Real-time image compression for high-speed particle tracking

King-Yeung Chan, Dominik Stich, and Greg A. Voth[a)]
Department of Physics, Wesleyan University, Middletown, Connecticut 06459

(Received 1 November 2006; accepted 16 January 2007; published online 27 February 2007)

High-speed particle tracking with digital video creates very large data rates and as a result experimenters are forced to make compromises between spatial resolution, temporal resolution, and the duration over which data is acquired. The images produced in particle tracking experiments typically contain a large amount of black space with relatively few bright pixels and this suggests the possibility of image compression. This paper describes a system for real-time compression of high-speed video. A digital circuit placed between the camera (500 Hz, 1280×1024 pixels) and frame grabber compresses data in real-time by comparing input pixels with a threshold value and outputs a vector containing the brightness and position of the bright pixels. In a typical experiment, the compression ratio for an image ranges from 100 to 1000 and varies dynamically depending on the number of filtered pixels. The reduced data rate makes it possible to write directly to the hard disk. While previously data could only be acquired for 6.5 s into 4 GB of dedicated video RAM, the new system could acquire full resolution data continuously for up to a week into a 600 GB hard drive. © *2007 American Institute of Physics.* [DOI: 10.1063/1.2536719]

I. INTRODUCTION

Optical particle tracking has become an important tool in the study of fluids and soft materials.[1–4] In the studies of turbulent fluids in our research group we use stereoscopic high-speed digital imaging to reconstruct three-dimensional (3D) trajectories of particles. However, this approach faces serious constraints due to the huge data rates produced by multiple high-speed video cameras. Many of the other approaches to particle tracking such as acoustic tracking,[5] confocal imaging,[3] and holographic imaging,[6] have similar problems with huge data rates. The core problem is that using images to extract 3D trajectories with high spatial resolution, high temporal resolution, and a long duration requires rapid acquisition of huge data sets.

A typical experimental configuration uses between two and four high speed cameras. We currently use Basler A504k cameras that record 1280×1024 pixel images at 500 Hz. (More expensive cameras have recently become available that, for example, can take 600×800 images at 6700 Hz.[7]) Any of these (complementary metal-oxide-semiconductor) CMOS cameras can achieve higher frame rates by decreasing the spatial resolution since the limiting factor is the data transfer rate. The data rate from each of our cameras is 625 MB/s, and storing this data stream is currently impractical with anything other than dedicated video RAM. We typically have 4 GB of RAM for each camera, which allows us to record only 6.5 s of data before waiting for the much slower transfer to hard disk. The large volume of data also requires a very long processing time. Other groups working on this problem have implemented various solutions to the data rate problem. Ott and Mann[8] accepted the low space and time resolution available from standard PAL video cameras. Voth *et al.*[9] built custom imagers that allowed one-dimensional (1D) projection imaging at 70 kHz. Oulette *et al.*[10] have decreased the spatial resolution of high speed video cameras to 256×256 pixels in order to achieve 27 kHz imaging rate.

In this article we describe a device that significantly reduces data volume and lengthens data duration in such experiments. It is an image compression circuit whose principle is based on the observation that images produced by digital cameras in particle tracking experiments usually contain a large amount of black space with localized bright spots indicating particle positions. We have implemented a simple and effective image compression scheme by disposing of the black space in an image and outputting only the location and brightness of bright pixels [Fig. 1].

The compression factor achieved by this circuit is encouraging. In a typical experiment, the factor of compression for an image is around 100–1000, depending on the number of bright particles present. Since the data transfer rate is significantly reduced, data can be directly written to the hard drive, whose memory capacity is much larger. The reduction in data volume and the increase in storage capacity enables the new system to record for up to a week instead of seconds. The time for data processing and analysis is also significantly reduced. We expect that systems similar to this can also be used effectively for time-resolved particle image velocimetry and other applications.

II. THE COMPRESSION CIRCUIT

A. Overview

This central piece of the system is a digital circuit situated between our Basler A504k high-speed camera and EPIX E4 frame grabber [Figs. 2 and 3]. It compresses data in real-time without using the processor(s) of the host computer, so

[a)]URL: http://www.gvoth.web.wesleyan.edu/lab.htm

5	31	**115**	**203**
9	29	**96**	**138**
14	23	32	25
6	17	21	8

[**115**, *2*, *0*]
[**203**, *3*, *0*]
[**96**, *2*, *1*]
[**138**, *3*, *1*]

FIG. 1. A conceptual diagram showing an image compressed into bright pixel vectors[(brightness, x, y)]. A threshold of 35 was used and the filtered brightness values are in bold type.

data size is reduced before they reach the processing machine.

