JOURNAL OF AEROSPACE COMPUTING, INFORMATION, AND COMMUNICATION
Vol. 3, February 2006

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

Haiqian Yu* , Miriam Leeser† and Gilead Tadmor‡
Northeastern University, Boston, MA 02115

and
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 field. Yet the intensive computational requirements of PIV limit its use almost exclusively to off-line processing, analysis and modeling. This paper proposes hardware implementation of the cross-correlation algorithm as a means to make real-time PIV available for closed-loop control. We introduce a real-time PIV system which exploits the low-level parallelism of the cross-correlation computation by implementing it with reconfigurable hardware. The system can process 15 *complete* image pairs per second, which is more than 70 times speedup over a sequential software implementation. Moreover, for high speed PIV cameras with the ability of capturing thousands of images per second, a selected area cross-correlation is implemented to shorten the processing time for each pair of images at the cost of decreased spatial resolution. This implementation is very suitable for model based high speed PIV analysis. Our hardware structure can easily be 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 measurement 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 measurement ability. It is clearly a highly desirable measurement tool in the emerging field of closed loop flow control. However, PIV's high computational complexity and resulting slow processing speed limits its use almost exclusively to off-line processing and modeling. We are interested in applying PIV for real-time feedback flow control, which

Received 6 June 2005; revision received ??; accepted for publication 12 October 2005.

* PhD student, ECE Dept., 440 DANA, 360 Huntington Ave, Boston, MA 02115, AIAA Student Member.
† Associate Professor, ECE Dept., 440 DANA, 360 Huntington Ave, Boston, MA 02115.
‡ Professor, ECE Dept., 440 DANA, 360 Huntington Ave, Boston, MA 02115, AIAA Member.
§ Assistant Research Associate, Dept. of Aeronautics, HQ USAFA/DFAN, Colorado Springs, CO 80840, AIAA Senior Member.

requires considerable computational speedup. This paper presents an implementation in reconfigurable hardware as a way to achieve the computational speedup.

Using PIV in real time systems is an emerging field of research, as illustrated by experiments on the circular cylinder wake at the US Air Force Academy.[3,4] The main motivation is to have multiple non-intrusive, far field sensing locations. Added advantages include the fact that these measurements are feasible even in hostile environments, such as those with high temperatures. Up until now, dynamic measurements have been restricted almost exclusively to traditional sensors, such as microphones and hot wire sensors, where the physical structure of the sensor accommodates a wide dynamic range. Yet, these standard sensing techniques deliver only a single sensing point per device, and are restricted almost exclusively to the wall shear layer, which reduces the sensitivity to details of far-field coherent structures. Even when far-field sensing is feasible with traditional sensors, such as with a hot wire, such sensing can be difficult and intrusive and is sensitive to harsh far field environments.

A common feature in the emerging field of feedback control in fluid flow systems using low dimensional models is that the information used for feedback decisions is based on the instantaneous values of the time varying coefficients of dominant expansion modes. It is clearly advantageous to estimate these values using multiple deep field measurements of the velocity field at selected points, as compared with a few wall mounted sensor measurements. Our research,[3–6] as well as the research of others,[7–9] details control algorithms where the availability of far field data is advantageous in feedback flow control. Real-time PIV therefore not only addresses a particular feedback algorithm, but enables a whole new class of applications.

The traditional stumbling block for the real time use of PIV in feedback control has been the unavailability of the fast processing needed to produce information useful for real-time control, such as the value of the velocity field at selected locations. In order for PIV to become useful for feedback control, it must become a dynamic technique, which can only be achieved by substantial speedup of the required processing of the PIV algorithm. The advantage of an FPGA implementation is that it makes this substantial speedup possible. FPGAs are particularly suited to the kind of processing needed for real-time PIV implementations. FPGAs can be configured for massive fine grained parallelism, which is the source of the speedup for PIV implementation. The degree of parallelism available in an FPGA implementation is not available using traditional microprocessors or DSPs.

