Kincses Z, Orzó L, Nagy Z, Mező Gy, Szolgay P High-speed, SAD based wavefront sensor architecture implementation on FPGA JOURNAL OF SIGNAL PROCESSING SYSTEMS 64:(3) pp. 279-290. (2011)

DOI: 10.1007/s11265-010-0487-4

High-speed, SAD based wavefront sensor architecture implementation on FPGA

Zoltán Kincses
Department of Electrical Engineering
and Information Systems,
University of Pannonia,
Veszprém, Hungary
kincsesz@vision.vein.hu

László Orzó[1], Zoltán Nagy[1], György Mező[2],
Péter Szolgay[1,3]
[1] Cellular Sensory and Wave Computing Laboratory
Computer and Automation Institute, HAS,
Budapest, Hungary
orzo@sztaki.hu,nagyz@sztaki.hu, szolgay@sztaki.hu
[2] Heliophysical Observatory, Debrecen, Hungary
gmezo@puma.unideb.hu
[3]also affiliated to: Faculty of Information
Technology, Pazmany Peter Catholic University,
Budapest, Hungary

***Abstract*— Wavefront aberrations caused by turbulent or rapidly changing media can considerably degrade the performance of an imaging system. To dynamically compensate these wavefront distortions adaptive optics is applied. We developed an affordable adaptive optic system which combines CMOS sensor and Liquid Crystal on Silicon (LCOS) display technology with the Field Programmable Gate Arrays (FPGA) devices parallel computing capabilities. A high-speed, accurate wavefront sensor is an elemental part of an adaptive optic system. In the paper, an efficient FPGA implementation of the Sum of Absolute Differences (SAD) algorithm, which accomplishes the correlation-based wavefront sensing, is introduced. This architecture was implemented on a Spartan-3 FPGA which is capable of real-time (>500 fps) measuring the incoming wavefront.**

Keywords: SAD, FPGA, wavefront sensor, real-time image processing

1 INTRODUCTION

The rapidly changing turbulent media distort the wavefront of the propagating light wave, thus considerably deteriorating the imaging system's performance. Wavefront sensors measure these wavefront distortions. An Adaptive Optic (AO) system using the wavefront measurement data can dynamically compensate these disturbances applying a deformable mirror (Micro-Electro-Mechanical Systems (MEMs) [1]) or other actuator devices [2]; this provides an aberration corrected system with increased imaging performance. Moreover, adaptive optic devices may be applied not only in large telescopes, but also in many other diverse fields from ophthalmology to telecommunication.

An AO system correcting ability is better if the delay from sensing the wavefront to correcting is smaller than evolution time of the medium being corrected for. This delay can be diminished by choosing faster components: sensors, actuators, and computing-controlling units. However, the communication bandwidth between the components also limits the achievable speed of a closed loop AO system.

Thus, we have developed a single board adaptive optic system, made up of a high-speed CMOS sensor, a very fast Liquid Crystal on Silicon (LCOS) display, and a Field Programmable Gate Array (FPGA) device for control and calculation purposes. This type of affordable AO system

can open a new scope of widespread applications of adaptive optics. Distortions of the incoming wavefront are determined by using the on-board CMOS sensor with the help of the FPGA, calculating the required corrections. At the end, the LCOS device displays the corrections, which can eliminate the incoming wavefront distortions.

Although several wavefront sensor architectures exist, here we applied a special version of the most popular Hartmann-Shack (HS) sensors. In a HS sensor the image of the incoming pupil is projected on a lenslet array. Each lens of the array forms a miniature image of the source object and divides the incoming aperture into sub-apertures. The images of these sub-apertures are detected by an area scan sensor. The shifts of these images from the reference positions (positions without aberrations) specify the local wavefront slopes at the locations of the corresponding sub-apertures (the wavefront is regarded locally flat but tilted). The overall shape of the wavefront extended for the whole pupil can be reconstructed by assembling all these local tilted surfaces.

For a point source object (such as a star), simple quad-cell based HS sensors can measure these shifts [3, 4]. However, in the case of extended objects (the Sun or the Moon), these shifts can be determined only with a higher resolution sensor from the correlations between the sub-aperture images and reference sub-aperture images. Correlation-based HS sensors have signal to noise ratio larger, than that of the quad cells [5], and they can be used also in the case of point source objects, but they require considerably more computing resources. Even for first order aberration compensations (tip-tilt) high-speed correlation trackers are frequently applied [6].

In the case of conventional applications (e.g. Solar observation), the wavefront has to be measured by using many relatively large resolution sub-apertures. Since the wavefront is dynamically changing at a very high rate (time scale of a few milliseconds), an appropriately fast, real-time correlation-based wavefront sensor is required [7].

Some parallel processing devices can fulfill the necessary computation at this speed. Even though modern CPUs, stream processors (a GPU) [8, 9, 10] or DSPs can provide the required computation power, the FPGA technology appears to be a very competitive alternative, due to its fast development. As the control of the sensor and actuator regularly needs the application of some programmable logic device, it seems to be advantageous to employ both of them to fulfill the requested computation tasks as well. Recently, several FPGA-based wavefront sensors and AO system architectures have been introduced [11, 12] and their attainable performance was studied comprehensively [13, 14].

Considering the limitation of the FPGA devices and special parameterization of the mandatory wavefront sensors, we have chosen the Sum of Absolute Difference (SAD) method to implement the required correlation like processing. Several efficient FPGA implementations of the SAD algorithm have emerged [15], due to the needs of the FPGA implementation of motion image compression algorithms.

Accordingly, the optimal matching position of two pictures can be determined by the SAD algorithm. First, the SAD values of the picture are calculated, and then the minimum of them is determined. By this method, the displacement of the images of all sub-apertures is determined with respect to a reference image which can be considered as an image of a wavefront with no local slope. There are several alternatives to assign a proper reference image: here we use a sample image of a central sub-aperture as reference. Otherwise, the average image of many sub-apertures can also be used as reference. The SAD values are computed through the next equation:

$$SAD_{k,l} = \sum_{i}^{S}\sum_{j}^{S}\left|P_{i,j} - R_{i+k,j+l}\right|,\ k,l\ <\ A, \qquad (1)$$

where S is the size of the sub-aperture, A the size of the SAD value array, P the sub-aperture pixel, and R the reference pixel. The size of the reference image is the sum of the sizes of the sub-aperture and the SAD value array minus 1. Practically, the chosen size of the SAD value array is smaller than the size of the sub-aperture.

