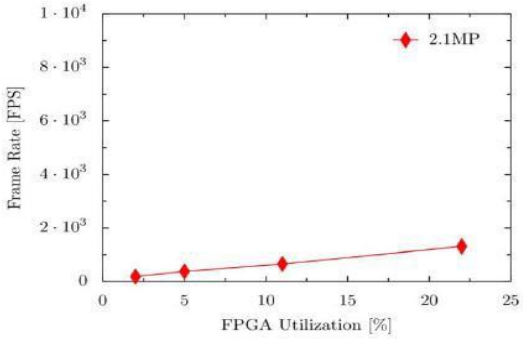

Figure 10. Achievable frame rate for an image with 2.1 Megapixels.

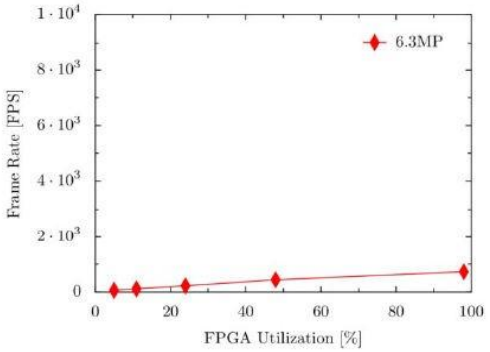

Figure 11. Achievable frame rate of maximum 800 fps for an image with 6.3 MegaPixels and a throughput of 5 Gigapixels per second.

TABLE II. SOFTWARE COALESCING:

Image size / # of slices	*Throughput (Gigapixels/s)*
256 × 256 / 4	6,62
1024 × 1024 / 8	10,51
2048 × 1024 / 4	13,40

THROUGHPUT IN GIGAPIXEL/SEC

Under these conditions the coalescing unit on the GPP achieves up to 13 GPixels/s, which is in the same order of magnitude as the FPGA implementation. In general, the irregular data structures and the sequential algorithm of the coalescing process are better suited for a GPP software implementation. Considering the performance results of both slice processing units and the coalescing unit on a heterogeneous platform, the conclusion is that the slice processing units should be implemented on a FPGAs and the coalescing unit should be implemented on a GPP in order to achieve maximum performance for big data image analytics. Achieving a throughput of 5 Gigapixel/s is equivalent to 0.4 Petabytes per day, so the implementation on a heterogeneous platform with a single mid-sized FPGA can be scaled up linearly with a several FPGAs and processor cores under the condition that several different images are processed in parallel. Beyond these performance considerations, mapping the irregular data structure and the sequential part of the algorithm, which is the coalescing algorithm, to the GPP is of advantage because of its higher processing frequency. For the regular data structures used in the slice processing units carrying out the parallel part of the algorithm, the FPGA achieves a higher throughput due to its parallel architecture.

V. CONCLUSION

Images and videos have the highest amount of volume among big data and therefore high performance image processing plays a very important role for image analytics. In this paper, we have investigated a parallel component labeling algorithm including feature extraction with a broad set of features as an important part of an image analytics framework. It is shown here that the performance of the proposed parallel component labeling algorithm with generalized feature extraction is accelerated and optimized if it is mapped to and executed on a heterogeneous hardware platform based on a fine-grained field-programmable gate array and a multi-core processor. For the parallelized connected component labeling (CCL) algorithm of this paper, the required memory compared to the requirement of algorithms known in the literature is reduced significantly on a heterogeneous hardware platform for typical image sizes even when compared to similar sliced parallel single pass CCL algorithms and architectures such that memory size nor memory bandwidth of the hardware platform has an impact on the performance. The basic structure of the algorithm and architecture is a set of parallel CCL units generating feature data of image slices being coalesced in a separate and subsequent coalescing unit. In order to achieve highest performance, it was shown that the parallel CCL units should be implemented on fine-grained FPGAs and the coalescing unit should be implemented in software on a multi-core processor. With the achieved throughput of 5 Gigapixels per second or 0.4 Petabytes per day the implementation on such a heterogeneous platform with a single mid-sized FPGA can be scaled up linearly with a plurality of FPGAs. Thus, the concept has been proven to be ideally suited for high performance image analytics for big data in motion.

