Journal of Biomedical Optics

SPIEDigitalLibrary.org/jbo

Dedicated hardware processor and corresponding system-on-chip design for real-time laser speckle imaging

Chao Jiang
Hongyan Zhang
Jia Wang
Yaru Wang
Heng He
Rui Liu
Fangyuan Zhou
Jialiang Deng
Pengcheng Li
Qingming Luo

SPIE

Dedicated hardware processor and corresponding system-on-chip design for real-time laser speckle imaging

Chao Jiang, Hongyan Zhang, Jia Wang, Yaru Wang, Heng He, Rui Liu, Fangyuan Zhou, Jialiang Deng, Pengcheng Li, and Qingming Luo
Huazhong University of Science and Technology, Britton Chance Center for Biomedical Photonics, Wuhan National Laboratory for Optoelectronics, Wuhan 430074, China

Abstract. Laser speckle imaging (LSI) is a noninvasive and full-field optical imaging technique which produces two-dimensional blood flow maps of tissues from the raw laser speckle images captured by a CCD camera without scanning. We present a hardware-friendly algorithm for the real-time processing of laser speckle imaging. The algorithm is developed and optimized specifically for LSI processing in the field programmable gate array (FPGA). Based on this algorithm, we designed a dedicated hardware processor for real-time LSI in FPGA. The pipeline processing scheme and parallel computing architecture are introduced into the design of this LSI hardware processor. When the LSI hardware processor is implemented in the FPGA running at the maximum frequency of 130 MHz, up to 85 raw images with the resolution of 640×480 pixels can be processed per second. Meanwhile, we also present a system on chip (SOC) solution for LSI processing by integrating the CCD controller, memory controller, LSI hardware processor, and LCD display controller into a single FPGA chip. This SOC solution also can be used to produce an application specific integrated circuit for LSI processing. [DOI: 10.1117/1.3651772]

Keywords: biomedical optics; image processing; speckle.

Paper 11384R received Jul. 20, 2011; revised manuscript received Sep. 10, 2011; accepted for publication Sep. 16, 2011; published online Oct. 31, 2011.

1 Introduction

Laser speckle imaging (LSI) is a technique used to produce a two-dimensional blood flow map *in vivo* without invasion and scanning. Its high temporal and spatial resolution enables that it is appropriate for the monitoring of the blood flow of cerebral cortices,[1–8] retina,[9–11] and skin[12,13] in the research. Real-time visualized LSI is also significant in the application of neurosurgery for monitoring the dynamic blood flow. However, it is difficult to achieve a real-time visualized LSI due to the heavy computation burden on the current personal computer platform. Therefore, optimized processing algorithms were developed[14,15] and some devices with powerful computing capability were also employed to build real-time LSI processing platforms. For instances, graphics processing unit (GPU) for parallel computing is employed for achieving a real-time blood flow visualization of high-resolution analysis.[16–18] Besides, a digital signal processor (DSP) is also used to accelerate the processing of LSI.[19]

With the increasing development of field programmable gate array's (FPGA) high integration density and speed level, this type of hardware is also introduced to perform high performance computing such as numerical simulation, imaging processing, etc. FPGA is an integrated circuit designed to be configured by the customer or designer after manufacturing, which can be used to implement any logical functions that an application specific integrated circuit (ASIC) can perform. FPGAs have already been used in processing of holographic,[20–26] optical coherent tomography,[27,28] fluorescence lifetime sensing,[29,30] Monte Carlo simulation,[31] etc. Different from the personal computer, GPU and DSP which implement the algorithm by running corresponding program codes including three standard stages: fetching instruction, translating, and performing, FPGA implements an algorithm by the dedicated designed digital circuit integrated in the FPGA chip. This type of dedicated circuit can be developed by employing hardware description language (HDL) such as VHDL and Verilog HDL.

We have designed a dedicated digital circuit for the real-time processing of laser speckle imaging in FPGA. This LSI hardware processor features computing the data using hardware circuit during the whole processing flow without needing any program code. The pipeline processing scheme and parallel hardware architecture are introduced into the design of this type of LSI hardware processor to further improve the computing performance. Compared with the GPU solution and DSP solution, this hardware-based LSI processor can achieve real-time processing at very low clock frequency and power dissipation. It is quite appropriate to be used to design a portable LSI system with high performance and stability. To our knowledge, no other group has ever reported this solution for LSI processing. Moreover, we also present a system on chip (SOC) solution for LSI processing by integrating the CCD camera controller, memory controller, LSI hardware processor, and LCD display controller into a single FPGA chip. This SOC solution has the potential to be ported to the ASIC platform for producing a

Address all correspondence to: Pengcheng Li, Huazhong University of Science and Technology, Britton Chance Center for Biomedical Photonics, Wuhan National Laboratory for Optoelectronics, Wuhan, 430074, China. Tel: 86-27-87792033; Fax: 86-27-87792034; E-mail: pengchengli@mail.hust.edu.cn.

1083-3668/2011/16(11)/116008/10/$25.00 © 2011 SPIE

low-cost, but high performance, chip for the LSI processing system.

2 Method

2.1 *Algorithms for Processing of LSI*

The processing of LSI can be divided into two steps. The first step is computing the local spatial contrast and the second step is converting the contrast to velocity value. The local spatial contrast c is estimated as the ratio of standard deviation to the mean intensity $\overline{I}$ of pixels within a small $n\times n$ sliding window (n is the length of the square window quantified as pixel) defined as[32]

$$C = \frac{\sigma_I}{\overline{I}} = \frac{\sqrt{\frac{\sum_{i=1}^{N}(I-\overline{I})^2}{N-1}}}{\overline{I}}. \tag{1}$$

In Eq. (1), σ_I is the standard deviation of intensity, i is the index of the pixels within the sliding window, and N is equal to $n\times n$ or the total number of pixels within the sliding window. Moreover, Eq. (1) can be replaced by another equivalent form with sums as

$$C = \frac{\sigma_I}{\overline{I}} = \frac{\sqrt{\frac{N\sum_{i=1}^{N} I_i^2 - \left(\sum_{i=1}^{N} I_i\right)^2}{N(N-1)}}}{\frac{\sum_{i=1}^{N} I_i}{N}}. \tag{2}$$

Because the mean velocity of blood flow is proportional to the camera's exposure time T divided by correlation time τ_c, the relationship between local spatial contrast and T/τ_c is defined as[32]

$$C = \sqrt{[\exp(-2x) - 1 + 2x]/(2x^2)}, \quad x = T/\tau_c. \tag{3}$$

According to Eq. (3), T/τ_c cannot be resolved directly, so the Newton iteration numerical method is usually employed. However, the iteration time is uncertain, so this method is quite time-consuming. Tom et al. proposed a "table" method to resolve this problem.[14] All the values of T/τ_c are previously stored in a table indexed by speckle contrast value with a specific interval. First, the program obtains the rough T/τ_c value with some statistical error according to the measured speckle contrast's closest value in the table. Then the Newton iteration algorithm starts to find the more precise value based on the table's value. This method improves the convergence speed greatly. Beside this, Cheng et al. proposed a simplified LSI analysis method.[11] When the camera's exposure time is further lager than the correlation time, T/τ_c is considered to be equal to $1/C^2$. This method avoids the iteration and enhances the performance greatly. Moreover, this simplified form is more hardware-friendly and easier to be constructed by hardware circuit in FPGA.

