Multiobjective Optimization of FPGA-Based Medical Image Registration

Omkar Dandekar[1,2], William Plishker[1,2], Shuvra Bhattacharyya[1]

Raj Shekhar[1,2]

[1]*Department of Electrical and Computer Engineering, University of Maryland, College Park, MD 20742 {omkar, plishker, ssb}@umd.edu*

[2]*Department of Diagnostic Radiology, University of Maryland, Baltimore, MD 21201 rshekhar@umm.edu*

Abstract

With a multitude of technological innovations, one emerging trend in image processing, and medical image processing, in particular, is custom hardware implementation of computationally intensive algorithms in the quest to achieve real-time performance. For reasons of area-efficiency and performance, these implementations often employ limited-precision datapaths. Identifying effective wordlengths for these datapaths while accounting for tradeoffs between design complexity and accuracy is a critical and time consuming aspect of this design process. Having access to optimized tradeoff curves can equip designers to adapt their designs to different performance requirements and target specific devices while reducing design time. This paper presents a multiobjective optimization strategy developed in the context of field-programmable gate array–based implementation of medical image registration. Within this framework, we compare several search methods and demonstrate the applicability of an evolutionary algorithm–based search for efficiently identifying superior multiobjective tradeoff curves. This strategy can easily be adapted to a wide range of signal processing applications, including areas of image and video processing beyond the medical domain.

1. Introduction

An emerging trend in real-time signal processing systems is to accelerate computationally intensive algorithmic components by mapping them to custom or reconfigurable hardware platforms, such as application-specific integrated circuits (ASICs) and field-programmable gate arrays (FPGAs). Most of these algorithms are initially developed in software using floating-point representation and later migrated to hardware using finite precision (e.g., fixed-point representation) for achieving improved computational performance and reduced hardware cost. These implementations are often parameterized, so that a wide range of finite precision representations can be supported [1] by choosing an appropriate wordlength for each internal variable. As a consequence, the accuracy and hardware resource requirements of such a system are functions of the wordlengths used to represent the internal variables. Determining an optimal wordlength configuration has been shown to be NP-hard [2] and can take up to 50% of the design time for complex systems [3]. Moreover, a single optimal solution may not exist, especially in the presence of multiple conflicting objectives. In addition, a new configuration generally needs to be derived when the design constraints are altered.

The problem of finding optimal wordlength configurations can be formulated as a multiobjective optimization, where different objectives — for example, accuracy and area — generally conflict with one another. Although this approach increases the complexity of the search, it can find a set of Pareto-optimized configurations representing strategically-chosen tradeoffs among the various objectives. This allows a designer to choose an efficient configuration that satisfies given design constraints and provides ease and flexibility in modifying the design configuration as the constraints change.

An optimum wordlength configuration can be identified by analytically solving the quantization error equation as described in [4-8]. This analytical representation, however, can be difficult to obtain for complex systems. Techniques based on local search or gradient-based search [9] have also been employed, but these methods are limited to finding a single feasible solution as opposed to an optimized tradeoff curve. An exhaustive search of the entire design space is guaranteed

978-0-7695-3307-0/08 $25.00 © 2008 IEEE
DOI 10.1109/FCCM.2008.50

to find Pareto-optimal configurations. Execution time for such exhaustive search, however, increases exponentially with the number of design parameters, making it unfeasible for most practical systems. Methods that transform this problem into a linear programming problem have also been reported [4], but these techniques are limited to cases in which the objectives can be modeled as linear functions of the design parameters. Other approaches based on linear aggregation of objectives may not find proper Pareto-optimal solutions when the search space is nonconvex [10]. Techniques based on evolutionary methods have been shown to be effective in searching large search spaces in an efficient manner [11, 12]. Furthermore, these techniques are inherently capable of performing multipoint searches. As a result, techniques based on evolutionary algorithms (EA) have been employed in the context of multiobjective optimization (SPEA2 [13], NSGA-II [14]).

This article presents a novel multiobjective optimization strategy developed in the context of FPGA-based implementation of medical image registration. The tradeoff between FPGA resources (area and memory) and implementation accuracy is explored, and Pareto-optimized solutions are identified. This analysis is performed by treating the wordlengths of the internal variables as design variables. We also compare several search methods for finding Pareto-optimized solutions and demonstrate the applicability of search based on evolutionary techniques for efficiently identifying superior multiobjective tradeoff curves. This optimization strategy can easily be adapted to a wide range of signal and image processing applications.

This paper is organized as follows. Section 2 provides background on image registration and outlines an architecture for its FPGA-based implementation. The formulation of the multiobjective optimization and various search methods to find Pareto-optimized solutions are described in Section 3. Section 4 presents experimental results and compares various search methods. In Section 5, related work for optimum wordlength search and multiobjective optimization is presented. Section 6 concludes the paper.

2. Image registration

Medical image registration is the process of aligning two images that represent the same anatomy at different times, from different viewing angles, or using different imaging modalities. This method attempts to find the transformation ($\hat{T}$) that optimally aligns a reference image (RI) with coordinates x, y, and z and a floating image (FI) under an image similarity measure ($\mathbb{F}$):

$$\hat{T} = \arg\max_{T} \mathbb{F}(RI(x,y,z), FI(T(x,y,z))). \quad (1)$$

Many image similarity measures, such as the sum of squared differences and cross correlation, have been used, but mutual information (MI) has recently emerged as the preferred similarity measure. MI-based image registration has been shown to be robust and effective in multimodality image registration [15]. However, this form of registration typically requires thousands of iterations (MI evaluations), depending on image complexity and the degree of initial misalignment between images. Castro-Pareja et al. [16] have shown that, calculation of MI for different candidate transformations is a factor limiting the performance of MI-based image registration. We have, therefore, developed an FPGA-based architecture for accelerated calculation of MI [17], which is capable of computing MI 40-times faster as compared to software implementation.

