In Proceedings of the IEEE Biomedical Circuits and Systems Conference, November 2006.

RECONFIGURABLE IMAGE REGISTRATION ON FPGA PLATFORMS

Mainak Sen[1], *Yashwant Hemaraj*[1,2], *Shuvra S. Bhattacharyya*[1], *Raj Shekhar*[2]

[1]Department of Electrical and Computer Engineering,
University of Maryland, College Park, MD, 20742, USA.
[2]Department of Diagnostic Radiology,
University of Maryland School of Medicine, Baltimore, Maryland, USA

ABSTRACT

Image registration is computationally intensive, and hence difficult to implement in real-time. In recent efforts, image registration algorithms have been implemented in field-programmable gate array (FPGA) technology to improve performance, while providing programmability and dynamic reconfigurability. In this paper, we present a novel architecture for dynamically-reconfigurable image registration, along with details on the methodology used to derive the architecture. Unlike previous FPGA implementations for image registration, the architecture developed in this paper tunes its parallel processing structure adaptively based on relevant characteristics of the input images.

1. INTRODUCTION

Image registration is a fundamental requirement in medical imaging and an essential first step for meaningful multimodality image fusion and accurate serial image comparison. It is also a prerequisite for creating population-specific atlases and atlas-based segmentation. Despite the existence of powerful algorithms and clear evidence of clinical benefits they can bring, the clinical utilization of image registration remains limited. The slow speed (i.e., long execution time) of fully automatic image registration algorithms especially for 3D images has much do with this lack of clinical integration and routine use.

This paper focuses on image registration algorithms that must be executed under real-time performance constraints. In some cases, for example, visual accessories in surgical applications must meet stringent performance criteria in order to provide adequate response and interactivity to surgeons. Hardware implementation is one way to speed-up applications over existing software implementations. However, designing hardware requires significantly higher turnaround time, and is more error prone compared to software implementation. Systematic methods based on precise application modeling abstractions and associated hardware mapping techniques are therefore desirable, since such methods make the design process more structured, while at the same time exposing opportunities for system-level performance optimization.

In this paper, we develop such a structured design methodology specifically in the context of image registration. Our approach starts with capturing the high level algorithm structure through a carefully-designed, coarse-grain dataflow model of computation. We then develop methods to analyze this dataflow representation to systematically provide a hardware implementation that dynamically optimizes its processing structure in response to the particular image registration scenario in which it operates.

Image registration algorithms can be mapped to field programmable gate arrays (FPGAs) for efficient execution as we have reported earlier [4]. Specifically, we reported a new architecture for mutual information-based rigid-body image registration, created a proof-of-concept implementation, and achieved greater than an order of magnitude speedup for registration of multimodality images (MR, CT and PET) of the human head, PET and CT images of the thorax and abdomen, and 3D ultrasound and SPECT images of the heart [15]. As a demonstration of single modality image registration, we used the accelerated implementation also for registration of pre- and post-exercise 3D ultrasound images of the heart [15].

Several clinical applications to benefit from the proposed work include whole-body PET/CT registration [14], virtual colonoscopy [3] and image registration tasks involving pre- and intra-operative images in the context of image-guided surgeries [5]. The overall benefits will extend to numerous other applications being developed by researchers worldwide.

In this paper, we build on our experience with architectures for image registration by developing and applying novel dataflow-based models and analysis methods of image registration applications. These methods provide a framework for mapping and high-level optimization of these applications onto embedded architectures. Using this framework, we develop a new dynamically reconfigurable architecture for image registration that optimizes its processing structure adaptively based on relevant characteristics of its input.

2. BACKGROUND

2.1 Dataflow modeling

In the dataflow model of computation, an application is represented as a directed graph in which vertices (*actors*) correspond to computational modules, and edges correspond to first-in, first-out buffers that queue data as it passes between actors. In coarse-grain dataflow, actors can be of arbitrary complexity — usually, coarse-grain actors represent computations of intermediate complexity, on the order of 10-100 lines of equivalent C code. Dataflow is a widely used in the design of signal processing applications because it is an intuitive mode for algorithm designers to work with, and it also exposes high-level application structure that is useful for analysis, verification, and optimization of implementations [1].

