Real-time Particle Image Velocimetry for Feedback Loops Using FPGA Implementation

Haiqian Yu*, Miriam Leeser† and Gilead Tadmor ‡
Boston, MA 02148
hyu{mel,tadmor}@ece.neu.edu

Stefan Siegel
US Air Force Academy, CO 80840

Digital Particle Image Velocimetry (PIV) is well established as a fluid dynamics measurement tool, being capable of non-intrusively and concurrently measuring a distributed velocity filed. Yet the intensive computational requirements of PIV limit its usage almost exclusively to off-line processing, analysis and modelling. This paper proposes hardware implementation of the cross-correlation algorithm as a means to make real-time PIV available for closed-loop control. This paper introduces a real-time PIV system which exploits the low-level parallelism of the cross-correlation computation by implementing it with reconfigurable hardware. The system processes 15 complete image pairs per second, which is more than 70 times speedup over a sequential software implementation. Moreover, our hardware structure can be easily expanded to a more parallel design for faster processing given sufficient hardware resources. This design can be reused with only minor modifications for different image sizes and interrogation areas.

I. Introduction

A. PIV

Particle image velocimetry (PIV) is a measuring technique for evaluating the velocity field in fluid flows. In a conventional PIV system,[1,2] small particles are added to the fluid and their movements are measured by comparing pairs of images of the flow, taken in rapid succession. The local fluid velocity is estimated by dividing the images into small interrogation areas and cross correlating the areas recorded in the two frames. Such systems are called double frame/single exposure systems.

PIV is an extremely useful method in fluid dynamics analysis because of its non-intrusive and concurrent measuring ability. It is clearly a highly desirable measurement tool in the emerging field of closed loop flow control. However, PIV's high computational complexity limits its usage almost exclusively to off-line processing and modelling. If PIV is to become widely applicable in feedback flow control, it is important to find new ways for computational speedup. This paper presents an implementation in reconfigurable hardware as a promising option.

B. FPGA

Field Programmable Gate Arrays(FPGAs) are a widely used reconfigurable hardware technology. Their fine grained parallelism can provide much faster processing for selected applications than a general purpose computer. Moreover, due to their reconfigurability, they are more flexible than Application Specific Integrated Circuits(ASICs). The key to the popularity of FPGAs is their ability to implement different circuits simply by being appropriately programmed.

Hardware speedups can be achieved by utilizing both temporal and spatial parallelism. Temporal parallelism maximizes the amount of time each piece of hardware is working while spatial parallelism enables

*PhD Student, ECE Dept., Northeastern University
†Associate Professor, ECE Dept., Northeastern University
‡Professor, ECE Dept., Northeastern University

multiple operations to run simultaneously. These two types of parallelism can greatly improve the performance of a design even if the processing frequency of the FPGA implementation is much lower than that of a general purpose computer.

Most FPGAs are composed of three fundamental components: logic blocks, I/O blocks and programmable routing.[3] The logic blocks in most modern FPGAs are built up from groups of Look-Up-Tables(LUTs) and registers with interconnection between them. Designs can be implemented in FPGA chips by configuring the LUTs and programming the routing.

C. Related Work

Digital PIV has proved to be very useful in fluid dynamics and related areas. Its applications include aerodynamics,[2,4] hydrodynamics, medical research[5] and micro-fluidics.[6] Real time requirements have given rise to a growing interest in real-time PIV systems. Research work as well as application specific commercial systems are being proposed.[7,8] Software processing, implemented in a standard DSP board or a PC, limits the number of interrogation areas processed in order to achieve even relatively moderate real-time requirements. In contrast, processing the entire image is possible in reconfigurable hardware. The first implementation of real-time application PIV was reported in.[7] That system runs at 10Hz for very small interrogation areas. Tsutomu et al.[9] and Toshihito et al.[10] have proposed a FPGA based real-time PIV system which can process 20 pairs of images per second using the Xilinx XC2V6000 chip. They exploit the redundant computation in cross-correlation for different interrogation areas in order to reduce the total number of operations. Therefore, the structure and performance of this design is very dependent of the size of the interrogation area. In contrast, the performance of the implementation presented in this research is independent of the design specifics.

