Implementing Image Processing Pipelines in a Hardware/Software Environment

Heather Quinn and Miriam Leeser
Northeastern University

Laurie Smith King
College of the Holy Cross

Abstract

Reconfigurable hardware devices, such as Field Programmable Gate Arrays (FPGAs), can be used to speed up image processing applications. Creating reconfigurable applications is not as straightforward as designing either software or hardware, since the application is intrinsically a hardware/software codesign. We have been developing a codesign environment that eases the process of creating reconfigurable applications for a Network of Workstations environment [1]. We have found that the performance of such an application is highly dependent on image size. While an application implemented on an FPGA can be one to two orders of magnitude faster than the application implemented in software, processing in hardware incurs additional costs that are not required for software. Some of these costs are hardware initialization costs, extra processing steps for easy processing of the border cases, and communication of the image to and from the reconfigurable device. The runtime of image processing applications varies with image size, so processing small images on an FPGA might not be efficient due to the additional overhead. Detailed profiling of hardware (with the extra costs) and software runtimes for a range of image sizes exposes a crossover point where all smaller images should be processed in software and all larger images should be processed in hardware. The goal of this project is to determine at runtime how to decide which implementation to execute for the optimal runtime for a given image.

Often a series of image processing transformations will be applied to an image. The series of transformations is called a pipeline and the individual transformations are called components. We assume that, for each component, there are hardware and software implementations and that the profiles for both implementations are known. We intend to predict profiles for pipelines so that the fastest implementation can be chosen at runtime. A pipeline profile is not easily characterized, though. A pipeline with N components will have 2^N implementations. Each implementation of a pipeline will have a different profile, because the extra costs are different for each combination of implementations. At any given image size, a different implementation might give the fastest runtime. Pipelines also have an extra associated cost, as the FPGA needs to be reprogrammed to satisfy a given implementation. Finally, a pipeline's profile is uniquely defined. Adding or removing even one component from the pipeline creates a completely different set of pipeline implementations and profiles. This problem of determining how to implement a pipeline is very close to the partitioning problem in hardware/software codesign.

Ideally, we will be able to predict the behavior of a pipeline from the hardware/software profiles of the components and profiling of the extra costs. Our initial assumption is that the components' runtimes and all extra costs are additive. Here are our assumptions about components and how components' implementations combine in pipelines:

1) All components are image in/image out.
2) Only one component is mapped to hardware at a time. The FPGA device needs to be reprogrammed for each hardware component being used.
3) Communication costs are incurred at the beginning and end of each hardware sub-sequence.
4) Some of the algorithms require that the edge be duplicated for processing the pixels on the boundary of the image. Software algorithms can duplicate the boundary when an edge pixel is being processed, but hardware algorithms are more efficient if the edge is duplicated before processing. We assume the edge duplication process (image padding) occurs in software before transferring the image into SRAM.
5) If there are back to back algorithms that require image padding, then there are these cases:

Case	Action	Implemented In
Hardware to Hardware	fix the image padding	Hardware
Hardware to Software	remove image padding	Software
Software to Hardware	pad the image	Software
Software to Software	no changes	N/A

An example of how these costs come into play in a pipeline are shown in Figure 1 for the pipeline Median Filter→Edge Detection. The software and hardware boundaries are denoted as light gray boxes. Every time a hardware or software boundary is crossed a communication cost is incurred. The other costs, such as padding the image and reprogramming the hardware, are shown as dark gray boxes. From this figure, we can see that assigning a component implementation to its hardware or software implementation is not a simple "yes/no" assignment, since all of the extra costs need to be weighed as well as hardware/software algorithm speedup. Figure 2 shows a table of software and hardware runtimes based on image sizes and the crossover point for the individual Median Filter and Edge Detection components. We can see from these equations that Edge Detection has a better hardware speedup than Median Filter, which gives Edge Detection a smaller crossover point than Median Filter. The profiles of the pipeline implementations show that an all hardware implementation, shown in Figure 1d, has the fastest runtime for images greater than 1680 pixels. There is a distinct crossover point at 1680 pixels and all images smaller run fastest if processed in an all software implementation, as shown in Figure 1a. The runtimes of the pipeline implementations can be seen in Figure 3. Given the rules discussed above for predicting the pipeline's profile, we were able to predict the behavior with up to 14% accuracy.

Choosing the proper implementation of a pipeline can have a direct effect of the performance of the pipeline. Overhead costs cannot be ignored. With known component profiles, the implementation of a pipeline can be predicted so that profiling the complete pipeline will not be necessary. We have presented a short example that shows how the Median Filter and Edge Detection components work together in an image processing pipeline and the extra costs incurred on the hardware/software boundaries.

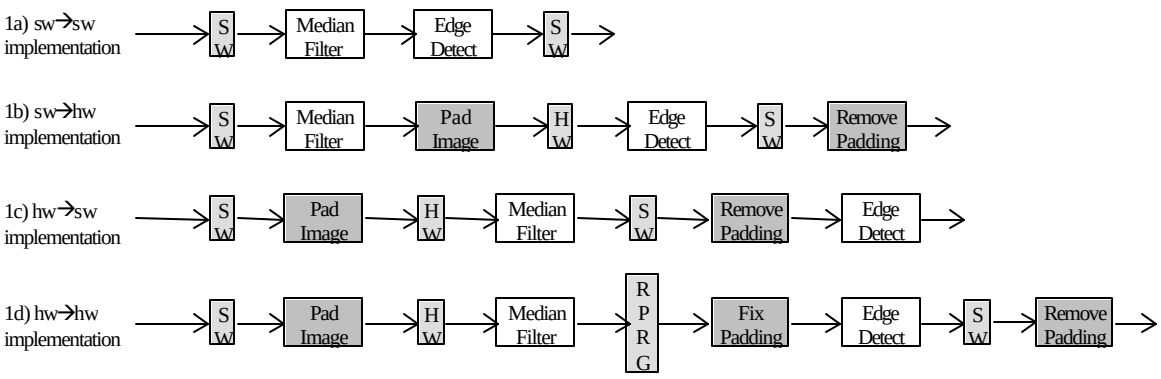

Figure 1: Four HW/SW Assignments of Median Filter → Edge Detection

Algorithm	Software Runtime (ms)	Hardware Runtime (ms)	Crossover Point
Median Filter	0.083x + 169.72	0.0025x + 532.97	3600 pixels
Edge Detection	0.1502x + 65.061	0.0032x + 357.93	1680 pixels

Figure 2: Software/Hardware Runtimes as a Function of the Number of Pixels (x) and the Crossover Point for Median Filter and Edge Detection

Permutation	Runtime (ms)
Software→Software	0.2356x + 134.87
Software→Hardware	0.0906x + 358.73
Hardware→Software	0.1519x + 253.69
Hardware→Hardware	0.0036x + 459.22

Figure 3: Runtime as a Function of the Number of Pixels (x) for Each Implementation of the Median Filter→ Edge Detection Pipeline

REFERENCES

[1] L.A. Smith King, Heather Quinn, Miriam Leeser, Demetris Galatopoullos, and Elias Manolakos, *Runtime Execution of Reconfigurable Hardware in a Java Environment*, International Conference on Computer Design (ICCD), September 2001, pp 380-5.