The structure of the paper is as follows: In Section 2 the overall structure of the FPGA-based adaptive optic system is outlined. Details of the wavefront sensing architecture implemented on FPGA are introduced in Section 3. In Section 4 our results are presented. Finally, conclusions are drawn in Section 5.

2 THE FPGA-BASED ADAPTIVE OPTIC SYTEM

Our FPGA-based adaptive optic system contains three main components. The first one is a very high frame-rate, mega pixel resolution CMOS image sensor, which is equipped with an appropriate lenslet array. The second one is a high-speed and resolution LCOS display aimed to fulfill the wavefront correction. The third one is an on-board FPGA, which is responsible for the control of the overall system and the calculation of the correction data. The error compensation of the sensor and data display using different linear, nonlinear and spatial filtering operations can also be accomplished in real-time by the built in FPGA. The adaptive optic system can be seen in Figure 1.

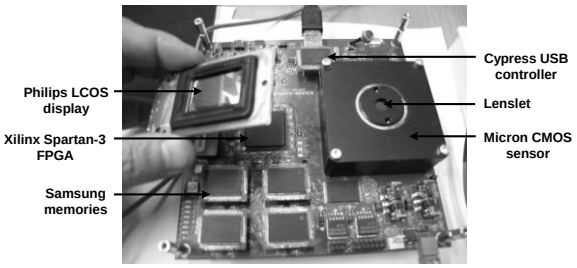

Figure 1. The adaptive optic system

The input image flow for the system is provided by a Micron MT9M413 CMOS sensor [16] whose resolution is 1280×1024 pixels (12μm×12μm in size). It is able to acquire 500 full-image frames per second. The sensor has digital output with ten parallel 10-bit wide data buses working at 60MHz to ensure the extremely large sensing speed; for this reason, it is often applied in high frame-rate cameras. This sensor is extended with a 32×32 array of micro-lenses, called lenslet array, and each of them covers 16×16 center part within a 32×32 sub-aperture on the sensor surface. The general structure of an N×M lenslet array covering S×S sub-apertures is depicted in Figure 2. This lenslet array and the corresponding sensor are applied to implement the HS wavefront sensor, which provides the required input of the adaptive optic system.

Optical geometry of the actual wavefront sensor depends on the aimed application. Parameterization issues are considered in the HS sensor literature [17].

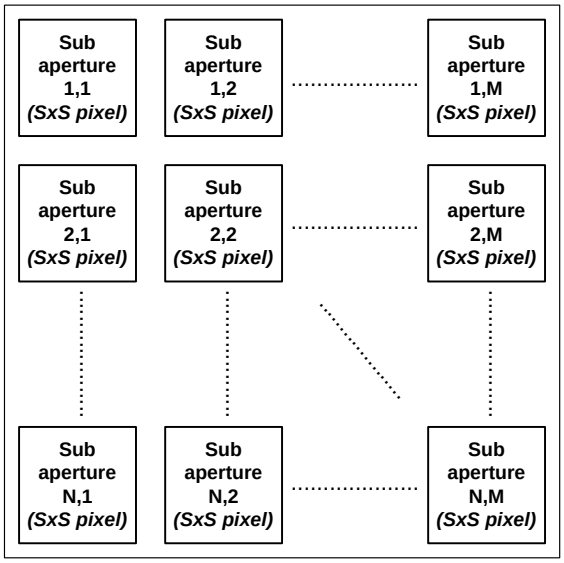

Figure 2. The general structure of an N×M lenslet array covering S×S sub-apertures

The correction is performed by a Philips DD720 LCOS display [18, 3] whose resolution is 1280×768 pixels with 20μm pixel sizes, which can display up to 540 full-image frames per second. With the help of this device, amplitude or phase modulation can be carried out by applying appropriate wave plates and polarizers. The required I/O bandwidth is approximately 750 megabytes/second, which implies that real-time data processing can be fulfilled only by using a parallel device.

Details and limitations of the considered zonal correction procedure, as it are even more intricate than wavefront sensing, is out of scope of this paper. However, it seems important to consider its achievable performance from the HS sensor speed and resolution requirements point of view.

The obvious choice to provide the appropriate control of the sensor and display devices is using a programmable logic device. Usually, the commercially available FPGA development boards do not make it easy to connect to high-performance imaging devices (cameras). Furthermore, the aforementioned 750 megabytes/second I/O bandwidth requirement is also a bottleneck of these FPGA boards. Consequently, a customized system was constructed, equipped with: 1) the previously introduced CMOS sensor, 2) LCOS display, and 3) Xilinx Spartan-3 XC3S4000 FPGA [19] whose density is 4 million system gates, and supplied by 96 18Kb BlockRAM and 96 18×18bit multipliers.

Although the system is designed as a stand-alone device for configuration and calibration purposes, a USB interface is required, and it is driven by an USB microcontroller. Many other standard steps, such as uploading sensor data, displaying bias patterns, downloading image correction and program parameters use this communication channel as well. Since there is not enough memory on the given FPGA to store a full image frame, external off-chip memories are required. Static RAMs are used, in which 8 full-image frames can be stored. Consequently, fixed pattern noise removal (FPN), flat field correction and LCOS manufacturing caused phase unevenness compensation can be carried out. The sensor and display device need different reference voltages, which are set by appropriate D/A converters. These D/A converters are programmed through a standard Serial Peripheral Interface (SPI). Furthermore, the temperature and power control of the LCOS display can be handled by a microcontroller, which communicates with the FPGA through a Universal Asynchronous Receiver Transmitter (UART) interface.

3 ARCHITECTURE IMPLEMENTED ON FPGA

Our primary goal is to implement a high-speed and highly parallel SAD architecture on the FPGA to

determine the displacement of the sub-apertures and to calculate the wavefront distortions, at rate of up to 500 full frames per second. Although the SAD algorithm is simple, its efficient FPGA implementation requires particular design considerations.