This circuit receives and transmits data through two pairs of serial Camera Link cables (one pair each for input/output). A standard chipset of Camera Link differential line receivers are implemented on the circuit board to convert signals from serial to parallel format to be read by the compression processor. Differential transmitters then encode the compressed data back into serial Camera Link format for output. These components are mounted on a custom-built circuit board that we designed using circuit design software provided by Pad2Pad,[11] who manufactured the board and mounted the components.

Situated in the middle of our circuit board is an Altera Cyclone FPGA (Field-Programmable Gate Array). This FPGA is capable of working above 200 MHz and, therefore, is easily able to handle the camera clock of 67.58 MHz. We chose to use a programmable chip because of its versatility and the parallel processing capabilities it provides. The chip is mounted on a Parallax hardware development kit which allows convenient upload of circuit layout files from the computer to the chip. The circuit layout was designed and tested using Quartus II, the free development software provided by Altera, using Verilog HDL (hardware description language). Another advantage of using an FPGA is that its programmability allows for convenient expansion and modification of the system.

The circuit was designed to receive and recreate the original signal format of the camera and frame grabber so that neither of them has to be modified internally to use this device. The frame grabber basically receives normal camera frames that contain encoded data instead of original pixels.

There are other ways that such a system could be implemented. It would be possible to do the compression with a dedicated digital signal processor. It may become possible to perform the compression on the CPU of a host computer. Currently this is unlikely to be reliable since current systems can just barely record data to memory using DMA over the PCI Express bus at the 650 MB/s rate needed. (The Epix E4 card we use just became available in 2006 and a firmware upgrade in the fall of 2006 was necessary to get the full 650 MB/s through DMA.) However, our view is that the most promising future developments for imaging systems, such as this, will be to follow the example of biological imaging systems that do the image compression in parallel at the imaging device. Our implementation using parallel logic for compression results in both a useful device and a step toward this goal.

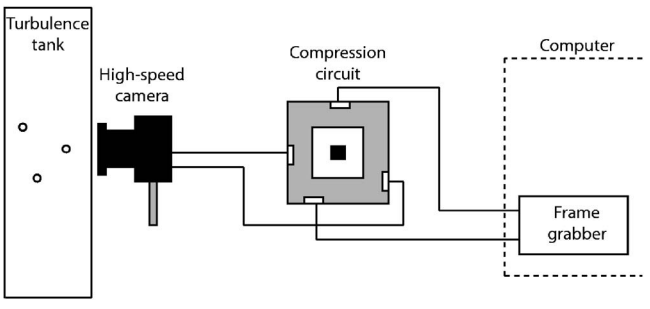

FIG. 2. A diagram showing the experimental setup.

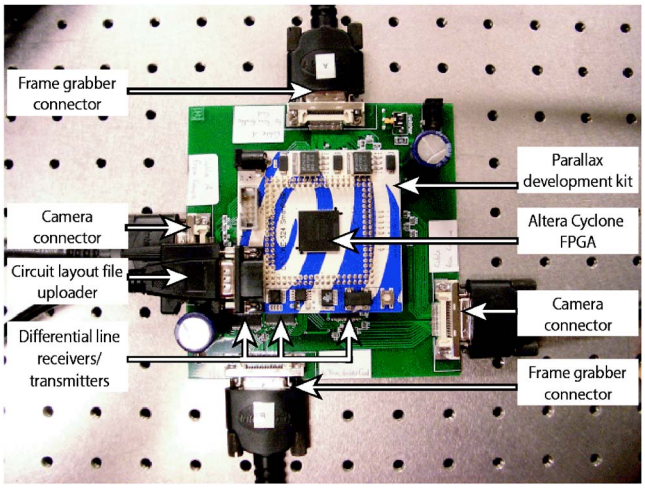

FIG. 3. A photograph of the compression circuit with annotations on its basic components.

B. Image processor architecture

All the computation in this device is done by the programmed circuit in the FPGA. Its function is to filter pixels according to a built-in threshold value and output the bright pixels as a vector of location and brightness (referred to below as "bright pixel vector"). A schematic diagram of the programmed circuit is shown in Fig. 4.