2.2 *LSI Algorithm Implemented in FPGA and the Design of the LSI Hardware Processor*

In this study, the simplified speckle imaging equation $1/C^2$ is used to estimate the relative velocity of blood flow. Equation (2) can be converted to be the blood flow equation as

$$\frac{1}{c^2} = \frac{N-1}{N} \times \frac{\left(\sum_{i=1}^{N} I_i\right)^2}{N\sum_{i=1}^{N} I_i^2 - \left(\sum_{i=1}^{N} I_i\right)^2}. \tag{4}$$

LSI only presents the relative flow map, so getting rid of the constant *(N-1) / N* of Eq. (4) has no effect on quantifying this type of relative flow map. A more simplified relative velocity value equation without the constant *(N-1) / N* is more hardware-friendly in FPGA defined as

$$v = \frac{\left(\sum_{i=1}^{N} I_i\right)^2}{N\sum_{i=1}^{N} I_i^2 - \left(\sum_{i=1}^{N} I_i\right)^2}, \tag{5}$$

where v stands for the new relative velocity value.

Our hardware circuit implemented for the dedicated LSI processor is based on Eq. (5). In this design, the pipeline processing scheme and parallel hardware architecture are employed to further enhance the computing performance by enlarging the data throughout. As illustrated in Fig. 1, the circuit of the LSI processor mainly consists of a 3-stage pipeline and 1 divider array. The modules of stage 1 to stage 3 in the pipeline are corresponding to calculate the numerator and denominator of Eq. (5) which are output by the module of stage 3 in a pipeline before the divider array. All the modules in the pipeline are running separately at the same time. For example, when module (c) of stage 2 are calculating the square value of the raw data accumulation out from module (a) of stage 1, the module (a) is calculating the new accumulation of pixels in the next adjacent sliding window. When the size of the sliding window is 5×5, each module in the pipeline performs a part task of a sliding window in five clock cycles. That means for each 5 cycles a pair of numerator and denominator of a sliding window will be output from the module of stage 3 in the pipeline waiting for a dividing operation. The divider unit of this LSI hardware processor is designed using the scheme of the shift-and-subtraction method. So completing a dividing operation costs more than five clock cycles. For instance, when the raw data width is 12 bits in binary and the relative velocity value is at the precision of 0.01 in decimal, one dividing operation costs 45 clock cycles. Therefore, in order to keep pace with the output speed for the numerator and denominator from the module of stage 3 in the pipeline, a divider array containing 9 hardware dividers are integrated into the LSI processor circuit. This parallel architecture consisting of multiple hardware dividers makes sure that each pair of numerator and denominator can be processed in time. For each five clock cycles, a relative velocity value will be output from one divider of the divider array. Besides, a gatherer module follows the divider array that gathers the relative velocity values from the dividers and outputs these values in chronological order.

The method of reading the pixels of raw image into the pipeline is illustrated in Fig. 2. At the rising edge of each clock cycle, a pixel value of a sliding window is read into the pipeline from raw image memory. The reading order is one pixel after one pixel in the vertical direction in a sliding window. We define that the sliding window's 5 pixels in the vertical direction make up a vertical strip. The two adjacent sliding windows have the same four vertical strips, so that only a new vertical strip needs

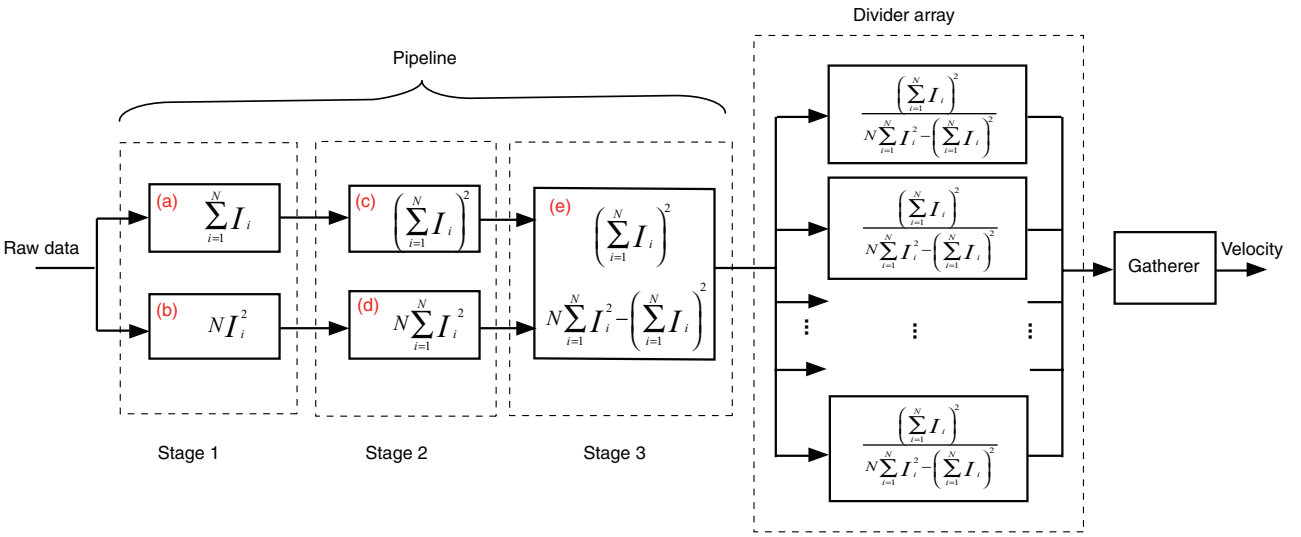


Fig. 1 The circuit architecture of LSI hardware processor mainly consists of a 3-stage pipeline and a divider array. Module (a) and module (b) of stage 1 in pipeline calculate the accumulation and the square multiplied with constant N of raw data within a sliding window separately. Module (c) and module (d) of stage 2 in pipeline calculate the square of the data out from module (a) and the accumulation of the data out from module (b) separately. Module (e) calculates the subtraction of the data out from module (c) and module (d). Besides, module (e) caches the accumulation of module (c) waiting for outputting with the subtraction value at the same time as numerator and denominator of Eq. (5). Divider array contains multiple divides for keeping pace with the output speed of module (e) in pipeline. Gatherer module gathers the relative velocity value out from dividers in divider array and outputs these values in chronological order.

to be read into the pipeline when computing the new adjacent is possible. Because module (a) and module (d) of the LSI hardware processor's circuit can cache the temporary results of the strips from 2 to 5 of the last sliding window and these strips are the first, the second, the third, and the fourth strip of the new adjacent sliding window at the same time. This scheme allows that computing a sliding window only needs to read the fifth vertical strip of the current sliding window into the pipeline, as the former four vertical strips are cached in the circuit when computing the last sliding window. Of course, there is no adjacent sliding window for the first sliding window in each line of the raw image, so the first one must read all five vertical strips while the following ones only need to read one vertical strip. With this method, computing a raw image with the resolution of $M \times N$ pixels needs $5 \times (M\text{-}4) \times N + 55$ clock cycles when the size of the sliding window is 5×5, where M is the number of rows of the raw image and N is the number of columns of the raw image. The number 55 is the latency for the last sliding window from having been read into the pipeline to outputting the relative velocity value in the hardware divider of the divider array.