PIV data processing may introduce additional bottlenecks such as the flow of raw data from the camera. This issue can be circumvented by embedding the FPGA processing in the camera. The volume of output can be dramatically reduced by only transmitting the estimated values of the velocity field at a small set (typically one to ten) of desired locations. Further reduction in the communication burden can be achieved if the computations associated with the dynamic feedback algorithm are implemented on the same board. An example is the dynamic estimation of the Fourier coefficients of efficient Galerkin expansions of the flow field, such as the Proper Orthogonal Decomposition (POD). However, the computations associated with PIV processing, to compute the desired values of the velocity field, remain the main bottleneck. Addressing this bottleneck, which is our goal here, is therefore a key enabler. Beyond real time applications, an FPGA based correlation algorithm implemented directly in the camera can benefit standard PIV measurements as well. Modern high speed cameras buffer all acquired images in camera memory before transferring them to the processor. Currently, the entire image data needs to be transferred from the camera to the processor, typically a PC. Even with the fastest interfaces commercially available, this transfer adds significant amounts of time to the overall workflow. If the images can be processed in real time in the camera, only the correlation results need to be transferred, which is a much faster process since the amount of data to be transferred is then 1-2 orders of magnitude smaller. More importantly, the smaller amount of data to be transferred will allow for continuous measurements that are not limited by the amount of memory available in the camera.

In the next section, we provide general background on FPGA architectures. FPGA implementations are particularly well suited to solving PIV due to their ability to provide pixel level processing in real time, and at relatively low cost. In addition, out implementation can easily be reconfigured for a particular PIV problem to match the parameters of the problem such as the size of the image, window size, amount of overlap between windows, etc. The implementation described in this paper processes pairs of images completely and in real-time. Our enhanced implementation can handle higher image acquisition speeds by finding only the velocity field in the neighborhood of areas of interest. The implementation can be reconfigured to change the location of these areas of interest during the application, thus allowing us to make multiple deep field measurements of the velocity field at selected points.

B. Field Programmable Gate Arrays

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.

For PIV, the advantage of using FPGAs is that the low level pixel processing required at high speed can be performed efficiently by exploiting the inherent parallelism in FPGA architectures. Both temporal and spatial parallelism is exploited. Temporal parallelism maximizes the amount of time each piece of hardware is working while spatial parallelism enables multiple operations to run simultaneously. These two types of parallelism can greatly improve the performance of a PIV implementation even if the processing frequency of the FPGA is much lower than that of a general purpose computer performing the same tasks.

FPGAs are composed of three fundamental components: logic blocks, I/O blocks and programmable routing.[10] The logic blocks are built up from groups of Look-Up-Tables (LUTs) and registers with interconnection between them. Each LUT can be configured to provide a simple combinational or sequential logic function when appropriately connected to the registers. Moreover, LUTs can be grouped together using multilevel programmable routing hierarchies to implement more complicated logic functions.[11] I/O blocks can configure each I/O pin for a different function thus providing an interface between the FPGA chip and peripheral components. By downloading different bitstreams which can configure the logic blocks, I/O blocks, and programmable routing, different designs can be implemented in FPGA chips. For PIV, such reconfiguration allows different image sizes, window sizes and overlap to be handled efficiently.

Increasing integration density and performance of integrated circuits makes high performance million gates FPGAs possible. Besides CLBs, I/O blocks and programmable routing, modern FPGAs usually have on-chip memory in the form of Random Access Memory (RAM) blocks, and some have embedded multipliers or even complete RISC processors on the chip. These extra available resources can further boost the performance of the implemented digital circuits. In this paper, we consider an FPGA structure composed of the three fundamental components with additional on-chip memory as our available FPGA hardware resources. The small but fast on-chip memory can be used in hardware implementation to form an efficient memory hierarchy to improve performance.

FPGAs provide an efficient hardware implementation platform using programmable technologies for pixel level processing such as that found in PIV. For example, our implementation can perform more than 32 multiply operations and 32 additions every clock cycle; thus exhibiting a very high level of spatial parallelism. In addition, the memory interface is designed to keep all the operations active all the time; thus exhibiting a high level of temporal parallelism. Due to this high level of parallelism, an FPGA is more efficient for this kind of low level pixel processing than other programmable architectures such as DSPs.

