ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com

International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 5, Special Issue 8, March 2018

EFFICIENT IMPLEMENTATION OF OBJECT DETECTION USING TEMPLATE MATCHING IN FPGA

S.Hariprasath[1], S.Nithya Lakshmi[2], S.Padma[3], M.Priyanga[4], G.Purvaja[5]
Assistant Professor, Department of EEE, Saranathan College of Engineering, Trichy, India[2,3,4,]
UG Student, Department of EEE, Saranathan College of Engineering, Trichy, India[2,3,4,5]

Abstract: **In this paper, FPGA implementation of object detection using template matching algorithm is described. In the proposed methods, for the template matching algorithm, the matching section of the source image with reference to the template image makes use of Sum of Absolute Difference (SAD)Algorithm. The architecture is designed to process source image and the given templateimage with resolution of 16×16 and 4×4 pixels respectively. In this work two techniques namelyrolling-index architecture forSAD computation and an optimized binary tree adder are proposed. For every moving window, the difference between source image and templateis calculated in parallel processing mode to compute the matched pixels within two clock cycles. In order to achieve area reduction, the adder scheme is modified. This Proposed System is Implemented using Verilog HDL 2005 version. Simulation is carried out by using imulated by Modelsim 6.4ctool and Synthesized by Xilinx 9.2 tool. The proposed system is implemented Spartan3XC3S200 TQ144 FPGA. The proposed design is efficient in terms of hardware resources.**

Index Terms—**Area overhead, error detection, FPGA design, template matching, sum of absolute difference**

I. INTRODUCTION

Template matching is a renowned technique in digital image processing for finding small parts of an image that match with a given template image. It can be used as a part in manufacturing process as a tool for quality control, a way to navigate a mobile robot or as a method to detect edges in images. Template Matching is a method for searching and finding the location of a template image in a larger image. It simply moves the template image over the input image (as similarto 2 dimensionalconvolutions) and compares the template and patch of input image under the window. It returns a gray scale image, where each pixel denotes how much does the neighbourhood of that pixel match with template. If input image is of size (WxH) and template image is of size (wxh), output image will have a size of (W-w+1, H-h+1). The parameters (W,H)are taken as width and height of the rectangle window that slides over the source image. The chosen rectangle is the region of template. In this work, Sum of Absolute Difference (SAD) technique is chosen to detect the matched pixels, since the SAD technique is a unpretentious operation and can be simplified as logical XOR operations and additions. In this proposed work, the objective is to design and implement the template matching algorithm in FPGA such that the resultant design is area optimized and also equitably operates at high speed. Thus, two techniques namely rolling-index based SAD architecture and optimized binary tree adder are proposed in this work.

The rest of the paper is structured in the following manner. The survey of the existing systems is presented in section II. The proposed system is described in section III. The SAD algorithm is described in section IV. The results are discussed in section V. Finally the future enhancement of the proposed design is described in section VI.

II. EXISTING SYSTEMS

In Advanced method for high speed template matching targeting FPGAs [1] authored by P. Elvert, T. Tiemerding, C. Diederich, and S. Fatikow, novel approaches to high-speed template matching on FPGAs was described. The validation of these approaches has shown that the resulting tracking quality and feasibility is highly dependent on the relative size of the template in regard to the object to track. The results show tracking uncertainties between one single pixel for low and hundreds of pixels for high resolution videos.

In another paper [2] titled as A two stage template matching algorithm and its implementation on FPGA by the authors H. Aktaş, R. Sever, and B. U. Töreyin, in order to decrease the computational cost and number of cycles in template matching algorithm, a novel two-stage algorithm is proposed. The Sum of Absolute Differences method is used for matching. The proposed algorithm is implemented on Field-Programmable-Gate-Array (FPGA). The algorithm is accelerated with the effective usage of Block RAMs distributed on FPGA. Thus, the proposed algorithm became fast enough for real time object tracking applications on

ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com

International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 5, Special Issue 8, March 2018

UAVs. In the paper [3] titled Real-time FPGA-based template matching module for visual inspection application written by the authors J.-Y. Chen, et al , how template matching enables localization of objects under inspection, but suffers from long computation due to high computation complexity is described. In this work, a real-time FPGA-based template matching module which accelerates time consuming normalized cross-correlation (NCC) template matching was presented.