3.1 The architecture of the overall system

The architecture implemented on FPGA has three main parts, as shown in Figure 3. The *Shuffle* unit is responsible for serializing the pixels required for the calculation of the SAD values. SAD values for each image are computed, and the smallest value is chosen by the *SAD* unit. This value and its 4-connected neighbors are transferred to a Xilinx MicroBlaze soft-core processor to determine the slope values at sub-pixel resolution motion vectors. In addition to these elements, a *CMOS controller*, a *Flat* and *FPN* unit, an *LCOS* unit, a *USB control* unit, a *Memory controller*, and a *Master Controller* unit are also essential. The *CMOS controller* receives the image data and controls the Micron CMOS sensor. The *LCOS* unit computes the correction terms and sends them to the LCOS display. The *FPN* unit eliminates the fixed pattern noise from the CMOS sensor whereas the flat field correction is performed by the *Flat* unit. The *USB controller* is responsible for the communication with the host computer connected to the FPGA board. The coordinates of the relevant pixels of the CMOS sensor are defined by the host computer, whereas the reference image is updated rarely by the MicroBlaze unit. The *Memory controller* handles the data transfer from and to the memory. The memories store data for the *Flat* and *FPN* units, while the *Master Controller* unit is responsible for the control of the overall system.

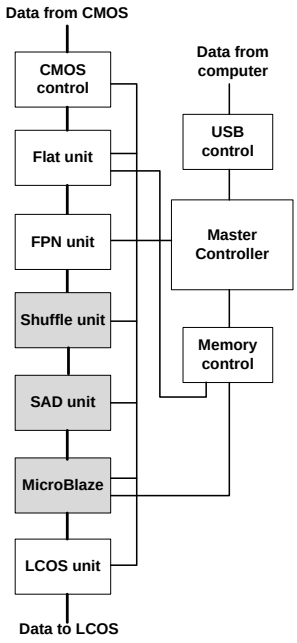

Figure 3. The implemented architecture of the overall system on FPGA (the main three parts are shaded)

3.2 The Shuffle unit

During the image capturing only the pixels of the sub-apertures are used; therefore, SAD values should be computed (to determine the wavefront) only in these areas, as shown in Figure 2. The relevant pixels with respect to the coordinates of the sub-apertures are defined by the host computer, by sending the upper left coordinates which are stored in the memory of the Pixel selection and Control unit. The selection and serialization of the incoming pixels from the CMOS sensor are performed by the *Shuffle* unit, whose architecture is shown in Figure 4.

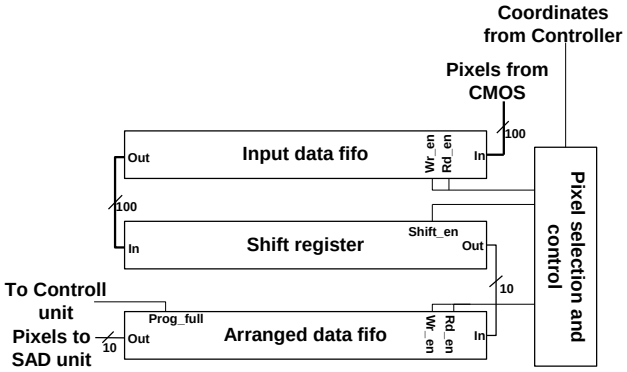

Figure 4. The architecture of the *Shuffle unit*

This architecture is built up from an *Input data FIFO*, an *Arranged data FIFO*, a parallel in serial out *Shift register* unit, and a *Pixel selection and control* unit. The *Input data FIFO* is a 100-bit input and 100-bit output FIFO, which is responsible for storing the 10×10-bit pixel data generated by the CMOS sensor in every clock pulse. This FIFO operates on the same clock frequency as the data bus of the CMOS sensor. The FIFO has to be deep enough to store a whole sub-aperture row (1280×P), in the worst case. The parallel in serial out *Shift register* unit converts the 100-bit input data to 10-bit output data by propagating only the relevant pixels (pixels of a certain sub-aperture) to the *Arranged data FIFO* and discarding the others. Therefore, only the pixels of the sub-apertures are stored in the *Arranged data FIFO*. This FIFO stores these pixels until all pixels in a certain row of this sub-aperture have arrived, and they are fed into the *SAD* unit serially. The parallel in serial out *Shift register* unit and the *Arranged data FIFO* operates on the clock frequency of the *SAD* unit. The *Pixel selection and control* unit controls the operation of the FIFOs and the parallel in serial out *Shift register* unit.

3.3 Operation of the SAD unit

The *SAD* unit is designed to compute Eq.(1) on the sub-apertures of the input image in real-time. Image data are sent by the CMOS sensor in a row-wise order. Therefore, S×S pixels should be stored to carry out the computation of the SAD values. Additionally, there are several sub-apertures in the row, which further increases the memory requirements, because the double buffering of the sub-apertures are required, and this is impractical in the case of large sub-apertures. The size of the required memory can be computed by the following formula:

$$s_{memory} = 2 \cdot S \cdot S \cdot p \cdot N \qquad (2)$$

where S is the sub-aperture size p is the bit width of the pixels and N is the number of sub-apertures.

Instead of storing the sub-aperture windows, the computation of the SAD values is rearranged to match the incoming data flow, whereas all partial SAD values are computed in parallel. When, for example, the first pixel ($P_{0,0}$) has arrived, the first term of Eq.(1) – $AD_{k,l,0,0}=|P_{0,0}-R_{k+i,l+j}|$ – is computed for all possible *k,l* values. When partial results, using the first row of the sub-aperture are computed, the SAD values are stored in a temporary buffer and the computation with the next sub-aperture starts. In this case AxA pixels should be stored, where

$$A = (r - S + 1) \cdot (r - S + 1) \qquad (3)$$

and r is the size of the reference image. To avoid overflow during the SAD computation the width of the partial results should be extended by $\log_2 \lceil S \cdot S \rceil$ bits. So in this case the memory requirement can be computed by the following formula:

$$s_{memory} = A \cdot A \cdot (p + \log_2 \lceil S \cdot S \rceil) \cdot N \qquad (4)$$

In our case relatively large reference images are used, with small shifts of the sub-apertures, thus S will be always larger than A. Therefore storing the partial SAD values requires considerably less memory than double buffering the incoming sub-aperture data. For example using the following typical values r=32, S=28, A=5, p=8, N=32, in this case double buffering of the sub-apertures requires 392 Kbits, while storing only the partial results requires only 14 Kbits memory.