C. Related Work

Digital PIV has proved to be very useful in fluid dynamics and related areas. Its applications include aerodynamics,[2,12] hydrodynamics, medical research[13] and micro-fluidics.[14] Real-time requirements have given rise to a growing interest in real-time PIV systems. Research work as well as application specific commercial systems have been proposed.[15,16] Software processing, implemented on 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, rather than a limited number of interrogation areas, is possible in reconfigurable hardware. The first implementation of real-time application of PIV was reported in Ref. 15. That system runs at 10 Hz for very small interrogation areas. Tsutomu et al.[17] and Toshihito et al.[18] have proposed a FPGA based real-time PIV system which can process 20 pairs of images per second using the Xilinx XC2V6000 chip. Their design processes the entire image, and processes windows that are up to 32×32 pixels in size, with 50% overlap. 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 here is independent of the design specifics.

Model-based feedback control design uses a small number of sensors to estimate the much larger number of states that characterize system dynamics. Whether explicitly or implicitly, the dynamic component of a feedback

compensator is used to estimate the state from these few measurements. The Luenberger observer[19] is a standard framework for real-time state estimation. The restriction on the number of sensors stems from considerations such as cost, weight, size, and physical accessibility. Considering the cylinder wake benchmark, we have shown[5,6] that a mere single velocity sensor suffices to reconstruct sufficient information for successful actuation.

However, in order to estimate all dominant modes in this flow field, or for flow fields of higher spatial complexity in general, more than a single sensor reading is necessary for state estimation. Cohen et al.[20] have shown that using a heuristic sensor placement scheme which locates sensors on extremes of the Proper Orthogonal Decomposition (POD) spatial modes minimizes the number of sensors required for a given estimation quality. For situations involving noisy measurements, considerations involving the signal to noise ratio distribution in the flow field can be used to further optimize sensor placement.[21] For flow state estimation of this kind, a real-time PIV system providing selected sensor readings in a spatially configurable fashion is ideal. Therefore, in most cases, not all the interrogation areas of the PIV field of view need to be computed to estimate the flow state. Dynamically choosing several areas of interest provides for even faster processing. There is no related work that we are aware of in hardware implementation of selected interrogation areas.

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

PIV is a well established technique.[1,2] In this section we provide the background needed to understand our PIV implementation and the design choices that were made in the implementation. 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*[22] 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 implementations 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, which is also called Direct-CC in this paper. The integral formulation of Direct-CC 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 $(-(m - n/2) \leq x \leq (m - n/2), -(m - n/2) \leq y \leq (m - n/2))$, 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 $F\{\}$ stands for Fourier transform.

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

The FFT based cross-correlation system is shown in Fig. 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. The FFT-CC requires more processing than the Direct-CC algorithm. Based on these observations we choose to proceed with the direct cross-correlation algorithm.

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 for estimating the peak position have been developed. For narrow correlation peaks, 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)).

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

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

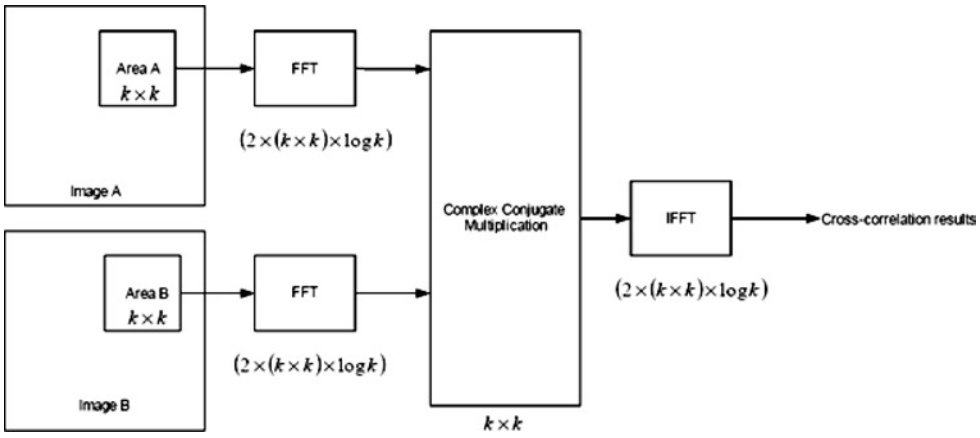

Fig. 1 FFT-based cross-correlation system diagram.