2.2 FPGA technology

In this paper, we target our hardware optimization framework to an FPGA device, the Altera Stratix EP1S10F780C5. A major advantage of FPGA technology is the potential for dynamic reconfiguration of the processing structure. In the context of FPGA implementation, dataflow is especially useful because it effectively exposes application concurrency, and facilitates configuration of and mapping onto parallel resources. This opens up design space exploration opportunities for meeting different user constraints, and achieving different implementation trade-offs. However, streamlining the use of dataflow technology is challenging because it requires careful mapping of application characteristics into the graphical and actor-based modeling abstractions of dataflow, and because the associated optimization issues, while exposed more effectively for signal processing applications compared to other modeling abstractions, are usually NP-complete to solve exactly [1]. This paper addresses these challenges for the image registration domain.

2.3 Forms of dataflow for signal processing

The synchronous dataflow (SDF) model [8] has strong compile time predictability properties, and is the most mature form of dataflow for signal processing system design. SDF-based hardware synthesis has been explored in [13][16]. However, the SDF model is highly restrictive for many computer vision applications because the model cannot handle data-dependent rates of data transfer between actors [11].

Various extensions and alternatives to SDF have been developed to provide for more flexible application modeling. For example, a cyclo-static dataflow (CSDF) [2] graph can accommodate multi-phase actors that exhibit different consumption and production rates during different phases, as long as the variations across phases form statically-known, periodic patterns. This provides for more flexibility, but still does not permit data-dependent production or consumption patterns.

More recently, a meta-modeling technique called homogeneous parameterized dataflow (HPDF) [12] was proposed in which actor behavior can be adapted in a structured way through dynamically-adjusted parameter values. While HPDF allows significant flexibility in dynamically changing actor behavior, the restrictions imposed in the model ensure that HPDF subsystems are homogeneous — in terms of the average rate at which their constituent actors execute — across any particular level in modeling hierarchy. This permits efficient scheduling and resource allocation for actors, as well as verification of bounded memory requirements and deadlock-free operation, which are useful safety properties to guarantee in embedded hardware and software systems.

HPDF is especially useful because it is a meta-modeling technique. Hierarchical actors in an HPDF model can be refined using any dataflow modeling semantics that provides a well-defined notion of subsystem iteration. For example, a hierarchical HPDF actor can have SDF, CSDF, or HPDF actors as its constituent modules.

3. IMAGE REGISTRATION

Image registration is the process of aligning multiple images that represent the same feature. Medical image registration concentrates on aligning two or more images that represent the same anatomy from different angles, and are obtained at different times. Image registration is a key feature for a variety of imaging techniques and there two main algorithmic approaches — *linear* and *elastic*. A linear transformation can be approximated by a combination of rotation, transformation and scaling coefficients while an elastic approach is based on nonlinear continuous transformations, and is implemented by finding correlations among meshes of control points. Our study concentrates on the linear approach.

Real-time image registration is essential in the medical field for enabling image-guided treatment procedures, and pre-operative treatment planning.

There are many methods for 3D image registration. Algorithms based on voxel similarity fulfill the above criteria better than feature-based approaches [7]. Of them, the most commonly used technique is image registration based on *mutual information* [9]. Mutual information (MI) methods are robust and can work effectively with multi-modal images.

3.1 MI-based image registration

MI-based image registration relies on maximizing the mutual information between two images. Mutual information is a function of two 3-D images and a transformation between them. The transformation matrix contains the information about the rotation, scaling shear and translations that need to be applied to one of the images in order to map it completely to the other image so that a one-to-one correspondence is established between the coordinates of the images. A cost function based on the mutual information is calculated from the individual and joint histograms. The transformation that maximizes the cost function is viewed as the optimum transformation. The goal MI-based image registration is then to find this optimal transformation:

$$\hat{T} = \arg max_T MI(RI(x,y,z), FI(T(x,y,z))) ,$$

Here, *RI* is the reference image, and *FI* is the floating image (the image that is being registered).