In what follows we will first discuss potential PIV implementation algorithms according to their computational complexity and ability to be parallelized. In Section III we introduce our closed-loop system setup and give details of our hardware implementation. Section IV presents the results and its performance compared with a software implementation. Section V concludes the paper and closes with thoughts about future work.

II. PIV Algorithms

There are two commonly used PIV methods to estimate local particle velocity: *Direct Cross-Correlation* (Direct-CC) and *FFT-based Cross-Correlation* (FFT-CC). Another method named *feature based tracking*[11] has been proposed to estimate the velocity field, but it is only effective in feature locations and thus is not considered for our implementation. In this section, we present both Direct-CC and FFT-CC algorithms and select Direct-CC for our hardware implementation, based on a comparison of computational complexity.

A. Direct-CC vs. FFT-CC

Both implementation start with a pair of same size particle images, recorded from a traditional PIV recording camera. For processing, the images need to be divided into small interrogation windows. The selected window size depends on the flow velocity and the time interval between the times the two images are taken. We call the window from the first image *Area A* and that from the second, *Area B*. We use $N \times N$ to represent the size of the images, and $m \times m$ and $n \times n$ to represent the sizes of *Area A* and *Area B*, respectively. We assume that images and interrogation areas are square and that $m > n$. Finding the best match between *Area A* and *Area B* can be accomplished through the use of the discrete cross-correlation function, whose integral formulation is given in Equation (1):

$$R_{AB}(x,y) = \sum_{i=0}^{n-1}\sum_{j=0}^{n-1} A(i+x, j+y)B(i,j) \tag{1}$$

Here, *x, y* is termed the *sample shift*. For each choice of sample shift *(x,y)*, the sum of the products of all overlapping pixel intensities produces one cross-correlation value $R_{AB}(x,y)$. By applying this operation for a range of shifts ($-\frac{m-n}{2} \le x \le \frac{m-n}{2}, -\frac{m-n}{2} \le y \le \frac{m-n}{2}$), a correlation plane of the size $(m-n+1)\times(m-n+1)$ is formed. A high cross-correlation value indicates a good match at this sample shift position. The peak value is used as an estimate of the local particle movement, yielding an estimate of the local velocity field.

FFT-CC takes advantage of the correlation theorem which states that the Fourier transform of the two functions' cross-correlation is the complex conjugate multiplication of their Fourier transforms. This is shown in Equation (2), where $\mathbf{F}\{\}$ stands for Fourier transform.

$$\mathbf{F}\{R_{AB}(x,y)\} = \mathbf{F}\{A(x,y)\} \cdot \mathbf{F}\{B(x,y)\}^* \tag{2}$$

The FFT based cross-correlation system is shown in Figure 1. For FFT-CC, it is required that the two inputs have the same size. Areas A and B must be padded with zeros before applying the FFT. We use $k \times k$ to represent the size of the zero-padded interrogation area. Labels under the FFT blocks represent the number of complex multiply operations.

For our application, we set $n = 32$ and $m = 40$ for the size of the interrogation areas. In comparing implementations, we ignore addition operations and count only multiplications. This is reasonable since multipliers require more hardware resources and computation time. For the Direct-CC algorithm, the total number of multiplications is $32 \times 32 \times 9 \times 9 = 82944$ for one interrogation area. For FFT-CC, we need to select $k = 64$ because $32 < m < 64$. The total number of multiplications is $(((32 \times 4) \times \log_2 64) \times 64) \times 2 \times 3 + 64 \times 64 \times 4 = 311296$, since each complex number multiplication requires four real number multiplications. Based on these observation we choose to proceed with the direct cross-correlation algorithm.

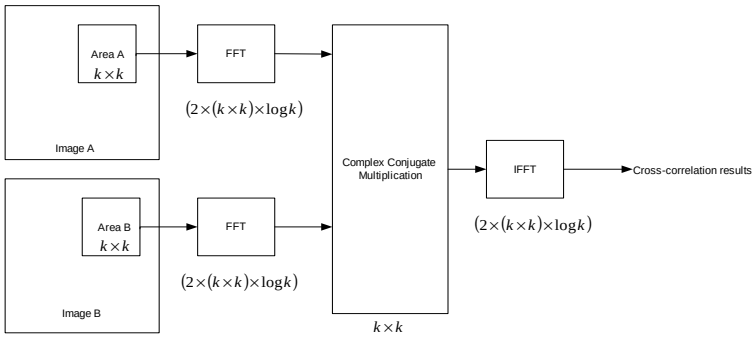

Figure 1. FFT-based Cross-Correlation System Diagram.

