High-Level Synthesis for Medical Image Processing on Systems on Chip: A Case Study

Fraser D Robinson*, Louise H Crockett†, William H Nailon‡, Robert W Stewart†
*Department of Biomedical Engineering, University of Strathclyde, Glasgow, UK
†Department of Electronic and Electrical Engineering, University of Strathclyde, Glasgow, UK
‡Department of Oncology Physics, Edinburgh Cancer Centre, Edinburgh, UK

***Abstract*—Adaptive radiotherapy is a technique intended to increase the accuracy of radiotherapy. Currently, it is not clinically feasible due to the time required to process the images of patient anatomy. Hardware acceleration of image processing algorithms may allow them to be carried out in a clinically acceptable timeframe. This paper presents the experiences encountered using high-level synthesis tools to design an accelerated segmentation algorithm for computed tomography images targeted for implementation on a System on Chip. Hardware coprocessors and their interfaces for optimal threshold generation and 3D mean filter algorithms were synthesised from C++ functions. Hardware acceleration significantly outperformed the software only implementation. The high-level synthesis tools allowed the rapid exploration of different design options. However, hardware design knowledge was still necessary in order to interpret the results effectively.**

I. Introduction

Adaptive radiotherapy aims to improve the accuracy of radiotherapy by adapting the original treatment plan based on computed tomography (CT) images of the patient at the time of treatment. Currently, this is not clinically feasible due to the time required to process the images and adapt the treatment plan. Hardware acceleration of these algorithms may allow them to be carried out in a clinically acceptable timeframe of the order of one minute.

Field programmable gate arrays (FPGA) have been shown to be capable of performing image processing algorithms faster than other types of processor[1], [2]. Hardware implementations of the image processing and dose calculation algorithms using FPGA may therefore make adaptive radiotherapy a clinical possibility. The relatively recent introduction of Systems on Chip (SoC), where a multi-core central processing unit (CPU) is closely integrated to FPGA fabric, present the possibility of achieving greater performance by allowing algorithms to be partitioned between hardware and software in order to maximize the benefits of each architecture.

A major drawback of hardware acceleration is the effort required to develop an efficient hardware design for the algorithm. The ability to partition sections of the design between hardware and software further increases the design options that need to be explored in order to produce the most efficient implementation. High-level synthesis tools aim to allow the designer to direct the synthesis of hardware from an algorithm description in a high-level software language. These tools are intended to increase the rate at which different design solutions can be explored. This is particularly useful for more complex algorithms such as those required for adaptive radiotherapy. It is the aim of the wider work of which this study forms a part, to accelerate algorithms for adaptive radiotherapy using hardware coprocessors.

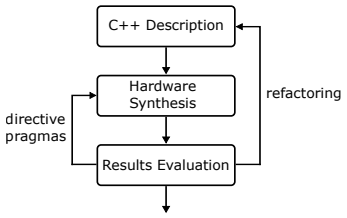

Fig. 1. High-level synthesis workflow.

This paper presents the experiences encountered using high-level synthesis tools to design an accelerated segmentation algorithm for segmenting CT images. The algorithm generated optimal thresholds and applied a local filtering operation. It was targeted for implementation on an SoC.

II. Methodology

The Xilinx SDSoC 2015.4 development environment, which incorporates the Vivado HLS 2015.4 tool, was used to develop a hardware accelerated implementation of a segmentation algorithm from a description written in C++. The design was targeted for implementation on a Xilinx Zynq Z7020 SoC, which integrates a dual-core CPU with an 85,000 logic cell FPGA.

CT image data of a phantom normally used for the purposes of quality assurance testing was obtained. The objective of the algorithm was to generate optimal thresholds in order to segment the CT image. The method for optimal threshold determination was based on Otsu's method[3]. A 3D mean filter was implemented to reduce noise in the image prior to applying the optimal thresholds to segment the image.

The Vivado HLS tool was used to synthesise hardware coprocessors for implementation in the FPGA fabric from C++ functions describing the threshold generation and mean filter algorithms. This process was an iterative one to achieve the best performance by directing the hardware synthesis through a combination of directive pragmas and code refactoring, as shown in figure 1.

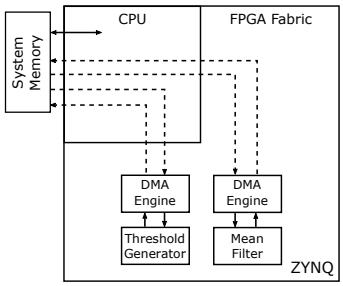

Fig. 2. Schematic diagram of segmentation design.

