International Journal of Modern Engineering Research (IJMER)
www.ijmer.com Vol. 3, Issue. 4, Jul - Aug. 2013 pp-2205-2214 ISSN: 2249-6645

Optimal Control Problem and Power-Efficient Medical Image Processing Using Puma

Himadri Nath Moulick[1], Moumita Ghosh[2]

[1]*CSE, Aryabhatta Institute of Engg& Management, Durgapur, PIN-713148, India*
[2]*CSE, University Institute Of Technology, (The University Of Burdwan) Pin -712104, India*

ABSTRACT: *As a starting point of this paper we present a problem from mammographic image processing. We show how it can be formulated as an optimal control problem for PDEs and illustrate that it leads to penalty terms which are non-standard in the theory of optimal control of PDEs. To solve this control problem we use a generalization of the conditional gradient method which is especially suitable for non-convex problems. We apply this method to our control problem and illustrate that this method also covers the recently proposed method of surrogate functional from the theory of inverse problems. Graphics processing units (GPUs) are becoming an increasingly popularplatform to run applications that require a high computation throughput.They are limited, however, by memory bandwidth and power and, assuch, cannot always achieve their full potential. This paper presents thePUMA architecture - a domain-specific accelerator designed specificallyfor medical imaging applications, but with sufficient generality to makeit programmable. The goal is to closely match the performance achievedby GPUs in this domain but at a fraction of the power consumption. Theresults are quite promising - PUMA achieves upto 2X the performance of a modern GPU architecture and has up to a 54X improved efficiency on a floating-point and memory-intensive MRI reconstruction algorithm.*

KEYWORDS: *generalized conditional gradient method, surrogate functional, image processing, optimal control of PDEs*

I. INTRODUCTION

For many years medical imaging has aimed at developing fully automatic, software based diagnostic systems. However, the success of those automatic systems is rather limited and the human expert is as much responsible for the final diagnosis as in previous years. Hence, growing effort has been devoted to enhancing the techniques for presenting the medical images as well as additional information. In Germany a particular effort is made in mammography, i. e. X-ray scans of the female breast for early detection of breast cancer. The process of examination by the medical experts is divided into a very short recognition phase (< 1 sec.) and a second verification phase (≈ 1 min.). During the recognition phase, the expert first recognizes the coarse features, then more and more fine features. Tests have shown, that the experts usually form their decisions during this very short recognition phase. Nevertheless, the verification phase is the more critical one. The critical and difficult cases, where the recognition phase does not end with a preliminary diagnosis, most often applies to women in the early stages of cancer. During the verification phase the expert shifts forwards and backwards, thereby alternating in examining small details and in catching an overall impression of the location of critical patterns within the organ. The advent of programmable graphics processing units, or GPUs, for general-purpose computing is one of the major steps taken in computing over the last few years. These GPGPUs which, in the past, have been predominantly used for gaming and advanced image and video editing are now being used by many developers to accelerate inherently parallel programs in several other domains. Indeed, considerable amounts time and engineering effort are often spent in order to modify programs so that they may run effectively on GPUs.Several different application domains observe remarkable speedups when using GPUs, including the following [18]:• 4X to 100X speedup on medical applications, such as biomedical image analysis, 3D reconstruction of tissue structures for large sets of microscopic images and accelerating MRI reconstructions.• 8X to 260X speedup on electronic design automation, such as power grid analysis and statistical static timing analysis.• 4X to 327X speedup on physics applications, such as weather prediction and astrophysics.• 11X to 100X speed up financial applications such as instrument pricing using Monte-Carlo methods.

