I. Introduction

FPGA based hardware accelerators have become more and more important for bioinformatics applications. These applications use wide range of algorithms, including searches in large databases, sequence alignment, statistical analysis and image processing. A part of these algorithms can be efficiently accelerated using FPGA devices.

Biological and biomedical experiments like microarray experiments [1], high-content screening (HCS) or cellular microscopy [2] result in large amount of image data. These data have to be processed to evaluate the experiments. There are different applications available for this purpose, for example the CellProfiler [3], which can process and analyze cellular microscopy images.

The final goal would be an FPGA based hardware accelerator for (biomedical) image processing purposes. This would include several reconfigurable execution units, which would enable the parallel processing of the data. By using partial dynamic reconfiguration [4], the execution units could be quickly reconfigured with the necessary algorithms.

This paper introduces a reconfigurable test system and two basic image processing functions implemented with FPGA. These functions are a 3 x 3 median filter and a 3 x 3 FIR filter, which can process 8 bit grayscale images.

II. The reconfigurable system

A. Hardware

The design is implemented on the Xilinx ML506 board. This board utilizes an XC5VSX50T FPGA (Virtex-5 SXT family), which is optimized for DSP and memory-intensive applications. It contains increased number of Block-RAMs and DSP48E slices and its capacity enables to implement more complex designs in the future. The ML506 board has wide range of communication interfaces to exchange data with the PC, for example serial ports, USB port, PCI-Express x1 interface and 10/100/1000 Ethernet interface. The Ethernet interface has been chosen because it requires less development effort and it provides enough communication speed. Fig. 1 shows the block diagram of the reconfigurable system. The system clock frequency is 100 MHz.

![Figure 1: Block diagram of the reconfigurable system](image)

The Virtex-5 SXT FGAs don’t contain embedded PowerPC processor therefore the MicroBlaze soft processor is used in the design. The 32 kB Block-RAM stores the software running on the
processor and it is connected to the MicroBlaze CPU through the LMB bus. The 256 MB external memory stores the original and the filtered images, as well as the configuration data for the reconfigurable modules. Because the limited number of the internal block-RAMs, using fast and high capacity external memory is the efficient way to store these data. The multi-port memory controller provides access to the external memory. The memory controller has 8 ports: one port is connected to the PLB bus, the other ports are available to the reconfigurable modules. The Ethernet MAC peripheral is necessary for the Ethernet communication. The PLB timer peripheral has two timer registers. One timer is required by the TCP/IP stack, the other timer is used to measure the reconfiguration and the filtering time.

The PLB ICAP peripheral is a master peripheral connected to the PLB bus. It handles the internal configuration access port (ICAP) of the Virtex-5 FPGA and can be used to update the configuration of the reconfigurable modules. Before starting the dynamic reconfiguration process, the start address and the data length have to be written into the proper registers. Then the peripheral reads the configuration data from the SDRAM and writes it into the ICAP. The maximum allowed clock frequency of the ICAP is 100 MHz. When the system clock frequency is greater than 100 MHz, the peripheral includes a DCM module to synthesize the 100 MHz ICAP clock and the synchronization logic required to connect the two sides working at different clock frequencies.

The PLB filter controller peripheral provides the connection between the system and the filter module in the reconfigurable region. The internal filter and peripheral registers can be accessed through the PLB slave interface. In the current design, the Xilinx CacheLink (XCL) interface has been chosen for data transfer between the filter and the external SDRAM. The XCL is a simple FIFO based point-to-point interface, it provides 1, 4, 8 or 16 word length read and write transfers. The filter controller has one read and one write XCL interface, which are connected to the multi-port memory controller.

B. Software applications

The main function of the software for the MicroBlaze processor is to handle the TCP/IP communication and to control the reconfiguration and the filtering. This software uses the uIP 1.0 TCP/IP protocol stack. A simple Windows application has been created for the PC side. This application can be used to download the configuration data and the images to the target system, modify the filter coefficients and read back the filtered images. It also implements the software version of the filters. This allows comparing the hardware and the software filtering times.

III. Implementation of the filters

The 2D median and the 2D FIR filtering algorithms are called local image processing algorithms, because they depend on data from a relatively small neighborhood compared to the image size. These algorithms use a sliding window (Fig. 2) and execute different operations on the pixels found in the window. The filters in this design use a 3 x 3 window and can process 8 bit grayscale images with a maximum size of 1024 x 1024 pixels. The image size is adjustable.

![Figure 2: Moving the window on the image](image1)

![Figure 3: Block diagram of the filters](image2)
The sliding window image buffer can be easily implemented in hardware as shown in Fig. 3. Registers store the pixels in the window, the remaining pixels (image width – 3) of the rows are stored in a dual-port Block-RAM. When a new data is written into the buffer, the window is shifted right one position.