In the paper [4] titled as FPGA-based template matching using distance transform authored by S. Hazel, A. Kugel, R. Manner, and D. M. Gavrilaa high-performance FPGA solution to generic shape-based object detection in images is proposed. The underlying detection method involves representing the target object by binary templates containing positional and directional edge information. A particular scene image is pre-processed by edge segmentation, edge cleaning and distance transforms. Matching involves correlating the templates with the distance-transformed scene image and determining the locations where the mismatch is below a certain user-defined threshold. In this paper we present a step by step implementation of the components of such object detection systems, taking advantage of the data and logical parallelism opportunities offered by FPGA architecture. The realization of a pipelined calculation of the pre-processing and correlation on FPGA is presented in detail.

In the paper[5] titled as Using template matching for object recognition in infrared video sequences authored by the authors I. Pham, R. Jalovecky, and M. Polasek
the similarity criteria normalized cross correlation is described. The cross correlation is not invariant to changes in image intensity such as lighting conditions, and the range of correlation coefficient is dependent on the size of the feature, while we can normalize for the effect of changing intensity and template size by using normalized cross correlation. The basic principle of the algorithm is based on the assumption the object is selected at time t with the centre of mass T and the greatest relative velocity between the camera and the target vct is always known.

III.PROPOSED SYSTEM

Template matching is a technique used for identifying any object in the source image which is similar to the template image. Template matching needs two primary components: source image and template image; and three sequential processing stages: gray scaling, binarization, and SAD. Gray scaling processes is called preprocessing stage. And its stored in RAM Architecture.

First step in the proposed system is gray scaling. The input colour image is converted into gray scale image by standard conversion procedure and the pixels are read as a text file. Matlab tool is utilized for this task. The Pixel Values of Template Image and Main Images are stored in Memory. After this process is finished, the pixels are read and SAD process on the input pixels is performed to complete the matching process.SAD is a calculation that sums the difference of gray levelper pixel, between source and template image. Let I(x,y) is the source image and T(x,y) is the template image thenSAD procedure is described by (1). Here, x and y represent the position of pixel coordinates (x,y). The smaller SAD value indicates that there is a highercoincidence of similarity between the template and source image. The SAD Design is implemented as the Processing element (PE) of each pixel.

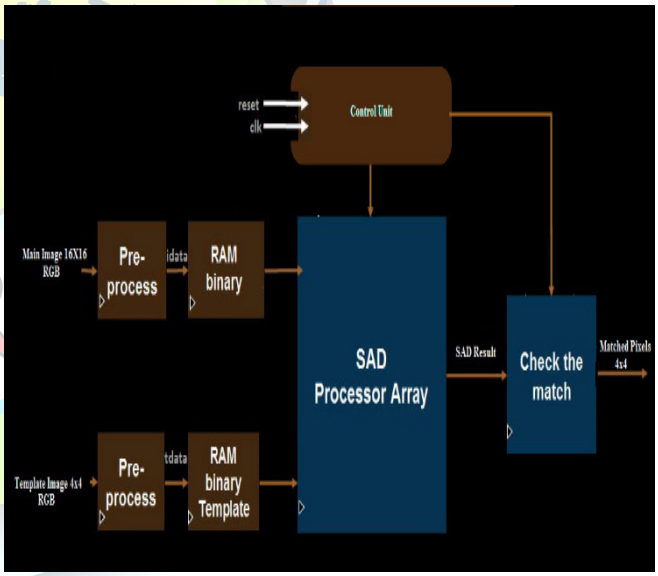

Fig 1: Proposed Template Matching System

The SAD is calculated to find the absolute difference between the present position of the image and template Images. If the pixel values are equal, then the SAD computation produces zero as the result. Based on repeating the aforementioned step continuously until the entire image is scanned by the moving window, the scanning processing is completed. Then by comparing the computed SAD values with the threshold, the best match of the template to the image patch is estimated. The comparator section gives the co-ordinate values of the matched Pixels.

ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com

International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 5, Special Issue 8, March 2018

The co-ordinates are given as input to the binarizer module which produces the binary image. The binarizer output values are stored in the memory. After all the pixels are scanned by the moving window, the values are exported to Matlab tool to view the matched result. The complete flow the proposed work is depicted in figure 2.

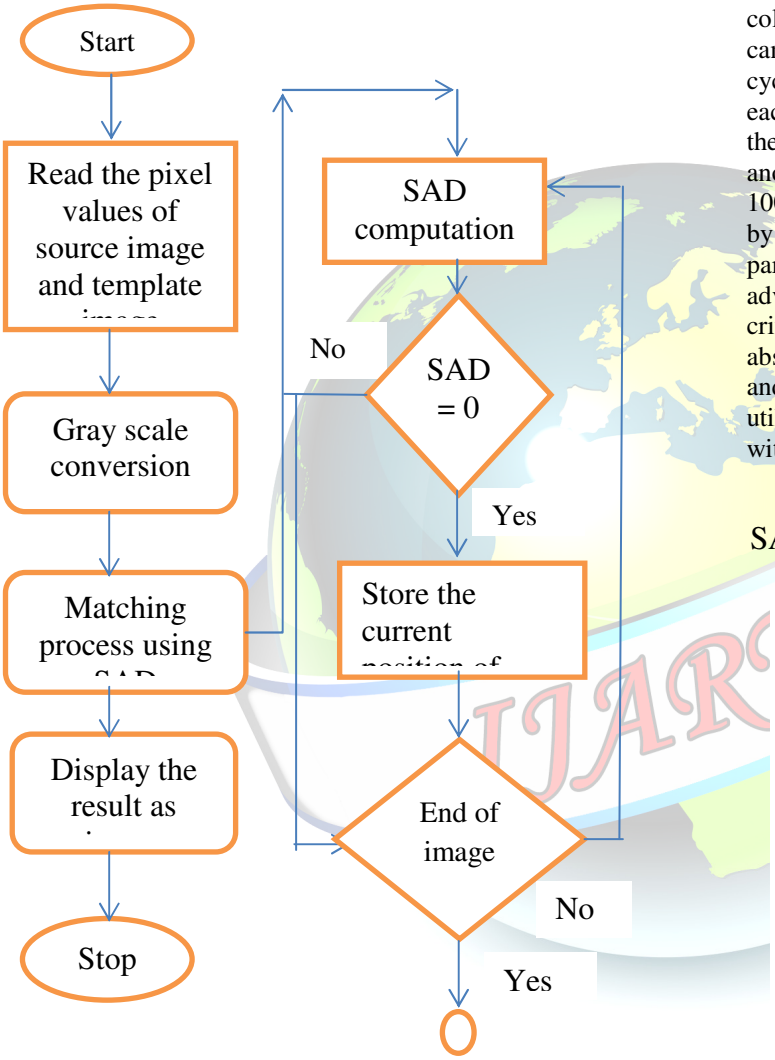

Figure 2 flowchart of the proposed method

IV. PROPOSED SAD ALGORITHM

In this work, a 2-D intra-level architecture called the Propagate Partial SAD is proposed. The architecture is composed of PE arrays with a 1-D adder tree in the vertical direction. Current pixels are stored in each PE, and two sets of continuous reference pixels in a row are broadcasted to PE arrays at the same time. In each PE array with a 1-D adder tree, distortions are computed and summed by a 1-D adder tree to generate one-row SAD. The row SADs are accumulated and propagated with propagation registers in the vertical direction The reference data of searching candidates in the even and odd columns are inputted by Ref. Pixels 0 and Ref. Pixels 1, respectively. After initial cycles, the SAD of the first searching candidate in the zero th column is generated, and the SADs of the other searching candidates are sequentially generated in the following cycles. When computing the last searching candidates in each column, the reference data of searching candidates in the next columns begin to be applied as input values through another reference input. Then, the hardware utilization is 100% except the initial latency. In Propagate Partial SAD, by broadcasting reference pixel rows and propagating partial-row SADs in the vertical direction, it provides the advantages of fewer reference pixel registers and a shorter critical path. The specific PE in Fig. 3 Estimates the absolute difference between the Cur_pixel of the search area and the Ref_pixel of the current macroblock. Thus, by utilizing PEs, SAD shown in as follows, in a macroblock with NxN size of can be evaluated:

$$SAD=\sum_{i=0}^{N-1}\sum_{j=0}^{N-1} x_{ij} - y_{ij} \quad \ldots\ldots (1)$$

