A Hardware Implementation to Compute Displacement of Moving Object in a Real Time Video

Kota Solomon Raju
Digital Systems Group
CSIR-CEERI, Pilani
Rajasthan, India

Dorothi Borgohain
Dept of ECE
Tezpur University
Assam, India

Manoj Pandey
Digital Systems Group
CSIR-CEERI, Pilani
Rajasthan, India

ABSTRACT

This paper presents the implementation of FPGA based real time object tracking system that is targeted to Spartan-6 IVK board. The presented work combines both hardware and software flow where the complex parts of the algorithm such as computing displacement is designed to HW modules. Other part of the algorithm such as video data fetching, window based weight calculation is targeted to software. In our work, we are using standard video format having resolution of 1280x720. The main idea of this work is to accelerate the performance of the object detection system by partitioning the algorithm between dedicated hardware IP core and software.

Keywords

Xilinx, EDK, Bhattacharya-Coefficient, real time object tracking.

1. INTRODUCTION

Video processing has become an important application domain of smart embedded system. Video processing is the key part of video surveillance system, security system, automatic traffic control and many more. For each of the area, object tracking is fundamental part of the system. Object tracking is used for identifying the trajectory of moving object in continuous video frame sequence. Real-time processing of the captured video data is necessary in this application. In other cases it is used to increase the quality and reduce the volume of information to be transmitted and stored. Moreover, real-time video processing is computationally demanding but often highly parallelizable, making it amenable to hardware implementations.

By using new features of platform FPGAs we propose a method using a video process controller to achieve real time object detection task especially for designs that uses the benefit of hardware software co-design where some part of algorithm has been implemented on dedicated hardware block and some on software.

As the complete system is being targeted on FPGA, the hardware software co-design concept may be useful where the complex computation part can be implemented as parallel running block. In proposed idea, displacement computation block has been chosen to implement on hardware as this block is increasing computation overhead. This block can be easily implemented on hardware with basic mathematical blocks like multiplier, divider and adder of floating point. These basic components make algorithm to be implemented on FPGA easily.

This paper is organized as follows. Some of the previous works are mentioned in Section 2. Section 3 presents brief description of system architecture. Algorithm analysis has been presented in Section 4. System analysis of the design has been discussed in Section 5. Experimental Setup has been presented in Section 6. The results and conclusion with future work have been presented in Section 7 and 8 respectively.

2. PREVIOUS WORK

D. Comaniciu et al. [3] proposed a new approach toward target representation and localization, the central component in visual tracking of non-rigid objects which is efficient, modular, has straightforward implementation and provides superior performance on image sequences.

Benjamin et al. [5] rather than using the standard, Epanechnikov kernel, they have used a kernel weighted by the Chamfer distance transform to improve the accuracy of target representation and localization, minimizing the distance between the two distributions in RGB color space using the Bhattacharyya coefficient. By processing real time images and communicating wirelessly with their robot, they could track moving images against complex, cluttered backgrounds.

Dorin Comaniciu et al. [4] proposed a technique that employs the mean shift analysis to derive the target candidate that is the most similar to a given, target model, while the prediction of the next target location is computed with a Kalman filter. It combines the advantages of both mean shift and Kalman filter and it can theoretically track an object correctly even when occlusion is present.

Cheng et al. [6] proposed an improved mean shift object tracking algorithm. A novel weights given method is given, which improves the kernel function. The method is that the pixels which near the centre of object are given biggest weights, and the pixels which at the edge of the object are given by exponent distribution as a result of occlusion. In order to hand the occlusion, the occlusion detecting method based on sub-block detecting is also established. The novel sub-block detecting algorithm is that the tracking window is divided into two parts, including right and left, and the similarity measure is calculated respectively. The improved mean shift algorithm can hand the occlusion effectively and track the moving object very well, and it can track moving object more powerful than the basic mean shift tracked.

All of the above work will take a longer time for tracking process to finish because the algorithm implementation in software is rather complex and also software implementation can process 15-22 frames per second.