II. MOTIVATION FROM MEDICAL IMAGING

This process can be supported by presenting the expert different versions of the original image during close up and normal sub phases. More precisely, the expert sees a version with contrast enhanced small details in a close up phase (‘fine scale’), while he sees an image which preserves all major edges but smoothes within regions (‘coarse scale’) during the normal phase. For enhancing fine details in mammography images a variety of algorithm have been proposed. Many of them are based on the wavelet transform due to its property of dividing an image into different scale representations; see for example [7] and references therein. In this work we deal with the development of an optimized presentation for one cycle of the verification phase. To put the problem in mathematical terms, we start with a given image y0 assumed to be a function defined on $\Omega := [0,1]^2$. The fine scale and the coarse scale image are denoted yf and yc respectively. Under the natural assumption of finite energy images wemodel them as functions in $L^2(\Omega)$The goal is, to produce a movie (i. e. atime dependent function) y: $[0,1] \rightarrow L^2(\Omega)$, from the given images y0, yfand yc such that• the movie starts in y0, i. e. y(0) = y0,

the movie sweeps to the fine scale image and to the coarse scale image,e. g. y(t) ≈ yffor t∈[.2, .4] and y(t) ≈ ycfor t∈[.6, .8],• the movie sweeps in a "natural" way.An example for a mammography image, a fine scale, and coarse scale imageis shown in Fig 1. As a first guess one could try to make a linear interpolationbetween the fine scale and the coarse scale representation. Thismethod has one serious drawback: It does not take the scale sweep into account,i. e. all fine details are just faded in rather than developing one afteranother.

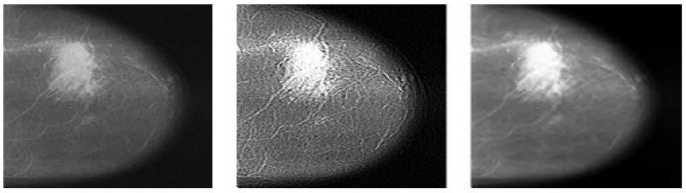

Fig 1: A mammography image. Left: original image y0, middle: fine scale image yf , right: coarse scale image yc
Modeling as an optimal control problem

III. PDES AND CONTROL PROBLEMS IN IMAGE PROCESSING

Parabolic partial differential equations are a widely used tool in image processing. Diffusion equations like the heat equation [14], the Perona-Malik equation [10] or anisotropic equations [13] are used for smoothing, denoising and edge enhancing. The smoothing of a given image $y_0 \in L^2(\Omega)$ with the heat equation isdone by the solution of the equation

$$\begin{aligned} y_t - \Delta y &= 0 \quad \text{in } [0,1] \times \Omega \\ y_\nu &= 0 \quad \text{on } [0,1] \times \partial\Omega \\ y(0) &= y_0 \end{aligned} \tag{1}$$

Where y_ stands for the normal derivative, i. e. we impose homogeneous Neumann boundary conditions.

The solution y: $[0,1] \to L^2(\Omega)$ gives a movie which starts at the image y0 and becomes smoother with time t. This evolution is also called scale space and is analyzed by the image processing community in detail since the 1980s. Especially the heat equation does not create new features with increasing time, see e. g. [5] and the references therein.
Thus, the heat equation is well suited to model a sweep from a fine scale image yf to a coarse scale image yc. Hence, we take the image yf as initial value. To make the movie y end at a certain coarse scale image yc instead of its given endpoint y(1) we propose the following optimal control problem:

$$\begin{aligned} \text{Minimize} \quad & J(y,u) = \frac{1}{2}\int_\Omega |y(1) - y_c|^2 dx + \frac{\alpha}{2}\int_0^1\int_\Omega |u|^2 \, dxdt \\ \text{subject to} \quad & y_t - \Delta y = u \quad \text{in } [0,1] \times \Omega \\ & y_\nu = 0 \quad \text{on } [0,1] \times \partial\Omega \\ & y(0) = y_f. \end{aligned} \tag{2}$$

In other words, the diffusion process is forced to end in yc with the help of a heat source u.