The input data from the camera is clocked at 67.58 MHz. In each clock cycle, ten pixels (8-bits per pixel) are received in parallel. The circuit also receives a *frame valid* bit indicating frame breaks, a *line valid* bit indicating line breaks, and three other camera configuration/trigger signals which are unchanged by the circuit. The pixels are transferred line by line in a left-to-right, top-to-bottom manner. The maximum resolution of a frame is 1280×1024 pixels.

The position counter [Fig. 4(a)] keeps track of the current pixel position according to the pixel clock, *frame valid* bit and *line valid* bit. The x position is counted in tens (0–127, at maximum resolution), since 10 pixels come in parallel in each clock cycle.

Ten parallel filters [Fig. 4(b)] compare incoming pixels with a preset threshold value [Fig. 4(c)]. If a pixel is higher than the threshold, the filter sends its brightness to the corresponding FIFO (first-in-first-out memory device) buffer [Fig. 4(d)]. Each filter also maintains a counter for how many bright pixels it has passed through in a frame.

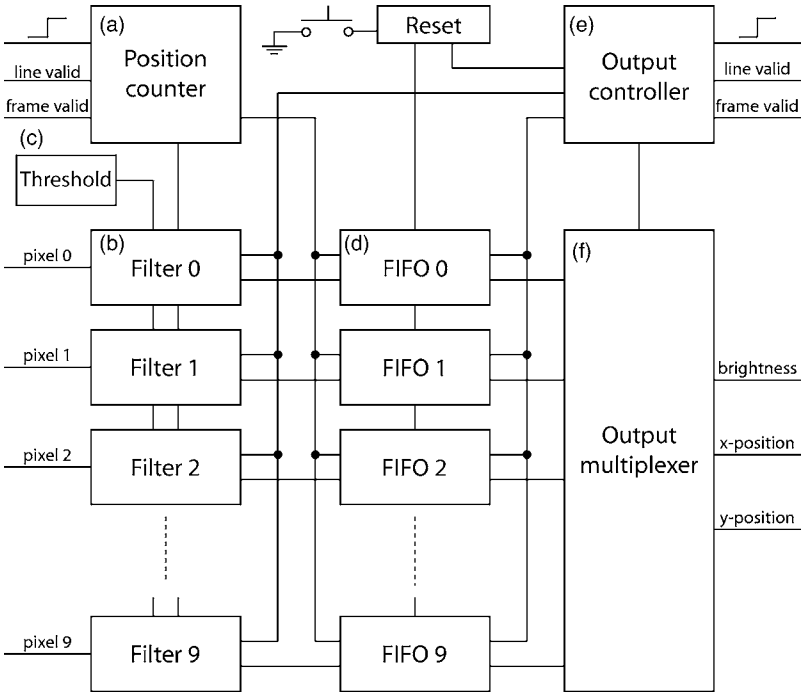

FIG. 4. A conceptual diagram of the circuit architecture used in the image processor.

Ten FIFOs [Fig. 4(d)] with a length of 1000 words and a width of 25 bits are used to store the brightness and location data. Each FIFO receives the brightness information from its corresponding filter module and the position from the signal counter, a total of 25 bits $[8(\text{brightness})+7(x)+10(y)]$. While the data in the FIFOs are being read out by the output controller [Fig. 4(e)], pixels from the next frame are being transferred in simultaneously. Therefore, the capacity of each FIFO is only half of its size, and the maximum number of bright pixels allowed from each filter in one frame is only 500. This means that not more than 5000 bright pixels can be stored from any frame, and in practice the limit is somewhat less than this because some FIFOs will overflow before others are filled.

The output controller [Fig. 4(e)] recreates a new camera frame containing the compressed data. It has to follow the same signal timing rules that the camera uses. At the end of each frame, it receives the number of bright pixels from the counters in the filters and begins to clock out the correct number of bright pixel vectors out of each FIFO. Since each FIFO contains data from two frames at the same time, the counters are essential for distinguishing between data from the current frame and from the next frame.

An output multiplexer [Fig. 4(f)] relays the data output from the FIFOs to the output bus. It is responsible for calculating the exact x coordinate for output, as the x position was only counted in tens up to this point. The number of bits required by the x position is now 11 (0–1279, at maximum resolution), which increases the output bits to 29 bits $[8(\text{brightness})+11(x)+10(y)]$.