2.1. FPGA-based implementation of mutual information calculation

During the execution of image registration using this architecture, the optimization process is executed from a host workstation. The host provides a candidate transformation, while the FPGA-based implementation applies it to the images and performs the corresponding MI computation. The computed MI value is then further used by the host to update the candidate transformation and eventually find the optimal alignment between the RI and FI. Figure 1 shows the top-level block diagram of the aforementioned architecture. The important modules in this design are described in the following subsections.

2.1.1. Voxel counter. Calculation of MI requires processing each voxel in the RI. In addition, because the implemented algorithm processes the images on a subvolume basis, RI voxels within a 3D neighborhood corresponding to an individual subvolume must be processed sequentially. The host programs the FPGA-based MI calculator with subvolume start and end addresses, and the voxel counter computes the address corresponding to each voxel within that subvolume in z–y–x order.

2.1.2. Coordinate transformation. The initial step in MI calculation involves applying a candidate transformation (T), to each voxel coordinate ($\vec{v}_r$) in the RI to find the corresponding voxel coordinates in the FI ($\vec{v}_f$). This is mathematically expressed as:

$$\vec{v}_f = T \cdot \vec{v}_r . \quad (2)$$

The deformation model employed is a six-parameter rigid transformation model and is represented using a 4×4 matrix. The host calculates this matrix based on the current candidate transformation provided by the

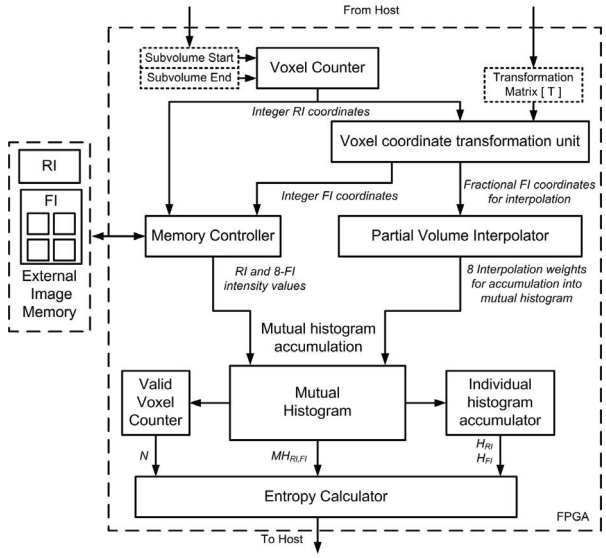


Figure 1: Top-level block diagram of FPGA-based architecture for MI calculation

optimization routine and sends it to the MI calculator. A fixed-point representation is used to store the individual elements of this matrix. The coordinate transformation is accomplished by a simple matrix multiplication.

2.1.3 Partial volume interpolation. The coordinates mapped in the FI space ($\vec{v}_f$) do not normally coincide with a grid point (integer location), thus requiring interpolation. Nearest neighbor and trilinear interpolation schemes have been used most often for this purpose; however, partial volume (PV) interpolation, introduced by Maes et al. [15] has been shown to provide smooth changes in the histogram values with small changes in transformation. The reported architecture consequently implements PV interpolation as the choice of interpolation scheme. $\vec{v}_f$ in general, will have both fractional and integer components and will land within an FI neighborhood of size $2 \times 2 \times 2$. The interpolation weights required for the PV interpolation are calculated using the fractional components of $\vec{v}_f$. Fixed-point arithmetic is used to compute these interpolation weights. The corresponding floating voxel intensities are fetched by the image controller in parallel using the integer components of $\vec{v}_f$. The image controller also fetches the voxel intensity corresponding to $\vec{v}_r$. The MH then must be updated for each pair of reference and floating voxel intensities (eight in all), using the corresponding weights computed by the PV interpolator.

2.1.4. Image memory access. The typical size of 3D medical images prevents the use of high-speed memory internal to the FPGA for their storage. Between the two images, the RI has more relaxed access requirements, because it is accessed in a sequential manner (in *z–y–x* order). This kind of access benefits from burst accesses and memory caching techniques, allowing the use of modern dynamic random access memories (DRAMs) for image storage. For the architecture presented, both the RI and FI are stored in separate logical partitions of the same DRAM module. Because the access to the RI is sequential and predictable, the architecture uses internal memory to cache a block of RI voxels. Thus, during the processing of that block of RI voxels, the image controller has parallel access to both RI and FI voxels. The RI voxels are fetched from the internal FPGA memory, whereas the FI voxels are fetched directly from the external memory.

The FI, however, must be accessed randomly (depending on the current transformation *T*) and eight FI voxels (a $2 \times 2 \times 2$ neighborhood) must be fetched for every RI image voxel to be processed. To meet this memory access requirement, the reported architecture employs a memory addressing scheme similar to the cubic addressing technique reported in the context of volume rendering [18]. A salient feature of this technique is that it allows simultaneous access to the entire $2 \times 2 \times 2$ voxel neighborhood. The reported architecture implements this technique by storing four copies of the FI and taking advantage of the burst mode accesses native to modern DRAMs. The image voxels are arranged sequentially such that, performing a size two burst fetches two adjacent 2×2 neighborhood planes, thus making the entire neighborhood available simultaneously. The image intensities of this neighborhood are then further used for updating the MH.