B. Sub-pixel Interpolation

By finding the position of the peak value of the cross-correlation plane we can determine the local displacement of particles and thus estimate the movement of fluid. The position of the correlation peak can be measured to sub-pixel accuracy using sub-pixel interpolation. Several methods of estimating the peak position have been developed. For narrow correlation peaks, using three adjacent values to estimate the correlation peaks is widely used and proven to be efficient. The most common three-point estimators are parabolic peak fit (Equation (3)) and Gaussian peak fit(Equation(4)).

$$\begin{cases} p_x = x + \frac{R_{(x-1,y)} - R_{(x+1,y)}}{2R_{(x-1,y)} - 4R_{(x,y)} + 2R_{(x+1,y)}} \\ p_y = y + \frac{R_{(x,y-1)} - R_{(x,y+1)}}{2R_{(x,y-1)} - 4R_{(x,y)} + 2R_{(x,y+1)}} \end{cases} \tag{3}$$

$$\begin{cases} p_x = x + \frac{lnR_{(x-1,y)} - lnR_{(x-1,y)}}{2lnR_{(x-1,y)} - 4lnR_{(x,y)} + 2lnR_{(x+1,y)}} \\ p_y = y + \frac{lnR_{(x,y-1)} - lnR_{(x,y+1)}}{2lnR_{(x,y-1)} - 4lnR_{(x,y)} + 2lnR_{(x,y+1)}} \end{cases} \tag{4}$$

We use parabolic peak fit in our implementation since it is more appropriate for hardware implementation and its accuracy is comparable with Gaussian peak fit.

III. System Implementation

A. Closed-Loop System

We use a similar setup as that described in,[8] with the distinction that we implement the cross-correlation and peak finding processes in reconfigurable hardware. To meet real-time processing requirements, the software implementation[8] must sacrifice spatial resolution because it cannot complete the required cross-correlation of all interrogation areas during the time interval between image updates. In our new implementation, reconfigurable hardware is used to replace the software that performs the cross-correlation, which is the most computationally intensive part of the calculation.

Our system setup is shown in Figure 2. The PIV camera is connected to a frame grabber. Data from the CCD camera streams into memory through the frame grabber. As soon as enough data has been acquired, the hardware processing unit starts cross-correlation and finding peaks in the correlation plane. The resulting velocity information is then sent to the host for post processing before it goes to the feedback control unit, to be used in closed loop actuation. A more compact design can send the velocity information directly to the feedback control unit to further shorten the delays. The frame grabber can acquire 15 image pairs per second and therefore our real-time requirement of processing speed is 15 pairs/second. The previous design,[8] which did not use reconfigurable hardware, investigated only a small number of interrogation areas to save processing time. Our hardware design processes all the interrogation areas at the speed of 15 image pairs per second.

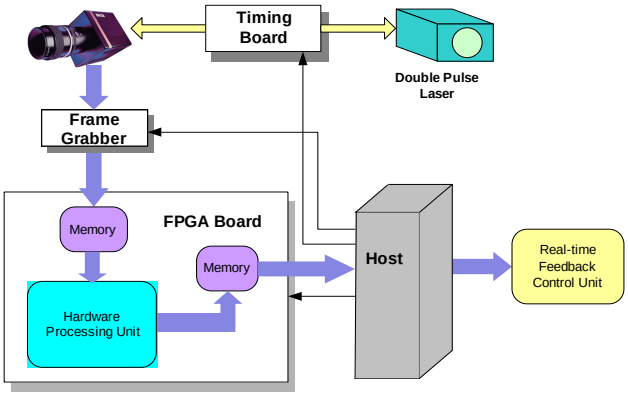

Figure 2. System Diagram.