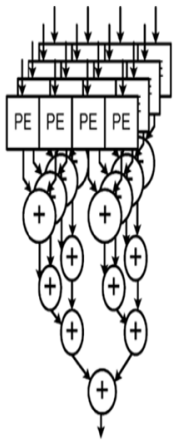

Figure 3. SAD Tree Architecture

The pixel example is shown in figure 4.

ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com

International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 5, Special Issue 8, March 2018

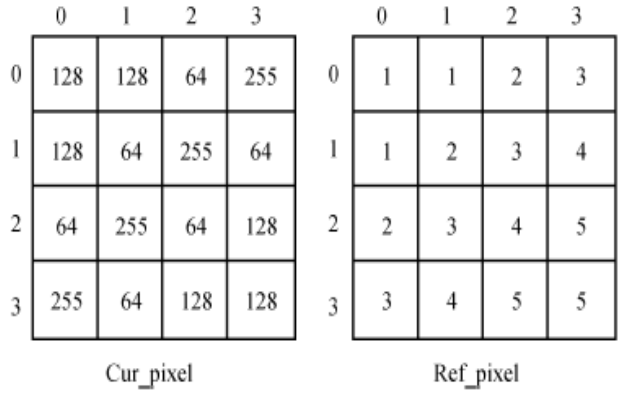

Figure 4. Pixel example for SAD computation

The concept of the proposed SAD Tree architecture. The proposed SAD Tree is a 2-D intra-level architecture and consists of a 2-D PE array and one 2-D adder tree with propagation registers Current pixels are stored in each PE, and reference pixels are stored in propagation registers for data reuse. In each cycle, current and reference pixels are inputted to PEs. Simultaneously, continuous reference pixels in a row are inputted into propagation registers to update reference pixels. In propagation registers, reference pixels are propagated in the vertical direction row by row. In SAD Tree architecture, all distortions of a searching candidate are generated in the same cycle, and by an adder tree, distortions are accumulated to derive the SAD in one cycle. In order to provide a high utilization and data reuse, the snake scan is adopted and reconfigurable data path propagation registers are developed in the proposed SAD Tree, which consists of five basic steps from A to E. The first step, A, fetches pixels in a row and the shift direction of propagation registers is downward. When calculating the last candidates in a column, one extra reference pixel is required to be inputted, that is, step B. When finishing the computation of one column, the

When finishing the computation of one column, the reference pixels in the propagation registers are shifted left in step C. Because the reference data have already been stored in the propagation registers, the SAD can be directly calculated. The next two steps, D and E, are the same as steps A and B except that the shift direction is upward. After finishing the computation of one column in the search range, we execute step C and then go back to step A.

V. RESULT AND DISCUSSION

PROPOSED MULTIPLIER RESULTS

Device utilization summary of the proposed system is given in table 1.

