Hindawi Publishing Corporation
International Journal of Reconfigurable Computing
Volume 2008, Article ID 738174, 17 pages
doi:10.1155/2008/738174

Research Article

Multiobjective Optimization for Reconfigurable Implementation of Medical Image Registration

Omkar Dandekar,[1,2] William Plishker,[1,2] Shuvra S. Bhattacharyya,[1] and Raj Shekhar[1,2]

[1] *Department of Electrical and Computer Engineering, University of Maryland, College Park, MD 20742, USA*
[2] *Department of Diagnostic Radiology and Nuclear Medicine, University of Maryland School of Medicine, Baltimore, MD 21201, USA*

Correspondence should be addressed to Raj Shekhar, rshekhar@umm.edu

Received 6 March 2008; Revised 11 September 2008; Accepted 27 November 2008

Recommended by Juergen Becker

In real-time signal processing, a single application often has multiple computationally intensive kernels that can benefit from acceleration using custom or reconfigurable hardware platforms, such as field-programmable gate arrays (FPGAs). For adaptive utilization of resources at run time, FPGAs with capabilities for dynamic reconfiguration are emerging. In this context, it is useful for designers to derive sets of efficient configurations that trade off application performance with fabric resources. Such sets can be maintained at run time so that the best available design tradeoff is used. Finding a single, optimized configuration is difficult, and generating a family of optimized configurations suitable for different run-time scenarios is even more challenging. We present a novel multiobjective wordlength optimization strategy developed through FPGA-based implementation of a representative computationally intensive image processing application: medical image registration. Tradeoffs between FPGA resources and implementation accuracy are explored, and Pareto-optimized wordlength configurations are systematically identified. We also compare search methods for finding Pareto-optimized design configurations and demonstrate the applicability of search based on evolutionary techniques for identifying superior multiobjective tradeoff curves. We demonstrate feasibility of this approach in the context of FPGA-based medical image registration; however, it may be adapted to a wide range of signal processing applications.

1. Introduction

In the field of real-time signal processing systems, acceleration of computationally intensive algorithmic components is often achieved by mapping them to custom or reconfigurable hardware platforms, such as field-programmable gate arrays (FPGAs). Often, multiple kernels in a single application can benefit from this approach to acceleration, requiring them to share a single fabric. This is particularly necessary in applications where multiple kernels share data and feed results to each other. For example, in medical imaging it has been shown that both image preprocessing [1–3] and image registration [4–6] can achieve high levels of speedup through hardware acceleration. To maximize the performance of an application and to optimize the fabric resource utilization, the kernels must be designed to meet their application requirements while balancing their resource consumption on the fabric. Application requirements often change at run time and strategies based on static design must try to identify a reasonable "average case" design configuration that accommodates all possible scenarios. Because this approach can be highly suboptimal and can result in significant under- or overutilization of the fabric in many scenarios, modern FPGAs are emerging with run-time reconfiguration capabilities. Self-monitoring FPGA implementations are able to adapt to variable application requirements and reconfigure their processing structures to better-suited design configurations [7]. This not only improves application performance but also results in more effective utilization of fabric resources. To exploit this technology, it is highly desirable that the designers provide quality design configurations that trade off application performance with fabric resources. Consequently, the primary focus of this work is to develop a framework that enables the designers to identify such optimized design configurations.

A common system parameter for trading off resource and performance is datapath wordlength. Typically, algorithms are first developed in software using floating-point representation and later migrated to hardware using finite precision (e.g., fixed-point representation) for achieving improved computational speed and reduced hardware cost. These implementations are often parameterized, so that a wide range of finite precision representations can be supported [8] 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 that minimizes the hardware implementation cost while satisfying a design criterion such as maximum output error has been shown to be nondeterministic polynomial-time (NP)-hard [9] and can take up to 50% of the design time for complex systems [10]. In addition, a single optimal solution may not exist, especially in the presence of multiple conflicting objectives. Moreover, a new configuration generally must be derived when the design constraints are altered.

An optimum wordlength configuration can be identified by analytically solving the quantization error equation as described previously by several authors [11–15]. This analytical representation, however, can be difficult to obtain for complex systems. Techniques based on local search or gradient-based search [16] 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 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 [11], 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 [17]. Techniques based on evolutionary methods have been shown to be effective in searching large search spaces in an efficient manner [18, 19]. Furthermore, these techniques are inherently capable of performing multipoint searches. As a result, techniques based on evolutionary algorithms (EAs) have been employed in the context of multiobjective optimization (SPEA2 [20], NSGA-II [21]). However, their application to solving wordlength optimization problems has been limited.

We formulate this problem of finding optimal wordlength configurations 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. In this work, we present this novel multiobjective optimization strategy and demonstrate its feasibility in the context of FPGA-based implementation of medical image registration. The tradeoff between FPGA resources (area and memory) and implementation accuracy is systematically 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, in the context of the chosen problem, the applicability of search based on evolutionary techniques for efficiently identifying superior multiobjective tradeoff curves. In comparison with the earlier reported techniques, our work captures more comprehensively the complexity of the underlying multiobjective optimization problem and demonstrates the applicability of our framework in finding superior Pareto-optimized solutions in an efficient manner, even in the presence of a nonlinear objective function.

This paper is organized as follows. Section 2 provides background on image registration and outlines an architecture for its FPGA-based implementation. We also highlight some strategies for parameterized design and synthesis of this architecture. Formulations for multiobjective optimization and various search methods to find Pareto-optimized solutions are described in Section 3. Section 4 describes experimental results, compares various search methods, and presents postsynthesis validation of the presented strategy. In Section 5, discussion on 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. Image registration is an active area of research, and over the last several decades numerous publications have outlined various methodologies to perform image registration and its applications. Maintz and Viergever [22] and Hill et al. [23] have presented a comprehensive summary of the range of the image registration domain. Several types of image registration are in routine use (see [22–25]); however, registration based on voxel intensities remains the most versatile, powerful, and inherently automatic way of achieving the alignment between two images. This approach, in general, 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 ($\mathbb{F}$):

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

Many image similarity measures, such as the sum of squared differences and cross-correlation, have been used, but over the last decade mutual information (MI) has emerged as the preferred similarity measure. MI is an information theoretic measure and is calculated as:

$$\mathrm{MI}(\mathrm{RI}, \mathrm{FI}) = h(\mathrm{RI}) + h(\mathrm{FI}) - h(\mathrm{RI}, \mathrm{FI}). \tag{2}$$

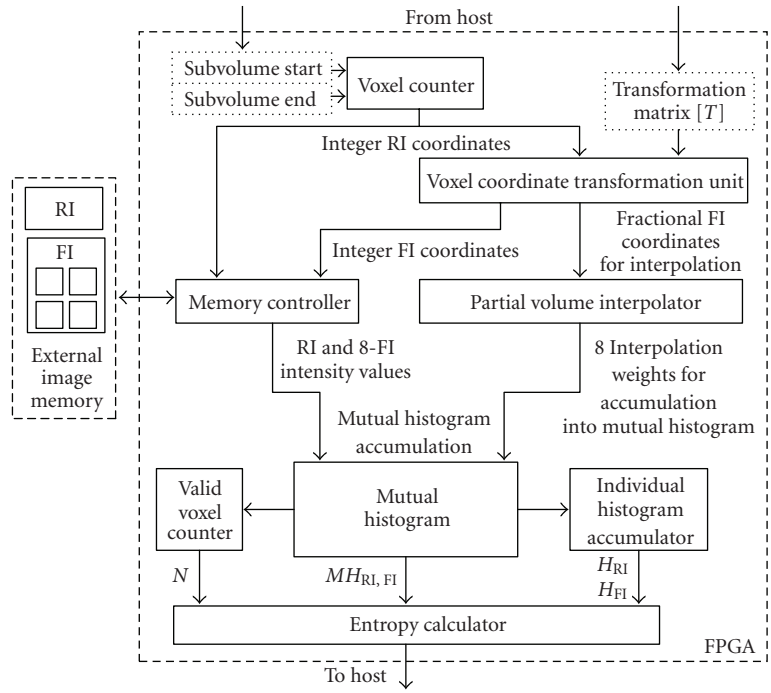


Figure 1: Top-level block diagram of FPGA-based architecture for MI calculation.