C. Output format

This circuit imitates the structure of a camera frame, outputting data in a ten-byte-per-cycle format. The byte arrangement in a clock cycle is shown in Fig. 5(b). Bytes 4–9 are zero, due to the limited number of output pins in the Parallax development kit used in this prototype circuit. Bytes 0–3 contain the bright pixel information whose breakdown is shown in the Fig. 5(c). A visualization of a compressed image is shown in Figs. 6(b) and 6(c). Data can potentially only occupy the top 40 lines in a compressed frame because 5000 bright pixel vectors fill 39.06 lines. When there are fewer bright pixels, fewer lines are occupied, since data after the last bright pixel vector will be zero. The black vertical columns in the frame are bytes 4–9 of each vector which are zero.

III. SUPPORTING SOFTWARE

A. Overview

A new software application was designed to interface with this new compression system and serve as an imaging application for researchers to visualize and record data using the new setup. Its primary function is to record the com-

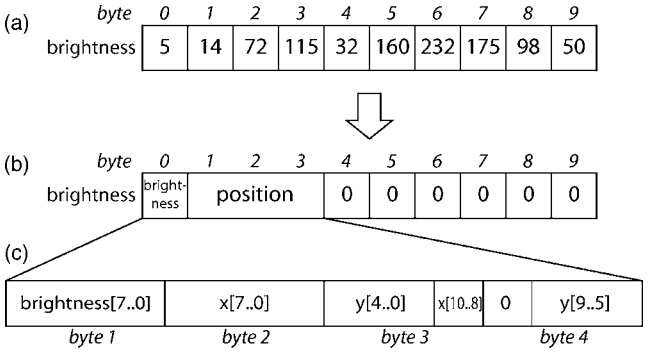

FIG. 5. A comparison between 10-byte arrays (data sent in one clock cycle) in uncompressed format and compressed format. (a) In an uncompressed array, each pixel occupies one byte. The pixel value is the brightness of the pixel. (b) In a compressed array, each bright pixel vector occupies 10 bytes. Bytes 1–3 contains information about the bright pixel vector. Bytes 4–9 are zero due to insufficient output of the compression circuit. (c) The breakdown of the first 4 bytes.

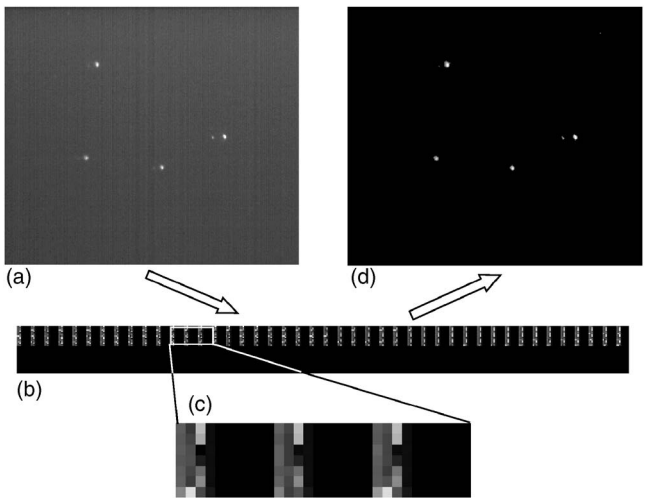

FIG. 6. Sample images (295×276 out of 1280×1024 pixels) showing stages of compression (threshold=100). (a) Original image. (b) A compressed frame, which is an encoded camera frame sent by the compression circuit. Black space at the bottom shows that there are no more bright pixels in that frame. (c) Zoomed view of bright "columns" in a compressed frame. They are 4 bytes wide and contain the bright pixel vectors. The other 6 bytes are zero. (d) Decompressed image. Decompression is done by software after recording.

pressed data to hard drive in real-time. It displays live decompressed images at 10 frames per second for visualization. It also communicates with the camera to configure basic settings such as frame rate, exposure, gain, and area of interest. It was developed in Visual C++ and uses the XCLIB library functions provided by the manufacturer of the frame grabber.

B. Real-time recording algorithm and further compression

The most important feature of this application is that it is capable of saving the compressed images to the hard drive in real-time. In the saving process, two additional stages of compression were done in real-time in the software.