Table 1. Device utilization summary

Device Utilization Summary				
Logic Utilization	**Used**	**Available**	**Utilization**	**Note(s)**
Number of 4 input LUTs	594	55,296	1%	
Logic Distribution				
Number of occupied Slices	353	27,648	1%	
Number of Slices containing only related logic	353	353	100%	
Number of Slices containing unrelated logic	0	353	0%	
Total Number of 4 input LUTs	594	55,296	1%	
Number of bonded IOBs	270	633	42%	
IOB Flip Flops	128			
Number of GCLKs	1	8	12%	
Total equivalent gate count for design	5,407			
Additional JTAG gate count for IOBs	12,960			

SYNTHESIS REPORT:

Timing constraint: Default OFFSET OUT AFTER for Clock 'clk'
Total number of paths / destination ports: 827024 / 12

-
Offset: 33.721ns (Levels of Logic = 18)
Source: PE10/C_Registered_0 (FF)
Destination: SAD<11> (PAD)
Source Clock: clk rising

Data Path: PE10/C_Registered_0 to SAD<11>
Gate Net
Cell:in->out fanout Delay Delay Logical Name (Net Name)

FDRE:C->Q 3 0.720 1.246 PE10/C_Registered_0 (PE10/C_Registered_0)
LUT2:I0->O 7 0.551 0.000 PE10/Madd_Nine_Bit_Sum_lut<0> (PE10/Nine_Bit_Sum<0>)
MUXCY:S->O 1 0.500 0.000 PE10/Madd_Nine_Bit_Sum_cy<0> (PE10/Madd_Nine_Bit_Sum_cy<0>)

ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com

International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 5, Special Issue 8, March 2018

XORCY:CI->O 4 0.904 1.256
PE10/Madd_Nine_Bit_Sum_xor<1>
(PE10/Nine_Bit_Sum<1>)
LUT4:I0->O 2 0.551 1.216
PE10/Madd_AD_xor<1>11 (AD10<1>)
LUT4:I0->O 2 0.551 1.072
instance_name/V5/FA2/Cout1 (instance_name/V5/C<1>)
LUT3:I1->O 2 0.551 1.072
instance_name/V5/FA3/Cout1 (instance_name/V5/C<2>)
LUT3:I1->O 2 0.551 1.072
instance_name/V5/FA4/Cout1 (instance_name/V5/C<3>)
LUT3:I1->O 2 0.551 1.072
instance_name/V5/FA5/Cout1 (instance_name/V5/C<4>)
LUT3:I1->O 2 0.551 1.072
instance_name/V5/FA6/Cout1 (instance_name/V5/C<5>)
LUT3:I1->O 2 0.551 0.945
instance_name/V5/FA7/Cout1 (instance_name/V5/C<6>)
LUT3:I2->O 2 0.551 1.216
instance_name/V5/FA8/Mxor_S_xo<1>1
(instance_name/A3<7>)

RTL synthesis outputs are shown in figure 5,6,7 respectively.