3.2 Computation of mutual information

Mutual information is calculated from individual and joint entropies using the following equations:

$$MI(RI, FI) = H(RI) + H(FI) - H(RI, FI) ,$$

$$H(RI) = -\Sigma p_{RI}(a)\log p_{RI}(a) , H(FI) = -\Sigma p_{FI}(a)\log p_{FI}(a) ,$$

$$\text{and } H(RI, FI) = -\Sigma p_{RI,FI}(a,b)\log p_{RI,FI}(a,b) ,$$

where $H(RI)$, $H(FI)$, $H(RI, HI)$ and $MI(RI, FI)$ denote the reference image entropy, floating image entropy, joint entropy and mutual information between the two images.

The mutual histogram represents the joint intensity probability distribution. The individual voxel intensity probabilities are the histograms of the reference and floating images in the region of overlap of the two images for an applied transformation.

MI calculation is a memory-intensive task that is not well suited to cache-based memory architectures. The calculation of mutual information starts with the accumulation of the mutual histogram values to the mutual histogram memory while every coordinate is being transformed (MH update stage). This is followed by the MI calculation stage where the values stored in the mutual histogram memory are used to find the individual and joint entropies described above.

In the MH update stage, voxel coordinates are transformed by a transformation matrix representing a linear transformation. Since the new coordinates do not always coincide with the location of a voxel in the reference image, a partial volume interpolation scheme [10] is used to update the MH memory based on eight interpolation weights for the eight neighboring voxels. It has been shown that the size of the mutual histogram can be selected as 64x64 for 8 bit images.

3.3 Optimization

The image registration algorithm calculates the transformation matrix for which the mutual information between the images is maximum. Initially, a small number of test transformations is applied. The values of these transformations and the MI values are stored in an *optimizer*. The optimizer outputs the values of the new transformation depending on the values of the mutual histogram in the previous iterations. Optimization of the transformation parameters depends on the nature of the images and the amount of misalignment between the two images. Some methods, such as the simplex method, provide faster convergence than the others. In the simplex method, in order to optimize a transformation with m parameters, the optimizer needs to store $(m+1)$ previous values. There is a trade-off between the convergence time and the complexity of the optimizer.

4. APPLICATION MODELING

In this section, we present a hierarchical dataflow representation of MI-based image registration in terms of the HPDF meta-modeling approach integrated with CSDF for modeling lower-level, multi-phase interactions between actors (see Section 2.3). Figure 1 shows our top level HPDF model of the application. Here, "$(m+1)D$" represents $(m+1)$ units of *delay*; each unit of delay is

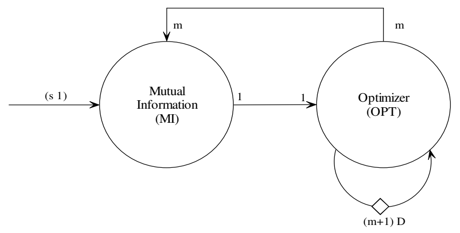

Fig 1. Top level model of image registration application.

analogous to the z^{-1} operator in signal processing, and is typically implemented by placing an initial data value on the corresponding dataflow edge. The MI actor consumes one data value (*token*) on every execution. This token contains co-ordinates of the reference image and the floating image. After s executions, where s denotes the size of the image, the MI actor produces the entropy between the reference and floating images. This value is then sent to the optimizer as a single token.

The optimizer, which stores the previous $(m+1)$ values to perform a simplex optimization of an m-parameter transformation vector, sends m tokens to the MI actor. Since m can vary depending on the number of parameters used to represent the desired transformation, the associated edge represents a variable-rate edge of the HPDF graph.

The internal representation of the hierarchical MI actor is shown in Figure 2. Here, “RI Mem” consumes one token (coordinates) and produces one token (intensity values at the input coordinates), and “Coordinate Transform” produces one token, which represents the transformed coordinates. If this voxel is valid (i.e., the voxel coordinate falls within the floating image coordinates boundary), it is passed on to the “Weight Calculator” (WC) and “FI Mem.”