REFERENCES

[1] U. Chandrasekhar, A. Reddy, and R. Rath, –A comparative study of enterprise and open source big data analytical tools,‖ *Conference on Information*

Communication Technologies (ICT), 2013 IEEE, pp.372–377.

[2] T. Eaton, D Deroos, ―Understanding big data,‖ in McGraw-Hill Companies, April 2012.

[3] C. P. Arkady B. Zaslavsky and D. Georgakopoulos, ―Sensing as aservice and big data,‖ *in Proceedings of the International Conference on Advances in Cloud Computing (ACC)*, July 2012.

[4] Hirzel, M.; Andrade, H.; Gedik, B.; Jacques-Silva, G.; Khandekar, R.; Kumar, V.; Mendell, M.; Nasgaard, H.; Schneider, S.; Soule, R.; Wu, K.-L., "IBM Streams Processing Language: Analyzing Big Data in motion," *IBM Journal of Research and Development* , vol.57, no.3/4, pp.7:1,7:11, May-July 2013

[5] Y. Demchenko, Z. Zhao, P. Grosso, A. Wibisono, and C. de Laat,―Addressing big data challenges for scientific data infrastructure,‖ *4th International Conference on Cloud Computing Technology and Science (CloudCom)*, 2012 IEEE, pp. 614–617.

[6] S. H. Ketan Paranjape and B. Masson, ―Heterogeneous computing inthe cloud: Crunching big data and democratizing hpc access for the life sciences,‖ in Intel White Paper.

[7] C. Computer, ―Hybrid-core: the big data computing architecture,‖ in Convey White Paper.

[8] D. Garlasu, V. Sandulescu, I. Halcu, G. Neculoiu, O. Grigoriu, M. Marinescu, and V. Marinescu, ―A big data implementation based on grid computing,‖ *in Roedunet International Conference (RoEduNet), 2013 11th*, pp. 1–4.

[9] Oracle, ―Big data for the enterprise,‖ in white Paper, 2013.

[10] K. Appiah, A. Hunter, F. Meng, and P. Dickinson, ―Accelerated hardware video object segmentation: From foreground detection to connected components labelling,‖ *in Computer vision and image understanding*, 2013, pp. 1282–1291.

[11] A. Rosenfeld and J. L. Pfaltz, ―Sequential operations in digital picture processing,‖ J.ACM, vol. 13, pp. 471–494, October 1966.

[12] D. Bailey and C. Johnston, ―Single pass connected components analysis,‖ *in Proceedings of Image and Vision Computing New Zealand* 2007, dec. 2007, pp. 282–287.

[13] N. Ma, D. Bailey, and C. Johnston, ―Optimised single pass connected components analysis,‖ *in ICECE Technology, 2008. FPT 2008. International Conference on*, dec. 2008, pp. 185 –192.

[14] D. Bailey, C. Johnston, and N. Ma, ―Connected components analysis of streamed images,‖ *International Conference on Field Programmable Logic and Applications*, sept. 2008, pp. 679 –682.

[15] V. Kumar, K. Irick, A. Al Maashri, and N. Vijaykrishnan, ―A scalable bandwidth aware architecture for connected component labeling,‖ *in VLSI ISVLSI), IEEE Computer Society Annual Symposium on*, july 2010, pp. 116 –121.

[16] M. Klaiber; L. Rockstroh; Z. Wang; Y. Baroud , S. Simon "A Memory-Efficient Parallel Single Pass Architecture for Connected Component Labeling of Streamed Images" *International Conference on Field-Programmable Technology (FPT)*, Seoul, Korea, Dec.2012.

[17] R. C. Gonzalez and R. E. Woods. Digital Image Processing (3rd Edition). Prentice Hall, August 2007, pp.741-76