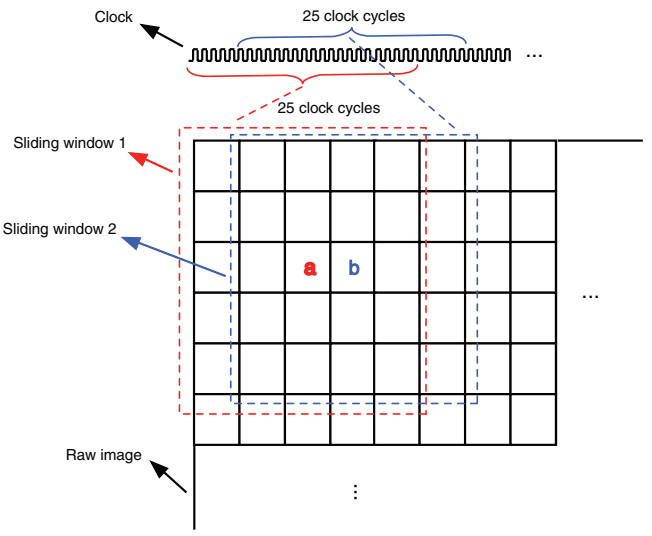


Fig. 2 The illustration describes the method of reading the raw data into the pipeline of the LSI hardware processor. The size of sliding window is 5×5. The small black rectangles stand for pixels of raw image. The colored rectangles with a dashed line stand for two adjacent sliding windows. Symbols a and b mark the center positions of sliding window 1 and sliding window 2. Clock sequence diagrams are also presented to tell how many cycles for reading each sliding widows' pixels into the pipeline.

2.3 *SOC Solution for LSI System*

Based on the LSI hardware processor described above, we further designed a SOC solution for the LSI system by integrating the CCD camera controller, memory controller, LSI hardware processor, and LCD display controller in a single FPGA chip. This SOC-based LSI system can achieve real-time data acquisition, processing, and display. The main architecture of this SOC design is illustrated in Fig. 3.

2.3.1 *CCD camera controller*

The CCD camera controller controls the work mode of the CCD camera and receives the image data captured by a CCD camera. Currently, the CCD camera in this design is OB-1280 (OVED, Shenzhen China) that only provides digital BT-656 video format (25 frames/s) converted from PAL format by a video analog-to-digital converter chip, so this controller module also needs to decode the BT-656 format image to the raw image data. A frame

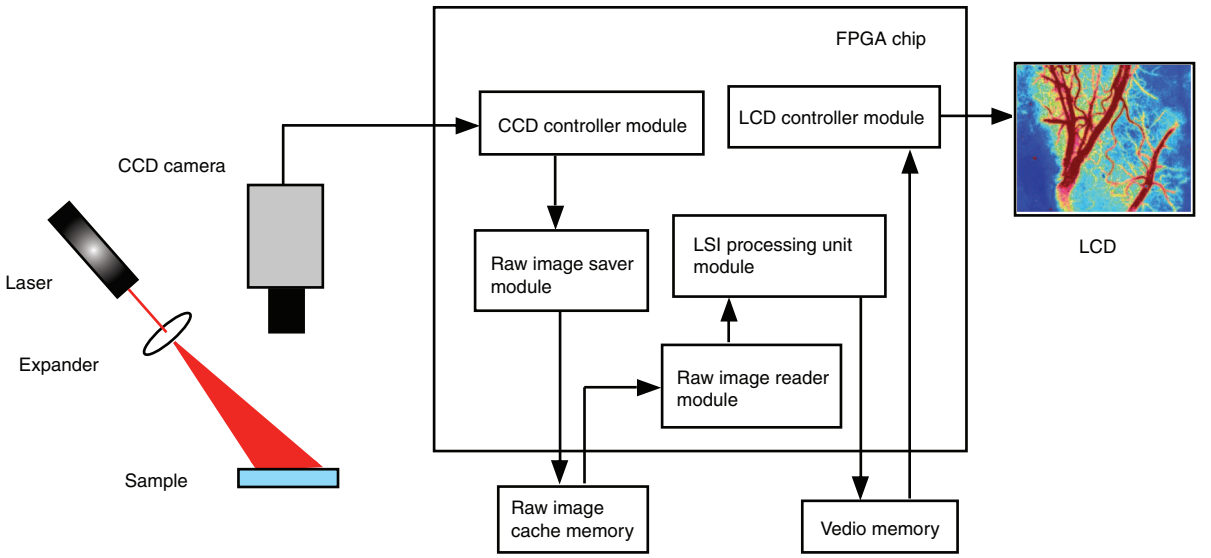


Fig. 3 The main architecture of the SOC solution for the LSI system. The direction of the arrow stands for the direction of the data that flows in the main modules.

of a video image organized in BT656 format contains the pixel data and some other data for a special mark. The pixel data complies with the 4:2:2 encoding parameter using the YCbCr color space. The marked data is used to locate the start position of each line and tells the odd or even field of the video image. As illustrated in Fig. 4, decoding the BT656 video to raw image format is mainly extracting the *Y* component that is the gray scale value of the video image. Besides, the camera controller should also cooperate with the saver module to relocate the pixel lines, for the pixel data values of adjacent lines are located in the even and odd fields separately in the BT656 frame. The current camera's resolution is 752×582 pixels. The camera controller module clips the original video frame to a sub-image with the resolution of 640×480 pixels in order to display well in a LCD with the same resolution employed in this SOC system. The raw image composed of the *Y* component can be used to perform LSI analysis. In future work, we will modify the function of this CCD camera controller module and directly control a high speed analog-to-digital converter chip to acquire image data from a high frame rate CCD image sensor chip rather than a commercial camera.

2.3.2 *Reader module and the saver module*

The memory controller consists of a saver module and a reader module. The saver module caches the raw image data out from a CCD controller module into the image memory and the reader module reads the raw image data into the LSI processor. The raw image cache memory is shared by the saver module and the reader module. To enlarge the data throughout and simplify the memory operating interface, the static random access memory (SRAM) chip is highly recommended. Of course, if the FPGA chip has enough memory resource, using the memory in the FPGA chip can further simplify the memory interface. In our current design, we use an independent SRAM chip as raw image cache, as the FPGA chip we use does not own a lot of memory resource.