2.1.5. Updating the mutual histogram. For a given RI voxel (*RV*), there are eight intensity pairs ($RV, FV_0 : FV_7$) and corresponding interpolation weights. Because the MH must be updated (read–modify–write) at these eight locations, this amounts to 16 accesses to MH memory for each RI voxel. This high memory access requirement is handled by using the high-speed, dual-ported memories internal to the FPGA to store the MH. The operation of updating the MH is pipelined and, hence, read-after-write (RAW) hazards can arise if consecutive transactions attempt to update identical locations within the MH. The reported design addresses this issue by introducing pre-accumulate buffers, which aggregate the weights from all conflicting transactions. Thus, all the transactions leading to a RAW hazard are converted into a single update to the MH, thereby eliminating any RAW hazards.

While the MH is being computed, the individual histogram accumulator unit computes the histograms for the RI and FI. These individual histograms are also stored using internal, dual-ported memories. The valid voxel

counter module keeps track of the number of valid voxels accumulated in the MH and calculates its reciprocal value. The resulting value is then used by the entropy calculation unit for calculating the individual and joint probabilities.

2.1.6. Entropy Calculation. The final step in MI calculation is to compute joint and individual entropies using the joint and individual probabilities, respectively. To calculate entropy, it is necessary to evaluate the function $f(p) = p \cdot ln(p)$ for all the probabilities. As each probability p takes values between [0,1], the corresponding range for the function $f(p)$ is $[-e^{-1},0]$. Thus, $f(p)$ has a finite dynamic range and is defined for all values of p. Several methods for calculating logarithmic functions in hardware have been reported [19], but of particular interest is the multiple lookup table (LUT)–based approach introduced by Castro-Pareja et al. [20]. This approach minimizes the error in representing $f(p)$ for a given number and size of LUTs and, hence, is accurate and efficient. Following this approach, the reported design implements $f(p)$ using multiple LUT–based piecewise polynomial approximation.

3. Multiobjective optimization

The aforementioned architecture is designed to accelerate the calculation of MI for performing medical image registration. We have demonstrated this architecture to be capable of offering execution performance superior to that of a software implementation [17]. The accuracy of MI calculation (and by extension, that of image registration) offered by this implementation, however, is a function of wordlengths chosen for the internal variables of the design. Similarly, these wordlengths also control the hardware implementation cost of the design. For medical applications, the ability of an implementation to achieve the desired level of accuracy is of paramount importance. It is, therefore, necessary to understand the tradeoff between accuracy and hardware implementation cost for this design and to identify wordlength configurations that provide effective tradeoffs between these conflicting criteria. This multiobjective optimization will allow a designer to systematically maximize accuracy for a given hardware cost limitation (imposed by a target device, for example) or minimize hardware resources to meet the accuracy requirements of a medical application.

The following section provides a formal definition of this problem and the subsequent section describes a framework for multiobjective optimization of FPGA-based medical image registration.

3.1. Problem statement

Consider a system Q that is parameterized by N parameters n_i ($i = 1, 2, ..., N$), where each parameter can take a single value from a corresponding set of valid values (v_i). Let the design configuration space corresponding to this system be S, which is defined by a set consisting of all N-tuples generated by the Cartesian product of the sets v_i, $\forall i$:

$$S = v_1 \times v_2 \times v_3 \times \cdots \times v_N . \quad (3)$$

The size of this design configuration space is then equal to the cardinality of the set S or, in other words, the product of cardinalities of the sets v_i:

$$|S| = |v_1| \times |v_2| \times |v_3| \times \cdots \times |v_N| . \quad (4)$$

For most systems, not all configurations that belong to S may be valid or practical. We therefore define a subset $\Im$ ($\Im \subset S$), such that it contains all the feasible system configurations. Now consider m objective functions ($f_1, f_2, \ldots, f_m$) defined for system Q, such that each function associates a real value for every feasible configuration $c \in \Im$.

The problem of multiobjective optimization is then to find a set of solutions that simultaneously optimize the m objective functions according to an appropriate criterion. The most commonly adopted notion of optimality in multiobjective optimization is that of Pareto optimality. According to this notion, a solution c^* is *Pareto optimal* if there does not exist another solution $c \in \Im$ such that $f_i(c) \leq f_i(c^*)$, for all i, and $f_j(c) < f_j(c^*)$, for at least one j.

Given a multiobjective optimization problem and a heuristic technique for this problem that attempts to derive Pareto-optimal or near-Pareto-optimal solutions, we refer to solutions derived by the heuristic as "Pareto-optimized" solutions.

3.2. Multiobjective optimization framework

Figure 2 illustrates the framework that we have developed for multiobjective optimization of the aforementioned architecture. There are two basic components of this framework. The first component is the search algorithm that explores the design space and generates feasible candidate solutions; and the second component is the objective function evaluation module that evaluates candidate solutions. The solutions and associated objective values are fed back to the search algorithm, so that they can be used to refine the search. These two components are loosely coupled so that different search algorithms can be easily incorporated into the framework. Moreover, the objective function evaluation module is parallelized using a message passing interface (MPI) on a 32-processor cluster. With this parallel implementation, multiple solutions can be evaluated in parallel, thereby increasing search

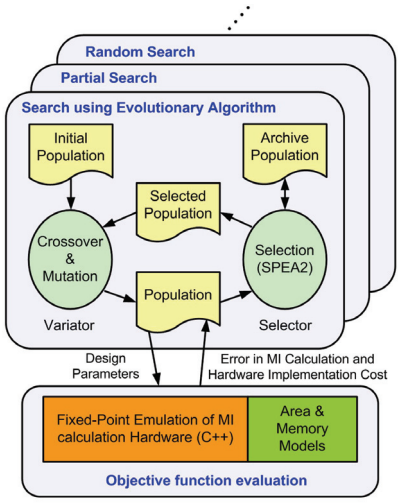


Figure 2: Framework for multiobjective optimization of FPGA-based image registration

performance. These components are described in detail in the following sections.