The operation of the *SAD* unit is demonstrated in details on a simple example, where a 3×3 sized lenslet array covering 3×3 sub-apertures are used. The arrangement of the lenslet array is depicted in Figure 2. To calculate the 3×3 SAD value array of a sub-aperture of the lenslet array, a 5×5 reference picture is required in order to obtain a SAD value for every single pixel. Elements of the sub-aperture, the reference picture, and the SAD results are enumerated in a row-wise order, as shown in Figure 5. According to Eqs.(5-7), pixel P_1 is required for the calculation of the first term of the SAD values but with a different reference pixel. Similarly, pixel P_2 is required for the calculation of the second term of all the nine SAD values as shown in Eqs.(5-7), and so on.

By this method, partial SAD values are computed in parallel. The computation of one sub-aperture can be carried out in A×A cycles.

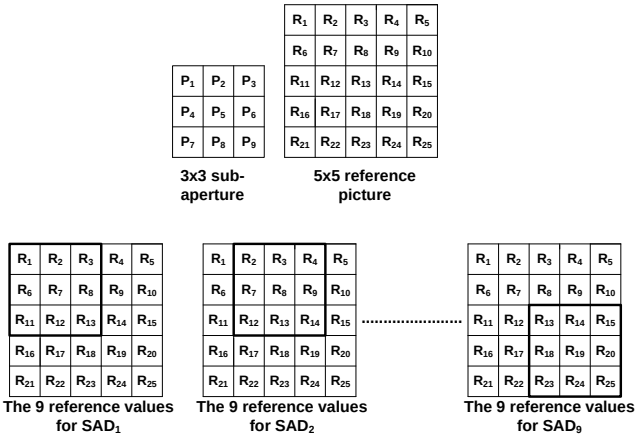

Figure 5. The 3×3 sub-aperture, the 5×5 reference picture, and the reference values required for the computation of SAD_1, SAD_2, …SAD_9

$$SAD_1 = |P_1 - R_1| + |P_2 - R_2| + |P_3 - R_3| + |P_4 - R_6| + |P_5 - R_7| + |P_6 - R_8| + |P_7 - R_{11}| + |P_8 - R_{12}| + |P_9 - R_{13}| \quad (5)$$

$$SAD_2 = |P_1 - R_2| + |P_2 - R_3| + |P_3 - R_4| + |P_4 - R_7| + |P_5 - R_8| + |P_6 - R_9| + |P_7 - R_{12}| + |P_8 - R_{13}| + |P_9 - R_{14}| \quad (6)$$

.....

$$SAD_9 = |P_1 - R_{13}| + |P_2 - R_{14}| + |P_3 - R_{15}| + |P_4 - R_{18}| + |P_5 - R_{19}| + |P_6 - R_{20}| + |P_7 - R_{23}| + |P_8 - R_{24}| + |P_9 - R_{25}| \quad (7)$$

3.4 The SAD unit

The main building blocks of the *SAD* unit are the *Reference Register* unit, the *Absolute Differences* unit, the *SAD Controller* unit, and the *Minimum Finder* unit, as can be seen in Figure 6. in case of 3×3 sub-aperture. The *Reference Register* stores the reference values and generates the appropriate reference window for a certain pixel of the sub-aperture. The *Absolute Differences* unit calculates the partial SAD or Absolute Difference (AD) values for the computation of the SAD. The *Minimum Finder* unit defines the smallest SAD value and 4-connected neighbors. *SAD Controller* unit controls the operation of the *SAD* unit. In addition to these elements, the *SAD* unit also contains an accumulator register array and some BlockRAMs. The size of the accumulator register array and the number of BlockRAMs is defined by the size of the SAD value array, whereas the depth of the BlockRAMs is specified by the number of the sub-apertures in the lenslet array rows. The accumulator unit is responsible for the addition of the computed AD results. The calculated AD results (SAD results in case of the last row of the sub-aperture) are shifted from the accumulator register array into the BlockRAMs at the end of the processing of the actual row of the sub-aperture.

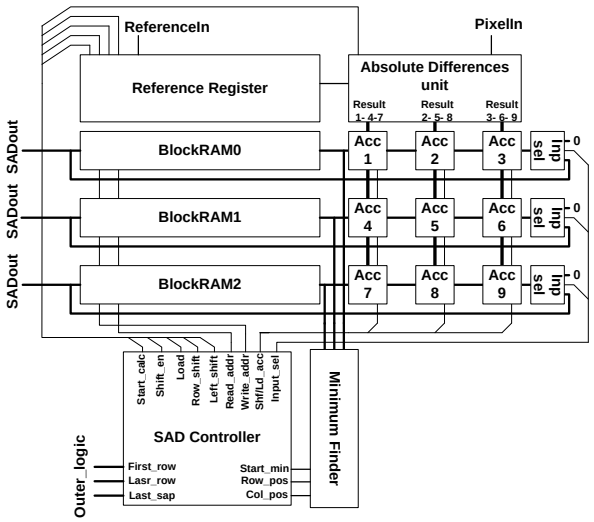

Figure 6. The architecture of the SAD unit in case of 3×3 sub-aperture

The *SAD* unit is started when a row of a sub-aperture arrives at the *Arranged data FIFO* unit (see Figure 4). The *Reference Register* unit is started by the *SAD Controller* unit to generate the required reference values for the calculation of the AD results when the first pixel has arrived. Meanwhile, the *Absolute Differences* unit is also started to calculate all of the required absolute difference values. This unit has only two clock cycles latency. Results are stored in the accumulator array. If the next pixel has arrived, all the AD values in connection with this pixel are computed and added to the partial results stored in the accumulator array. At the end of the actually processed row of the sub-aperture, the content of the accumulator is shifted into the BlockRAMs. For the first row of the sub-aperture, the accumulator array is filled with zeros; in the other cases, the partial results computed with the previous row of the next sub-aperture are loaded into the accumulator. This is necessary for the accumulator to be up-to-date, and hold the partial results for next sub-aperture. The reinitialization of the values in the *Reference Register* is also performed during this time.