In this equation, $h(\mathrm{RI})$ and $h(\mathrm{FI})$ are the individual entropies and $h(\mathrm{RI}, \mathrm{FI})$ is the mutual entropy of the images to be registered. These entropies are further calculated as:

$$\begin{aligned} h(\mathrm{RI}) &= -\sum p_{\mathrm{RI}}(x) \cdot \ln\left(p_{\mathrm{RI}}(x)\right), \\ h(\mathrm{FI}) &= -\sum p_{\mathrm{FI}}(x) \cdot \ln\left(p_{\mathrm{FI}}(x)\right), \\ h(\mathrm{RI}, \mathrm{FI}) &= -\sum \sum p_{\mathrm{RI,FI}}(x) \cdot \ln\left(p_{\mathrm{RI,FI}}(x)\right). \end{aligned} \tag{3}$$

Here, the notations p_{RI}, p_{FI}, and $p_{\mathrm{RI,FI}}$ represent the individual probability distribution function (PDF) of RI, individual PDF of FI, and the mutual PDF of RI and FI, respectively. These distributions are estimated from the individual and mutual histograms of the images to be registered. Additional details about computation of MI, its properties, and its application to image registration can be found in an article by Pluim et al. [25].

MI-based image registration has been shown to be robust and effective in multimodality image registration [24]. 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. [4] 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 [6] that is capable of computing MI 40 times faster than software implementation. The transformation model (T in (1)) employed by this architecture is a locally rigid-body model consisting of three-dimensional (3D) translations and rotations. Consequently, the analysis presented in this article pertains to locally rigid transformations. However, it must be noted that hierarchical rigid-body transformations can be used to represent deformable (nonrigid) transformation models as demonstrated in the volume subdivision-based approach reported by Walimbe and Shekhar [26].

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 sections.

2.1.1. Voxel Counter

Calculation of MI requires processing (fetching the voxel from the image memory, performing coordinate transformation, and updating the mutual histogram (MH)) 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:

$$\vec{v}_f = T \cdot \vec{v}_r. \tag{4}$$

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 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. [24], 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

Images from different modalities (computed tomography (CT), magnetic resonance imaging (MRI), positron emission tomography (PET), etc.) have different native resolution, typical image dimensions, and dynamic range. Despite these variations, dimensions of most medical images are smaller than $512 \times 512 \times 512$ and are supported by our architecture. The dynamic range of these images is indicated by the number of bits used to represent the intensity (gray value) at every voxel. For MI-based registration, however, these images are typically converted to 7- or 8-bit representation as a part of image preprocessing. This is done to prevent dispersion of the MH and leads to improved quality of image registration. After this preprocessing step, all the gray values in the images are used for image registration.

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 [27] and image registration [4]. 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) and associated 3D neighborhood in the FI ($FV_0 : FV_7$), 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 preaccumulate buffers, which aggregate the weights from all conflicting transactions. Thus, all the transactions leading to an 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 reciprocal value of the number of valid voxels in the histogram is calculated by using successive subtraction operations. This operation takes N clock cycles (where N is the fractional wordlength of the reciprocal value) and must be performed only once per every MI calculation. The resulting value is then used by the entropy calculation unit for calculating the individual and joint probabilities and subsequently entropies as described in (3).

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 on values within $[0, 1]$, 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 [28], but of particular interest is the multiple lookup table (LUT)-based approach introduced by Castro-Pareja and Shekhar [5]. 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.

2.2. Parameterized Architectural Design

Implementations of signal processing algorithms using microprocessor- or DSP-based approaches are characterized by a fixed datapath width. This width is determined by the hardwired datapath of the underlying processor architecture. Reconfigurable implementation based on FPGAs, in contrast, allows the size of datapath to be customized to achieve better tradeoffs among accuracy, area, and power. Moreover, this customization can also change at run time to accommodate varying design requirements. The use of such custom data representation for optimizing designs is one of the main strengths of reconfigurable computing [29]. It has been contended that the most efficient hardware implementation of an algorithm is the one that supports a variety of finite precision representations of different sizes for its internal variables [8]. In this spirit, many commercial and research efforts have employed parameterized design style for intellectual property (IP) cores [30–34]. This parameterization capability not only facilitates reuse of design cores but also allows them to be reconfigured to meet design requirements.

During the design of the aforementioned architecture, we adopted a similar design style that allows configuration of the wordlengths of the internal variables. Hardware design languages such as VHDL and Verilog natively support hierarchical parameterization of a design through use of *generics* and *parameters*, respectively. This design style takes advantage of these language features and is employed for the design of all the modules described earlier. We highlight the main features of this design style using illustrative examples. Consider a design module with two input variables that computes an output variable through arithmetic manipulation of the input variables. The wordlength of the input variables (denoted by IP1_WIDTH, IP2_WIDTH) and that of the output variable (denoted by OP_WIDTH) are the design parameters for this module. The module can then be parameterized for these design variables as illustrated in Figure 2(a).

In a pipelined implementation of an operation, a module may have multiple internal pipeline stages and corresponding intermediate variables. Wordlengths chosen for these intermediate variables can also impact the accuracy and hardware requirements of a design. In our implementation scheme, we do not employ any rounding or truncation for the intermediate variables, but deduce their wordlengths based on the wordlengths of the input operands and the arithmetic operation to be implemented. For example, multiplication of two 8-bit variables will, at the most, require a 16-bit-wide intermediate output variable. A parameterized implementation of this scenario is illustrated in Figure 2(c). Sometimes it is also necessary to instantiate a vendor-provided or a third-party IP core, such as a first in/first out (FIFO) module or an arithmetic unit, within a design module. In such cases, we simply pass the wordlength parameters down the design hierarchy to configure the IP core appropriately and thereby maintain the parameterized design style (see, e.g., Figure 2(b)).

When signals cross module boundaries, the output wordlength and format (position of the binary point) of the source module should match the input wordlength and format of the destination module. This is usually achieved through use of a rounding strategy and right- or left-shifting of the signals. Adopting "rounding toward the nearest" strategy to achieve wordlength matching is expected to introduce the smallest error but requires additional logic resources. In our design, we therefore implement truncation (or "rounding toward zero" strategy), while the signal shifting is achieved through zero padding. Both these operations are parameterized and take into account the wordlengths and the format at the module boundaries (see, e.g., Figure 2(c)). Thus, this parameterized design style enables the architecture to support multiple wordlength configurations for its internal variables.