In *Kota et al. [7]* the complete system has been implemented on the hardware but the algorithm has been computed by the processor only, that gives an extra computation over head as well as decrement in frame rate. Above parameters affects the overall system performance.

3. SYSTEM ARCHITECTURE

Proposed architecture given in Figure 1, camera has been used as an input source that captures the real time video stream and passes it to the video processing board. FPGA Mazzanine Card (FMC) IMAGEOV has been used as main interface in between camera or DVI-module and video processing board that provides the development of video applications. SPARTAN6 FPGA has been used to develop an environment that allows the user to accelerate performance of video processing applications at low cost, low power industrial imaging systems. DDR has been used as an external memory to buffer the real time video stream. HyperTerminal has been used for user interaction. DVI-monitor has been used to display the processed data.

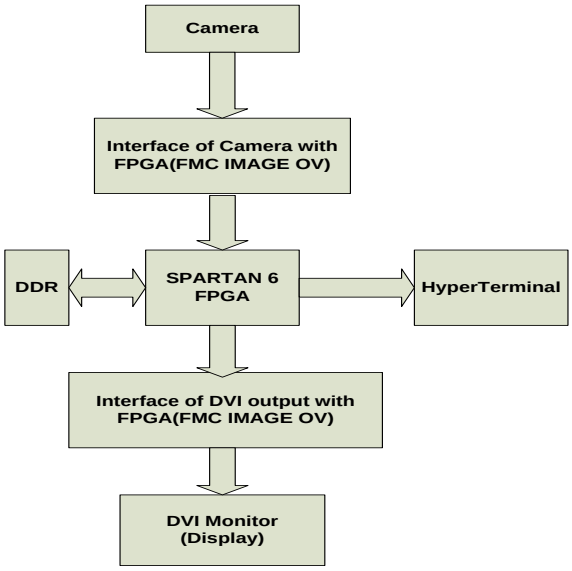

Figure 1: Proposed System Architecture

4. ALGORITHMS ANALYSIS

In Mean Shift algorithm, a feature space is chosen in the first frame to characterize the target model represented by its probability density function (pdf) 'q' centered at spatial location 0. In the second frame, candidate model is defined at location 'y' characterized by pdf p(y). Thus,

Target model[3]:

$$q_u = C\sum_{i=1}^{n} C\,k(||x_i||^2)\partial(b(x_i) - u'') \quad (1)$$

Candidate model [3] :

$$p(y) = C_h \sum_{i=1}^{n_h} k(\left|\frac{y-x_i}{h}\right|^2)\partial(b(x_i) - u) \quad (2)$$

Assuming size of the model to be normalized with kernel radius h=1.

C, C_h are the normalization constant given by

$$C = [\sum_{i=1}^{n} k(||x_i||)^2]^{-1} \quad (3)$$

$$C_h = \frac{1}{\sum_{i=1}^{n_h}\left(\left\|\frac{y-x_i}{h}\right\|^2\right)} \quad (4)$$

'∂' is the Kronecker delta function.

'k' is the Epanechnikov kernel, used to yield smoothed set of values in mean shift calculation that gives smaller weight to pixels farther from the centre that to avoid occlusion or interference from the background.

Distance between two discrete distributions is,

$$d(y) = \sqrt{1 - \rho[p(y), q]} \quad (5)$$

$\rho[p(y), q]$ is the Bhattacharya co-efficient[2] given by

$$\rho[p(y), q] = \sum_{u=1}^{n_h} w_i k(\left|\frac{y-x_i}{h}\right|^2 \quad (6)$$

Where,

$$w_i = \sum_{u=1}^{m} \sqrt{\frac{q_u}{p_u(y_0)}}\,\delta[b(x_i) - u] \quad (7)$$