The SAD value calculation of the sub-aperture is finished when the last row of the sub-apertures is processed. The minimum of the SAD values and its 4-connected neighbors are determined by the *Minimum Finder* unit, which also specifies the locations of these values in the sub-aperture. When all of the SAD values are computed the results can be read from the BlockRAM memories in a column-wise order. The smallest values in each column are determined in parallel by using a tree of comparators, while the minimum of the whole area is determined serially. Therefore the latency of the *Minimum Finder* depends on the size of the SAD value array. After

the computation of the SAD values of the last sub-aperture in the actual row, the minimum values and the 4-connected neighborhood can be read out from the BlockRAMs, according to the locations specified by the *Minimum Finder* unit. These values are sent to the Xilinx MicroBlaze processor for further processing.

The implemented *SAD* unit operates on twice higher clock frequency than the data bus of the CMOS sensor. The size of the reference image, which defines the size of the SAD value array, the size, and number of the sub-apertures is parametrizable in the Very High Speed Integrated Circuits Hardware Description Language (VHDL) description of the unit, according to the wavefront sensing algorithm, where 8×8 to 32×32 sub-apertures are required. Additionally, several *SAD* units can work in parallel by slicing the input image and using more *Shuffle* units.

3.5 The Reference Register unit

The *Reference Register* stores the reference values and generates the appropriate reference window for a certain pixel of the sub-aperture. The S×S reference window moves from left to right during the computation, as shown in Figure 5. In order to make the implementation simpler and to use the shift register resources in the FPGA, the reference window is fixed and the reference values are shifted in our system. Utilizing the shift registers requires less area and routing resources than using a simple register array. For further optimization, the non-used part of the reference array is stored in BlockRAM. The required nine reference values in the example above are always placed on the upper left side of the reference array in a 3×3 window. The architecture of the *Reference Register* unit is shown in Figure 7. in case of a 5×5 reference picture. This unit contains a shift register array to store and shift the reference values. The size of the array is defined by the size of the reference image. The array is filled up with the reference values in the initialization phase. To compute the SAD values the data stored in the reference array should be shifted circularly, with respect to the position of the incoming pixels. Thus, the *Reference Register* unit has three operation modes: 1) load, 2) left shift, 3) row shift. In the load mode, the reference values are loaded into the bottom right register (Reg_{25}) and shifted left in the register chain pixel-wise; data from the leftmost registers (Reg_{21}, Reg_{16}, Reg_{11}, Reg_{6}, Reg_{1}) are shifted to the rightmost registers (Reg_{20}, Reg_{15}, Reg_{10}, Reg_{5}) a row above. In the left shift mode, values in the registers are shifted circularly in the same row. Finally, the row shift mode is very similar to the load mode, except in the case of the lower right register (Reg_{25}) which is loaded with the contents of the upper left register (Reg_{1}).

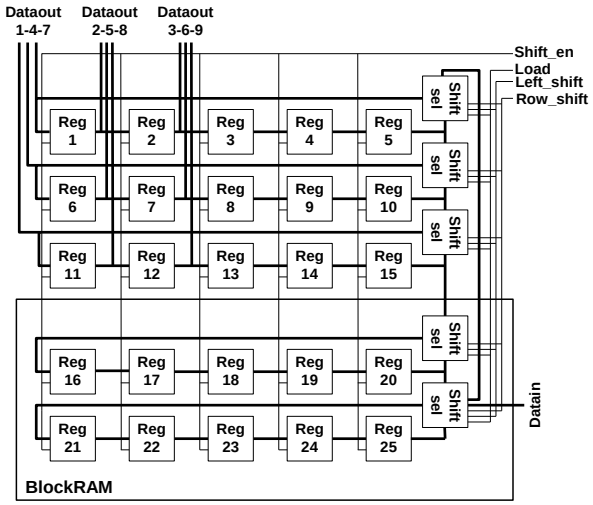

Figure 7. The architecture of the *Reference Register* unit for the case of a 5×5 reference image

The detailed steps of moving the reference values can be seen in Figure 8, in case of the aforementioned example. In the 1st clock cycle, when the first pixel of the first sub-aperture has arrived, the reference values are not shifted, because they are in the proper position. In the 2nd clock cycle, the reference values are shifted one position to the left, as can be seen in Figure 8. In the 3rd clock cycle, when the third pixel of the first sub-aperture is available, also a left shift of the reference values is carried out. Before the calculation of the AD results of the second sub-aperture, the reference values should be shifted back to the initial position. This is done under the 4th, 5th clock cycle, as shown in Figure 8. In the 6th clock cycle the calculation of the first row of the second sub-aperture may start. During this calculation the reference values are shifted, as described previously. In the 11th clock cycle the calculation of the first row of the third sub-aperture may start. In this case, the reference values are not only shifted left, but also started to shift up. This is required to prepare the proper reference window for the processing of the next row of the first sub-aperture. After the 13th clock cycle, an initialization phase is also required until all values from the second row are shifted up to the first row of the array.