3.2.1. Design parameters. As described in the earlier section, the architecture performs MI calculation using a fixed-point datapath. As a result, the accuracy of MI calculation depends on the precision (wordlength) offered by this datapath. The design parameters in this datapath define the design space and are identified and listed along with the corresponding design module (see Figure 1) in Table 1.

A fixed-point representation consists of an integer part and a fractional part. The number of bits assigned to these two parts are called the integer wordlength (IWL) and fractional wordlength (FWL), respectively. The collective number of bits allocated to these parts control the range and precision of the fixed-point representation. For this architecture, the IWL required for each design parameter can be deduced from the range information specific to the image registration application. For example, in order to support translations in the range of [–64, 63] voxels, 7 bits of IWL (with 1 bit assigned as a sign bit) are required for the translation parameter. We used similar range information to choose the IWL for all the parameters, and these values are reported in Table 1. The precision required for each parameter, which is determined by its FWL, is not known a priori. We, therefore, determine this by performing multiobjective optimization using the FWL of each parameter as a design variable. In our experiments, we used the design range of [1, 32] bits for FWLs of all the parameters. The optimization framework can support different wordlength ranges for different parameters, which can be used to account for additional design constraints, such as, for example, certain kinds of constraints imposed by third-party intellectual property.

The entropy calculation module is implemented using a multiple LUT–based approach and also employs fixed-point arithmetic. However, this module has already been optimized for accuracy and hardware resources, as described in [20]. The optimization strategy employed in [20] uses an analytical approach that is specific to entropy calculation and is distinct from the strategy presented in this work. This module, therefore, does not participate in the multiobjective optimization framework of this paper, and we simply use the optimized configuration identified earlier. This further demonstrates the flexibility of our optimization framework to accommodate arbitrary designer- or externally-optimized modules.

3.2.2. Search algorithms. An exhaustive search that explores the entire design space is guaranteed to find all Pareto-optimal solutions. However, this search can lead to unreasonable execution time, especially when the objective function evaluation is computationally intensive. For example, with four design variables, each taking one of 32 possible values, the design space consists of 32^4 solutions. If the objective function evaluation takes 1 minute per trial (which is quite realistic for multiple MI calculation using large images), the exhaustive search will take 2 years. Consequently, we considered other search methods as described below.

The first method is *partial search*, which explores only a portion of the entire design space. For every design variable, the number of possible values it can take is reduced by half by choosing every alternate value. A complete search is then performed in this reduced search space. This method, although not exhaustive, can effectively sample the breadth of the design space. The second method is *random search*, which involves randomly generating a fixed number of feasible solutions. For both of these methods, Pareto-optimized solutions are identified from the set of solutions explored.

Table 1: Design variables for FPGA-based architecture. Integer wordlengths are determined based on application-specific range information, and fractional wordlengths are used as parameters in the multiobjective optimization framework

Architectural Module	Design Variable	Integer wordlength (IWL) (bits)	Fractional wordlength (FWL) range (bits)
Voxel coordinate transformation	Translation vector	7	[1,32]
	Rotation matrix	4	[1,32]
Partial volume interpolation	Floating image address	27	[1,32]
Mutual histogram accumulation	Mutual histogram bin	25	[1,32]

The third method is performing a search using evolutionary techniques. EAs have been shown to be effective in efficiently exploring large search spaces [11, 12]. In particular, we have employed SPEA2 [13], which is very effective in sampling from along an entire Pareto-optimal front and distributing the solutions generated relatively evenly over the optimal tradeoff surface. Moreover, SPEA2 incorporates a fine-grained fitness assignment strategy and an enhanced archive truncation method, which further assist in finding Pareto-optimal solutions. The flow of operations in this search algorithm is shown in Figure 2.

For the EA-based search algorithm, the representation of the system configuration is mapped on to a "chromosome" whose "genes" define the wordlength parameters of the system. Each gene, corresponding to the wordlength of a design variable *i*, is represented using an integer *allele* that can take values from the set v_i, described earlier. Thus, every gene is confined to wordlength values that are predefined and feasible for a given design variable. The genetic operators for crossover and mutation are also designed to adhere to this constraint and always produce values from set v_i, for a gene *i* within a chromosome. This representation scheme is both symmetric and repair free and, hence, is favored by the schema theory [21], and is computationally efficient, as described in [22].

3.2.3. Objective functions. We consider the hardware implementation cost and the error in MI calculation to be the conflicting objectives that must be minimized for our FPGA implementation problem. The FPGA implementation cost has two components: the first is the amount of logic resources (number of LUTs) required by the design, and the second is the internal memory consumed by the design. We treat these as independent objectives in order to explore the synergistic effects between these complementary resources. The logic resources required for a given feasible configuration are estimated by using the FPGA area models developed by Constantinides et al. [4, 23]. The memory requirement of a configuration is calculated by an analytical expression parameterized on the design wordlengths. This architecture-specific analytical expression is accurate and derived from the size of memory elements (e.g., FIFOs, memory for MH) instantiated by various design modules.