III.1: Adaption to image processing: The above described problem is classical in the theory of optimal control of PDEs, though not well adapted to image processing. The solution of this problem may have several drawbacks: The control u will be smooth due to the regularization and have a large support. This will result in very smooth changes in the image sequence y and, more worse, in global changes in the whole image. To overcome these difficulties, different norms can beused for regularization. A widely used choice in image processing is to use Besov norms because they are appropriate to model images. Besov norms can be defined in different ways, e. g. in terms of moduli of smoothness [12] or in terms of Littlewood-Paley decompositions [6]. Here we take another viewpoint and define the Besov spaces via norms of wavelet expansions [2, 9].For a sufficient smooth wavelet ψ the Besov semi norm of a function f on a set $M \subset \mathbf{R}^d$ is defined as

$$|f|^q_{B^s_{p,q}(M)} = \sum_j \left(2^{sjp} 2^{j(p-2)d/2} \sum_{i,k} |\langle f, \psi_{i,j,k}\rangle|^p \right)^{q/p} \tag{3}$$

Where j is the scale index, k indicates translation and i stand for the directions. The Besov space Bsp, q (M) is defined as the functions f∈Lp (M) that has a finite Besov semi norm. See [6, 9] for a more detailed introduction to wavelets and Besov spaces.

III.2: The solution of the PDE and the control-to-state mapping: The solution of the heat equation is a classical task. If we assume that the initial value yf is in L2(Ω) and the control u is in L2([0, 1] ×) the solution y is in L2(0, 1,H1(Ω)) ∩ C([0, 1],L2(Ω)). Especially y is continuous with respect to time and the point evaluation y(1) makes sense, see e. g. [8]. In our

case the solution operator $u \mapsto y$ is affine linear, due to the nonzeroinitial value. We make the following modifications to come back to a linear problem: We split the solution into two parts. The non-controlled part yn is the solution of

$$\begin{aligned} y_t^n - \Delta y^n &= 0 \\ y^n(0) &= y_f \end{aligned}$$

and the homogeneous part y^h is the solution of

$$\begin{aligned} y_t^h - \Delta y^h &= u \\ y^h(0) &= 0 \end{aligned} \tag{4}$$

(Both with homogeneous Neumann boundary conditions). Then the solution operator G : $u \mapsto y^h$ of equation (1) is linear and continuous from L2([0, 1],L2(Ω)) to L2(0, 1,H1(Ω)) ∩ C([0, 1],L2(Ω)). With the help of the point evaluation operator we have the control-to-state mapping K: $u \mapsto$ yh (1) linear and continuous from L2 ([0, 1],L2(Ω)) to L2(Ω). Then the solution is y = yn + yh and we can focus on the control problem for yh. Together with the thoughts of the previous subsection we end up with the following minimization problem:
Minimize

$$J(u) = \frac{1}{2}\|Ku - y_c + y^n(1)\|^2_{L^2(\Omega)} + \alpha |u|^p_{B^s_{p,p}([0,1]\times\Omega)}. \tag{5}$$

III.3. Solution of the optimal control problem: The minimization of the functional (2) is not straightforward. The no quadratic constraint leads to a nonlinear normal equation which can not be solved explicitly. Here we use a generalization of the conditional gradient method for the minimization.

IV. THE GENERALIZED CONDITIONAL GRADIENT METHOD

The classical conditional gradient method deals with minimization problems of the form

$$\min_{u \in C} F(u), \tag{6}$$

here C is a bounded convex set and F is a possible non-linear function. One notices that this constrained problem can actually be written as an "unconstrained" one with the help of the indicator functional

$$I_C(u) = \begin{cases} 0 & u \in C \\ \infty & u \notin C \end{cases}.$$

With Φ = IC, problem (3) thus can be reformulated as

$$\min_{u \in H} F(u) + \Phi(u). \tag{7}$$

To illustrate the proposed generalization, we summarize the key properties of F and Φ: F is smooth while Φ may contain non-differentiable parts. The minimization problem with Φ alone is considered be solved easily while the minimization of F is comparatively hard. The influence of Φ is rather small in comparison to F. With these assumptions in mind, the conditional gradient method can also be motivated as follows. Let u ∈H be given such that _(u) <∞. We like to find an update direction by a linearized problem. Since Φ is not differentiable, we only linearize F:

$$\min_{v \in H} \langle F'(u)|v\rangle + \Phi(v). \tag{8}$$

The minimizer of this problem serves as an update direction.So this "generalized conditional gradient method" in the (n+1)–st step reads as follows: Let un∈H be given such that Φ (un) <∞.