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 [6]. The accuracy of MI calculation (and, by extension, that of image registration) offered by this implementation, however, is a function of the 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

```
--Declaration of a parameterized entity
entity Module1 is
 generic (
  IP1_WIDTH : INTEGER;
  IP2_WIDTH : INTEGER;
  OP_WIDTH : INTEGER);
 port (
  s1 : IN STD_LOGIC_VECTOR (IP1_WIDTH-1 DOWNTO 0);
  s2 : IN STD_LOGIC_VECTOR (IP2_WIDTH-1 DOWNTO 0);
  o1 : OUT STD_LOGIC_VECTOR (OP_WIDTH-1 DOWNTO 0));
end Module1;
```

(a)

```
--Instantiation of a vendor supplied
--IP-core, scfifo. The width of the
--FIFO is set to be equal to that of
--the signal to be buffered (o1).
fifo1 : scfifo
 generic map (
  LPM_NUMWORDS => (2**LOG2_DEPTH),
  LPM_WIDTH => OP_WIDTH,
  LPM_WIDTHU => LOG2_DEPTH)
 port map (
data => o1,...);
```

(b)

```
--Declaration of an intermediate variable with appropriate wordlength
signal i1 : STD_LOGIC_VECTOR (IP1_WIDTH+IP2_WIDTH-1 DOWNTO 0);
i1 <= s1 * s2; --Arithmetic operation (multiplication)

--Truncation of LSBs: Performed if (IP1_WIDTH+IP2_WIDTH) >= OP_WIDTH
o1 <= i1(IP1_WIDTH+IP2_WIDTH-1 DOWNTO IP1_WIDTH+IP2_WIDTH-OP_WIDTH);

--Signal-shifting: Performed if IP1_WIDTH+IP2_WIDTH) < OP_WIDTH
o1 <= i1 & CONV_STD_LOGIC_VECTOR(0,OP_WIDTH-(IP1_WIDTH+IP2_WIDTH));
```

(c)

Figure 2: Parameterized architectural design: (a) declaration of a parameterized entity; (b) an example instantiation of a vendor-supplied IP-core; (c) usage of parameterized internal variables and an example of truncation and signal shifting, performed at the module boundaries.

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 a design and to identify wordlength configurations that provide effective tradeoffs between these conflicting criteria. This multiobjective optimization allows a designer to systematically maximize accuracy for a given hardware cost limitation (e.g., imposed by a target device) or minimize hardware resources to meet the accuracy requirements of a medical application.

Section 3.1 provides a formal definition of this problem, and Section 3.2 describes a framework developed 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 $(i = 1, 2, \ldots, N)$, where each parameter can take on 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, $\forall i$:

$$S = v_1 \times v_2 \times v_3 \times \cdots \times v_N. \tag{5}$$

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

the cardinalities of the sets v_i:

$$|S| = |v_1| \times |v_2| \times |v_3| \times \cdots \times |v_N|. \quad (6)$$

For most systems, not all configurations that belong to S may be valid or practical. We, therefore, define a subset $\mathfrak{I}$ ($\mathfrak{I} \subseteq S$), 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 $c \in \mathfrak{I}$.

The problem of multiobjective optimization is then to find a set of solutions that simultaneously optimizes 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 $c \in \mathfrak{I}$ such that $f_i(c) \leq f_i(c^*)$, for all i, and $f_j(c) < f_j(c^*)$, for at least one j. The solution c^* is also called a *nondominated* solution because no other solution dominates (or is superior to) solution c^* as per the Pareto-optimality criteria. The set of Pareto-optimal solutions, therefore, includes all nondominated solutions.

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.

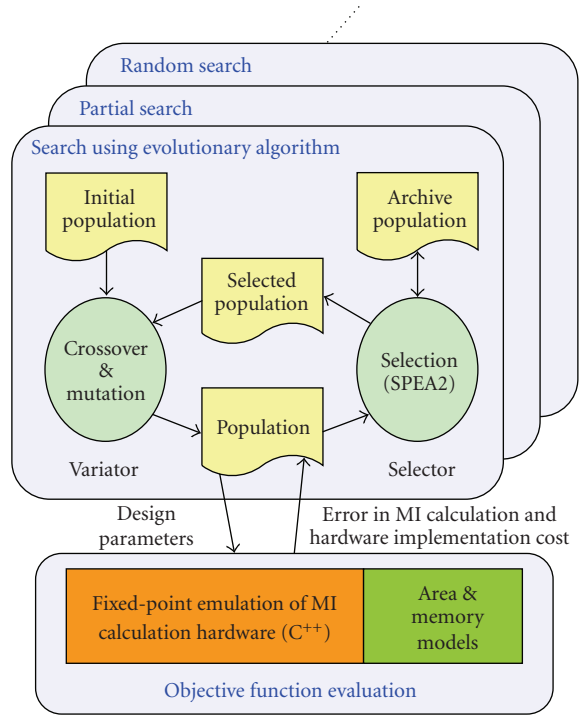


Figure 3: Framework for multiobjective optimization of FPGA-based image registration.

3.2. Multiobjective Optimization Framework

Figure 3 illustrates the framework that we have developed for multiobjective optimization of the aforementioned architecture. The framework has two basic components. The first is the search algorithm that explores the design space and generates feasible candidate solutions; the second 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 performance. These components are described in detail in the following sections.

3.2.1. Design Parameters

As described in Section 2.1, 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 numbers of bits assigned to these two parts are called the integer wordlength (IWL) and fractional wordlength (FWL), respectively. The individual numbers 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 IP.

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 previously [5]. The optimization strategy employed in this earlier work 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

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	9	[1, 32]
Mutual histogram accumulation	Mutual histogram bin	25	[1, 32]

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. Even with the 32-processor cluster that we employed and assuming linear speedup, exhaustive search for a four-variable system will require about 3.5 weeks. This highlights the infeasibility of exhaustive search even for a system with a relatively small number of design variables. Consequently, we have considered alternative 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.

The third method is performing a search using evolutionary techniques. EAs have been shown to be effective in efficiently exploring large search spaces [18, 19]. In particular, we have employed SPEA2 [20], which is quite 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 3.

For the EA-based search algorithm, the representation of the system configuration is mapped onto 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 cross-over 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 [35] and is computationally efficient, as described by Kianzad and Bhattacharyya [36].

3.2.3. Objective Function Models and their Fidelity

Search for Pareto-optimized configurations requires evaluating candidate solutions and determining Pareto-dominance relationships between them. This can be achieved by calculating objective functions for all the candidate solutions and by relative ordering of the solutions with respect to the values of their corresponding objective functions. We consider the error in MI calculation and the hardware implementation cost to be the conflicting objectives that must be minimized for our FPGA implementation problem. We model the FPGA implementation cost using 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. Because of the size of the design space and limitations resulting from execution time, it is not practical to synthesize and evaluate each solution. We, therefore, employ models for calculating objective functions to evaluate the solutions. The quality of the Pareto-optimized solutions will then depend on the fidelity of these objective function models.

The error in MI calculation can be computed by comparing the MI value reported by the limited-precision FPGA implementation against that calculated by a double-precision software implementation. For this purpose, we have utilized a bit-true emulator of the hardware. This emulator was developed in C++ and uses fixed-point arithmetic to accurately represent the behavior of the limited-precision hardware. It supports multiple wordlengths for internal variables and is capable of accurately calculating the MI value corresponding to any feasible configuration. We have verified its equivalence with the hardware implementation for a range of configurations and image transformations.

This emulator was used to compute the MI calculation error. The MI calculation error was averaged for three distinct image pairs (with different image modality combinations) and for 50 randomly generated image transformations. The same sets of image pairs and image transformations were used for evaluating all feasible configurations.

The memory required for a configuration is primarily needed for intermediate FIFOs, which are used to buffer internal variables, and the MH memory. For example, a 64-word-deep FIFO used to buffer a signal with a wordlength of b will require $64 \times b$ bits of memory. In our architecture, the depth of the FIFOs and the dimensions of the MH are constant, whereas their corresponding widths are determined by the wordlength of the design parameters. Using these insights, we have developed an architecture-specific analytical expression that accurately represents the cumulative amount of memory required for all internal FIFOs and MH. We used this expression to calculate the memory requirement of a configuration.

For estimating the area requirements of a configuration, we adopt the area models reported by Constantinides et al. [11, 37]. These are high-level models of common functional units such as adders, multipliers, and delays. These models are derived from the knowledge of the internal architecture of these components. Area cost for interconnects and routing is not taken into account in this analysis. These models have been verified for the Xilinx Virtex series of FPGAs and are equally applicable to alternative FPGA families and for application-specific integrated circuit (ASIC) implementations. These models have also been previously used in the context of wordlength optimization [11, 37, 38].