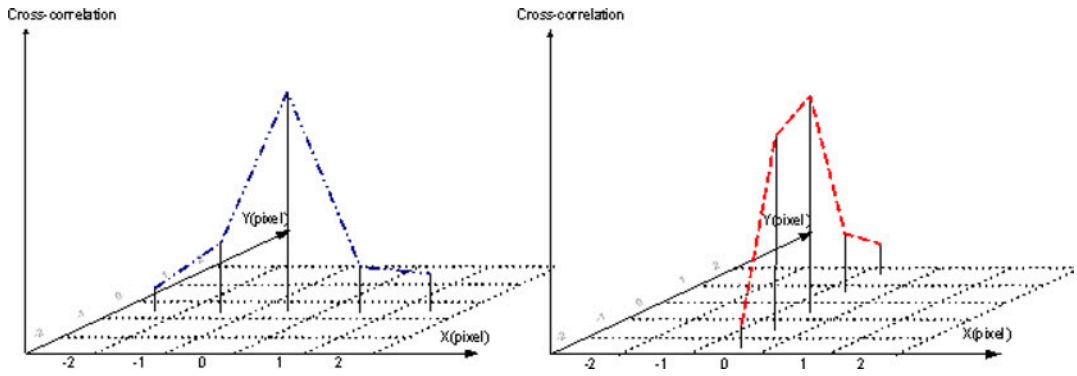

Fig. 2 Parabolic peak fit sub-pixel interpolation.

We chose to use parabolic peak fit in our implementation. Gaussian peak fit is commonly accepted to be more accurate than parabolic peak; however this comes at a cost of more complex hardware implementation. Parabolic peak fit can be implemented with shifts and adds, which are relatively inexpensive hardware operations. The additional logarithmic functions in the Gaussian peak fit formula consume a great deal of hardware area. Our current implementation achieves the same accuracy as Ref. 2. Figure 2 shows an example of parabolic peak fit sub-pixel interpolation. In this figure, the cross-correlation peak value appears at position ($x = 0$, $y = 0$). After considering the neighboring cross-correlation value using the parabolic peak fit method, we adjust the position of the peak value to ($x = -0.1$, $y = -0.4$).

III. System Implementation

A. Closed-Loop System

We use a similar setup as that described in Ref. 16, 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[16] 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 Fig. 3. The PIV camera is connected to a frame grabber. Data from the CCD camera streams into memory on the FPGA board through the frame grabber. As soon as enough data has been acquired, the

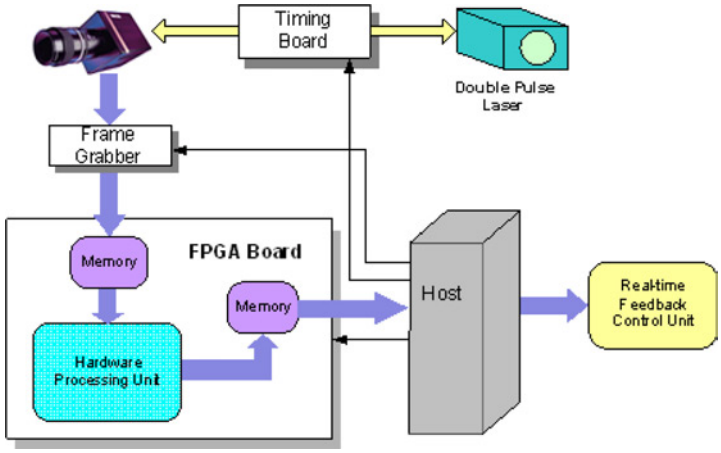

Fig. 3 System diagram.

hardware processing unit starts cross-correlation and finding peaks in the correlation plane. The processing begins before the second image has been completely acquired. 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 delay. 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,[16] 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.

B. Hardware

Our target board, Firebird, is a commercial computing engine from Annapolis Micro Systems, Inc.[23] It has 5 on-board memory banks (36MB in total) and one Xilinx Virtex2000E[24] 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 by using Direct Memory Access (DMA). Figure 4 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 5 shows the block diagram of our FPGA implementation. The frame grabber puts data from the camera into the on-board memory through the I/O interface and memory interface of the FPGA. All the on-board memory banks are dual-port Static RAMs so that pixel data can be read out from the memory to the processing unit while the frame grabber is streaming data to the memory. We use *on-chip* memory to store one pair of interrogation areas because accessing *on-chip* memory has a much shorter delay than accessing *on-board* memory. Figure 5 also shows that we actually use two copies of *on-chip* memory to store the interrogation areas and the temporary results. We use one copy of *on-chip* memory to store a pair of interrogation areas. The next pair of interrogation areas can be loaded from *on-board* memory to the other copy of *on-chip* memory while we are still processing the current interrogation area. These two copies of *on-chip* memory change their role after the processing of one interrogation area processing is complete. By doubling the size requirement of the *on-chip* memory, we can overlap the processing and data loading time for better speedup.