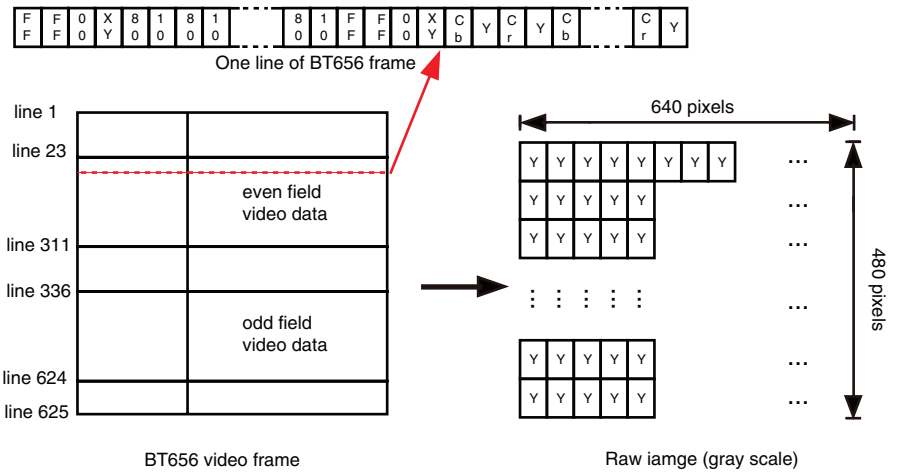


Fig. 4 Extracting the *Y* components (gray scale value) from BT656 video frame to make up a raw image that can be used to perform LSI processing.

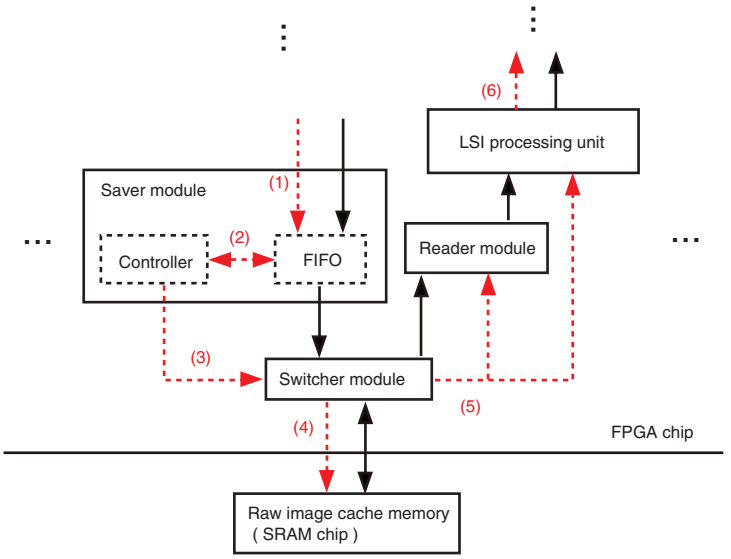


Fig. 5 The architecture for sharing the raw image cache memory by saver module and reader module. The red dashed line arrows stand for the handshaking signal or control signal. The black solid line arrows stand for data signal. The direction of the arrow stands for the transferring direction of the handshaking signal, control signal, and data. The red number in the bracket is used to mark the nearby handshaking signal or control signal.

The scheme of sharing the raw image cache memory is illustrated in Fig. 5. The direction of the red dashed line arrow stands for the transferring direction of handshaking signals or control signals, while the direction of the black solid line arrow stands for the transferring direction of data between modules. The saver module consists of a first in, first out (FIFO) and a small controller sub-module. Usually a FIFO is a small fast memory with one writing operation port and one reading operation port. It allows the data that comes in first to go away first. This feature of the FIFO can be used for buffering data flow between modules that runs at a different speed such as the camera controller and the switcher module in Fig. 5. The camera controller writes the raw image data to the FIFO continuously through the writing port. The control signal (1) from the camera controller module (not illustrated in Fig. 5) is used for controlling the writing operation of the FIFO. A controller sub-module in the saver module is able to continuously monitor how much data has been in the FIFO and controls the reading operation module of FIFO through control signal (2). When more than a line of pixel data of the raw image has been buffered in the FIFO, the controller sub-module will start to transfer a line of pixel data to the raw image cache memory through the switcher module by controlling the reading operation port of the FIFO. One issue that deserves to be mentioned is that the controller sub-module must notify the switcher module to release the data channel connecting the raw image cache memory and reader module through signal (3) before starting to transfer the data in the FIFO to the raw image cache memory. Of course, the switcher module will notify the reader module and LSI processing unit module to stop working and protect the current status through signal (5) once it receives the notification of saver module. The moment the data channel between the raw image cache memory and reader module is released, the data channel between saver module and the raw image cache memory is built at once. And a line of pixel data is transferred into the raw image cache memory in a burst mode during a short time slice. When transferring is finished, the switcher module disconnects the data channel between saver module and the raw image cache memory to rebuild the data channel between the reader module and the raw image cache memory. And the switch module also notifies that the reader module and LSI processing unit can work again through control signal (5).

Another issue that must be noted is that not every moment can be made use of by the reader module to read raw data into a LSI processing unit when the data channel between saver module and the raw image cache memory is disconnected, because the reader module must wait until at least a complete valid sliding window's raw data has been already stored in the raw image cache memory. After this, data are buffered in the raw image cache, the switcher module notifies the reader module and LSI processing unit module to start work. This design allows that the part of the data of a frame that already comes into raw image cache memory can be processed at once, while the subsequent part of the data of this frame may be still buffered in the FIFO or even has not been read into the LCD controller module yet. In other words, during the process of acquiring a frame of raw image, the LSI processing of the raw data part of this same frame that comes into raw image cache memory early is performed at the same time. Actually, when a frame of raw image is all transferred into the raw image cache memory, the LSI processing of this frame of raw image is also nearly completed with the current frame rate of the CCD camera (25 frames/s). Because the LSI processing is much faster than the raw image capturing, the reader module and LSI hardware processing unit module often have to stop to wait for the subsequent data due to the low frame rate of this CCD camera.