We further evaluated the fidelity [39] of these area models using a representative module, PV interpolator, from the aforementioned architecture. This module receives the fractional components of the FI address and computes corresponding interpolation weights. We varied the FWL of the FI address from 1 to 32 bits and synthesized the module using the Altera Stratix II and Xilinx Virtex 5 as target devices. For a meaningful comparison, the settings for the analysis, synthesis, and optimization algorithms (e.g., settings to favor area or speed) for the design tools (Altera Quartus II and Xilinx ISE) were chosen to be comparable. After complete synthesis, routing, and placement, we recorded the area (number of LUTs) consumed by the synthesized design. This process was automated by using the Tcl scripting feature provided by the design tools and through the parameterized design style described earlier. We then compared the consumed area against that predicted by the adopted area models for all FWL configurations. The results of this experiment are presented in Figure 4. These results indicate that the area estimates (number of LUTs) predicted by the model are comparable to those obtained through physical synthesis for both the target devices. For quantitative evaluation, the fidelity of the area models was calculated as follows:

$$\text{Fidelity} = \frac{2}{N(N-1)}\left(\sum_{i=1}^{N-1}\sum_{j=i+1}^{N} F_{ij}\right), \tag{7}$$

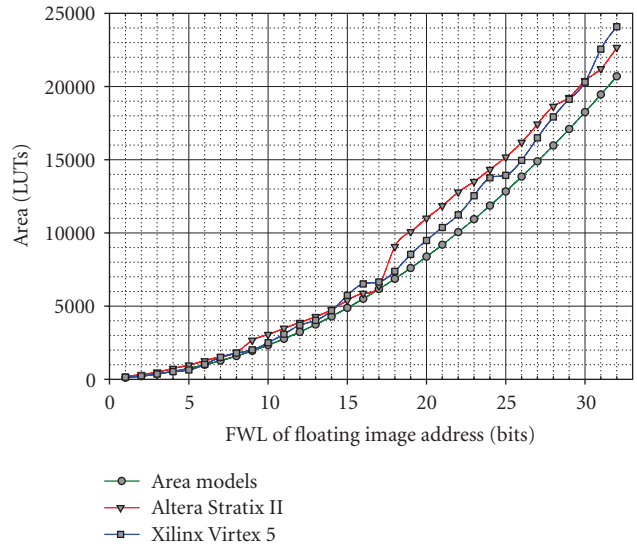


Figure 4: Comparison of the area values predicted by the adopted area models with those obtained after physical synthesis.

where

$$F_{ij} = \begin{cases} 1, & \text{if } \operatorname{sign}(S_i - S_j) = \operatorname{sign}(M_i - M_j), \\ 0, & \text{otherwise.} \end{cases} \tag{8}$$

In this equation, the M_is represent the values predicted by the area models; the S_is represent the values obtained after physical synthesis. The fidelity of the area models when evaluated with respect to the synthesis results obtained for both Altera and Xilinx devices was 1, which corresponds to maximum ("perfect") fidelity.

An interesting observation is that in some cases the high-level area models underestimate by as much as 25% the number of LUTs required. This can be explained by the fact that these models were calibrated using previous generation devices [11, 37]. It must be, however, noted that the Pareto-dominance relationship between the design configurations is maintained as long as the relative ordering (with respect to an objective function such as area) between two design configurations is preserved. Using more accurate area models will certainly improve the absolute prediction of area requirements corresponding to a given design configuration but, as such, will not affect the relative ordering of a set of design configurations. Designing accurate area models that take into account the latest devices, cross-vendor FPGA architectures, special-purpose computational units, and various synthesis optimizations is nevertheless important and will be a topic of a future investigation. The perfect fidelity we achieved for the current area models indicates that the relative ordering of FWL configurations with respect to their area requirements is consistent for the model and synthesized designs. These results further validate the applicability of using the aforementioned area models for multiobjective optimization.

Table 2: Number of solutions explored by search methods.

Search method	Number of solutions explored
Partial search	65 536
Random search	6000
EA-based search	6000

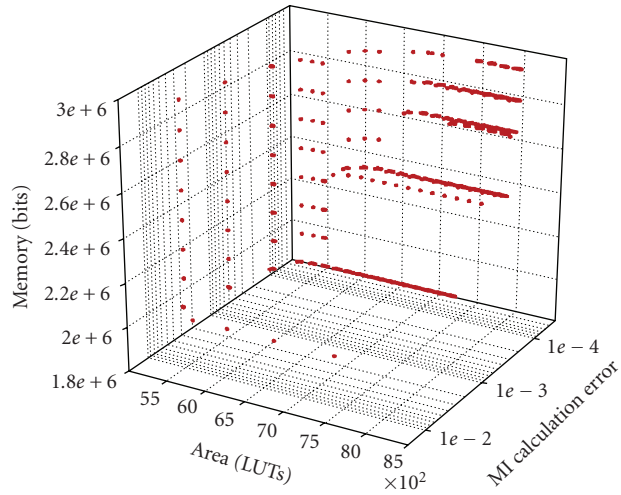
Table 3: Parameters used for EA-based search.

Parameter	Value
Population size	200
Number of generations	30
Cross-over probability	1.0
Mutation probability	0.06

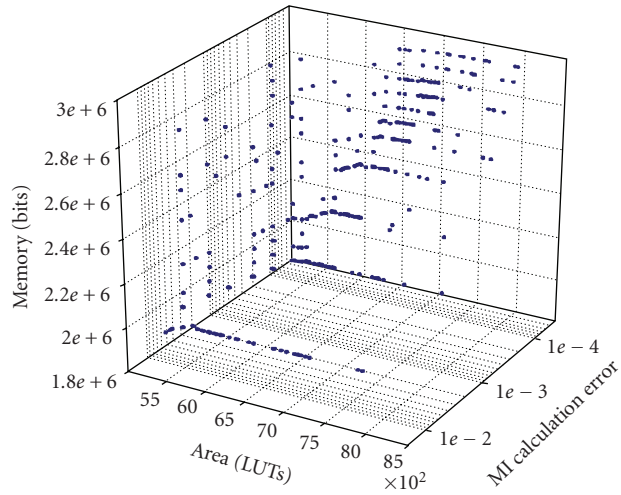
4. Results

We performed multiobjective optimization of the aforementioned architecture using the search algorithms outlined in Section 3. 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. These parameters were identified experimentally. For example, using a population size of 100 yielded similar search results; however, the diversity of the solutions found in the objective space was relatively poor. Similarly, increasing maximum number generations beyond 30 did not yield a significant improvement in the quality of the search solutions. The cross-over and mutation operators were chosen to be one-point cross-over 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.

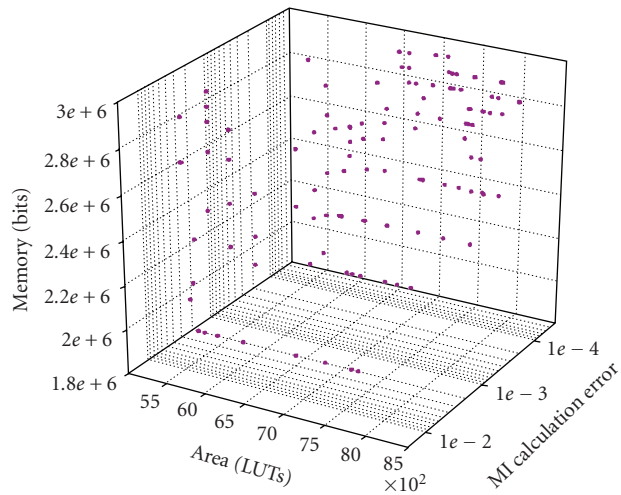
The solution sets obtained by each search method were then further reduced to corresponding nondominated solution sets using the concept of Pareto optimality. As described earlier, the objectives considered for this evaluation were the MI calculation error and the memory and area requirements of the solutions. Figure 5 shows the Pareto-optimized solution set obtained for each search method. Qualitatively, the Pareto front identified by the EA-based search is denser and more widely distributed and demonstrates better diversity than other search methods. Figure 6 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 more than 10 times faster than that required for the partial search.