Figure 6 shows the detail of the pipeline stages of our correlation part in Fig. 5. 32 multipliers are selected for our 32×32 interrogation area B. The rectangles between stages are registers for storing intermediate results. The numbers shown in the figure represent 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. 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 the cross-correlation process. The results are available after only a few clock cycles delay.

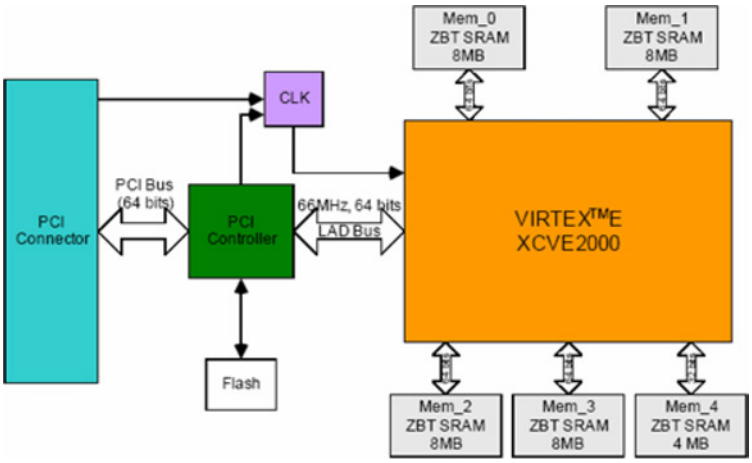

Fig. 4 FireBird block diagram (from Annapolis Micro Systems).

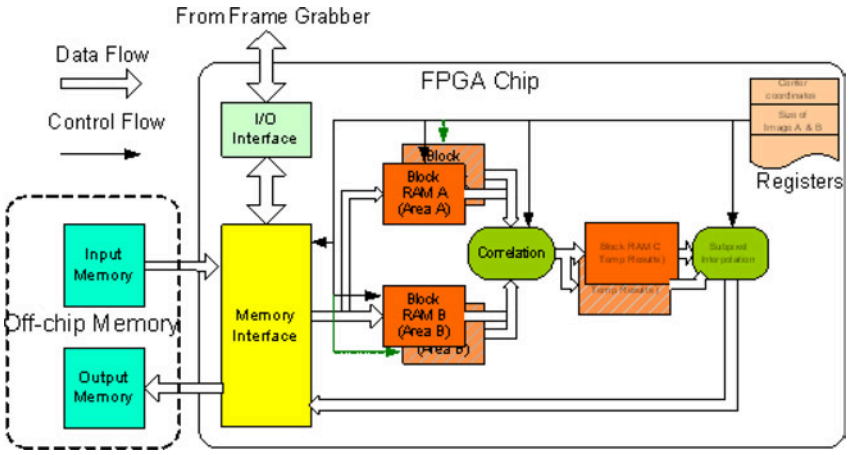

Fig. 5 Block diagram of FPGA implementation.

To efficiently use the memory bandwidth between the *on-board* memory banks and the FPGA chip, we grouped several pixels together into one memory address. In our implementation, the pixel bit width is 8 while the memory bank bit width is 64. That means we can maximally group 8 pixels in one memory address so that for each read cycle, 8 pixels can be loaded from the *on-board* memory to the *on-chip* memory. Considering the 50% interrogation window overlap factor, we group 4 pixels in one *on-board* memory location to avoid a more complicated controller while at the same time increasing the memory bandwidth usage by a factor of 4. The data width in *on-chip* memory is configured to be 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.