2.3.3 *LSI processing unit module and the LCD controller module*

The LSI processing unit module containing a LSI hardware processor converts the raw data into the relative velocity value based

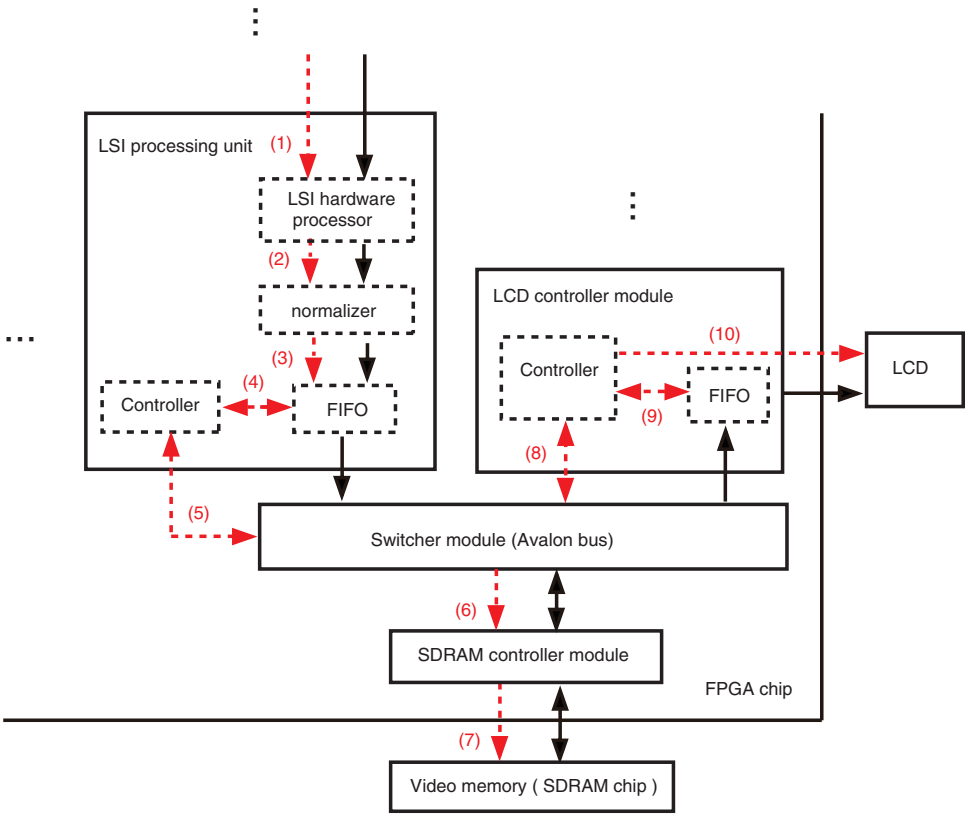


Fig. 6 The architecture of operating the video memory by a LSI processing unit module and LCD controller module. The red dashed line arrows stand for the handshaking signal or control signal. The black solid line arrows stand for the data signal. The direction of the arrow stands for the transferring direction of the handshaking signal, control signal, and data. The red number in the bracket is used to mark the nearby handshaking signal or control signal.

on the algorithm described in Sec. 2.2 of this paper. The main architecture of the LSI processing unit module is illustrated in Fig. 6. The control signal (1) is the same one as control signal (5) in Fig. 5 for determining whether the LSI hardware processor runs or stops. The raw data in the front end of the LSI hardware processor comes from the reader module. The reader module feeds the LSI hardware processor with the raw data with the method illustrated in Fig. 2. Because the logic for transferring data to video memory is much simpler than that of the saver module (it does not need to do the work of converting the separated odd field and even field video data into a regular raw image as the saver module), we put it together with the LSI hardware processor in the LSI processing unit module. The relative velocity values from the LSI processor are normalized into color index values before transferring to video memory for display. So a normalizing logic sub-module is placed after the back end of the LSI hardware processor. It is controlled by the back end logic of the LSI hardware processor through signal (2). Each color index value is transferred into FIFO through the writing port controlled by the back-end logic of the normalizer module through control signal (3). The controller sub-module of the LSI processing unit module is always monitoring how much data has been in the FIFO through signal (4). When more than a line of pixels (color index value) is already in the FIFO, the controller sub-module starts to apply data channel from the switcher module through a handshake signal (5) to transfer the pixels to video memory. However, the controller sub-module cannot always get the data channel immediately, since the LCD controller module may be transferring data at the same moment. In this condition, the controller sub-module in the LSI processing unit module must wait until the LCD controller releases the data channel. So the depth of the FIFO in the LSI processing unit must be large enough to avoid overflowing and losing data.

Different from the switcher module completely designed by ourselves for severing saver module and reader module, the switcher module connecting video memory to LSI hardware processing unit module and LCD controller module is Altera Avalon bus module.[33] The interfaces of the LSI processing unit module and LCD controller module for connecting the switcher module must comply with the interface standard of the Avalon bus. Actually, the LSI processing unit module and LCD controller module act as master nodes in the Avalon bus, which can launch data transfer actively. The direct memory access logic circuit in the controller sub-modules of the two modules can directly access the video memory through the Avalon bus. The Avalon bus module acts as the arbitrator to allocate data channel to video memory for the LSI processing unit module and LCD controller module. The video memory is a synchronous dynamic random access memory (SDRAM) chip and it cannot be connected to the interface of an Avalon bus directly, so a SDRAM controller module provided as intellectual property (IP) core by Altera is used to bridge the SDRAM chip and the Avalon bus. Signal (6) is used to control the SDRAM controller module from the Avalon bus module and signal (7) is used to control

the SDRAM chip from the SDRAM control module. Moreover, a NIOS II soft CPU core must be added when using the Avalon bus in the Altera Quartus II development (not illustrated in Fig. 6). The soft core designed by HDL can be reconfigured to be integrated in the FPGA chip with our designed logic circuits. In the current design, this CPU nearly does nothing except for very small initialization work of some modules when the system starts, because the raw data acquiring, LSI processing, and display are completely done by the hardware modules. Maybe we can design more user-friendly interfaces using this CPU in a future work.

The LCD controller module reads the blood flow image in the video memory and displays it in a LCD. It mainly contains a controller sub-module and a FIFO. The controller sub-module mainly has two functions. One is monitoring the current quantity of rest pixels in the FIFO through signal (9). When the quantity of the pixels is lower than a specified threshold, the controller sub-module in the LCD controller module starts to apply for data channel from switcher module to read a new line of pixels through signal (8). The other is reading pixel data out from FIFO and displaying them in the LCD in a line-by-line scanning way. Make sure that the refresh rate of the scanning is not lower than 60 Hz in order to avoid screen flicker. The detailed scanning regulation can refer an open LCD controller IP core project.[34]

3 Result

We test the processing performance of our LSI hardware processor. In the Quartus II development environment and a low-cost Cyclone II FPGA chip from Altera, the maximum work frequency of the LSI hardware processor reaches about 130 MHz. That means this type of hardware processor can process around 85 frames of raw image with resolution of 640×480 per second when the size of the sliding window is 5×5. For another often-used sliding window size 7×7, the processing performance will be comprised to be around 60 frames/s for the raw image with the same size of 640×480 pixels. If higher speed level FPGA chips are used, such as the ones of the Stratix family from Altera, more performance enhancement could be expected.