(a) Partial search

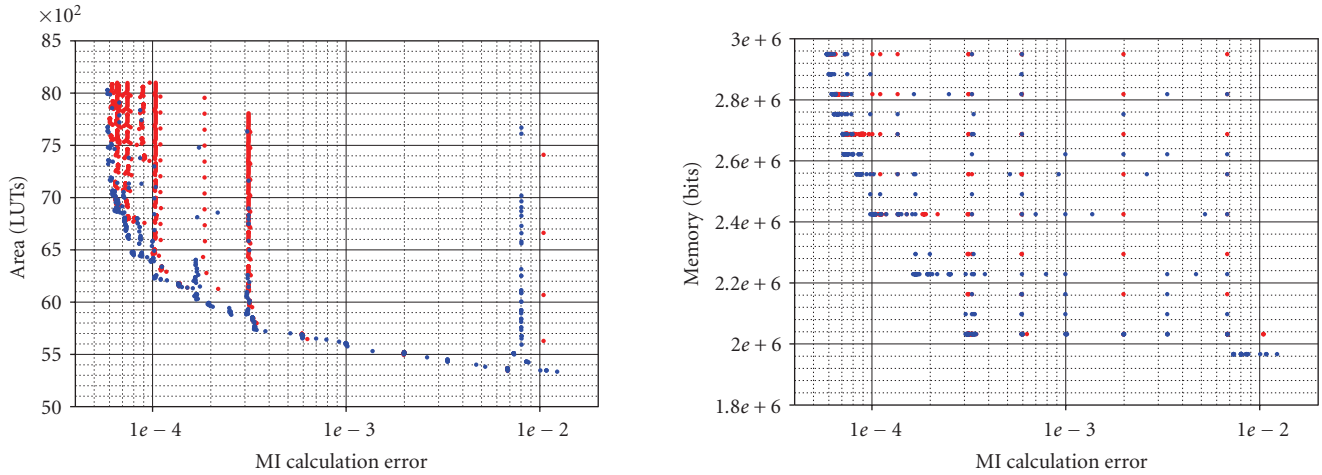


(b) EA-based search



(c) Random search

Figure 5: Pareto-optimized solutions identified by various search methods.



(a) Area versus MI calculation error

(b) Memory versus MI calculation error

Figure 6: Qualitative comparison of solutions found by partial search and EA-based search.

4.1. Metrics for Comparison of Pareto-Optimized Solution Sets

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 between different solution sets. We use the ratio of non-dominated individuals (RNIs) to judge the quality of a given solution set, and the diversity of a solution set is measured using the cover rate. These performance measures are similar to those reported by Zitzler and Thiele [40] 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 nondominated solutions. Let the union of these two sets be P_U. Furthermore, let P_{ND} be a set of all nondominated solutions in P_U ($P_{\text{ND}} \subseteq P_U$). The RNI for the solution set P_i is then calculated as:

$$\text{RNI}_i = \frac{|P_i \bigcap P_{\text{ND}}|}{|P_{\text{ND}}|}, \tag{9}$$

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 7. 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 the region between the minimum and maximum of an objective function as being divided into an arbitrary number of partitions. The cover rate is then calculated as the ratio of the number of partitions that is covered (i.e., 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:

$$C_k = \frac{N_k}{N}, \tag{10}$$

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

$$C = \frac{1}{m}\sum_{k=1}^{m} C_k. \tag{11}$$

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 is unknown for our targeted 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/decades 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 8. 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

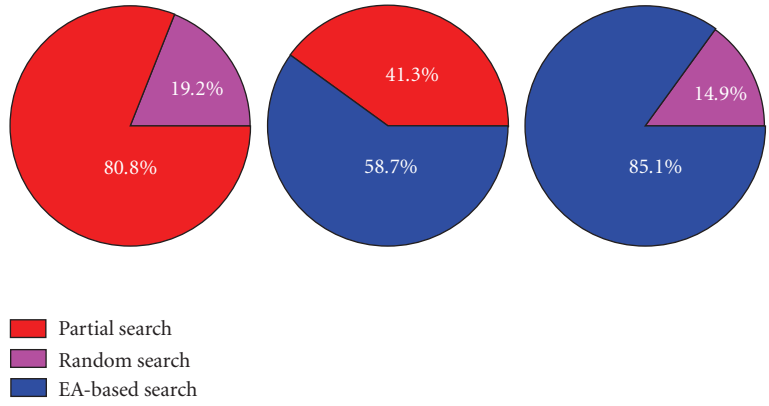


Figure 7: Comparison of search methods using the ratio of nondominated individuals (RNIs).

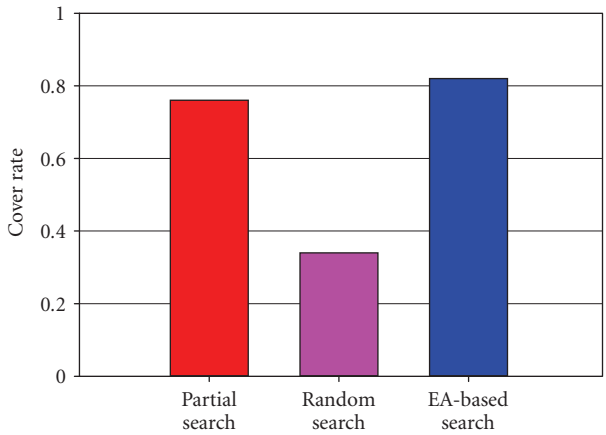


Figure 8: Comparison of search methods using cover rate.

compared with the partial search, using only 10% of the execution time.

4.2. Accuracy of Image Registration

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 nonmonotonic behavior with respect to the wordlength of design variables. Another important aspect is that the desired accuracy of registration depends on the application in which image registration is employed. For example, during an image-guided medical procedure high registration accuracy might be desired, whereas in a simple visualization task, slightly inaccurate image registration may be tolerated. Furthermore, in a multiresolution image registration approach slightly inaccurate (but, hardware resource-efficient) design configuration can be employed at the initial levels and a more accurate (but perhaps requiring more hardware resources) design configuration can be used at later levels. Thus, image registration accuracy is a constraint from an application perspective and, as such, is not used to guide the exploration of the design space. Instead, we used error in the MI calculation, which is relatively less application- and data-dependant, as an objective function.

Once the Pareto-optimized tradeoffs between MI calculation error and hardware resources are obtained through the presented approach, a system designer could evaluate the performance of these Pareto-optimized design configurations in the context of a specific target application. This can be done by using a set of sample image pairs acquired for that target application. To demonstrate the feasibility of this approach, we selected CT-CT registration as an example application. We randomly selected five clinical image pairs for this analysis and registered them using design configuration corresponding to each Pareto-optimized solution. These image pairs had the dimensions of $256 \times 256 \times 212$–$335$ voxels and the resolution of 1.4–$1.7\,\text{mm} \times 1.4$–$1.7\,\text{mm} \times 1.5\,\text{mm}$. This image registration was performed using the aforementioned bit-true simulator. The result of registration was then 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, which establish the relationship between MI calculation error and the registration error specific to this application of CT-CT registration, are reported in Figure 9. It must be noted that each point in this plot represents a valid design configuration. 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. These relationships can then be used to identify a design configuration in order to achieve desired registration accuracy for this example application of CT-CT registration. Similar relationships specific to a target application (e.g., PET-CT registration) can be generated using the aforementioned approach.