As mentioned above, bytes 4–9 of each encoded bright pixel vector are zero. Additional compression was done by writing only the first 4 bytes out of 10 bytes to file, effectively removing all the zero bytes in each bright pixel vector. Another layer of compression involves the recording algorithm detecting the size of the bright pixel array in each image and writing only the necessary data to the hard drive.

The recording algorithm is also responsible for detecting data overflow when the number of bright pixels exceeds 5000, the memory limit of the 10 FIFOs on the processor. The overflowed frame will be tagged in the file. It also detects missed frames which occur if the speed of recording cannot catch up with the frame rate.

The compressed data is stored in a custom video file format (.cpv, compressed video). Each .cpv file starts with a file header that includes recording information such as original image size. To differentiate between frames, each frame is enclosed between a 32-bit frame header indicating the number of that frame, and a 32-bit end-of-frame marker. The data in each frame consists of 4-byte blocks, which are byte 0–3 of a bright pixel vector [Fig. 5(c)].

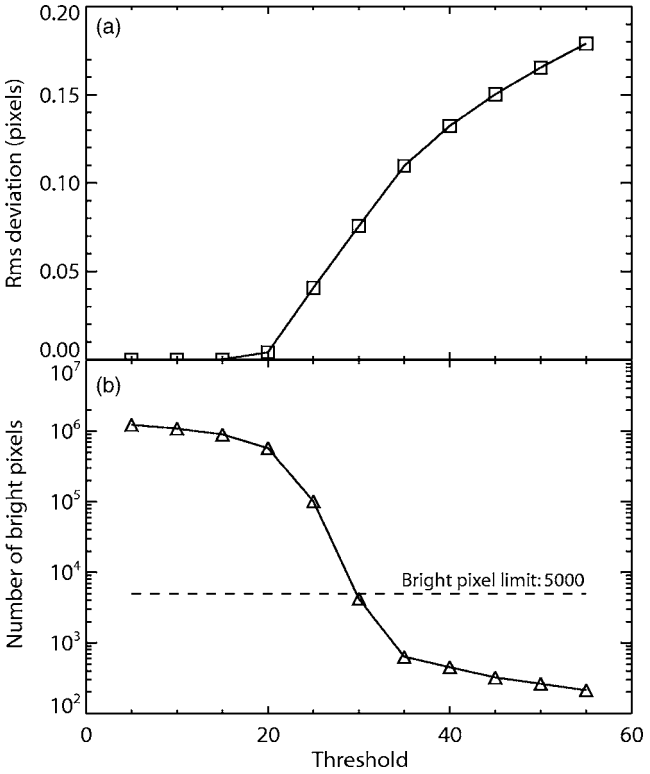

FIG. 7. (a) Deviation between particle positions measured from compressed images and the original positions measured from uncompressed images for several different values of the threshold. Only particles with a peak brightness above 80 were considered. (b) The number of filtered pixels for each threshold. The dashed line shows the maximum number of filtered pixels allowed in the current circuit, which is 5000.

IV. TEST RESULTS AND DATA ACCURACY

Test data was collected with the turbulence tank setup in our laboratory. Runs of 900 000 frames at 500 fps (30 min) at maximum resolution (1280×1024 pixels) were recorded. Each compressed file occupied around 1.4–2 GB. If this experiment were completed with the original system, the total file size needed was 900 000×1280×1024bytes=1099 GB. Therefore, these experiments achieved a compression ratio between 550 and 800. The average number of particles in view during these experiments is 16, and the compression ratio is expected to depend on particle density.

The most important breakthrough achieved by this system is the capability of recording data over a long period of time without sacrificing spatial or temporal resolution. Long data sets are essential to allow analysis of slow phenomena and to improve statistical convergence.

Since dim pixels around a particle that were below the threshold value are discarded, it is important to check the accuracy of the new compression system in locating particles. A statistical test was done to find the difference between locations of particles obtained from full images and compressed images. A series of 100 original images were filtered in software, producing the same effect as the compression circuit. Particle locations were found from the compressed images and compared to locations found from the original images. Figure 7(a) shows that the error in locating particles increases as the threshold increases, since more and more bright pixels are being cut off. For this data set, a

threshold of 35 yields an accuracy of 0.11 pixel, and the total number of bright pixels is well below the system limit [Fig. 7(b)]. Accuracy of about 0.1 pixel is typical for digital imaging of tracer particles, so this result indicates that position accuracy with this system is acceptable as long as the threshold is kept at a reasonably low level.