The error in MI calculation is computed by comparing the MI value reported by the limited-precision FPGA implementation against that calculated by a double-precision software implementation. We have developed a parameterized, bit-true emulation of the FPGA-based architecture that is capable of calculating the MI value corresponding to any feasible configuration for a given image transformation. This implementation was used to compute the MI calculation error. The MI calculation error was averaged for three distinct image pairs and for 50 randomly generated image transformations. The same set of image pairs and image transformations were used for evaluating all feasible configurations.

4. Results

We performed multiobjective optimization of the aforementioned architecture using the search algorithms outlined in the previous section. To account for the effects of random number generation, the EA-based search and random search were repeated five times each, and the average behavior from these repeated trials is reported. The number of solutions explored by each search algorithm in a single run is reported in Table 2. The execution time of each search algorithm was roughly proportional to the number of solutions explored, and the objective function evaluation for each solution took approximately 1 minute using a single computing node. As expected, the partial search algorithm explored the largest number of solutions. The parameters used for the EA-based search are listed in Table 3. The crossover and mutation operators were chosen to be one-point crossover and flip mutator, respectively. For a fair comparison, the number of solutions explored by the random search algorithm was set to be equal to that explored by the EA-based algorithm.

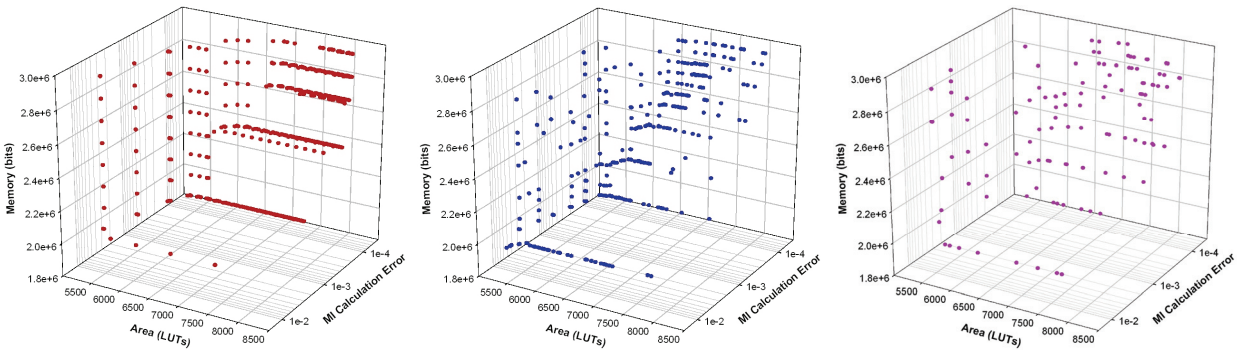
The solution sets obtained by each search method were then further reduced to corresponding nondominated solution sets using the concept of Pareto optimality. As described earlier, the objectives considered for this evaluation were the MI calculation error and the memory and area requirements of the solutions. Figure 3 shows the Pareto-optimized solution set obtained for each search method. Qualitatively, the Pareto front identified by the EA-based search is denser and widely distributed and demonstrates better diversity than other search methods.

Table 2: Number of solutions explored by search methods

Search Method	Number of solutions explored
Partial search	65,536
Random search	6,000
EA-based search	6,000

Table 3: Parameters used for EA-based search

Parameter	Value
Population size	200
Number of generations	30
Crossover probability	1.0
Mutation probability	0.06



(a) Partial search (b) EA-based search (c) Random search

Figure 3: Pareto-optimized solutions identified by various search methods

Figure 4 compares the Pareto fronts obtained by partial search and EA-based search by overlaying them and illustrates that the EA-based search can identify better Pareto-optimized solutions, which indicates the superior quality of solutions obtained by this search method. Moreover, it must be noted that the execution time required for the EA-based search was less than 10% of that for the partial search.

Quantitative comparison of the Pareto-optimized solution sets is essential in order to compare more precisely the effectiveness of various search methods. As with most real-world complex problems, the Pareto-optimal solution set is unknown for this application. We, therefore, employ the following two metrics to perform quantitative comparison. We use the ratio of non-dominated individuals (RNI) to judge the quality of the solution set, and the diversity of the solution set is measured using the cover rate. These performance measures are similar to those reported in [24] and are described below.

The RNI is a metric that measures how close a solution set is to the Pareto-optimal solution set. Consider two solution sets (*P1* and *P2*) that each contain only non-dominated solutions. Let the union of these two sets be P_U. Furthermore, let P_{ND} be a set of all non-dominated solutions in P_U ($P_{ND} \subset P_U$). The RNI for the solution set P_i is then calculated as:

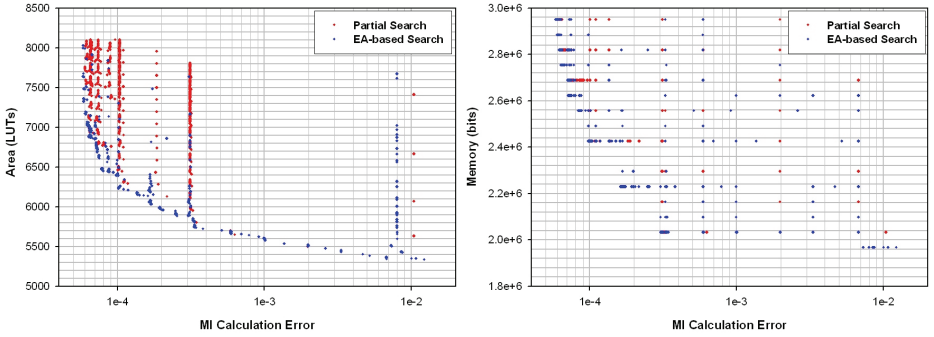
$$RNI_i = \frac{|P_i \cap P_{ND}|}{|P_{ND}|}, \tag{5}$$

where $|\cdot|$ is the cardinality of a set. The closer this ratio is to 100%, the more superior the solution set is and the closer it is to the Pareto-optimal front. We computed this metric for all the search algorithms previously described, and the results are presented in Figure 5. Our EA-based search offers better RNI and, hence, superior quality solutions to those achieved with either the partial or random search.