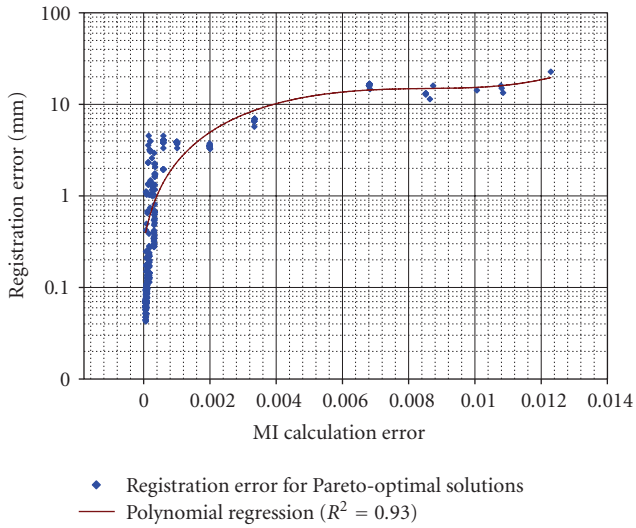


Figure 9: Relationship between MI calculation error and image registration error for an example application of CT-CT registration.

4.3. Postsynthesis Validation

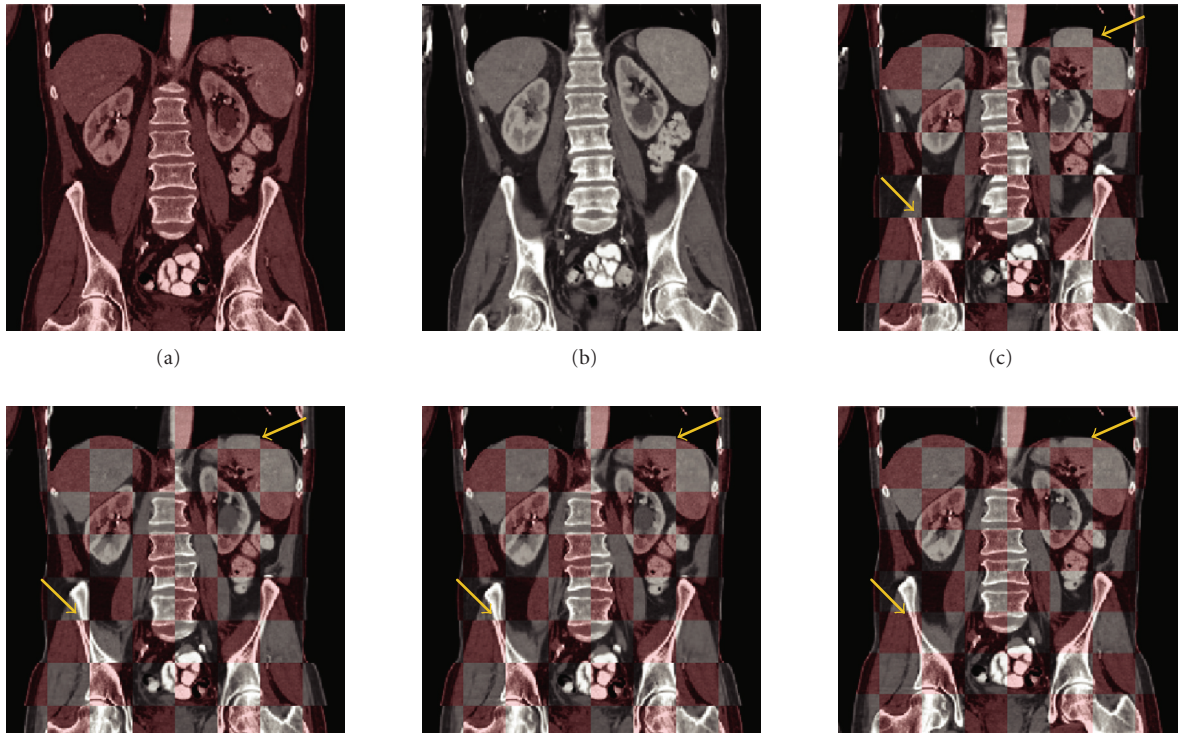
We performed further validation of the presented multiobjective optimization strategy through physical design synthesis. We identified three solutions from the Pareto-optimized set obtained using the EA-based search and synthesized the aforementioned architecture with configurations corresponding to these solutions. These solutions were identified with no specific clinical application in mind, but such that the tradeoff between various objective functions (MI calculation error, area, and memory) can be readily appreciated. Figure 9 reports the registration accuracy (calculated using the bit-true emulator that we developed) for all the Pareto-optimized design configurations. The system designer will have access to all the Pareto-optimized design configurations along with their expected MI-calculation error and hardware resource requirements, and, as such, can select a design configuration to meet the requirements of a given application.

These three configurations, which offer gradual tradeoff between hardware resource requirement and error in MI calculation, are listed in the first column of Table 4. The wordlengths associated with each configuration correspond to the FWLs of the design variables identified in Table 1. The design was synthesized for these configurations and the resulting realizations were implemented using an Altera Stratix II EP2S180F1508C4 FPGA (Altera Corporation, San Jose, Calif, USA) on a PCI prototyping board (DN7000K10PCI) manufactured by the Dini Group (La Jolla, Calif, USA). We then evaluated the performance of the synthesized designs and compared it with that predicted by the objective function models. The results of this analysis are summarized in Table 4 and are described below.

The error in MI calculation was computed by comparing the MI value reported by the limited-precision FPGA implementation against that calculated by a double-precision software implementation. The MI calculation error was averaged for three distinct image pairs and for 50 randomly generated image transformations for each pair. These image pairs and the associated transformations were identical to those employed in the objective function calculation. In this case, the average MI calculation error obtained by all the design configurations was identical to that predicted by the objective function model. This is expected because of the bit-true nature of the simulator used to predict the MI calculation error. We repeated this calculation with a different set of three image pairs and 50 randomly generated new transformations associated with each image pair. The MI calculation error corresponding to this setup is reported in the second column of Table 4. The small difference when compared with the error predicted by the models is explained by the different sets of images and transformations used. The area and memory requirements corresponding to each configuration after synthesis are reported in columns three and four of Table 4, respectively. For comparison, we have also included the values predicted by the corresponding objective function models in parenthesis. It must be noted that for all three configurations, the relative ordering based on Pareto-dominance relationships with respect to each objective function is identical for both postsynthesis and model-predicted values.

We also evaluated the accuracy of image registration performed using the implementation corresponding to each design configuration. For this analysis, we considered the same five CT image pairs described above. As reported earlier, these image pairs had dimensions of 256 × 256 × 212–335 voxels and resolution of 1.4–1.7 mm × 1.4–1.7 mm × 1.5 mm. The image registration results for one of those image pairs are illustrated in Figure 10. The result of registration between the remaining image pairs was also qualitatively similar. The registration error was calculated by comparing the obtained registration results with that obtained using double-precision software implementation. The mean and standard deviations of the registration error corresponding to each configuration are reported in Table 4. Good correlation is seen between the MI calculation error and the registration error, reinforcing the results presented in Section 4.2.