5. SYSTEM ANALYSIS

As shown in Figure 2, the input video generated by Image sensor has been transferred to Camera Input IP Core. This IP core decodes the BT656 format generated by camera, to generate synchronization signals like horizontal synchronization signal (HSYNC), vertical synchronization signal (VSYNC) and active video signal (ACTIVE_VIDEO) and formats the input frames to XSVI (Xilinx Standard Video Interface) bus standard. The Video Detect IP core used with the system does not alter the video but monitors the VSYNC and ACTIVE_VIDEO signals generated by camera input IP core to determine the resolution of the active video region of the incoming frames. This IP core also generates Video DMA compatible bus interface that writes active video data to the external memory. The Video DMA (VDMA) IP core along with the Video Frame Buffer Controller (VFBC) performs the active video transfer to or from the external memory. These IP cores can be configured via the MicroBlaze processor used in system as soft core. The GENLOCK port indicates where the first instance of VDMA writes the incoming frames. The second instance of VDMA reads video frames from memory based on the GENLOCK information. After that the histogram calculation block run by processor, gets the pixel data from DDR and takes the RGB values, each of 8-bits. It takes the pre-calculated kernel values and finds out the histogram of the target and the candidate model and the compute displacement IP computes the displacement of the target object in each frame. Since the output frame rate is higher than the input frame rate, frames are duplicated when necessary. The Video Generate IP core, controlled by processor, configures video timing for the output. This IP core takes an XSVI bus interface as input and optionally drives the pins of the DVI output interface.

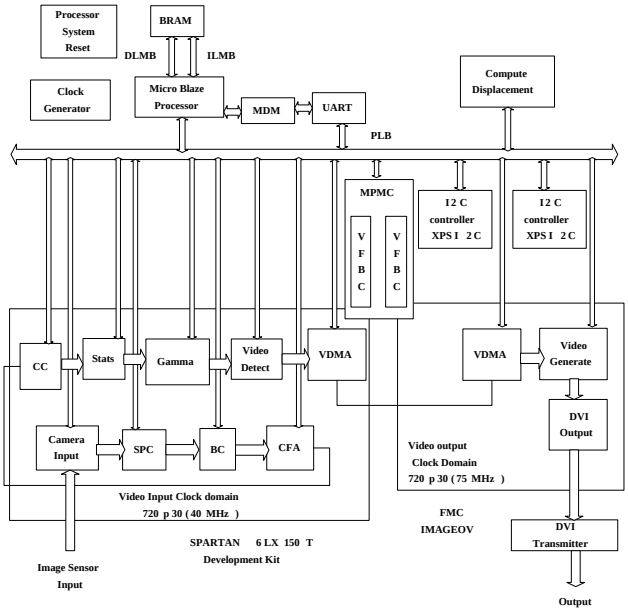

Figure 2: Base platform block diagram

As shown in Figure 3, weight of the data and kernel derivative has been computed by the processor and stored in BRAM1 and BRAM2 respectively. These BRAM has been integrated with the compute displacement IP. The process controller inside the IP core controls the data flow from one block to another and itself get controlled by the processor.

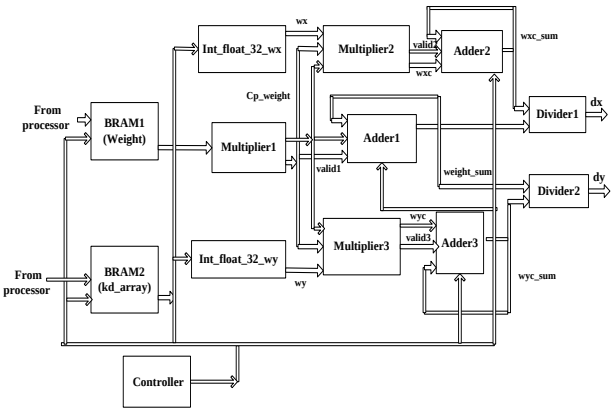

Figure 3: Block diagram of Compute Displacement

When the process controller is enabled by the MicroBlaze, it pass data to the Multiplier1 block from BRAM1 and BRAM2 and product (Cp_weight) has been calculated. Along with the Multiplier1, int_float_32_wx and int_float_32_wy blocks get enabled by the process controller. Multiplier2, Multiplier3 and Adder1 will be executed as input will be passed to them. Adder2 and Adder3 start their execution as they get input from Multiplier2 and Multiplier3 respectively. Process controller enables the Divider1 and Divider2 when all the data from BRAM1 and BRAM2 has been read. Finally, the displacement of the object will be computed.