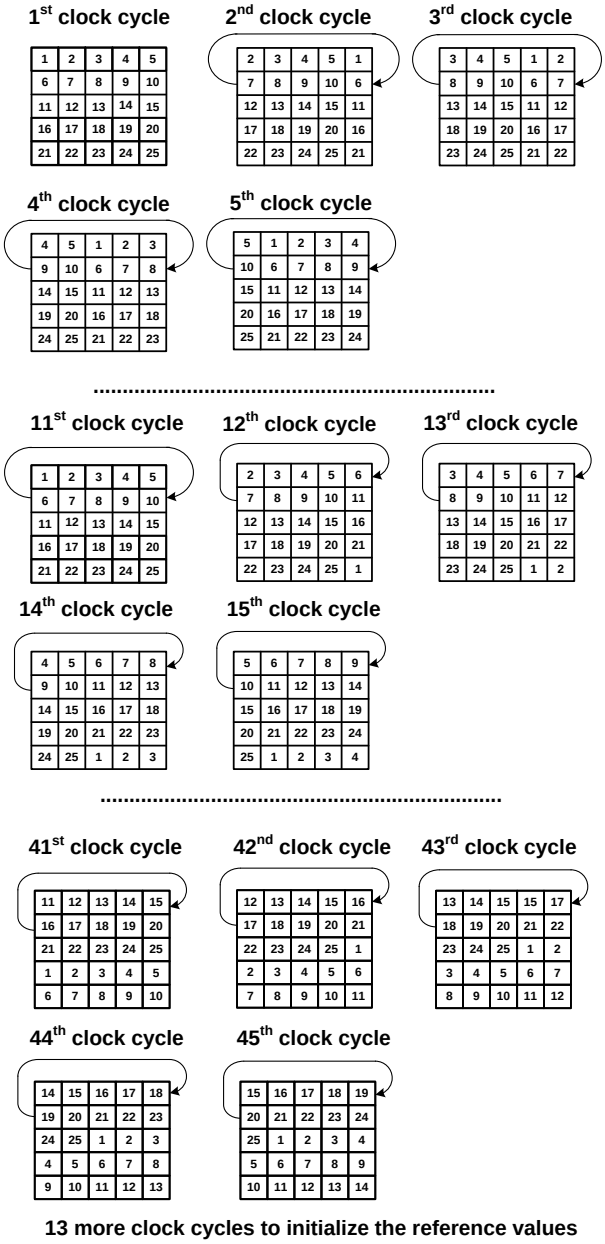

Figure 8. The data movement in the 5×5 reference register array

This process is fulfilled under the 14th, 15th, clock cycle. The process described previously from the 1st to the 15th clock cycle is repeated until the last row of the last sub-aperture is available in the 41st clock cycle. At this point a row shift is carried out under the 41st, 42nd, 43rd, 44th, 45th clock cycles for positioning the reference values. After the 45th clock cycle, the first half of the array and the second half of the array are in swapped positions, as represented in Figure 8. Consequently, these values should be shifted back to their initial positions in order to start the processing of the next sub-aperture row. This process here requires 13 clock cycles.

3.6 The Absolute Differences unit

The *Absolute Differences* unit is responsible for calculating the absolute differences between the reference and the input images. This unit is built up from an array of processing elements. The number of the processing elements is defined by the size of the sub-aperture. Therefore, all AD values can be calculated in parallel. The structure of the processing element can be seen in Figure 9.

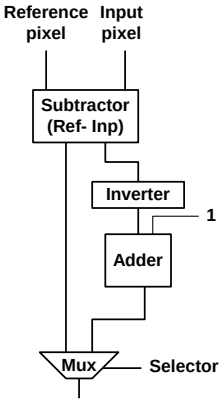

Figure 9. The structure of a processing element in the *Absolute Differences* unit

A processing element first calculates the difference between its inputs, then inverts the result, and adds 1 to it, generating this way the 2's complement form of the calculated difference. Finally, using a *Selector* signal – according to the MSB bit of the difference – the 2's complement as the absolute value of the difference or the difference is presented on the output. This unit has only two clock cycle latency.

4 RESULTS

An FPGA-based adaptive optic system was constructed and the architecture of the wavefront sensor system was implemented on a Spartan-3 XC3S4000 FPGA by using VHDL. The main block of this system is the *SAD* unit, which is responsible for the calculation of the SAD values. The implemented *SAD* unit is fully parametrizable with respect to the size and the number of sub-apertures. The unit can be parameterized in the VHDL code of the unit before the synthesis process, according to the wavefront sensing algorithm where 8×8, to 32×32 sub-apertures are required. The size of the reference image, which defines the size of the SAD value array, is also parametrizable.

The maximal size of the sub-aperture is limited by the area requirement of *SAD* units on the FPGA.

The number of the applicable sub-apertures is bounded by the row length of the CMOS sensor. The 18Kb BlockRAMs used in the *SAD* unit is large enough to store the entire row of the computed SAD values. The size of the sub-apertures is important, since it defines the required number of BlockRAMs and other logic resources on the FPGA. The performance of the *SAD* unit is investigated in case of different sub-aperture and reference image sizes. Static timing analysis of the placed and routed designs show that the speed of the *SAD* unit can attain the 120 MHz operating frequency in all cases. The Flip-Flop resource utilization of the implemented *SAD* unit on the Spartan-3 XC3S4000 FPGA can be seen in Figure 10.

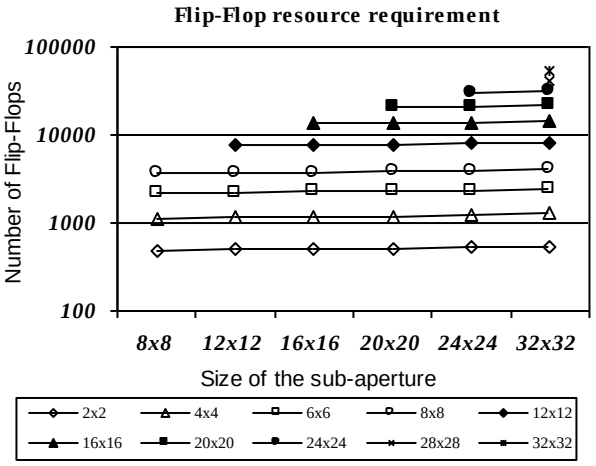

Figure 10. The Flip-Flop resource requirement of the *SAD* unit in case of different size of SAD value array

Figure 10 shows that the Flip-Flop resource requirement of the *SAD* unit increases quadratically, according to the size of the sub-apertures, but it does not depend on the size of the SAD value array. This is the cost of the high-level parallelism. If the size of the sub-aperture increases, then the number of required Flip-Flops also extends, because of the increasing number of accumulator registers, reference registers, and adders. The same behavior can be obtained in case of the 4-input LUT and BlockRAM resource requirement of the *SAD* unit. In order to achieve even faster performance, several *SAD* units can be used in parallel. The number of realizable *SAD* units in case of the Spartan-3 XC3S4000 and Virtex-4 XC4VLX200 can be seen on Table 1. For a fair comparison the Virtex-4 family was chosen, because the architecture of Virtex-5 and Virtex-6 families is considerably different from that of the Spartan-3. The number of realizable *SAD* units is determined by BlockRAM resources on the FPGA in case of SAD value arrays smaller than 12×12. For larger SAD value arrays, the bottleneck is the number of available Flip-Flops on the device.