Our design is limited by the availability of *on-chip* memory of the Xilinx Virtex2000E. Currently, we duplicate the pipelined structure shown in Fig. 6 so that two interrogation areas are processed in parallel. Such a design can

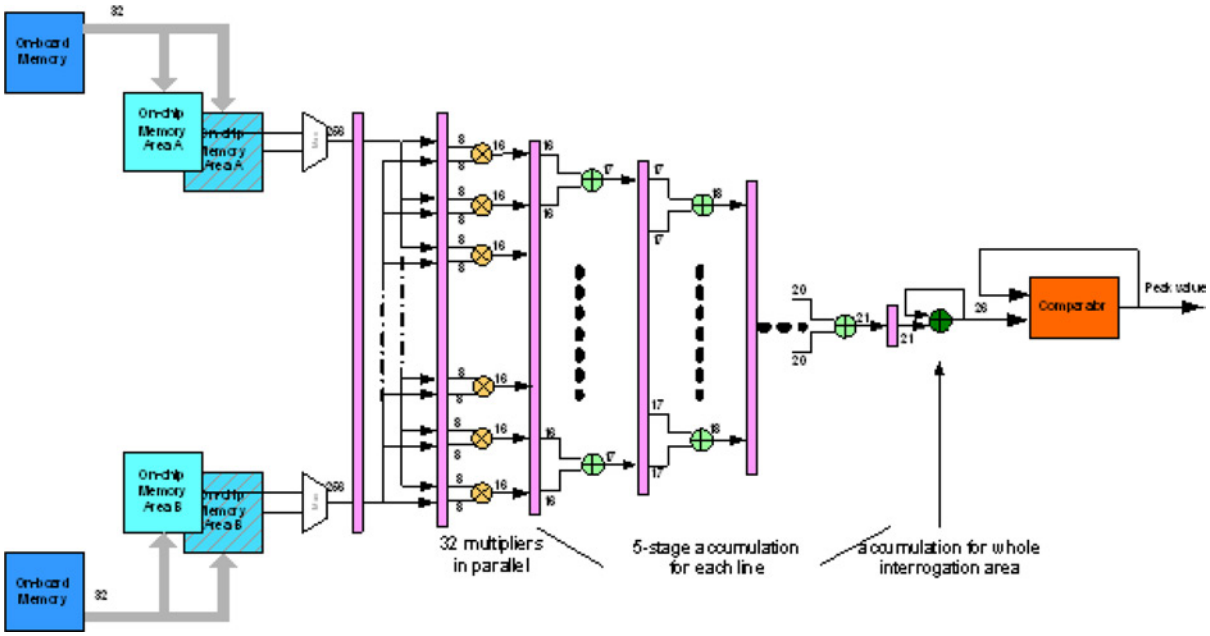

Fig. 6 Pipelined structure.

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 connecting the frame grabber directly to the on-board memory, we can process data right off the camera. Processing can start even before an entire pair of images is captured.

Our implementation 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.

A. Complete Interrogation

Our results show that for the same cross-correlation and sub-pixel interpolation algorithm, software using fixed-point arithmetic running on an Intel(R) 1.5 GHz Xeon requires 3.4 seconds while the FPGA implementation using an Annapolis Micro Systems' FireBird board takes only 0.047 seconds (Fig. 7(a)). 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.

B. Selected Area Interrogation

In the current context, the advantage of using only a few interrogation areas is primarily computational. Time is of the essence in feedback control, and computation time must be short when compared with the time constants of controlled processes. Given the short time constants typical of flow dynamics, the feasible scope of PIV-in-the-loop is directly associated with the ability to accelerate PIV processing, as well as subsequent control and command computations. It is therefore of obvious interest to explore the computation time advantages of minimizing the number of interrogation points.

We have implemented selected area correlation with the same hardware setup. For each pair of images, 8 interrogation areas are investigated per image pair and the corresponding velocity estimates are sent to the host. Users can change the positions of the interrogation areas according to the model requirements each time a new pair of images is acquired. The total processing time for one pair of images is only 235 μs for 8 interrogation areas. If the data can be transferred to the FPGA on-board memory in time, we can process a total of more than 4000 image pairs per second. This approach is extremely useful for systems requiring high speed processing. Figure 7(b) compares the processing time of complete interrogation and selected area interrogation.

