Heterogeneous Systems Architectures

Laboratory project

Design and implementation of an image processing heterogeneous digital system

V1.0 – Nov 11th, 2024

Revision history

Date	Notes	
Nov 11th, 2024	First release V1.0	jca@fe.up.pt, jcf@fe.up.pt

1. Objectives

The objective of the project is to design a heterogeneous hardware-software system for the ZedBoard development platform, which implements an image processing application given as a C program. The system should be optimized for performance while constrained to the logic resources available in the Zynq device of the target platform.

The work is divided into four main tasks, corresponding to 4 accomplishment goals:

i) Analyze the software application and study the potential of acceleration when implementing selected tasks (functions, combinations of functions or sections of code) as custom hardware accelerators.

ii) For the selected application tasks, adapt the source code and design the hardware processing system using High-Level Synthesis and explore adequately the solution design space to optimize the tradeoff between performance and logic complexity.

iii) Build a hardware-software system integrating the custom hardware blocks built in ii) with the ARM9 processing system of the Zynq device; the objective at this stage is to assemble a first functional HW/SW system, before optimizing the data transfer mechanisms between the software and the hardware domains.

iv) Optimize the data transfer processes between the software application and the custom hardware accelerator (using DMA transfers, data streaming interfaces or AXI4 data burst transactions…).

Details on the application are presented in section 3.

2. Grading

The final grading will be based on the level of realization of the 4 goals listed above, with the following relative weights. Each task will be evaluated between 0 (poor) to 5 (excellent).

task		weight
i)	Application analysis	20%
ii)	HLS design synthesis	30%
iii)	System integration and implementation	30%
iv)	System optimization	20%

3. The C application

The target application implements a sequence of image processing tasks on 8-bit per pixel gray-scale images to detect edges and generate an output binary image (1 bit per pixel). The input image is read as a text file created by a Matlab/Octave script. The size of the image to process can be configured in the software C application and the Matlab scripts. However, the image size will be constrained to 512 x 512 for the implementation of the hardware accelerator, due to memory limitations.

The sequence of images below illustrates the intermediate results of the sequence of the image processing tasks: the original image is blurred 2 times with a 3x3 low-pass average kernel; the bright is increased 1.5 times; the contrast is enhanced; an edge detection 5x5 kernel is applied;

the image is binarized with a fixed threshold (128); an erode function is applied to remove isolated white pixels (these images are located in folder **./dat**).

original
(datain.png)

blur (2X)
(imageblur.png)

increase bright 1.5X
(imagebright.png)

enhance contrast
(imageconstrast.png)

edge detection & saturate
(imageedgedet.png)

binarize & erode
(dataout.png)

The image processing application is implemented in program **aship.c**. The program reads a text file with the input image (gray scale, 8 bits per pixel) and writes the output image to a text file. The input text file is created from a bitmap image by running the Matlab script **im2text.m** and the output text file containing the output image can be converted again to a bitmap image with the script **text2im.m** (reads **dataout.txt** and creates image file **dataout.png**) or **test2ims.m** (reads all ***.txt** files and generates the corresponding ***.png** image files)

Alternatively, the C code can use the include file **datain.h** also created by the Matlab script **im2text.m** to embed the input image in the C source code as constant data (comment out the directive **#define LOAD_FROM_FILE** to ignore all the file operations and use the image included in the C code). This will be needed when running this application in the Zynq platform because there is no support for file handling.

Compile the program with the command **gcc aship.c –o aship** or just running **make**. The set of files provided in directory **./data** contain some 512x512 pixel sample images. If the source image is larger than 512x512 pixel a smaller 512x512 pixel window can be defined in **im2text.m** by selecting the pixel position of the top-left corner of the 512x512 window.

Run the program with **aship datain.txt dataout.txt** and then run the Matlab script **text2im.m** to display the output image and create a PNG image file.

The archive provided for this project is organized in 4 folders:

./doc	documentation files (includes this guide and the Vitis HLS user guide (UG1399)
./aship	C source files of the image processing application (compile with *make*)
./data	image file and text files created/used by the Matlab/Octave scripts
./Matlab	Matlab/Octave scripts to convert between bitmap images and text files used by the application

4. The project

The objective if this project is to build a hardware/software heterogeneous system to optimize the performance of the C application, adding a custom accelerator to the Zynq processing system. The custom IP blocks that implement the custom accelerator will be designed using the XILINX Vitis HLS high-level synthesis design flow.

The HLS optimization should consider various strategies to explore adequately the solution design space: code reorganization (eg. splitting, merging or specializing functions), choosing appropriate data types, loop optimization, interface design and partitioning arrays, either temporary arrays or the arrays used for the CPU interface.