Sub-aperture size in pixels (S×S)	Spartan-3 XC3S4000	Virtex-4 XC4VLX200
8×8	*11*	*37*
12×12	*7*	*23*
16×16	*4*	*13*
20×20	*3*	*9*
24×24	*2*	*6*
28×28	*1*	*4*
32×32	*1*	*3*

TABLE 1. THE REALIZABLE NUMBER OF *SAD* UNITS (REFERENCE PICTURE SIZE IS (2S-1)×(2S-1))

The number of clock cycles that a *SAD* unit requires to calculate the SAD values changes according to the size of the sub-apertures, as shown in Table 2 in case of maximal SAD value array size; thus, the number of real-time manageable sub-apertures is different. It also shows the number of sub-apertures and the area of the sub-apertures compared to the surface of the CMOS sensor which can be computed in real-time using one *SAD* unit. One *SAD* unit running at 120MHz clock frequency can handle 2171 8×8 sub-apertures on each frame of the CMOS sensor in real-time. However, this is only 42.6% of the surface of the CMOS sensor, but it requires only 8.72% of the FPGA resources. Therefore, using three *SAD* units on the Spartan-3 FPGA, the whole surface of the CMOS sensor can be processed in real-time. By using 32×32 sub-apertures, 127 sub-apertures can be handled in real-time. This makes it possible to process the 39.64% of the entire surface of the CMOS sensor. Using higher performance Virtex-4 XC4VLX200, three *SAD* units like this can be implemented, as shown on Table 1.

Sub-aperture size in pixels (S×S)	Required number of clock cycles	Maximum number of sub-apertures/frame	Percentage of the CMOS surface
8×8	*120*	*2171*	*42.60%*
12×12	*156*	*941*	*41.35%*
16×16	*496*	*522*	*40.78 %*
20×20	*780*	*331*	*40.39%*
24×24	*1128*	*228*	*40.10%*
28×28	*1540*	*167*	*39.86%*
32×32	*2016*	*127*	*39.64%*

TABLE 2. THE TEST RESULTS OF ONE SAD UNIT (REFERENCE PICTURE SIZE IS (2S-1)×(2S-1))

Additionally, the clock frequency of the *SAD* unit on this architecture can be increased to 230 MHz, according to the results of the static timing analysis. Therefore, applying this FPGA, also the whole surface of the CMOS sensor can be processed in real-time using 32×32 sub-apertures.

Our results are compared to a correlation-based wavefront sensor system described in [20]. This system is implemented on a Xilinx Virtex-4 SX35-10, in which the processing core operates on 100MHz clock frequency. For better comparison, also the clock frequency of our *SAD* unit is decreased to 100MHz. The comparison of the *SAD* unit implemented on Spartan-3 XC3S4000 and the correlation-based system is shown in Table 3.

	CMOS area: 256x256 Sub-aperture: 8x8 (in pixels)		**CMOS area: 512x512 Sub-aperture:16x16 (in pixels)**	
	Corr.	**SAD**	**Corr.**	**SAD**
Slices	*5431*	*2769*	*9932*	*10186*
LUTs	*10161*	*3435*	*18607*	*12093*
Flip-flops	*2096*	*3615*	*5981*	*13615*
Time	*2.89ms*	*1.267ms*	*18.1ms*	*5.238ms*
AT	***15.69***	***3.51***	***179.76***	***53.35***

TABLE 3. COMPARISON OF THE DIFFERENT IMPLEMENTATIONS.

The results show that the Area*Time (AT) parameter of our SAD-based wavefront sensor system is smaller than the correlation-based system described in [20]. In case of 8×8 sub-apertures, the AT parameter of our *SAD* unit is 22% of the correlation-based system. This ratio increases when larger sub-apertures are used. In 16×16 case, the performance difference is 29% between the two types of implementation. Thus SAD based solution provides similar accuracy [21] but it can be computed more efficiently. The performance of our system can be further increased by using more *SAD* units. Additionally, its operating frequency is also higher, especially in case of the Virtex-4.

Even though FPGA-based SAD implementations [15, 22, 23] have been published in different journals and books, systems like this were, generally, constructed on Altera FPGAs. Consequently the comparison of the Altera-based solutions with our Xilinx-based system may be not reliable, due to the architectural differences, but the overall performance of our implementation seems to out-perform them. However, our special purpose SAD architecture shows superior performance (16x16 sub-aperture: 10,186 slices and 496 clock cycles comparing to the published 9,478 slices and 1,600 clock cycles) with respect to the comparable motion estimation processor SAD implementations [23].

5 CONCLUSION

Our wavefront sensor system is based on the high-speed and highly parallel SAD calculation unit implemented on a Spartan-3 FPGA. This system can be used with a wide range of applicable lenslet arrays, since it is fully parametrizable with respect to the size and the number of sub-apertures and the reference image size. The performance of the system was tested on sub-apertures and reference images of different sizes, which defines the size of the SAD values array.

The results show that the resource requirement of the introduced SAD calculation unit increases quadratically according to the size of the sub-apertures which is the price of the high-level parallelism. Fortunately the size of the SAD value array does not influence the area requirement significantly. Considering 8×8 sub-apertures, the entire surface of the CMOS sensor can be processed in real-time, using three *SAD* units on the Spartan-3 XC3S4000 FPGA. In case of higher resolution sub-apertures, the real-time processable CMOS surface is reduced. Notwithstanding, applying higher performance FPGAs (Virtex-4) makes it possible to handle the whole area of the CMOS sensor real-time. In case of 8×8 sub-apertures 4.4 times performance gain can be achieved compared to the correlation-based wavefront sensor architecture. This difference is further increased when larger sub-apertures are used (see in Table 3). Our parallel SAD FPGA-based implementation architecture provides an efficient high-speed wavefront sensor whose performance is higher than that of the counterparts applied so far.