The performance of the resultant design configuration in terms of its raw clock rate is an important measure of the quality of a design. This clock rate directly affects the maximum voxel throughput that can be achieved by the design and, consequently, has an impact on the execution speed of image registration. The speed of a design configuration depends on, among other factors, the wordlengths of the design parameters. For example, performing arithmetic and memory operations using parameters with wider wordlengths may incur additional latency. As a result, design configurations employing design parameters with wider wordlengths may be slightly slower, although more accurate, than design configurations with shorter wordlengths. To provide some insights about this phenomenon, we recorded the maximum clock rate achieved by each of the design configurations we identified for synthesis. This represents the maximum postsynthesis frequency at which the design can operate and is reported in the last column of Table 4. These



(a) (b) (c)

(d) (e) (f)

Figure 10: Results of image registration performed using the high-speed, reconfigurable implementation: (a) and (b) two distinct poses; (c) fusion of (a) and (b) using a checkerboard pattern. The misalignment between images is evident at the edges of the squares within the checkerboard pattern; (d)–(f) fusion images after registration using the identified design configurations. These configurations offer progressively reduced image registration error (3.82 mm, 1.57 mm, and 0.45 mm, resp.) and result in correspondingly improved image alignment. The arrows indicate representative regions with misalignment that are better aligned after registration.

results indicate that the Pareto-optimized designs are not unreasonably slow and that their performance is comparable to that achieved (200 MHz) for a user-optimized design reported earlier [6].

This postsynthesis validation further demonstrates the efficacy of the presented optimization approach for reconfigurable implementation of image registration. It also further demonstrates how the approach enables a designer to systematically choose an efficient system configuration to meet the registration accuracy requirements for a reconfigurable implementation.

5. Discussion

With the need for real-time performance in signal processing applications, an increasing trend is to accelerate computationally intensive algorithms using custom hardware implementation. A critical step in going to a custom hardware implementation is converting floating-point implementations to fixed-point realizations for performance reasons. This conversion process is an inherently multidimensional problem because several conflicting objectives, such as area and error, must be simultaneously minimized. By systematically deriving efficient tradeoff configurations, one can not only reduce the design time [10] but can also enable automated design synthesis [41, 42]. Moreover, these tradeoff configurations allow designers to identify optimized, high-quality designs for reconfigurable computing applications. Our work presented in this paper develops a framework for optimizing tradeoff relations between hardware cost and image processing accuracy in the context of FPGA-based medical image registration.

Earlier approaches to optimizing wordlengths used analytical approaches for range and error estimations [11–15]. Some of these have used the error propagation method (e.g., see [14]), whereas others have employed models of worst-case error [12, 15]. 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 [43, 44]. 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 [16, 45]. 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 that we have

Table 4: Validation of the objective function models using postsynthesis results. The wordlengths in a design configuration correspond to the FWLs of the design variables identified earlier (see Table 1).

Design configuration	Objective functions postsynthesis value (predicted value)			Registration error (mean ± standard deviations, mm)	Design speed (f_{max}) (MHz)
	MI calculation error	Area (no. of LUTs)	Memory (Mbits)		
$\{5, 6, 4, 9\}$	2.4×10^{-3} (2.1×10^{-3})	6527 (5899)	2.23 (2.23)	3.82 ± 1.24	211
$\{8, 9, 7, 12\}$	5.3×10^{-4} (5.2×10^{-4})	7612 (6754)	2.45 (2.45)	1.57 ± 0.69	197
$\{9, 12, 10, 17\}$	7.7×10^{-5} (7.8×10^{-5})	10356 (8073)	2.81 (2.81)	0.45 ± 0.16	184

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 [42]. Also, some of the methods based on heuristics do not support different degrees of fractional precision for different internal variables [12]. 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 [38]. This approach, however, is prone to finding suboptimal solutions when the search space is nonconvex [17]. Some methods have also attempted to model this problem as a sequence of multiple mono-objective optimizations [46]. 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 [11]. 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 [18, 19] but have not been extensively applied to finding optimal wordlength configurations. One of the earlier attempts at using multiobjective EA formulation for wordlength optimization was reported by Istepanian and Whidborne [47]. This approach employed a simplistic model for hardware complexity and was limited to linear systems only. Leban and Tasic [48] also reported EA-based wordlength optimization of adaptive filters. However, this work was limited to mono-objective optimization only. More recently, Han et al. [49] reported EA-based multiobjective wordlength optimization for a filtering application. This work, however, considered only linear objective functions and lacked postsynthesis validation. 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 nonlinear objective function. Moreover, our optimization framework supports multiple search algorithms and objective function models and may be extended to a wide range of other signal processing applications. A preliminary version of the work presented in this article is published in [50]. This paper represents an enhanced and more thorough version of that work. New developments that we have incorporated into this paper include elaborating on the parameterized architectural design, evaluating the fidelity of the objective function models, and verifying the applicability of the proposed methodology through postsynthesis validation. In summary, this work has presented a framework that is capable of performing multiobjective wordlength optimization and identifying Pareto-optimized design configurations even in the context of nonlinear and complex objective functions. Through postsynthesis validation, this work has also demonstrated the feasibility of such a multiobjective optimization framework in the context of a representative image processing application, medical image registration.

6. Conclusion

One of the main strengths of reconfigurable computing over general-purpose processor-based implementations is its ability to utilize more streamlined representations for internal variables. This ability can often lead to superior performance and optimized fabric utilization in reconfigurable computing applications. Given this advantage, it is highly desirable to automate the derivation of optimized design configurations that can be switched among at run time. Toward that end, this paper has presented a framework for multiobjective wordlength optimization of finite-precision, reconfigurable 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 nonlinear objective functions. The evaluation that we have 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 have presented is not application-specific and, with additional work, may be extended to a multitude of other signal processing applications.

Acknowledgments

This work was supported by the U.S. Department of Defense (TATRC) under Grant DAMD17-03-2-0001. The authors thank Dr. Nancy Knight for her help in editing and refining this manuscript. The authors also thank the journal's reviewers for their feedback and suggestions in improving this manuscript.

References

[1] C. R. Castro-Pareja, O. Dandekar, and R. Shekhar, "FPGA-based real-time anisotropic diffusion filtering of 3D ultrasound images," in *Real-Time Imaging IX*, vol. 5671 of *Proceedings of SPIE*, pp. 123–131, San Jose, Calif, USA, January 2005.

[2] O. Dandekar, C. R. Castro-Pareja, and R. Shekhar, "FPGA-based real-time 3D image preprocessing for image-guided medical interventions," *Journal of Real-Time Image Processing*, vol. 1, no. 4, pp. 285–301, 2007.

[3] S. Venugopal, C. R. Castro-Pareja, and O. Dandekar, "An FPGA-based 3D image processor with median and convolution filters for real-time applications," in *Real-Time Imaging IX*, vol. 5671 of *Proceedings of SPIE*, pp. 174–182, San Jose, Calif, USA, January 2005.

[4] C. R. Castro-Pareja, J. M. Jagadeesh, and R. Shekhar, "FAIR: a hardware architecture for real-time 3-D image registratio," *IEEE Transactions on Information Technology in Biomedicine*, vol. 7, no. 4, pp. 426–434, 2003.

[5] C. R. Castro-Pareja and R. Shekhar, "Hardware acceleration of mutual information-based 3D image registration," *Journal of Imaging Science and Technology*, vol. 49, no. 2, pp. 105–113, 2005.

[6] 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, no. 2, pp. 116–127, 2007.

[7] M. L. Silva and J. C. Ferreira, "Support for partial run-time reconfiguration of platform FPGAs," *Journal of Systems Architecture*, vol. 52, no. 12, pp. 709–726, 2006.

[8] G. A. Constantinides, P. Y. K. Cheung, and W. Luk, "The multiple wordlength paradigm," in *Proceedings of the 9th Annual IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM '01)*, pp. 51–60, Rohnert Park, Calif, USA, April-May 2001.

[9] G. A. Constantinides and G. J. Woeginger, "The complexity of multiple wordlength assignment," *Applied Mathematics Letters*, vol. 15, no. 2, pp. 137–140, 2002.

[10] H. Keding, M. Willems, M. Coors, and H. Meyr, "FRIDGE: a fixed-point design and simulation environment," in *Proceedings of Design, Automation and Test in Europe (DATE '98)*, pp. 429–435, Paris, France, February 1998.

