Kincses Z, Vörösházi Zs, Nagy Z, Szolgay P, Laviniu T, Gacsádi A Investigation of area and speed trade-offs in FPGA implementation of an image correlation algorithm In: Corinto F, Pazienza GE

DOI: 10.1109/CNNA.2012.6331455

13th International Workshop on Cellular Nanoscale Networks and their Applications (CNNA), Turin, Italy Aug. 29-31, 2012: CNNA 2012. Konferencia helye, ideje: Torino, Olaszország, 2012.08.29-2012.08.31. Los Alamitos: IEEE, 2012. pp. 1-5.

Investigation of area and speed trade-offs in FPGA implementation of an image correlation algorithm

Z. Kincses[1], Zs.Vörösházi[2],
[1]Institute of Informatics, University of Szeged
Szeged, H-6701
kincsesz@inf.u-szeged.hu
[2]Dept. of Electrical Engineering, University of Pannonia
Veszprém, H-8200
voroshazi@virt.uni-pannon.hu

Z. Nagy[3], P. Szolgay[3], S. Gacsádi[4],
[3]Cellular Sensory and Optical Wave Computing Laboratory,
Hungarian Academy of Sciences
Budapest, H-1111
[4]Department of Electronics and Telecommunications,
University of Oradea, Romania
410087

***Abstract*—In this paper an image correlation algorithm is implemented on FPGA architecture for assisted movements of visually impaired persons or driving systems. Taking into account the limitations of FPGA devices and the special requirements of the correlation based image matching algorithm a semi-parallel implementation method is used. This provides an optimal trade-off between area and speed of the implemented algorithm. Several key issues are investigated and discussed related to the speed, area, and accuracy.**

I. INTRODUCTION

In general, determination of the correlation coefficient between two images requires high computing power which is proportional to the size of the template image (kernel). It is desirable that the template image size should be large enough to contain relevant information. For real-time computation of the correlation coefficients, the majority of the operations can be executed on a CNN algorithm as described in [1]. However, analog CNN VLSI implementations have relatively low precision (7-8 bits) and sensitive to the changes of temperature and supply voltage. For these reasons an FPGA-based implementation is chosen.

II. CORRELATION: MATHEMATICAL BACKGROUNDS

Similarity matching between two gray-scale images can be classified into feature-based and intensity based-methods. In case of the intensity based methods different metrics or procedures can be applied, such as: Euclidean distance, Sum of Absolute Differences (SAD), Mean Absolute Differences (MAD), Sum of Squared Differences (SSD) or Normalized Cross Correlation (NCC) [2].

The NCC in (1) can be easily derived from the SSD: maximizing the correlation is equivalent to minimizing the

sum of squared difference, which effectively gives more weight to large differences. Let us consider a test image $\Phi(m,n)$: $R^2 \rightarrow R$ with dimension $M \times N$.

$$CORR(i,j) = \frac{\sum_{p=1}^{P}\sum_{q=1}^{Q}\left[K(p,q)-\overline{K}\right]\cdot\left[\Lambda(p,q)-\overline{\Lambda}\right]}{\sqrt{\sum_{p=1}^{P}\sum_{q=1}^{Q}\left[K(p,q)-\overline{K}\right]^2}\cdot\sqrt{\sum_{p=1}^{P}\sum_{q=1}^{Q}\left[\Lambda(p,q)-\overline{\Lambda}\right]^2}} \quad (1)$$

where $K(p,q)$: $R^2 \rightarrow R$ denotes the template image (correlation kernel) with dimension $P \times Q$ ($p \in [1,P]$, $q \in [1,Q]$),

$\Lambda(p,q)$: $R^2 \rightarrow R$ represents the actual image region from test image $\Phi(m,n)$ compared to the kernel $K(p,q)$, with dimension $P \times Q$, ($p \in [1,P]$, $q \in [1,Q]$), respectively.

$\overline{K}$: is the mean intensity value of the template image

$\overline{\Lambda}$: is the mean intensity value of the test image.