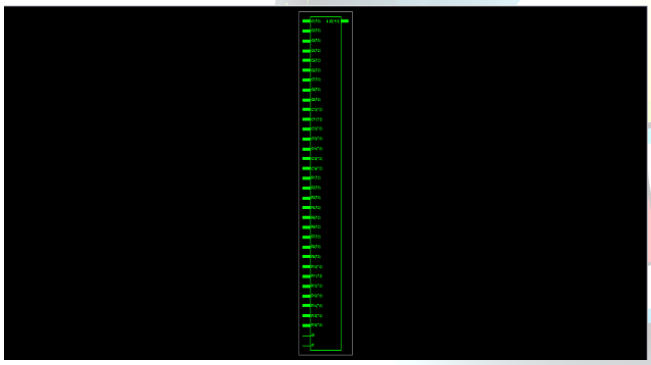

Figure 5. Main view

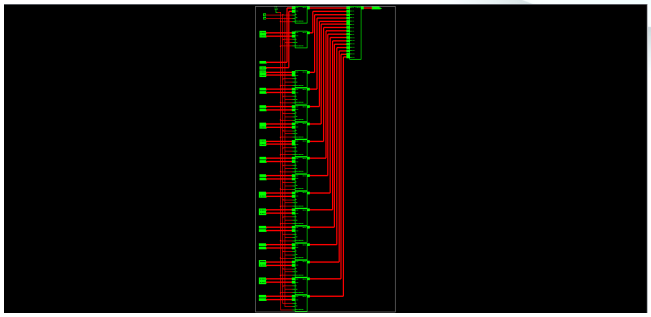

Figure 6. Inner view

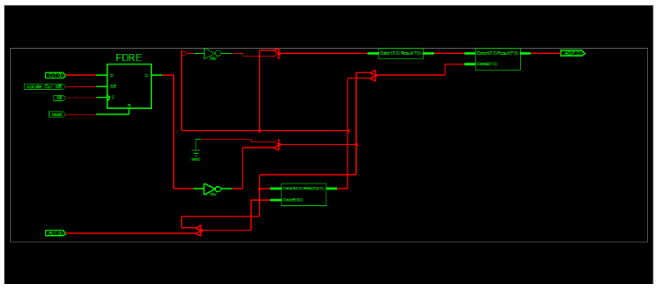

Figure 7. PE view

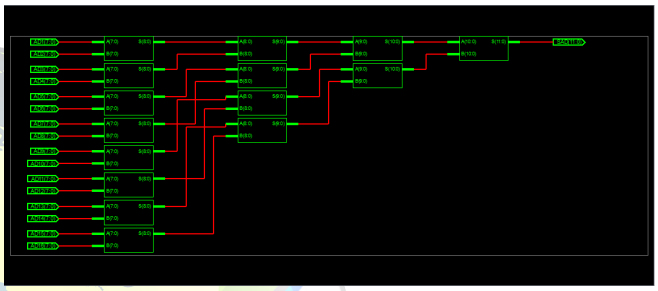

Figure 8. SAD adder Tree

VI. CONCLUSION

The proposed PE array for SAD processor occupies only 12% of the Spartan 3E chip. The delay achieved is 30.721ns. The designed system is operating at fairly high speed for real time applications. By incorporating speed optimization techniques such as pipelining mechanism, the speed can be further increased.

REFERENCES

[1] P. Elvert, T. Tiemerding, C. Diederich, and S. Fatikow, "Advanced method for high speed template matching targeting FPGAs," in Proc. Of Int. Symp. on Ophotomechatronics Technologies, pp. 33-37, November2014.
[2] H. Aktaş, R. Sever, and B. U. Töreyin, "A two stage template matching algorithm and its implementation on FPGA," in Proc. of 2015, 23rd Signal Processing and Communications Applications Conf., pp. 2214-2217, May 2015.
[3] J.-Y. Chen, et al, "Real-time FPGA-based template matching module for visual inspection application," in Proc. of 2012 IEEE/ASME Int. Conf. on Advanced Intelligent Mechatronics, pp. 1072-1076, July 2012.
[4] I. Pham, R. Jalovecky, and M. Polasek, "Using template matching for object recognition in infrared video

ISSN 2394-3777 (Print)
ISSN 2394-3785 (Online)
Available online at www.ijartet.com

International Journal of Advanced Research Trends in Engineering and Technology (IJARTET)
Vol. 5, Special Issue 8, March 2018

sequences," in Proc. Of 2015 IEEE/AIAA 34th Digital Avionics Systems Conf., pp. 8C5-1–8C5-9, September 2015.

[5] S. Hazel, A. Kugel, R. Manner, and D. M. Gavrila, "FPGA-based template matching using distance transform," in Proc. of 10th Annual IEEE Symp. on Field-Programmable Custom Computing Machines, pp. 89-97, April 2002.

[6] T. Adiono, M. D. Adhinata, N. Prihatiningrum, R. Disastra, R. V. W. Putra, and A. H. Salman, ''An architecture design of SAD based template matching for fast queue counter in FPGA,'' in Proc. of The 2016 Int. Symp. on Intelligent Signal Processing and Communication Systems, pp. 1-4, October 2016. (accepted)