[11] 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, no. 10, pp. 1432–1442, 2003.

[12] A. Nayak, M. Haldar, A. Choudhary, and P. Banerjee, "Precision and error analysis of MATLAB applications during automated hardware synthesis for FPGAs," in *Proceedings of Design, Automation and Test in Europe (DATE '01)*, pp. 722–728, Munich, Germany, March 2001.

[13] A. V. Oppenheim, R. W. Schafer, and J. R. Buck, *Discrete-Time Signal Processing*, Prentice Hall, Upper Saddle River, NJ, USA, 2nd edition, 1999.

[14] M. Stephenson, J. Babb, and S. Amarasinghe, "Bitwidth analysis with application to silicon compilation," in *Proceedings of the ACM SIGPLAN Conference on Programming Language Design and Implementation*, pp. 108–120, Vancouver, Canada, June 2000.

[15] S. A. Wadekar and A. C. Parker, "Accuracy sensitive word-length selection for algorithm optimization," in *Proceedings of the IEEE International Conference on Computer Design (ICCD '98)*, pp. 54–61, Austin, Tex, USA, October 1998.

[16] H. Choi and W. P. Burleson, "Search-based wordlength optimization for VLSI/DSP synthesis," in *Proceedings of the 7th IEEE International Workshop on VLSI Signal Processing*, pp. 198–207, La Jolla, Calif, USA, October 1994.

[17] 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, no. 1, pp. 63–69, 1997.

[18] M. S. Bright and T. Arslan, "Synthesis of low-power DSP systems using a genetic algorithm," *IEEE Transactions on Evolutionary Computation*, vol. 5, no. 1, pp. 27–40, 2001.

[19] 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, no. 4, pp. 390–401, 2002.

[20] E. Zitzler, M. Laumanns, and L. Thiele, "SPEA2: improving the strength Pareto evolutionary algorithm for multiobjective optimization," in *Proceedings of Evolutionary Methods for Design, Optimisation and Control with Applications to Industrial Problems (EUROGEN '01)*, pp. 95–100, Athens, Greece, September 2001.

[21] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan, "A fast and elitist multiobjective genetic algorithm: NSGA-II," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 2, pp. 182–197, 2002.

[22] J. B. A. Maintz and M. A. Viergever, "A survey of medical image registration," *Medical Image Analysis*, vol. 2, no. 1, pp. 1–36, 1998.

[23] D. L. G. Hill, P. G. Batchelor, M. Holden, and D. J. Hawkes, "Medical image registration," *Physics in Medicine & Biology*, vol. 46, no. 1, p. 1, 2001.

[24] 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, no. 2, pp. 187–198, 1997.

[25] J. P. W. Pluim, J. B. A. Maintz, and M. A. Viergever, "Mutual-information-based registration of medical images: a survey," *IEEE Transactions on Medical Imaging*, vol. 22, no. 8, pp. 986–1004, 2003.

[26] V. Walimbe and R. Shekhar, "Automatic elastic image registration by interpolation of 3D rotations and translations from discrete rigid-body transformations," *Medical Image Analysis*, vol. 10, no. 6, pp. 899–914, 2006.

[27] M. Doggett and M. Meissner, "A memory addressing and access design for real time volume rendering," in *Proceedings of the IEEE International Symposium on Circuits and Systems (ISCAS '99)*, pp. 344–347, Orlando, Fla, USA, May-June 1999.

[28] 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, no. 1, pp. 33–45, 1996.

[29] T. J. Todman, G. A. Constantinides, S. J. E. Wilton, O. Mencer, W. Luk, and P. Y. K. Cheung, "Reconfigurable computing: architectures and design methods," *IEE Proceedings: Computers and Digital Techniques*, vol. 152, no. 2, pp. 193–207, 2005.

[30] Altera Corp., "Altera IP Megacore Library," http://www.altera.com/literature/lit-ip.jsp.

[31] W. Luk, S. Guo, N. Shirazi, and N. Zhuang, "A framework for developing parametrised FPGA libraries," in *Proceedings of the 6th International Workshop on Field-Programmable Logic Smart Applications, New Paradigms and Compilers (FPL '96)*, pp. 24–33, Darmstadt, Germany, September 1996.

[32] W. Luk and S. McKeever, "Pebble: a language for parametrised and reconfigurable hardware design," in *Proceedings of the 8th International Workshop on Field-Programmable Logic Smart Applications, New Paradigms and Compilers (FPL '98)*, pp. 1–9, Tallinn, Estonia, August 1998.

[33] Xilinx Inc., "Xilinx Core Generator," http://www.xilinx.com/ise/products/coregen_overview.pdf.

[34] J. Zhao, W. Chen, and S. Wei, "Parameterized IP core design," in *Proceedings of the 4th International Conference on ASIC*, pp. 744–747, Shanghai, China, October 2001.

[35] T. Back, U. Hammel, and H. P. Schwefel, "Evolutionary computation: comments on the history and current state," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 3–17, 1997.

[36] V. Kianzad and S. S. Bhattacharyya, "Efficient techniques for clustering and scheduling onto embedded multiprocessors," *IEEE Transactions on Parallel and Distributed Systems*, vol. 17, no. 7, pp. 667–680, 2006.

[37] 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*, pp. 219–228, Napa, Calif, USA, April 2002.

[38] K. Han and B. L. Evans, "Optimum wordlength search using sensitivity information," *EURASIP Journal on Applied Signal Processing*, vol. 2006, Article ID 92849, 14 pages, 2006.

[39] N. K. Bambha and S. S. Bhattacharyya, "A joint power/performance optimization technique for multiprocessor systems using a period graph construct," in *Proceedings of International Symposium on System Synthesis (ISSS '00)*, pp. 91–97, Madrid, Spain, September 2000.

[40] E. Zitzler and L. Thiele, "Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach," *IEEE Transactions on Evolutionary Computation*, vol. 3, no. 4, pp. 257–271, 1999.

[41] 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, no. 1, pp. 39–57, 2005.

[42] 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*, vol. 47, no. 9, pp. 840–848, 2000.

[43] S. Kim, K. Kum, and W. Sung, "Fixed-point optimization utility for C and C++ based digital signal processing programs," *IEEE Transactions on Circuits and Systems II*, vol. 45, no. 11, pp. 1455–1464, 1998.

[44] K. 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, no. 8, pp. 921–930, 2001.

[45] M. A. Cantin, Y. Savaria, and P. Lavoie, "A comparison of automatic word length optimization procedures," in *Proceedings of the IEEE International Symposium on Circuits and Systems*, pp. 612–615, Phoenix, Ariz, USA, May 2002.

[46] 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, no. 4, pp. 416–422, 2002.

[47] R. S. H. Istepanian and J. F. Whidborne, "Multi-objective design of finite word-length controller structures," in *Proceedings of the Congress on Evolutionary Computation (CEC '99)*, pp. 1–68, Washington, DC, USA, August 1999.

[48] M. Leban and J. F. Tasic, "Word-length optimization of LMS adaptive FIR filters," in *Proceedings of the 10th Mediterranean Electrotechnical Conference (MALECON '00)*, pp. 774–777, Lemesos, Cyprus, May 2000.

[49] K. Han, A. G. Olson, and B. L. Evans, "Automatic floating-point to fixed-point transformations," in *Proceedings of the 40th Asilomar Conference on Signals, Systems, and Computers (ACSSC '06)*, pp. 79–83, Pacific Grove, Calif, USA, October 2006.

[50] O. Dandekar, W. Plishker, S. S. Bhattacharyya, and R. Shekhar, "Multiobjective optimization of FPGA-based medical image registration," in *Proceedings of the IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM '08)*, pp. 183–192, Stanford, Calif, USA, April 2008.