The test image is scanned pixel-by-pixel and overlapped by a kernel as a sliding window to calculate the matching degree for each pixel. The matching degree between the template $K(p,q)$ and a region $\Lambda(p,q)$ from the test image is obtained by computing the correlation coefficient, which indicates how well the pattern matches the contents of that region (compared image). Equation (1) can be rewritten as follows:

$$CORR^2(i,j) = \frac{\left\{\overline{\left[K(p,q)-\overline{K}\right]\cdot\left[\Lambda(p,q)-\overline{\Lambda}\right]}\right\}^2}{\overline{\left[K(p,q)-\overline{K}\right]^2}\cdot\overline{\left[\Lambda(p,q)-\overline{\Lambda}\right]^2}} \quad (2)$$

Using the above transformation in (2), the square root operation can be eliminated by replacing with a multiplication of the two images, while the division can be replaced by a multiplication of reciprocal expression. In this case we must consider that high values of correlation coefficient will result the situation where the two images are completely anti-correlated.

III. IMPLEMENTATION

In this section the FPGA-based implementation of the correlation expression will be investigated according to (2). There are three different ways to implement the image correlation algorithm: a sequential, a semi-parallel and a massively/fully-parallel approaches.

In the first case, the correlation coefficients $CORR^2(i,j)$ are computed serially by one processing element. Although, it has the smallest area requirement, it provides the slowest solution. Therefore it cannot be used for real-time implementations. In case of fully-parallel solution, the correlation coefficients are computed by a massively parallel array processor. Hereby, this solution gives the highest computing performance, but its area requirement will be quadratically increasing depending on the size of the kernel image. Therefore, this solution can only be applied in such cases where the size of the kernel image is sufficiently small [3]. In the third case, the semi-parallel solution provides a good trade-off between the first two implementations. Several processing units arranged in a row can calculate the correlation coefficients parallel in a row-wise order. Using this latter method the area requirement of the architecture is increased proportionally to the template size, so relatively large kernel images (64×64, 128×128) can be handled in nearly real-time.

To implement the correlation formula in (2) the semi-parallel solution was chosen. The overall architecture of this solution is shown in Fig. 1. The four main components are the Memory Controller unit, the Kernel Memory unit, the Correlator Core unit and the Mean Calculator unit. The Memory Controller unit provides an interface for an external memory, and transfer data to and from the Mean Calculator and Correlator Core units. The Kernel Memory unit stores the $K(p,q)-\overline{K}$ and $\left[K(p,q)-\overline{K}\right]^2$ values which considered as constants.

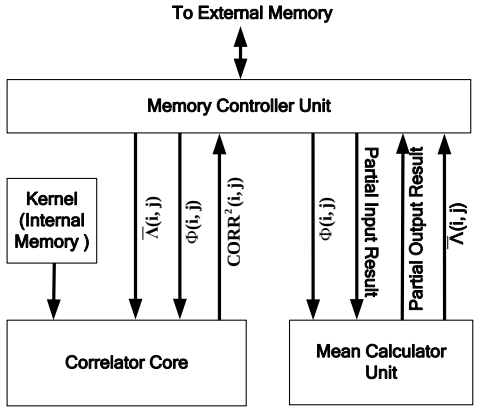

Figure 1. Architecture of the overall system

The Mean Calculator unit (Fig. 2) computes the mean intensity of the $(i, j)^{th}$ element compared image ($\overline{\Lambda}$), using the cumulative sum method. In this method the row-wise sum of the compared image is computed first then the sum of these results is calculated column-wise. Finally these values should be divided by $P \times Q$ to get the mean intensity of $\Lambda(p,q)$. Size of the template image is constant therefore the division can be done by the multiplication with the reciprocal of this value.