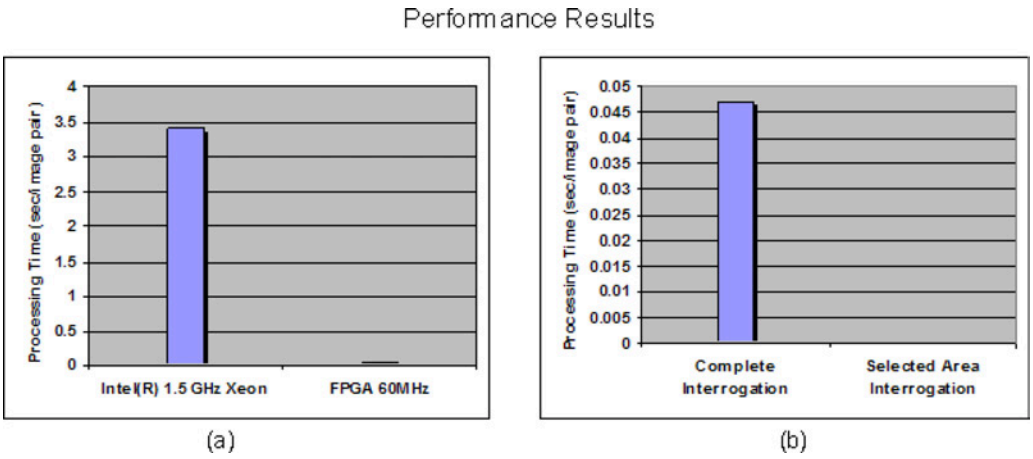

Fig. 7 (a) Hardware vs. Software; (b) Complete interrogation vs. Selected area interrogation.

To achieve the speed of 4000 image pairs per second, we need to make sure two other requirements are met. First, the image data flow from frame grabber to the memory on the FPGA board must be fast enough to transfer 4000 images pairs per second. Second, the interrogation area data flow from memory to the FPGA chip must be fast enough to transfer 8×4000 interrogation area pairs per second. According to the parameters in our experiment, the bottleneck will be the data flow rate from the frame grabber to the memory on the FPGA board. For example, a high speed camera with the ability to acquire 4000 pairs of 1008×1016 images per second would require the data flow rate to be 65 Gbits/sec from the frame grabber to the memory on the FPGA board for real-time processing. In this case, it would be better to put the hardware processing unit on the camera to avoid the huge data transfer between the camera and the FPGA board. Several camera manufacturers are investigating this option.

Our current experimental setup can measure dynamic velocity fields of complete images with a maximal frequency up to 10 Hz. By reducing the spatial resolution to 8 interrogation areas per image pair, the FPGA we are currently using has the potential to measure the flow with frequency up to 2000 Hz. In our experiment, the only part of the processing implemented in the FPGA chip is the cross-correlation, because this is the bottleneck in our current real-time system. Higher speed, real-time PIV is possible with the availability of more advanced FPGA chips. The entire processing, including data acquisition, timing control, feedback, etc. can potentially be implemented on the FPGA board. By doing so, we can greatly reduce the data exchange between different modules and obtain even more speedup. This will enable real-time control and feedback based on PIV.

V. Conclusion

Real-time PIV is one of the very few non-intrusive means of implementing multi-sensor estimation of flow fields for feedback control. 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. Selected area correlation is also implemented and 8 interrogation areas per image pair can be processed at the speed of 4000 pairs of images per second. The design presented in this paper can be easily used for other applications with only minor modifications. In the future, we plan to integrate the hardware processing part into the camera so that instead of transferring the huge amount of image data from the camera, we can transfer only the velocity data out of the camera for much faster real-time PIV applications.

Acknowledgments

This research was supported in part by National Science Foundation Grants CCR-0208791 and CCR-0410246.

References

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

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

[3]Siegel, S., Cohen, K., and McLaughlin, T., Feedback Control of a circular cylinder wake in a water tunnel experiment. 42nd AIAA Aerospace Sciences Meeting, Reno, NV AIAA paper 2004-0580, 2004.

[4]Siegel, S., Cohen, K., and McLaughlin, T., "Experimental Variable Gain Feedback Control of a circular cylinder wake," AIAA Fluid Dynamics Conference Portland, OR AIAA paper 2004-2611, June 2004.

[5]Gerhard, J., Pastoor, M., King, R., Noack, B. R., Dillmann, A., Morzy'nski, M. and Tadmor, G., "Model-based control of vortex shedding using low-dimensional Galerkin models," In *33^{rd} AIAA Fluids Conference and Exhibit*, Paper 2003-4262. Orlando, Florida, U.S.A., June, 2003.