B. Hardware

Our targeting board, Firebird, is a commercial computing engine from Annapolis Micro Systems, Inc.[12] It has 5 on-board memory banks (36MB in total) and one Xilinx Virtex2000E[13] FPGA chip. The FPGA chip can access the on-board memory through a memory interface and the host can access the memory through the FPGA chip or Direct Memory Access(DMA). Figure 3 shows the block diagram of the FireBird. The memory banks on the FireBird are called *on-board* memory while memory in the FPGA chip is called *on-chip* memory. We discuss the differences of these two types of memories and how we organize them for better performance below.

Figure 4 shows the pipeline stages of our accumulation part. The rectangles between stages are registers for storing intermediate results. The numbers shown in the figure represents the bit widths for each stage. The bit widths in our design are carefully chosen to guarantee no errors are introduced in the accumulation stages. The data streams into on-board memory from the CCD camera through the frame grabber. We use on-chip memory to store one pair of interrogation areas because accessing on-board memory has a much longer delay than accessing on-chip memory. For each interrogation area, we load *Area A* and *Area B* from *on-board* memory to *on-chip* memory, stream these interrogation area data into the pipeline stages, and complete cross-correlation process. The results are available after only a few clock cycles delay. Two *on-chip* memories are used to store one interrogation area so that we can load the next interrogation area in parallel with processing the current interrogation area. 32 multipliers are selected for our 32×32 interrogation area B. 4 pixels are grouped in *on-board* memory locations for a higher memory bandwidth. The data width in *on-chip* memory is 256 bits so that in one read operation, we can access one line of data in one clock cycle for 32 simultaneously multiply operations. With sufficient hardware resources, this structure can be duplicated to process several interrogation areas in parallel thus achieving an even higher speedup.

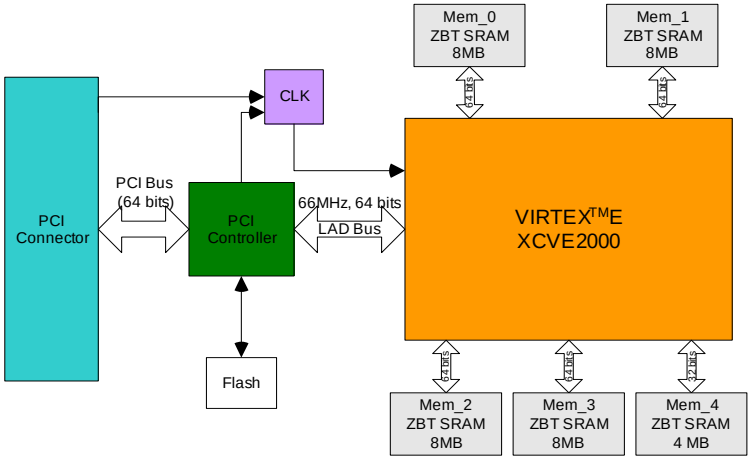

Figure 3. FireBird Block Diagram.

Our design is limited by the availability of *on-chip* memory of the Xilinx Virtex2000E. Currently, we duplicate the pipelined structure shown in Figure 4 so that two interrogation areas are processed in parallel.

Such a design can meet the closed-loop system requirements of 15 pairs of images/second. For applications where particle movement is faster, a faster data stream is required thus leading to more computations. An FPGA chip with a larger on-chip memory would enable a more parallel design in such systems.

IV. Performance

Our FPGA implementation improves performance in two ways. First, the parallel and pipelined design greatly reduces the image processing time. Second, by implementing the hardware processing unit in a closed-loop system, we can process data right off the camera. Processing can start even before an entire pair of images are captured.

Our application has two input images of size 1008×1016. The interrogation window of *Area A* is 40×40 and of *Area B*, 32×32. Interrogation windows have 50% overlap. We implement sub-pixel interpolation using the parabolic peak fit algorithm.