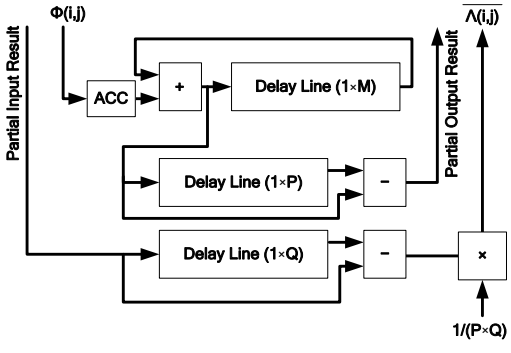

Figure 2. The Mean Calculator unit

The final correlation values $CORR^2(i,j)$ are calculated by the Correlator Core unit (Fig. 3). In the expression (2) the

numerator and denumerator are computed in parallel way by this unit. According to the size of the kernel image a new correlation value is computed in every P clock cycles.

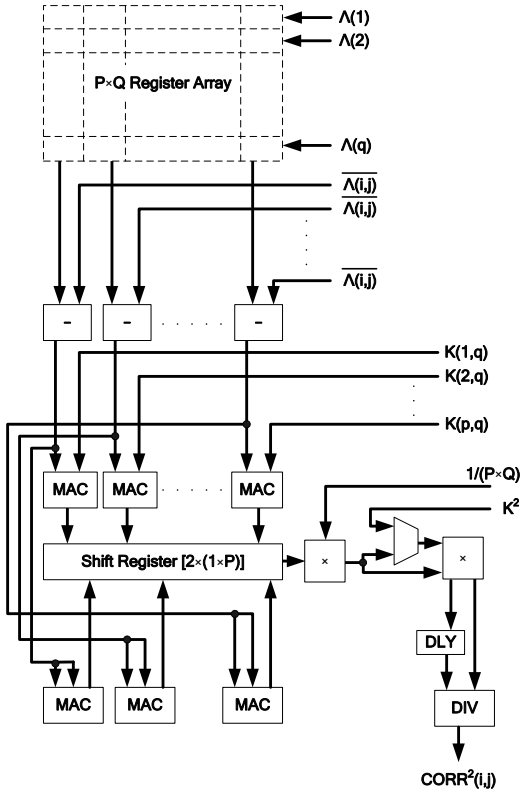

Figure 3. TheCorrelator Core Unit

The area requirement and also the performance of the proposed architecture were estimated. Supposing the size of the test image is 512×512, while the size of the kernel is changed from 16×16 to 256×256, the results are summarized on Table 1.

TABLE I. AREA AND SPEED ESTIMATION

	Kernel image size				
	16x16	***32x32***	***64x64***	***128x128***	***256x256***
Slice	576	1201	3116	9901	29811
BRAM	41	81	167	344	714
DSP (MAC)	66	130	386	770	1538
FPS	42	22	11	5	3

Our results are compared to an existing fully-parallel implementation [3]. The comparison is made with a test image having 256×256 pixel size and a 20×20 pixel-sized kernel. In our proposed semi-parallel implementation the required processing time is about 7.2 ms, while the architecture in [3] the processing time is only 0.163 ms. However, our solution consumes only 82 DSP MAC slices, while in case of fully-parallel implementation [3] 5-times more dedicated MAC blocks are required.

ACKNOWLEDGMENT

This research project supported by the János Bolyai Research Scholarship of the Hungarian Academy of Sciences and OTKA Grant No. K84267.

REFERENCES

[1] T. Laviniu, A. Gacsádi, I Gavriluţ V. Tiponuţ "A CNN Based Correlation Algorithm to Assist Visually Impaired Persons" Signals, Circuits and Systems (ISSCS), Iasi, Romania, 10th International Symposium on pp. 1–4, June 2011

[2] Donald G. Bailey "Design for Embedded Image Processing on FPGAs" 2011, Wiley-IEEE Press

[3] Almudena L., Luis E. "High performance FPGA-based image correlation" J. Real-Time Image Proc., Vol. 2, Special Issue, pp. 223-233, Springer, 2007