Now since all voxels may not be valid, r tokens are produced from the “Is Valid” actor. This actor also produces r tokens on the edge that connects it to “MH Memory” — specifically, it passes a token from “RI Mem” only if a valid voxel results from the transformation on input coordinates. For every input token in WC and FI Mem, eight output tokens are produced (as described in Section 3.2). The corresponding eight intensity locations in the “MH Memory” are updated based on the tokens produced by the WC actor.

After all coordinates are processed, which occurs during the the first $8r$ *phases* of the MH Memory actor, a 64x64 block of tokens is sent to the Entropy Calculator actor, which consumes all of these tokens, and produces a single token that contains the entropy value corresponding to the transformation applied.

The overall model shows potential for parallel hardware mapping at various levels of abstraction. For example, extensive “intra-pixel” (within the processing structure for a single pixel) parallelism is possible for the MH memory and adder. The dataflow model also exposes inter-pixel parallelism, which leads to another set of useful parallel implementation considerations. The architecture developed in this paper applies both intra- and inter-pixel parallelism, and balances these forms of parallelism adaptively based on input characteristics.

5. ACTOR IMPLEMENTATION

The lowest level (non-hierarchical) actors in our dataflow-based design are implemented in Verilog. As an illustration of Verilog-based actor in our design, Figure 3 shows the code corresponding to the Adder actor. An interesting point to note in this code example is that by analyzing the dataflow behavior, we can ensure that the interface code between the adder and the weight calculator places the correct weight at every clock cycle in the input buffer labeled ‘weight’. This illustrates how using dataflow as a high-level modeling abstraction helps to structure the hardware implementation process, and makes the hardware description language (HDL) code modular and reliable.

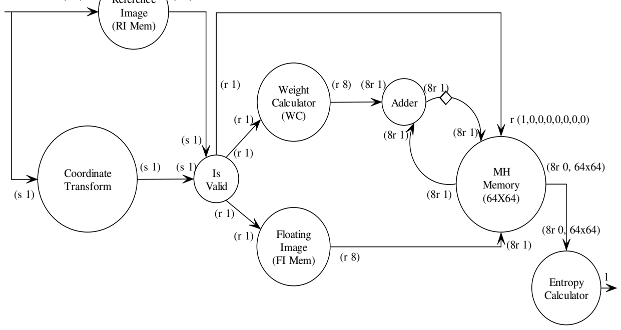

Fig 2. Dataflow model of Mutual Information subsystem.

6. DYNAMIC RECONFIGURATION

In this section, we elaborate on the dynamic reconfigurability of our proposed architecture and present results of our design.

Based on earlier studies [6] on hardware implementation of image Registration algorithm, we see that the image registration application has a memory bottleneck in the “MH memory” part of the MI calculator (see Figure 2). One improvement to alleviate this bottleneck is to map the “MH memory” actor into eight parallel hardware modules and provide separate “adder” modules for these eight memory modules.

A dataflow representation of the resulting architecture is shown in Figure 4. As evident from Figure 4, this architecture reduces the processing time but increases the memory requirements. This aspect is elaborated on in [6], which develops the dataflow model and memory management aspects in depth for a statically-configured version of our design (i.e., a version that is fixed beforehand, and does not employ dynamic reconfiguration).

In addition to the degree of memory parallelism that is employed, another parameter that affects the run-time of our architecture is the percentage of valid voxels (PVV) that results from a transformation on the floating image. The PVV is input-dependent. As the PVV increases, the run-time increases. This trend is illustrated in Figure 6.

```
/* global definitions in top.v */
reg [imsize+fracwidth-1:0] mh [0:4096];
reg [imsize+fracwidth-1:0] edgeweights [0:numweights-1];

/*one example module */
module mhupdate
#(parameter imsize = 8,
parameter fracwidth = 8,
parameter numweights = 8,
parameter lognumweights = 3)
(input [imsize-1:0] rival,fival,
input [imsize+fracwidth-1:0] weight,
input clk);
reg [11:0] currval;
reg [lognumweights:0]counter;

always @(posedge resetall)
   counter <= 0;

always @(posedge clk)
   begin
      if(counter < numweights) begin
         mh[currval] <= mh[currval] + weight;
         currval[5:0] <= rival[imsize-1:imsize-6];
         currval[11:6] <= fival[imsize-1:imsize-6];
         counter <= counter + 1;
      end
      else
         counter <= 0;
   end
```