The SDSoC environment was used to generate efficient interfaces between the hardware coprocessors and the rest of the system. Similarly to the Vivado HLS tool, different interfacing options were explored through a combination of directive pragmas and code refactoring, as shown in figure 1. This allowed options such as interface port and data caching to be explored. The Zynq has two types of interface port available for transferring significant amounts of data; the accelerator coherency port (ACP) and the AXI FIFO interfaces (AFI). The ACP is capable of the same data throughput as a single AFI, but provides cache coherent access to system memory. The AFI do not provide cache coherency, however there are four of these available, which may be able to produce higher data throughput than the single ACP. Two different types of direct memory access (DMA) engines were investigated for transferring data between system memory and hardware. The simpler of the two DMA engines requires that the data is located in physically contiguous memory. The more complex DMA engine, the scatter-gather DMA, is able to access data distributed throughout system memory. When using the AFI with the simple DMA, the physically contiguous areas of system memory used to store data can be marked as either cacheable or non-cacheable. Figure 2 shows a schematic diagram of the system.

III. Results

Table I shows the algorithm execution times for the interface configurations tested. It also shows the average execution time for the algorithm being executed purely in software on the Zynq's CPU. It can be seen that all of the implementations using hardware acceleration significantly outperform the software implementation with the worst case being 13 times faster. The best performing implementation used the simple DMA engine and the ACP to produce around a 60 times speedup over software and a 4.5 times speedup over the worst performing hardware implementation.

TABLE I
Average algorithm execution times

Software		887.43ms	
Hardware			
DMA	ACP	AFI cached	AFI non-cached
Simple	14.82ms	16.72ms	66.81ms
Scatter-Gather	17.51ms	20.42ms	n/a

IV. Discussion

The high-level synthesis tools used here allowed the rapid exploration of different design options to implement hardware coprocessors to accelerate the segmentation algorithm. The Vivado HLS tool is incorporated within the SDSoC environment and the two complement each other. The Vivado HLS tool was used to produce optimal hardware designs at the C++ function level, while the SDSoC tool was used to optimise the design at a system level. While it can be seen from the results presented here that every implementation utilising the hardware coprocessors significantly outperformed the software implementation, it is also clear that the best performing implementation displays a substantial improvement over the other hardware coprocessor implementations. This highlights the advantages of fully exploring the design options with regards to software/hardware interfacing.

The Vivado HLS tool enabled a variety of hardware designs to be produced quickly through the specification of directive pragmas and code refactoring. This is particularly useful for complex algorithms where the burden of manually developing multiple solutions for comparison would be arduous. However, where the desired hardware structure is known, utilising existing pre-optimised blocks may produce more advantageous results and save time. This example also highlights the importance of hardware design knowledge when using high-level synthesis tools in order to interpret the results of synthesis and to know when a more efficient solution is possible.

V. Conclusion and Future Work

The use of high-level synthesis tools can greatly increase the rate at which efficient hardware designs can be produced. This is particularly the case for designs targeted for implementation on SoC devices where the range of design options is expanded by the ability to partition the design between hardware and software. High-level synthesis tools are not yet a panacea for good hardware design, however. Hardware design knowledge is still necessary in order to interpret the results effectively and guide the tools to synthesise the optimal design.

It is intended to extend this work by accelerating further algorithms pertinent to adaptive radiotherapy using hardware coprocessors and apply these to patient image data.

Acknowledgements

This work was generously supported by the Engineering and Physical Sciences Research Council grant number 1531683.

References

[1] F. Grüll and U. Kebschull, "Biomedical image processing and reconstruction with dataflow computing on FPGAs," in *24th Int. Conf. on Field Programmable Logic and Applications (FPL), Munich*. IEEE, Sep. 2014, pp. 1–2.

[2] O. Dandekar and R. Shekhar, "FPGA-Accelerated Deformable Image Registration for Improved Target-Delineation During CT-Guided Interventions," *IEEE Trans.Biomed.Circuits and Syst.*, vol. 1, no. 2, pp. 116–127, Jun. 2007.

[3] N. Otsu, "A Threshold Selection Method from Gray-Level Histograms," *IEEE Trans. Syst. Man Cybern.*, vol. 9, no. 1, pp. 62–66, 1979.