A more detailed performance benchmark list is presented in Table 1. The performance of the CPU version is also listed as a comparison. The program written in C++ language running in the CPU implements the same algorithm as the FPGA version described in Sec. 2.2 of this paper. The C++ program using single float precision was developed in the Microsoft Visual Studio 2005 in Windows XP operating system and the CPU is Intel E7300 (dual core) running at the frequency of 2.67 GHz (only one core is used). Considering that the maximum work frequency of the LSI hardware processor varies in a different speed level FPGA chip, the precise clock cycles of processing a specific size raw image are presented, because the clock cycles will not vary for different FPGA chips when the size of the raw image is fixed, while the time consumption will varies in different FPGA chips. Therefore, the number of clock cycles is more important when evaluating the processing efficiency of the LSI hardware processor. However, the time consumptions are also presented in order to compare the performance with the personal computer's CPU. For a fixed size raw image, the higher the maximum work frequency is, the less time consuming. For example, the maximum work frequency of our LSI hardware processor can be 130 MHz in an Altera Cyclone II FPGA chip EP2C35F484C6 and 192 MHz in an Altera Stratix II FPGA chip EP2S30F484C3. The time consumption values listed in the table are computed in the case that the LSI hardware processor running at the frequency of 130MHZ in a Cyclone II FPGA chip (EP2C35F484C6) in current design. Sliding window sizes of 5×5 and 7×7 are used to record the performance parameters in Table 1.

We also compared the computing precision of LSI hardware processor with that of the C++ program using single float precision on a personal computer. Figure 7 presents the blood flow maps of mouse cerebral cortex that are processed in two different platforms. The relative difference map is also presented. It is demonstrated that the relative difference is roughly lower than 0.001. This is because the relative velocity value out of the

Table 1 FPGA versus CPU processing time consumptions for various image sizes.

	5×5 window size				7×7 window size			
Image size (pixels)	FPGA clocks	FPGA time/ms	CPU time/ms	Speed-up	FPGA clocks	FPGA time/ms	CPU time/ms	Speed-up
320×240	377655	2.9	7.7	2.65	524219	4.0	10.3	2.58
480×320	758455	5.8	15.6	2.69	1055099	8.1	20.8	2.57
640×480	1523255	11.7	31.5	2.69	2123579	16.3	42.1	2.58
800×600	2384055	18.3	49.3	2.69	3326459	25.6	66.2	2.59
1024×768	3911735	30.1	81.7	2.71	5462075	42.0	109.5	2.61
1280×1024	6528055	50.2	137.5	2.74	9121339	70.2	184.6	2.63
1392×1040	7210615	55.5	150.4	2.71	10075355	77.5	202.4	2.61
1600×1200	9568055	73.6	201.3	2.74	13372859	102.9	268.4	2.69

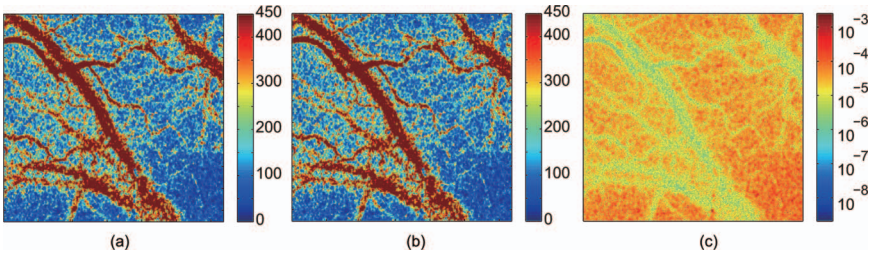
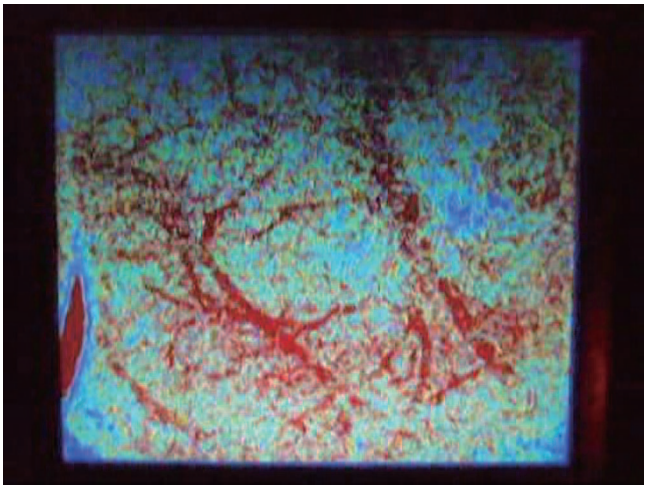


Fig. 7 The LSI blood flow map of the mouse cerebral cortex processed in a different platform and comparison: (a) The LSI blood flow map processed by a LSI hardware processor in FPGA; (b) the LSI blood flow map processed by a program written in C++ running on a personal computer with single precision; (c) the relative difference map in logarithm format between image (a) and image (b).

LSI hardware processor is at the precision of 0.01 in decimal. Actually, higher precision can be realized in our LSI hardware processor by adding a more logical resource to improve the computing precision of the hardware divider. But this precision is enough for practical use, as finally we usually normalize these float-type velocity values into integer color index for displaying in the monitor. One issue that needs to be noted is that the raw image of this blood map is placed in the raw image cache memory artificially rather than captured by the CCD camera equipped in the SOC design. It is convenient to compare the precision with personal computer's CPU using the same raw image. The size of this raw image is 640×480 pixels and the pixel's data width is 12 bits. The sliding window size for processing this raw image is 5×5.

A demonstration of real-time monitoring the rat's cerebral cortex blood flow is presented in Video 1 using the FPGA-based LSI SOC system. A small piece of skull was removed in the head of the rat to make a window to monitor. During the whole process, the rat was in narcosis. The camera's frame rate is 25 frames/s with the resolution of 640×480 pixels and the data width of raw image pixel is 8 bits. Each frame of the blood flow map in the LCD screen comes from the result of real-time processing a single frame raw image captured by the CCD camera. So the display rate in the LCD screen is also 25 frames/s. Pseudocolor is used to show the big or small of the blood flow. The redder the area is, the bigger the blood flow is in the corresponding area. The bluer the area is, the smaller the blood flow is. In the video picture, we can see that: the cortex areas with obvious blood vessels are reddest; the cortex areas with unobvious blood capillary are interposed; the areas without blood vessels are nearly blue. The whole video of the LCD screen was recorded by a commercial entertainment camera.



Video 1 Single-frame excerpt from the video of real-time monitoring of the cerebral cortex blood flow of a rat (AVI, 4.9 MB).
[URL: http://dx.doi.org/10.1117/1.3651772.1]

4 Discussion

In the current years, several solutions for accelerating the processing of LSI are reported.[14,16,17,19] The GPU solution is quite popular. We mainly made some comparisons between our FPGA hardware solution and the GPU solution[17] of Owen et al. including the easiness of system integration, processing speed, and some other trade-offs.