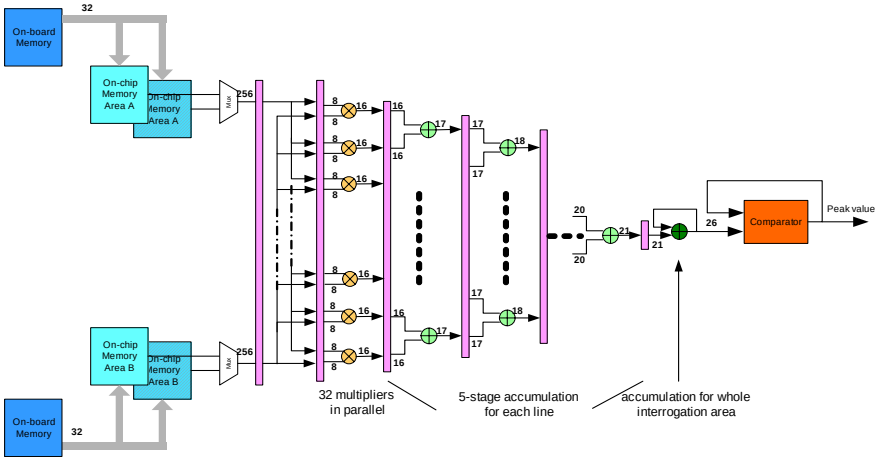

Figure 4. Pipelined Structure.

Our results show that for the same cross-correlation and sub-pixel interpolation algorithm, software using fixed-point running on an Intel(R) 1.5GHz Xeon requires 3.4 seconds while the FPGA implementation using an Annapolis Micro Systems' FireBird board takes only 0.047 seconds. The speedup of data transfer is not so easy to estimate since it depends on the memory type, the way data is transferred, etc. Still, we can safely say that the overall speedup is more than 70 times for our current hardware structure. This speedup can be further improved with a more parallel structure.

V. Conclusion

Real-time PIV is required in most closed loop systems. Cross-correlation is very computationally intensive, thus software implementation cannot meet the real-time requirements. Our reconfigurable hardware implementation can process the entire image with a speed of 15 pairs of images/second, more than 70 times speedup over a software implementation. The design presented in this paper can be easily used for other applications with only minor modifications. In the future, we plan to integrate this design into a closed-loop feedback control system.

Acknowledgements

This research was supported in part by by National Science Foundation Grant CCR-0208791.

References

[1]Adrian, R., "Particle-imaging technique for experimental fluid mechanics," *Annual Reviews in Fluid Mechanics*, Vol. 23, 1991, pp. 261–304.

[2]Raffel, M., Willert, C., and Kompenehans, J., *Particle Image Velocimetry*, Springer-Verlag, Berlin, Germany, 1998.

[3]Trimberger, S. M., editor, *Field-Programmable Gate Array Technology*, Kluwer Academic Publishers, 1994.

[4]Willert, C., Raffel, M., and Kompenhans, J., "Recent applications of particle image velocimetry in large-scale industrial wind tunnels," *ICIASF '97*, Sept. 1997, pp. 258–266.

[5]Hochareon, P., Manning, K., Fontaine, A., Deutsch, S., and Tarbell, J., "Development of high resolution particle image velocimetry for use in artificial heart research," *EMBS/BMES Conference*, Oct. 2002, pp. 1591–1592.

[6]Meinhart, C., Wereley, S., and Santiago, J., "PIV Measurements of a Microchannel Flow," *Experiments in Fluids*, Vol. 27, 1999, pp. 414–419.

[7]Arik, E. B. and Carr, J., "Digital Particle Image Velocimetry System for Real-time Wind Tunnel Measurements," *ICIASF '97*, Sept. 1997, pp. 267–277.

[8]Siegel, S., Cohen, K., and McLaughlin, T. E., "Real-time Particle Image Velocimetry for Closed-Loop Flow Control Studies," *41th AIAA Aerospace Sciences Meeting*, 2003.

[9]Maruyama, T., Yamaguchi, Y., and Kawase, A., "An Approach to Real-Time Visualization of PIV Method with FPGA," *FPL2001*, Jan. 2001, pp. 601–606.

[10]Fujiwara, T., Fujimoto, K., and Maruyama, T., "A Real-Time Visualization System for PIV," *FPL2003*, Sept. 2003, pp. 437–447.

[11]Verestoy, J. and and, D. C., "Digital PIV: A Chanllenge for Feature Based Tracking," *Proc. 23rd Workshop of the Austrian Pattern Recongnition Group*, 1999, pp. 165–174.

[12]Annapolis Micro Systems Inc., *FireBird^{TM} Reference Manual*.

[13]www.xilinx.com, *Virtex^{TM}-E 1.8 V Field Programmable Gate Arrays*, Xilinx Inc., July 2002.