6. EXPERIMENTAL SETUP

The Xilinx Embedded Development Kit (EDK) has been used to implement the system on Industrial Video processing Kit (IVK) board Spartan 6 LX150T. Figure 4 shows the board and the terminals used to test the design.

EDK is a suite of tools and Intellectual Property (IP) that facilitates to design a complete embedded system for implementation in a Xilinx Field Programmable Gate Array (FPGA) device. It generates the bitstream by merging software and hardware part and also configures the FPGA.

The Mean Shift algorithm is divided into different functions which are linked and coded in C language. Compute displacement which calculates the Bhattacharyya co-efficient is created as a custom IP.

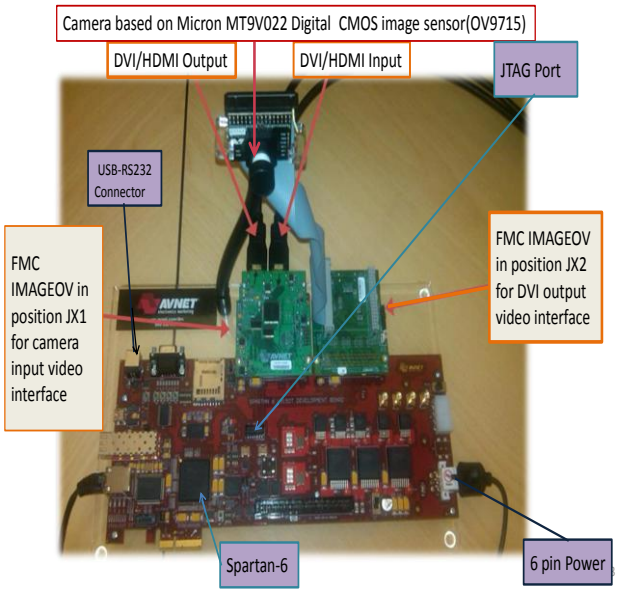

Figure 4: Call out diagram of Spartan 6 IVK board

The video resolution used is 1280x720P @ 30Hz of 24 bits per pixel. This resolution may be configured by the embedded processor and can be modified to support other resolutions (limited by the image sensor used) by tuning related parameter of I2C based camera control IC.

7. RESULTS

Compute displacement IP has been successfully simulated in ModelSim and the simulation result has been shown in Figure 5. It computes the x, y co-ordinates of the object in the next frame. In simulation the pixel values of the stored video frames has been provided to the custom IP and the result has been compared with the equations discussed above in the algorithm analysis section.

After verifying the simulation results, this IP core has been integrated with the system for real time video processing. Device utilization summary of the system has been shown in Table 1. The experiment shows that frame processing delay has been minimized by using dedicated hardware for the computation. This delay has been compared in real time environment with the system where the computation is performed by the processor.