Fig 3. Example code (partial) of the Adder from Figure 2

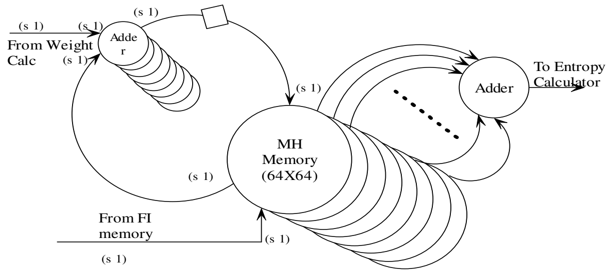

Fig 4. Parallel memory dataflow structure.

Furthermore, as the PVV increases, memory access becomes more of a bottleneck, and gradually, it becomes more performance-effective to trade-off inter-pixel parallelism in the architecture for intra-pixel parallelism in the form of multiple (parallel) memories that alleviate the memory bottleneck. This trend is demonstrated by the data in Figure 5, which compares the performance, for different PVV values, of a 1 voxel-8 memory architecture (intra-pixel parallelism) to a 7 voxel architecture with 1 memory module per voxel (inter-pixel parallelism) architecture. The value of 7 is selected here because for the targeted FPGA device, the area of a 1 voxel-8 memory architecture is around 7 times that of a 1 voxel-1 memory architecture. The units of performance in Figure 5 are nanoseconds per voxel per co-ordinate transform and the frequencies of operation of the different memory architectures vary between 74 MHz and 85 MHz for various configurations.

We note in Figure 5, considering the area constraint, performance of 1 voxel-1 memory architecture is better than that of a 1 voxel-8 memory architecture, however this trend changes as the voxel validity percentage increases. Therefore, our image registration architecture monitors the PVV metric at run-time and dynamically reconfigures the architecture from inter-pixel parallelism mode to intra-pixel parallelism mode once the transition point of around 50% PVV is observed. This can be viewed as a periodic (once per image), PVV-driven re-scaling of the subsystem shown in Figure 4.

Note that the optimal transition point is in general image-dependent, and our use of a fixed value of 50% as a transition point is therefore a heuristic approach. Dynamically determining the transition point is a useful topic for further investigation.

7. CONCLUSION

In this paper, we have presented the motivation, derivation, and FPGA mapping of an architecture for dynamically-reconfigurable image registration. We have demonstrated the ability of the architecture to strategically adapt its parallel processing configuration in response to relevant image characteristics, and for this purpose we have formulated the PVV metric, which represents the percentage of valid voxels that results from a transformation on the given floating image. Useful directions for further work include integration of the modeling insights and dynamic reconfiguration techniques developed in this paper with relevant design aspects of the FAIR architecture [4], which provides for efficient statically-configured image registration hardware.

Voxel Validity	Performance of 7 1 voxel-1 memory	Performance of 1 1voxel-8 memory
10%	6.39/7 = 0.91	2.54
50%	17.8/7= 2.54	2.91
90%	27.82/7 = 3.97	2.5
100%	30.08/7 = 4.29	2.33

Fig 5. Comparison of intra- versus inter-pixel parallelism modes for different PVV values.

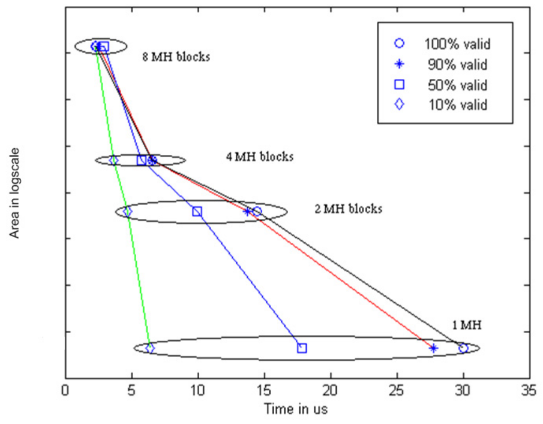

Fig 6. Trade-offs for different PVV values.