V. IMPLEMENTATION AND COSTS

The powerful tools available for high-speed digital circuit design make it straightforward and inexpensive to develop and test a system like this. The three design phases (the hardware program, the printed circuit board design, and the software application) each took about six months of effort by a Master's level student who was learning the necessary computer languages and programs as needed.

We faced several issues during development of this system. One challenge for us was to learn the rules of high-speed digital trace layout and power supply buffering. We designed the board based on guidance from Ref. 12 and never had a problem with this aspect of the design. On the other hand, the company that fabricated and stuffed the initial board did poor work in the soldering, and we ended up losing over a month sending that board back and forth. We also had difficulties with operating system latencies that caused missed frames during writing to the hard drive, but this problem has not occurred since we began using the EPIX E4 frame grabber that uses a 4× PCI Express bus.

The cost of the compression circuit is minimal compared with the cost of the cameras themselves. Currently, a Basler A504k camera with E4 frame grabber is about $10 000, not including the host PC. We are in the process of fabricating six more compression boards, and are spending about $600 each. About half of this is for the custom circuit board fabrication and stuffing, and the other half is for the Parallax development kits which contain the Altera programmable chips.

VI. DISCUSSION

This system demonstrates the possibility of real-time hardware-level image preprocessing of high speed video from particle tracking experiments. It significantly improves data acquisition efficiency by achieving a high compression ratio and expanding experimental duration. This system uses a commercially available camera and frame grabber, and the custom designed image processing board is relatively inexpensive to build.

There exist many possibilities for usage and expansion of this device. The use of a programmable chip in this circuit allows additional functions to be added to this device to perform more complicated signal preprocessing. For instance, one can implement more advanced compression schemes, or even implement feature-finding algorithms to locate the bright particles or objects in real-time. Potentially, this system can be implemented in many situations that involve high-speed recording of simple images such as time-resolved particle image velocimetry or x-ray imaging. The current limit of the system to images with less than 5000 bright pixels can be extended by replacing the Altera Cyclone with an FPGA of larger capacity. We are currently working on an upgrade to an Altera Stratix chip that could raise the FPGA storage limit above 100 000 pixels; of course, there may then be other factors such as hard drive write speed that will limit the number of pixels that can be stored. Eventually, the image compression step could be integrated into the imaging device itself to eliminate the bottleneck formed by data offload from the imager. This could allow the imaging frame rate to increase by a factor similar to the compression ratios achieved.

ACKNOWLEDGMENTS

This work was supported by Wesleyan University, the Alfred P. Sloan Foundation, and NSF Grant DMR-0547712. The authors thank Jim Johnson for help with data collection.

[1] A. LaPorta, G. A. Voth, A. Crawford, J. Alexander, and E. Bodenschatz, Nature **409**, 1017 (2001).
[2] G. A. Voth, G. Haller, and J. P. Gollub, Phys. Rev. Lett. **88**, 254501 (2002).
[3] E. R. Weeks, J. C. Crocker, A. C. Levitt, A. Schofield, and D. A. Weitz, Science **287**, 627 (2000).
[4] E. C. Rericha, C. Bizon, M. D. Shattuck, and H. L. Swinney, Phys. Rev. Lett. **88**, 014302 (2002).
[5] N. Mordant, P. Metz, O. Michel, and J. F. Pinton, Phys. Rev. Lett. **87**, 214501 (2001).
[6] B. Tao, J. Katz, and C. Meneveau, Phys. Fluids **12**, 941 (2000).
[7] This spec is for the Phantom v7s: http://www.visionresearch.com. Many other combination of specs are available, but this camera has one of the highest sustained pixel readout rates available.
[8] S. Ott and J. Mann, J. Fluid Mech. **422**, 207 (2000).
[9] G. A. Voth, A. L. Porta, A. M. Crawford, E. Bodenschatz, C. Ward, and J. Alexander, Rev. Sci. Instrum. **72**, 4348 (2001).
[10] N. T. Ouellette, H. Xu, and E. Bodenschatz, Exp. Fluids **40**, 301 (2006).
[11] http://www.pad2pad.com.
[12] H. Johnson and M. Graham, *High Speed Digital Design* (Prentice-Hall, Englewood Cliffs, NJ, 1993).