The cover rate estimates the spread and distribution (or diversity) of a solution set in the objective space. Consider that the region between the minimum and maximum of an objective function is divided into an arbitrary number of partitions. The cover rate is then calculated as the ratio of the number of partitions that are covered (that is, there exists at least one solution with an objective value that falls within a given partition) by a solution set to the total number of partitions. The cover rate (C_k) of a solution set, for an objective function (f_k) can then be calculated as:

$$C_k = \frac{N_k}{N}, \tag{6}$$

where N_k is the number of covered partitions and N is the total number of partitions. If there are multiple objective functions (m, for example), then the net cover rate can be obtained by averaging the cover rates for each objective



(a) Area vs. MI calculation error (b) Memory vs. MI calculation error

Figure 4: Qualitative comparison of solutions found by partial search and EA-based search

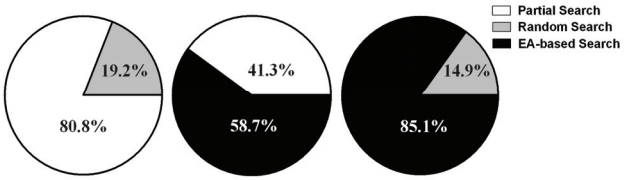


Figure 5: Comparison of search methods using the ratio of non-dominated individuals (RNI)

function as:

$$C = \frac{1}{m}\sum_{k=1}^{m} C_k \,. \qquad (7)$$

The maximum cover rate is 1, and the minimum value is 0. The closer the cover rate of a solution set is to 1, the better coverage and more even (more diverse) distribution it has. Because the Pareto-optimal front was unknown for this application, the minimum and maximum values for each objective function were selected from the solutions identified by all the search methods. We used 20 partitions/decade for MI calculation error (represented using a logarithmic scale), 1 partition for every 50 LUTs for the area requirement, and 1 partition for every 50 Kbits of memory requirement. The cover rate for all the search algorithms described earlier was calculated using the method outlined above, and the results are illustrated in Figure 6. The EA-based search offers a better cover rate, which translates to better range and diversity of solutions when compared with either partial or random searches. In summary, our EA-based search outperforms the random search and is capable of offering more diverse and superior quality solutions when compared with the partial search, using only 10% of the execution time.

An important performance measure for any image registration algorithm, especially in the context of medical imaging, is its accuracy. We did not choose registration accuracy as an objective function because of its dependence on data (image pairs), the degree of misalignment between images, and the behavior of the optimization algorithm that is used for image registration. These factors, along with its execution time, in our experience, may render registration accuracy as an unsuitable objective function, especially if there is non-monotonic behavior with respect to the wordlength of design variables.

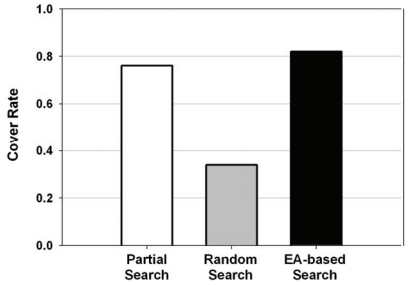


Figure 6: Comparison of search methods using cover rate

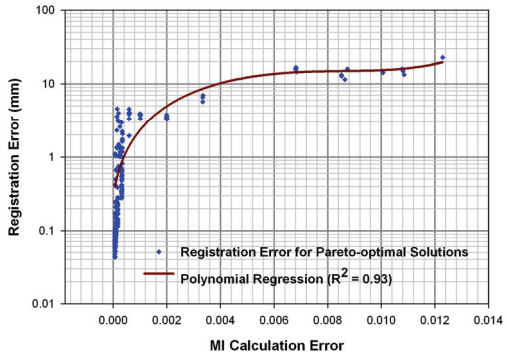


Figure 7: Relationship between MI calculation error and resulting image registration error

Instead, we evaluated the affect of error in MI calculation on the image registration accuracy for a set of image pairs. This analysis was performed using three computed tomography image pairs for the Pareto-optimized solutions identified by all of the search algorithms that we experimented with. Image registration was performed using limited-precision configurations corresponding to each solution, and the result was compared with that obtained using double-precision software implementation. Registration accuracy was calculated by comparing deformations at the vertices of a cuboid (with size equal to half the image dimensions) located at the center of the image. The results of this analysis are illustrated in Figure 7. As expected, there is a good correlation between the MI calculation error and the accuracy of image registration. This demonstrates that optimized tradeoff curves between MI calculation error and hardware cost, as identified by our reported analysis, can be used to represent the relationships between registration accuracy and hardware cost with high fidelity. This analysis also provides better insight about the sensitivity of image registration accuracy to various design parameters. Moreover, this will enable a designer to systematically choose an efficient system configuration to meet the registration accuracy requirements of specific clinical applications.