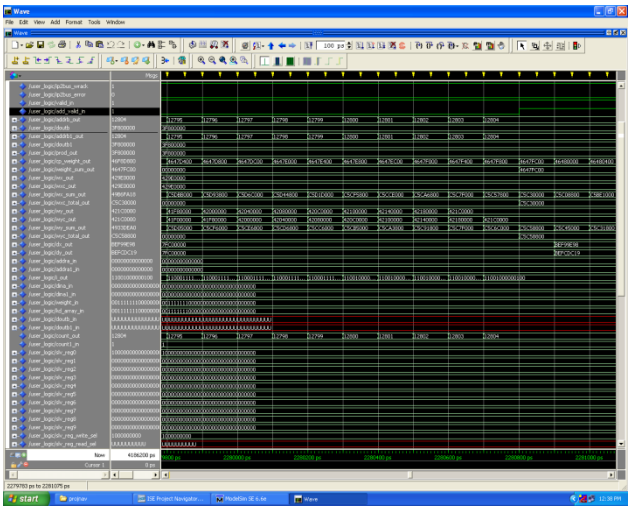

Figure 5: Simulation result in ModelSim

Table 1: Device Utilization Summary of the System

Device Utilization Summary			
Slice Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slice Registers	12,639	184,304	6%
Number used as Flip Flops	12,621		
Number used as Latches	2		
Number used as Latch-thrus	12		
Number used as AND/OR logics	4		
Number of Slice LUTs	21,543	92,152	23%
Number of occupied Slices	7,775	23,038	33%
Number of bonded IOBs	95	396	23%
Number of RAMB16BWERs	115	268	42%
Number of RAMB8BWERs	5	536	1%

8. CONCLUSION AND FUTURE WORK

The idea of partitioning the object tracking algorithm into Hw/Sw co design has been implemented and found to be very effective in performance. Results obtained in this work compared with existing works, proves that the concept has reduced the processing delay of the real time video frame.

The presented work may be extended by designing dedicated custom IP core for complete algorithm and software may be used only to control the IPs as well as display resolution. After implementation of complete algorithm in hardware, partial reconfiguration method may be used to minimize the utilization of device resource with negligible performance overhead.

9. ACKNOWLEDGMENT

We would like to thank Mr. Pramod Tanwar, Scientist-'C', CSIR-CEERI for his valuable suggestion and time to time discussions. At last but not least, we would like to thank Dr. Chandra Shekhar, Director, CSIR-CEERI, Pilani for giving opportunity to complete this work.

10. REFERENCES

[1] Alper Yilmaz, Omar Javed, Mubarak Shah, "Object Tracking- A Survey", ACM Computing Surveys, vol. 38, No. 4, Article 13, pp.145-190, December 2006

[2] Xiaohe Li, Taiyi Zhang, Xiaodong Shen and Jiancheng Sunb, "Object tracking using an adaptive Kalman filter combined with mean shift" UNC-Chapel Hill, TR 95-041,vol. 40, No. 7,pp 234-246, July 24, 2006

[3] D. Comaniciu, V. Ramesh, P. Meer, "Kernel-based object tracking," IEEE Trans. On Pattern Analysis and Machine Intelligence, pp. 564-575, Dec 2003.

[4] D. Comaniciu and P. Meer, "Mean shift: A robust approach toward feature space analysis," IEEE Trans. Pattern Anal. Machine Intel., vol. 24, no. 5, pp. 603–619,Dec 2002.

[5] Benjamin Gorry, Zezhi Chen, Kevin Hammond, Andy Wallace, and Greg Michaelson, "Using Mean-Shift Tracking Algorithms for Real-Time Tracking of Moving Images on an Autonomous Vehicle Testbed Platform" ,*International Conference on Intelligent Robotics and Manufacturing Automation, Venice, Italy, 2007* , World Academy of Science, Engineering and Technology (PWASET) , pp-45-48, *November 23-25, 2007*

[6] Tao Liu; Xiao-ping Cheng, "Improved mean shift algorithm for moving object tracking," *Computer Engineering and Technology (ICCET), 2010 2nd International Conference on* , vol.1, no., pp.V1-575,V1-578, 16-18 April 2010

[7] Kota Solomon Raju, Gargi Baruah, Manipati Rajesham and Palash Phukan, " Computing Displacement of Moving Object in a Real Time Video using EDK", International Conference on Computing, Communications, Systems And Applications(ICCCSA) Hyderabad, 30th-31st March 2012,pp 76-79; ISBN: 978-81-921580-8-2

[8] Spartan-6 Industrial Video Processing Kit – EDK Reference Design Tutorial, www.em.avnet.com/spartan6video.