Generally speaking, the GPU solution is easier to be integrated into the whole system in contrast to the FPGA solution, because the chip-level circuit and the board-level circuit have already been designed by corresponding vendors. Besides, the software for supporting the running environment is also provided by the GPU vendors. After installing the commercial GPU board in the personal computer through PCI-E interface and corresponding supporting library files, the main work is focusing on developing the specified application. Compared with the GPU solution that the hardware is already prepared, the FPGA solution needs to do a lot of basic work. We must design the FPGA circuit board ourselves by adding the CCD camera interface, memory chips, LCD display interface, and some other input–output interfaces around the FPGA chip. After the hardware platform is built, the subsequent work is to develop the specified logic for LSI processing. Besides the main work of the LSI hardware processor, some other logic must be developed for assisting the LSI hardware processor including the camera controller module, switcher module, etc. So the FPGA solution is more troublesome in the perspective of the developers of LSI.

In the paper of Owen et al., they presented a performance benchmark table in which the time consumptions of GPU-based LSI algorithm and the speed-up in contrast to CPU version are listed. Processing a raw image with a typical size of 640×480 pixels takes 5.3 ms on a low-cost GPU (Nvidia Geforce 8800GTS) when the sliding window size is 5×5. This time includes transferring the raw image onto the GPU, executing processing kernels, and transferring the processed image back to main memory. The two transferring operations are necessary for each frame on the GPU, so these time consumptions

are also included in the whole time consumption. In the FPGA platform, the camera controller module directly delivers the raw image data into the raw image cache memory in which the LSI processing unit module can directly access the raw data through a reader module. The LSI processing and the raw image capturing are nearly running in parallel. Moreover, the processed relative velocity values are transferred to video memory directly. Therefore, the time consumption of the FPGA-based LSI solution can nearly only include the processing time. Processing a raw image with a typical size of 640×480 pixels takes 11.7 ms in a Cyclone II FPGA chip we use when the sliding window size is 5×5. Generally speaking, the GPU LSI solution can obtain higher performance than FPGA solution due to the higher peak flops.

Although the system integration is more difficult than that of the GPU-based one and the performance is also lower, the FPGA-based LSI solution also has a lot of superiority.

Firstly, it is easier to realize a real time on-line processing than GPU when a high speed camera is used such as the one with the speed of around 100 frames/s and the typical resolution of 640×480 pixels (the width of pixel value is 12 bits). That requires transferring around 60 Mbytes/s to the personal computer from the camera in the GPU solution. The transferring of such a large amount of data is very difficult to realize by USB or Ethernet method. Even though a 1000-base Ethernet is used whose theoretical maximum speed is 125 Mbytes/s, the personal computer and the embedded computer inside the camera can hardly reach or maintain the speed of 60 Mbytes/s due to huge computing burden of running network protocol stack. Therefore, although the GPU can process hundreds of raw image per second, there is hardly a corresponding camera that can transfer such a great amount of data to a personal computer. But in the FPGA platform, the camera controller module's logical circuit can directly acquire raw image data from an image sensor by controlling an AD chip which converts the analogy signal of CCD image sensor to digital format. This data transmission method is able to maintain a high speed frame speed such as 60 M bytes/s. If a better speed level FPGA chip than Cyclone II one is used, it is able to realize a high speed real-time LSI processing system with the data throughout of around 100 frames 640×480 raw image using the SOC solution described above.

Secondly, the FPGA LSI solution has the potential to expand the number of LSI hardware processing units in order to further improve the processing capability. To achieve this increased solution, some extra logic circuits for scheduling these multiple LSI hardware processing unit need to be added. Of course, a FPGA chip that has adequate logic resource for this solution should be selected. But there is little flexibility on hardware of GPU, because the whole circuit architecture can no longer be modified by the user after it is manufactured. Moreover, multiple CCD cameras can be organized to work together in controlling a single FPGA chip by further modifying the current solution due to the reconfigurable future of FPGA. This solution will be very useful when multiple separated areas are needed to be monitored at the same time. Therefore, FPGA solution's flexibility allows us to design our dedicated hardware architecture.

Finally, the reason why FPGA LSI solutions performance is lower than that of the GPU version is mainly because the maximum work frequency is very low in contrast to the GPU's system frequency. Most FPGA generally runs at the frequency of no more than 300 MHz, while GPU runs at the frequency of around 1 GHz. Although the low work frequency leads to some performance loss, it also brings low-power dissipation. Besides, the whole size of the FPGA LSI system can be designed much smaller than that of the GPU version. Therefore, the FPGA LSI solution is quite appropriate to be used to design a compact handheld LSI system with high performance and stability due to the small size and low dissipation.

5 Conclusion

In this paper, we present a dedicated digital circuit for the real-time processing of laser speckle imaging in FPGA and a SOC solution-based LSI system that also has the potential to be ported to ASIC for producing a low-cost but high performance LSI processing integrated chip. Our next plan is to further improve this LSI hardware processor's performance by optimizing the architecture and design a faster CCD camera module for the SOC-based LSI system.

Acknowledgments

This work is supported by the program for New Century Excellent Talents in University (Grant No. NCET-08-0213), Science Fund for Creative Research Group of China (Grant No.61121004), the National Natural Science Foundation of China (Grant Nos. 30970964, 30800339, 30801482, and 30800313), and the Ph.D. Programs Foundation of Ministry of Education of China (Grant No. 20090142110054).

References

1. A. K. Dunn, H. Bolay, M. A. Moskowitz, and D. A. Boas, "Dynamic imaging of cerebral blood flow using laser speckle," *J. Cereb. Blood Flow Metab.* **21**(3), 195–201 (2001).
2. M. Li, P. Miao, J. Yu, Y. Qiu, Y. Zhu, and S. Tong, "Influences of hypothermia on the cortical blood supply by laser speckle imaging," *IEEE Trans. Neur. Sys. Rehab. Eng.* **17**(2), 128–134 (2009).
3. P. Li, S. Ni, L. Zhang, S. Zeng, and Q. Luo, "Imaging cerebral blood flow through the intact rat skull with temporal laser speckle imaging," *Opt. Lett.* **31**, 1824–1826 (2006).
4. C. Ayata, A. K. Dunn, Y. Gursoy-Zdemir, Z. Huang, D. A. Boas, and M. A. Moskowitz, "Laser speckle flowmetry for the study of cerebrovascular physiology in normal and ischemic mouse cortex," *J.Cereb. Blood Flow Metab.* **24**(7), 744–755 (2004).
5. A. J. Strong, E. L. Bezzina, P. J. B. Anderson, M. G. Boutelle, S. E. Hopwood, and A. K. Dunn, "Evaluation of laser speckle flowmetry for imaging cortical perfusion in experimental stroke studies: quantitation of perfusion and detection of peri-infarct depolarisations," *J. Cereb. Blood Flow Metab.* **26**(5), 645–653 (2006).
6. A. J. Strong, P. J. Anderson, H. R. Watts, D. J. Virley, A. Lloyd, E. A. Irving, T. Nagafuji, M. Ninomiya, H. Nakamura, and A. K. Dunn, "Peri-infarct depolarizations lead to loss of perfusion in ischaemic gyrencephalic cerebral cortex," *Brain* **130**(4), 995–1008 (2007).
7. X. Sun, P. Li, W. Luo, S. Chen, N. Feng, J. Wang, and L. Qingming, "Investigating the effects of dimethylsulfoxide on hemodynamics during cortical spreading depression by combining laser speckle imaging with optical intrinsic signal imaging," *Lasers Surg. Med.* **42**, 649–655 (2010).
8. X. Sun, Y. Wang, S. Chen, W. Luo, P. Li, and Q. Luo, "Simultaneous monitoring of intracellular pH changes and hemodynamic response during cortical spreading depression by fluorescence-corrected multimodal optical imaging," *NeuroImage* **57**(3), 873–884 (2011).
9. J. D. Briers and A. F. Fercher, "Retinal blood-flow visualization by means of laser speckle photography," *Invest. Ophthalmol. Vis. Sci.* **22**(2), 255–259 (1982).