5. Related work

With the need for real-time performance in image processing applications, and in many other types of signal processing, an increasing trend is to accelerate computationally intensive algorithms using custom hardware implementation. A critical step in this process is to convert floating-point implementations to fixed-point versions for performance reasons. This conversion process is an inherently multidimensional problem, as several conflicting objectives, such as area and error, have to be simultaneously minimized. By systematically deriving

efficient tradeoff configurations, one can not only reduce the design time [3] but can also enable automated design synthesis [25, 26]. Our work presented in this paper has developed a framework for optimizing tradeoff relations between hardware cost and implementation error in the context of FPGA-based medical image registration.

Earlier approaches to optimizing wordlengths used analytical approaches for range and error estimation [4-8]. Some of these have used the error propagation method (e.g., see [7]), whereas others have employed models of worst-case error [5, 8]. Although, these approaches are faster and do not require simulation, formulating analytical models for complex objective functions, such as MI, is difficult. Statistical approaches have also been employed for optimizing wordlengths [27, 28]. These methods employ range and error monitoring for identifying appropriate wordlengths. These techniques do not require range or error models. However, they often need long execution times and are less accurate in determining effective wordlengths.

Some published methods search for optimum wordlengths using error or cost sensitivity information. These approaches are based on search algorithms such as "Local," "Preplanned," and "Max-1" search [9, 29]. However, for a given design scenario, these methods are limited to finding a single feasible solution, as opposed to a multiobjective tradeoff curve. In contrast, the techniques we presented in this paper are capable of deriving efficient tradeoff curves across multiple objective functions.

Other heuristic techniques that take into account tradeoffs between hardware cost and implementation error and enable automatic conversion from floating-point to fixed-point representations are limited to software implementations only [26]. Also, some of the methods based on heuristics do not support different degrees of fractional precision for different internal variables [5]. In contrast, our framework allows multiple fractional precisions, supports a variety of search methods, and thereby captures more comprehensively the complexity of the underlying multiobjective optimization problem.

Other approaches to solve this multiobjective problem have employed weighted combinations of multiple objectives and have reduced the problem to mono-objective optimization [30]. This approach, however, is prone to finding suboptimal solutions when the search space is nonconvex [10]. Some methods have also attempted to model this problem as a sequence of multiple mono-objective optimizations [31]. The underlying assumption in this approximation, however, is that the design parameters are completely independent, which is rarely the case in complex systems. Modeling this problem as an integer linear programming formulation has also been shown to be effective [4]. But this approach is limited to cases in which the objective functions can be represented or approximated as linear functions of design variables.

EAs have been shown to be effective in solving various kinds of multiobjective optimization problems [11, 12] but have not been extensively applied to finding optimal wordlength configurations. An exception is the work of [32], which employs mono-objective EAs. In contrast, our work demonstrates the applicability of EA-based search for finding superior Pareto-optimized solutions in an efficient manner, even in the presence of a non-linear objective function. Moreover, our optimization framework supports multiple search algorithms and objective function models; and can easily be extended to a wide range other signal processing applications.

6. Conclusion

This paper has presented a framework for multiobjective optimization of finite precision FPGA implementations. This framework considers multiple conflicting objectives such as hardware resource consumption and implementation accuracy, and systematically explores tradeoff relationships among the targeted objectives. Our work has also further demonstrated the applicability of EA-based techniques for efficiently identifying Pareto-optimized tradeoff relations in the presence of complex and non-linear objective functions. The evaluation performed in the context of FPGA-based medical image registration demonstrates that such an analysis can be used to enhance automated hardware design processes, and efficiently identify a system configuration that meets given design constraints. Furthermore, the multiobjective optimization approach that we present is quite general, and can be extended to a multitude of other signal processing applications.

7. Acknowledgments

This work was supported by the U.S. Department of Defense (TATRC) under grant DAMD17-03-2-0001.

8. References

[1] G. A. Constantinides, P. Y. K. Cheung, and W. Luk, "The Multiple Wordlength Paradigm," in *The 9th Annual IEEE Symposium on Field-Programmable Custom Computing Machines, FCCM '01.* , 2001, pp. 51-60.

[2] G. A. Constantinides and G. J. Woeginger, "The complexity of multiple wordlength assignment," *Applied Mathematics Letters,* vol. 15 (2), pp. 137-140, 2002.