[6]Tadmor, G., Noack, B. R., Dillmann, A., Gerhard, J., Pastoor, M., King, R., and Morzy'nski, M., "Control, observation and energy regulation of wake flow instabilities," In 42^{nd} IEEE Conference on Decision and Control 2003, pp. 2334–2339. WeM10-4, Maui, HI, U.S.A., December, 2003.

[7]Rowley, C. W., Juttijudata, V. and Williams, D. R., "Cavity flow control simulations and experiments," AIAA paper 2005-0292, 43^{rd} AIAA Aerospace Sciences Meeting, Jan 2005.

[8]Afanasiev, K., and Hinze, M., "Adaptive control of a wake flow using proper orthogonal decomposition," *Shape Optimization & Optimal Design*, edited by Cagnol, J., Polis, M. P. and Zolesio, J.-P., Lecture Notes in Pure and Applied Mathematics, Marcel Dekker, New York, Vol. 216, 2001, pp. 317–332.

[9]Collis, S. S. Joslin, R. D., and Seifert, A., "Vassilis Theofilis, Issues in active flow control: theory, control, simulation, and experiment," *Progress in Aerospace Sciences*, Vol. 40, 2004, pp. 237–289.

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

[11]Ingo Cyliax, "Learning the Ropes: The FPGA Tour." *Circuit Cellar Online.* URL: http://www.circuitcellar.com/library/ropes/1199/c119lrpdf.pdf, November 1999. [cited August, 2005.]

[12]Willert, C. Raffel, M., and Kompenhans, J., "Recent applications of particle image velocimetry in large-scale industrial wind tunnels," *International Congress on Instrumentation in Aerospace Simulation Facilities*., September 1997, pp. 258–266.

[13]Hochareon, P., Manning, K. B., Fontaine, A. A., Deutsch, S., and Tarbell, J. M., Development of high resolution particle image velocimetry for use in artificial heart research. *International Conference of the IEEE Engineering in Medicine and Biology Society* , October 2002, pp. 1591–1592.

[14]Meinhart, C. D., Wereley, S. T., and Santiago, J. G., "PIV measurements of a microchannel flow," *Experiments in Fluids*, Vol. 27, 1999, pp. 414–419.

[15]Engin B. Arik and John Carr., "Digital particle image velocimetry system for real-time wind tunnel measurements," *International Congress on Instrumentation in Aerospace Simulation Facilities*, September 1997, pp. 267–277.

[16]Stefan Siegel, Kelly Cohen, and Thomas E. McLaughlin, "Real-time particle image velocimetry for closed-loop flow control studies." *41st AIAA Aerospace Sciences Meeting*, AIAA paper 2003-0920, 2003.

[17]Tsutomu Maruyama, Yoshiki Yamaguchi, and Atsushi Kawase, "An approach to realtime visualization of PIV method with FPGA," International Conference on Field-Programmable Logic and Applications, January 2001, pp. 601–606.

[18]Toshihito Fujiwara, Kenji Fujimoto, and Tsutomu Maruyama, "A real-time visualization system for PIV," 13th International Conference on Field Programmable Logic and Applications, Lisbon, Portugal. September 2003, pp. 437–447.

[19]Luenberger, D. G., Observers for multivariable systems. *IEEE Transactions on Automatic Control*, Vol. AC-11, 1965, pp. 190–197.

[20]Kelly Cohen, Stefan Siegel, and Thomas McLaughlin, "A heuristic approach to effective sensor placement for modeling of a cylinder wake," *Accepted for publication in Computers and Fluids*, August 2003.

[21]Kelly Cohen, Stefan Siegel, Dave Wetlesen, Jeff Cameron, and Aaron Sick, "Effective sensor placements for the estimation of proper orthogonal decomposition mode coefficients in von karman vortex street," *Journal of Vibration and Control*, Vol. 10, December 2004, pp. 1857–1880.

[22]Judit Verestoy, Dmitry Chetverikov, and Marcell Nagy, "Digital PIV: A challenge for feature based tracking, Proceeding, 23rd Workshop of the Austrian Pattern Recognition Group, 1999, 165–174.

[23]*FireBirdTM Reference Manual, Revision 3.0* Annapolis Micro Systems Inc., Annapolis Maryland, 2000.

[24]*VirtexTM-E 1.8 V Field Programmable Gate Arrays*. Xilinx Inc., San Jose, CA. July 2002. URL: www.xilinx.com. [cited August, 2005.]