REFERENCES

[1] S. S. Bhattacharyya, R. Leupers, and P. Marwedel. Software synthesis and code generation for DSP. *IEEE Trans. on Circuits and Systems — II: Analog and Digital Signal Processing*, 47(9):849-875, September 2000.
[2] G. Bilsen, M. Engels, R. Lauwereins, and J. A. Peperstraete. Cyclo-static dataflow. *IEEE Trans. on Signal Processing*, 44(2):397-408, February 1996.
[3] C. R. Castro-Pareja, B. Daly, and R. Shekhar. Elastic registration using 3D chainmail. In *Proc. of SPIE (Medical Imaging)*, 2006.
[4] C. Castro-Pareja, J M. Jagadeesh, and R. Shekhar. FAIR: A hardware architecture for real-time 3-d image registration. *IEEE Trans. on Information Technology in Biomedicine*, 7(4):426-434, 2003.
[5] O. Dandekar, V. Walimbe, K. Siddiqui, and R. Shekhar. Image registration accuracy with low-dose CT: How low can we go? In *Proc. of IEEE International Symposium on Biomedical Imaging*, pages 502-505, 2006.
[6] Y. Hemaraj, M. Sen, R. Shekhar, and S. S. Bhattacharyya. Model-based mapping of image registration applications onto configurable hardware. In *Proc. IEEE Asilomar Conference on Signals, Systems, and Computers*, October, 2006. To appear.
[7] M. Holden, D. Hill, E. Denton, J. Jarosz, T. Cox, T. Rohlfing, J. Goodey, and D. Hawkes. Voxel similarity measures for 3D serial MR brain image registration. *IEEE Trans. on Medical Imaging*, pages 94-102, 2000.
[8] E. Lee and D. Messerschmitt. Synchronous Data Flow. *Procs of the IEEE*, September 1987.
[9] J. P. W. Pluim, J. B. A. Maintz, M. A. Viergever, Mutual Information based Registration of Medical Images: A Survey. *IEEE Trans on Medical Imaging*, 2003.
[10] F. Maes, D. Vandermeulen and P. Suetens, Medical image registration using mutual information, *Proc. IEEE* 19, 1699 (2003).
[11] M. Sen, I. Corretjer, F. Haim, S. Saha, J. Schlessman, S. S. Bhattacharyya, and W. Wolf. Computer vision on FPGAs: Design methodology and its application to gesture recognition. In *Proc. IEEE Workshop on Embedded Computer Vision*, pages CD-ROM version, 8 pages, San Diego, California, June 2005.
[12] M. Sen, S. S. Bhattacharyya, T. Lv, and W. Wolf. Modeling image processing systems with homogeneous parameterized dataflow graphs. In *Proc. International Conference on Acoustics, Speech, and Signal Processing*, pages V-133-V-136, March 2005.
[13] M. Sen and S. S. Bhattacharyya. Systematic exploitation of data parallelism in hardware synthesis of DSP applications. In *Procs of the International Conference on Acoustics, Speech, and Signal Processing*, pages V-229-V-232, May 2004.
[14] R. Shekhar, V. Walimbe, S. Raja, V. Zagrodsky, M. Kanvinde, G. Wu, and B. Bybel. Automated three-dimensional elastic registration of whole-body PET and CT from separate or combined scanners. *Journal of Nuclear Medicine*, 46(9):1488-1496, 2005.
[15] R. Shekhar R, V. Zagrodsky, C. R. Castro-Pareja, V. Walimbe, and J. M. Jagadeesh. High-speed registration of three- and four-dimensional medical images by using voxel similarity. *RadioGraphics*, 23(6):1673-1681, 2003.
[16] M. Williamson. Synthesis of Parallel Hardware Implementations from Synchronous Dataflow Graph Specifications. *Ph.D. Thesis, University of California at Berkeley,* May 1998.