[3] H. Keding, M. Willems, M. Coors, and H. A. M. H. Meyr, "FRIDGE: a fixed-point design and simulation environment," in *Proceedings of Design, Automation and Test in Europe*, 1998, pp. 429-435.
[4] G. A. Constantinides, P. Y. K. Cheung, and W. Luk, "Wordlength optimization for linear digital signal processing," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems,* vol. 22 (10), pp. 1432-1442, 2003.
[5] A. Nayak, M. Haldar, A. Choudhary, and P. A. B. P. Banerjee, "Precision and error analysis of MATLAB applications during automated hardware synthesis for FPGAs," in *Proceedings of Design, Automation and Test in Europe* 2001, pp. 722-728.
[6] A. V. Oppenheim, R. W. Schafer, and J. R. Buck, *Discrete-time signal processing*, 2nd ed. Upper Saddle River, N.J.: Prentice Hall, 1999.
[7] M. Stephenson, J. Babb, and S. Amarasinghe, "Bidwidth analysis with application to silicon compilation," *ACM SIGPLAN Conference on Programming Language Design and Implementation,* vol. 35 (5), pp. 108-120, 2000.
[8] S. A. Wadekar and A. C. Parker, "Accuracy sensitive word-length selection for algorithm optimization," in *Proceedings of International Conference on Computer Design: VLSI in Computers and Processors, ICCD '98.* , 1998, pp. 54-61.
[9] H. Choi and W. P. Burleson, "Search-based wordlength optimization for VLSI/DSP synthesis," in *Proceedings of IEEE Workshop on VLSI Signal Processing, VII*, 1994, pp. 198-207.
[10] I. Das and J. E. Dennis, "A closer look at drawbacks of minimizing weighted sums of objectives for Pareto set generation in multicriteria optimization problems," *Structural and Multidisciplinary Optimization,* vol. 14 (1), pp. 63-69, 1997.
[11] M. S. Bright and T. Arslan, "Synthesis of low-power DSP systems using a genetic algorithm," *IEEE Transactions on Evolutionary Computation,* vol. 5 (1), pp. 27-40, 2001.
[12] C. L. Valenzuela and P. Y. Wang, "VLSI placement and area optimization using a genetic algorithm to breed normalized postfix expressions," *IEEE Transactions on Evolutionary Computation,* vol. 6 (4), pp. 390-401, 2002.
[13] E. Zitzler, M. Laumanns, and L. Thiele, "SPEA2: Improving the Strength Pareto Evolutionary Algorithm for Multiobjective Optimization," in *EUROGEN 2001 - Evolutionary Methods for Design, Optimisation and Control with Applications to Industrial Problems*, 2001, pp. 95-100.
[14] K. Deb, A. Pratap, S. Agarwal, and T. A. M. T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation,* vol. 6 (2), pp. 182-197, 2002.
[15] F. Maes, A. Collignon, D. Vandermeulen, G. Marchal, and P. Suetens, "Multimodality image registration by maximization of mutual information," *IEEE Transactions on Medical Imaging,,* vol. 16 (2), pp. 187-198, 1997.
[16] C. R. Castro-Pareja, J. M. Jagadeesh, and R. Shekhar, "FAIR: a hardware architecture for real-time 3-D image registration," *IEEE Transactions on Information Technology in Biomedicine,* vol. 7 (4), pp. 426-434, 2003.
[17] O. Dandekar and R. Shekhar, "FPGA-Accelerated Deformable Image Registration for Improved Target-Delineation During CT-Guided Interventions," *IEEE Transactions on Biomedical Circuits and Systems,* vol. 1 (2), pp. 116-127, 2007.
[18] M. Doggett and M. Meissner, "A memory addressing and access design for real time volume rendering," in *IEEE International Symposium on Circuits and Systems*, 1999, pp. 344-347.
[19] D. M. Mandelbaum and S. G. Mandelbaum, "A fast, efficient parallel-acting method of generating functions defined by power series, including logarithm, exponential, and sine, cosine," *IEEE Transactions on Parallel and Distributed Systems,* vol. 7 (1), p. 33, 1996.
[20] C. R. Castro-Pareja and R. Shekhar, "Hardware acceleration of mutual information-based 3D image registration," *Journal of Imaging Science and Technology,* vol. 49 (2), pp. 105-113, 2005.
[21] T. Back, U. Hammel, and H. P. Schwefel, "Evolutionary computation: comments on the history and current state," *IEEE Transactions on Evolutionary Computation,* vol. 1 (1), pp. 3-17, 1997.
[22] V. Kianzad and S. S. Bhattacharyya, "Efficient techniques for clustering and scheduling onto embedded multiprocessors," *Transactions on Parallel and Distributed Systems,* vol. 17 (7), pp. 667-680, 2006.
[23] G. A. Constantinides, P. Y. K. Cheung, and W. Luk, "Optimum wordlength allocation," in *Proceedings of 10th Annual IEEE Symposium on Field-Programmable Custom Computing Machines*, 2002, pp. 219-228.
[24] E. Zitzler and L. Thiele, "Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach," *IEEE Transactions on Evolutionary Computation,* vol. 3 (4), pp. 257-271, 1999.
[25] G. A. Constantinides, P. Y. K. Cheung, and W. Luk, "Optimum and heuristic synthesis of multiple word-length architectures," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems,* vol. 13 (1), pp. 39-57, 2005.
[26] K. Kum, J. Kang, and W. Sung, "AUTOSCALER for C: an optimizing floating-point to integer C program converter for fixed-point digital signal processors," *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing,* vol. 47 (9), pp. 840-848, 2000.
[27] S. Kim, K.-I. Kum, and W. Sung, "Fixed-point optimization utility for C and C++ based digital signal processing programs," *IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing,* vol. 45 (11), pp. 1455-1464, 1998.
[28] K.-I. Kum and W. Sung, "Combined word-length optimization and high-level synthesis of digital signal processing systems," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems,* vol. 20 (8), pp. 921-930, 2001.
[29] M. A. Cantin, Y. Savaria, and P. Lavoie, "A comparison of automatic word length optimization procedures," in *IEEE International Symposium on Circuits and Systems* 2002, pp. 612-615.
[30] K. Han and B. Evans, "Optimum wordlength search using sensitivity information," *EURASIP Journal on Applied Signal Processing vol. 2006, Article ID 92849,* pp. 1-14, 2006.
[31] T. Givargis, F. Vahid, and J. Henkel, "System-level exploration for Pareto-optimal configurations in parameterized system-on-a-chip," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems,* vol. 10 (4), pp. 416-422, 2002.
[32] M. Leban and J. F. Tasic, "Word-length optimization of LMS adaptive FIR filters," in *10th Mediterranean Electrotechnical Conference* 2000, pp. 774-777.