The rapid development of the FPGA technology offers the possibility to construct intelligent, programmable, very high-frame rates, and high-resolution cameras, diminishing the required communication bandwidth by on-camera processing in the control systems where high performance imaging sensors are used.

6 ACKNOWLDGEMENT

The research was supported by the OTKA Grant No.: K 61965. Hardware engineering work, including final PCB design, was accomplished by CORTEX Ltd. We express our gratitude to Andras Radványi, and special thanks are due to Zsolt Vörösházi and Beáta Vajda who helped us to complete the paper, and the helpful comments and suggestions of the reviewers are kindly acknowledged.

7 REFERENCES

[1] Dayton, D., Gonglewski, J., Restaino, S., Martin, J., Philips, J., Hartman, M., Kervin, P. Snodgress, J. Browne, S., Heimann, N., Shilko, M., Pohle, R., Carrion, B., Smith, C. and Thiel, D., "Demonstration of new technology MEMS and liquid crystal adaptive optics on bright astronomical objects and satellites," Opt. Express 10, 1508-1519 (2002).

[2] Richards, K., Rimmele, T., Hill, R., Chen, J., "High speed low latency solar adaptive optics camera", Proc. SPIE, 5171, 316-325 (2004).

[3] Zhaoliang Cao, Lifa Hu, Dayu Li, and Li Xuan, "Adaptive optics imaging system based on a high-resolution liquid crystal on silicon device," Opt. Express 14, 8013-8018 (2006).

[4] D. W. de Lima Monteiro, G. Vdovin, P. M. Sarro, "High-speed wavefront sensor compatible with standard CMOS technology, " Sensors and Actuators A: Physical,, 109 (3), 220-230 (2004)

[5] Poyneer, L.A., Palmer, D.W., LaFortune, K. N., Bauman, B., "Experimental results for correlation-based wavefront sensing," Advanced Wavefront Control: Methods, Devices, and Applications III. Proc. SPIE, 5894, 207-220 (2005).

[6] Chang-Hui Rao, Wen-Han Jiang, Cheng Fang, Ning Ling, Wei-Chao Zhou,Ming-De Ding, Xue-Jun Zhang, Dong-Hong Chen, Mei Li, Xiu-Fa Gao and Tian Mi "A Tilt-correction Adaptive Optical System for the Solar Telescope of Nanjing University" Chin. J. Astron. Astrophys. 3 (6), 576–586 (2003).

[7] Serati, S., Xiaowei, X., Mughal, O., Linnenberger A., "High-resolution phase-only spatial light modulators with sub-millisecond response," Proc. SPIE, 5106, 138-145 (2003).

[8] Rodríguez-Ramos, L. F., Marichal-Hernández, J.G., Rosa, F., "Modal Fourier wavefront reconstruction on graphics processing units," Advances in Adaptive Optics II. Proc. SPIE, 6272, 627215 (2006).

[9] Marichal-Hernandez, G., Rodriguez-Ramos, J.M., and Fernando Rosa, J., "Modal Fourier wavefront reconstruction using graphics processing units," Journal of Electronic Imaging 16(2), 023005 (2007).

[10] Rosa, F.L., Marichal-Hernandez, J.G., Rodriguez-Ramos, J.M., "Wavefront phase recovery using graphic processing units (GPUs)," Optics in Atmospheric Propagation and Adaptive Systems VII. Proc. SPIE, 5572, 262-272 (2004)

[11] Saunter C.D., Love G.D., Johns, M., Holmes, " FPGA technology for high speed, low cost adaptive optics" Proc. SPIE 5th International Workshop on Adaptive Optics in Industry and Medicine (2005).

[12] Rodríguez-Ramos, L.F., Viera, T., Herrera, G., Gigante, J.V., Gago, F., Alonso, Á., "Testing FPGAs for real-time control of adaptive optics in giant telescopes," Advances in Adaptive Optics II. Proc. SPIE, 6272, 62723X (2006).

[13] Rodríguez-Ramos, L.F., Viera, T., Gigante, J.V., Gago, F., Herrera, G., Alonso, Á., Descharmes, N., "FPGA adaptive optics system test bench," Astronomical Adaptive Optics Systems and Applications II. Proc. SPIE, 5903, 120-128 (2005).

[14] Saunter C.D. and Love, G.D., "Low cost, high speed control for adaptive optics," Proc. SPIE 6th International Workshop on Adaptive Optics for Industry and Medicine. (2007).

[15] Wong, S., Vassiliadis, S., Cotofana, S., "A sum of absolute differences implementation in FPGA hardware" Euromicro Conference Proc. 28, 183–188 (2002).

[16] http://www.microdisplay.com

[17] Jason Porter, Hope Queener, Julianna Lin, Karen Thorn, Abdul A. S. Awwal "Adaptive Optics for Vision Science: Principles, Practices, Design and Applications" Wiley, (2006)

[18] http://holoeye.com/download_area.html

[19] http://www.xilinx.com/

[20] Trujillo J.S., Valido, M.R., Rodríguez Ramos, L.F., Boemo, E., Rosa, F., Rodríguez Ramos, J.M., "Real time phase-slopes calculations using FPGAs", Proceedings of SPIE, 7015.

[21] R. Sridharan, A. Raja Bayanna and P. Venkatakrishnan "Simulations of Solar AO Systems" Springer Berlin, Science with Adaptive Optics (2005)

[22] Ambrosch K., Humenberger, M., Kubinger, W., Steininger, A., "SAD-Based Stereo Matching Using FPGAs", Springer, Embedded Computer Vision, (2008)

[23] Li, B.M. and Leong, P.H. "Serial and Parallel FPGA-based Variable Block Size Motion Estimation Processors" Journal of Signal Processing Systems 51, 77–98 (2008).