As the window moves through the image, its center covers a smaller area than the original image, which results a smaller filtered image size. This problem can be solved in different ways. In this design, the filters process an expanded image shown in Fig. 2. It contains the original image and its edges are the copy of the edges of the original image. This expanded image can be easily created in hardware by adding a multiplexer to the image buffer input, which can select the following write operations:

- reloading the previously written row into the buffer (at the 2nd and last rows)
- reloading the previously written pixel into the buffer (at the 2nd and last columns)
- writing the next pixel of the original image into the buffer (otherwise)

A. 2D median filter

Median filtering can efficiently reduce or remove “salt & pepper” noise from the images. Its edge-preserving nature makes it useful in cases when edge blurring is undesirable. Median filters sort the pixels in the window by intensity and the center element of the window is selected as the output.

In this design, the median element is determined by the following way. At first, each row of the window is sorted (stage A). In the next step, each column of the window is sorted (stage B). At last, the elements in the secondary diagonal are sorted (stage C). The median element in the secondary diagonal will be the median element of the original inputs. The fully parallel and pipelined sorting network can be seen in Fig. 4.

B. 2D FIR filter

The operation executed by the FIR filters is convolution, which is defined in Eq. (1). The output of the filter is the weighted sum of the elements in the window.

\[ f(x, y) = \sum_{i=-1}^{1} \sum_{j=-1}^{1} W_{ij} \cdot image(x + i, y + j) \]  

(1)

By selecting the appropriate weights, convolution can implement low-pass, high-pass and band-pass frequency domain filters. Low-pass filters use positive weights and are used for image smoothing. High-pass filters use a positive center weight and negative outer weights and are used to enhance high frequency components in an image such as edges and fine detail. All of the filter coefficients (\(W_{ij}\) weights) are adjustable.

The parallel implementation of a 3 x 3 FIR filter requires nine multipliers and eight adders, both are registered at the outputs. The adders are organized in tree form to reduce the pipeline stages. The last pipeline stage is the saturation unit, which saturates the output when overflow occurs. Negative output values are possible when there are negative \(W_{ij}\) weights. These output values are meaningless and are replaced with zeroes.
IV. Results

Table 1 shows the logic utilization of the reconfigurable region of the FPGA. The reconfigurable region is larger than the required size, only 24% of the logic resources are used by the filters. Decreasing the reconfigurable region size would shorten the reconfiguration time. Unfortunately, in case of more complex functions, more resources are required and these functions may not fit into the smaller reconfigurable regions. The size of the reconfigurable regions can be optimized when more image processing modules will be available.

<table>
<thead>
<tr>
<th>Module name</th>
<th>Configuration file size (bytes)</th>
<th>Reconfigurable region utilization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Median filter</td>
<td>146396</td>
<td>688 / 2880 (24%)</td>
</tr>
<tr>
<td>FIR filter</td>
<td>142779</td>
<td>473 / 2880 (16%)</td>
</tr>
</tbody>
</table>

Table 2 shows the reconfiguration and filtering time. As for the values in the “theoretical” columns of the table, it is assumed that every new input data is processed in one clock cycle. The actual values are about three times greater than the theoretical values. Because the filters are pipelined, they can process new data at every clock cycle. Therefore the bottleneck is the memory access and it have to be optimized in future designs to achieve better results. If the hardware and the software filtering times are compared, the FPGA design performs better. In case of 2D median filtering, the speed-up is 20x. In case of 2D FIR filtering, the speed-up is 8.6x. The ratio of the reconfiguration and the filtering times highly depends on the image size. To reduce the number of reconfigurations, it is suggested to process as many images as possible with a module before the module is reconfigured.

<table>
<thead>
<tr>
<th>Module name</th>
<th>Reconfig. time @ 100 MHz</th>
<th>Filtering time (512 x 512 image size)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>theoretical</td>
<td>actual</td>
</tr>
<tr>
<td>Median filter</td>
<td>0.366 ms</td>
<td>1.05 ms</td>
</tr>
<tr>
<td>FIR filter</td>
<td>0.357 ms</td>
<td>1.03 ms</td>
</tr>
</tbody>
</table>

Fig. 5 shows some images used for testing the filters. The task is to enhance the first noisy image. This process is very sensitive to the noise. When the noise is not removed, it will be also enhanced as shown in the second image. The median filter can be used to remove the noise from the first image as shown in the third image. The FIR filtered third image (final result) can be seen in the last image.

![Figure 5: Input and filtered images](image_url)

References

[3] CellProfiler cell image analysis software manual
URL: http://www.cellprofiler.org