10. H. Cheng, Y. Yan, and T. Q. Duong, "Temporal statistical analysis of laser speckle images and its application to retinal blood-flow imaging," *Opt. Express* **16**(14), 10214–10219 (2008).
11. H. Cheng and T. Q. Duong, "Simplified laser-speckle-imaging analysis method and its application to retinal blood flow imaging," *Opt. Lett.* **32**(15), 2188–2190 (2007).
12. Y. C. Huang, N. Tran, P. R. Shumaker, K. Kelly, E. V. Ross, J. S. Nelson, and B. Choi, "Blood flow dynamics after laser therapy of port wine stain birthmarks," *Lasers Surg. Med.* **41**(8), 563–571 (2009).
13. B. Choi, W. Jia, J. Channual, K. M. Kelly, and J. Lotfi, "The importance of long-term monitoring to evaluate the microvascular response to light-based therapies," *J. Invest. Dermatol.* **128**(2), 485–488 (2008).
14. W. J. Tom, A. Ponticorvo, and A. K. Dunn, "Efficient processing of laser speckle contrast images," *IEEE Trans. Med. Imaging* **27**(12), 1728–1738 (2008).
15. T. M. Le, J. S. Paul, H. Al-Nashash, A. Tan, A. R. Luft, F. S. Sheu, and S. H. Ong, "New insights into image processing of cortical blood flow monitors using laser speckle imaging," *IEEE Trans. Med. Imaging* **26**(6), 833–842 (2007).
16. S. Liu, P. Li, and Q. Luo, "Fast blood flow visualization of high-resolution laser speckle imaging data using graphics processing unit," *Opt. Express* **16**, 14321–14329 (2008).
17. O. Yang, D. Cuccia, and B. Choi, "Real-time blood flow visualization using the graphics processing unit,"*J. Biomed. Opt.* **16**(1), 016009 (2011).
18. C. Jiang, P. Li, and Q. Luo, "High speed parallel processing of biomedical optics data with PC graphic hardware," in *Proceedings of SPIE-OSA-IEEE Asia Communications and Photonics*, SPIE-OSA-IEEE, Shanghai (2009).
19. X. Tang, N. Feng, X. Sun, P. Li, and Q. Luo, "Portable laser speckle perfusion imaging system based on digital signal processor," *Rev. Sci. Instrum.* **81**, 125110 (2010).
20. S.-I. Satake, G. Sorimachi, N. Masuda, and I. Tomoyoshi, "Special-purpose computer for Particle Image Velocimetry," *Comp. Phys. Commun.* **182**, 1178–1182 (2011).
21. T. Ito, N. Masuda, K. Yoshimura, A. Shiraki, T. Shimobaba, and T. Sugie, "Special-purpose computer HORN-5 for a realtime electroholography," *Opt. Express* **13**(6), 1923–1932 (2005).
22. Y. Abe, N. Masuda, H. Wakabayashi, Y. Kazo, T. Ito, S.-I. Satake, T. Kunugi, and K. Sato, "Special purpose computer system for flow visualization using holography technology," *Opt. Express* **16**(11), 7686–7692 (2008).
23. Y. Ichihashi, N. Masuda, M. Tsuge, H. Nakayama, A. Shiraki, T. Shimobaba, and T. Ito, "One-unit system to reconstruct a 3-D movie at a video-rate via electroholography," *Opt. Express* **17**(22), 19691–19697 (2009).
24. Y. Ichihashi, H. Nakayama, T. Ito, N. Masuda, T. Shimobaba, A. Shiraki, and T. Sugie, "HORN-6 special-purpose clustered computing system for electroholography," *Opt. Express* **17**(16), 13895–13903 (2009).
25. N. Masuda, T. Ito, K. Kayama, H. Kono, S.-I. Satake, T. Kunugi, and K. Sato, "Special purpose computer for digital holographic particle tracking velocimetry," *Opt. Express* **14**(2), 587–592 (2006).
26. Y.-H. Seo, H.-J. Choi, J.-S. Yoo, and D.-W. Kim, "Cell-based hardware architecture for full-parallel generation algorithm of digital holograms," *Opt. Express* **19**(9), 8750–8761 (2011).
27. A. E. Desjardins, B. J. Vakoc, M. J. Suter, S.-H. Yun, G. J. Tearney, and B. E. Bouma, "Real-time FPGA processing for high-speed optical frequency domain imaging," *IEEE Trans. Med. Imaging* **28**(9), 1468–1472 (2009).
28. T. E. Ustun, N. V. Iftimia, R. D. Ferguson, and D. X. Hammera, "Real-time processing for Fourier domain optical coherence tomography using a field programmable gate array," *Rev. Sci. Instrum.* **79**, 114301 (2008).
29. D.-U. Li, E. Bonnist, D. Renshaw, and R. Henderson, "On-chip, time-correlated, fluorescence lifetime extraction algorithms and error analysis," *J. Opt. Soc. Am. A* **25**(5), 1190–1198 (2008).
30. D.-U. Li, B. Rae, R. Andrews, J. Arlt, and R. Henderson, "Hardware implementation algorithm and error analysis of high-speed fluorescence lifetime sensing systems using center-of-mass method," *J. Biomed. Opt.* **15**(1), 017006 (2010).
31. W. C. Y. Lo, K. Redmond, J. Luu, P. Chow, J. Rose, and L. Lilge, "Hardware acceleration of a Monte Carlo simulation for photodynamic therapy treatment planning," *J. Biomed. Opt.* **14**(1), 014019 (2009).
32. J. W. Goodman, *Statistical Optics*, Wiley-Interscience, New York (1985).
33. "Avalon Interface Specifications," Altera (2011).
34. "VGA/LCD Controller," http://opencores.org/project,vga_lcd.