1. Determine the solution of (5) and denote it vn.
2. Set sn as a solution of

$$\min_{s\in[0,1]} F(u_n + s(v_n - u_n)) + \Phi(u_n + s(v_n - u_n)).$$

3. Let $u_{n+1} = u_n + s_n(v_n - u_n).$

To ensure existence of a solution in Step 1 we state the following condition:Assumption 1 Let the functional Φ : H →]−∞,∞] be proper, convex, lower semi-continuous and coercive with respect to the norm. Standard arguments from convex analysis yield the existence of a minimize in Step 1 of the algorithm [4]. So if F is G^ateaux-differentiable in H, the algorithm is well-defined. The convergence of the generalized conditional gradient method is analyzed in detail by the authors in [1]. The main result there is the following theorem.

Theorem 2 Let Φ satisfy Assumption 1 and let every set Et = {u ∈H | Φ (u) ≤ t} be compact. Let F be continuously Fr´echet differentiable, let F +_ be coercive and u0 be given such that Φ (u0) <∞. Denote (un) thesequence generated by the generalized conditional gradient method.Then every convergent subsequence of (un) converges to a stationarypoint of F + Φ At least one such subsequence exists.Two remarks are in order: First, we notice that the theorem is also validif the functional F is not convex. Second, the theorem only gives convergenceto a stationary point which may seem unsatisfactory, specially if one wantsto minimize non-convex functions. But this does not have to be a drawback,as we will see in the next section:

1. Application: Here we show the application of the above described methodology. Since the effects can be seen more clearly in artificial images, we will not use original images. The artificial images we used are shown in Fig 2.

Fig 2: Images used for illustration. Left: fine scale image, right: coarse scale image.

For illustration we use the values p = 1 and s = 3/2 + " > 3/2 in theminimization problem (2), since this is close to the BV -norm and we have $B_{1,1}^{3/2+\varepsilon}([0,1]\times\Omega) \subset L^2([0,1]\times\Omega)$ compactly.

The results are presented in Figure 3. The figure shows a comparison of the linear interpolation, the pure result of the application of the heat equation and the result of the optimal control problem. One sees that the linear interpolation is only fading out the details. In the uncontrolled result(middle column) the details are vanishing one after another but the process does not end in the desired endpoint. The result of the optimal control problem (right column) exhibits both a nice vanishing of the details and end in the given endpoint.

V. THE ADVANTAGES OF GPUs

GPUs have many appealing hardware features. Firstly, they lend themselves very well to both thread-level and data-level parallelism.Thread-level parallelism (TLP) is exploited by having a large numberof independent processing elements (PEs) on the GPU, each with itsown set of functional units (FUs) and local storage. Individual threadscan quite cleanly be assigned, either statically by the programmer ordynamically by the hardware, to each of these PEs and inter-threadcommunication is made possible by some form of interconnect fabricor through local storage such as caches. Programs with a large amountof data-level parallelism (DLP) can make use of vector-SIMD units inthese PEs which allow a single instruction to perform an operationon several data at the same time. DLP can also be extracted inprograms with compute-intensive loops that have little or no interiteration dependencies by executing operations from different iterationswithin a single SIMD instruction.Secondly, GDDR RAM and its increasingly fast successor's haveallowed for GPUs to have access to an immense amount of memorybandwidth. The AMD Radeon HD 4870 - the first GPU to supportGDDR5 memory - has a memory bandwidth of up to 115 GB/s.Above all, GPUs are commodity hardware products commonly availableas a part of many desktop and laptop computers. The tools toprogramthem are also easily available; NVIDIA's Compute UnifiedDevice Architecture (CUDA) package, for example, is free to downloadfrom their website [15]. CUDA is a general purpose parallel computingarchitecture which consists of the CUDA instruction set and the computeengine in the GPU. It provides a small set of extensions to the C programming language, which enables straightforward implementation of parallel algorithms on the GPU. CUDA also supports scheduling the computation between CPU and GPU, such that serial portions of applications run on the CPU and parallel portions are mapped to the GPU. Individual cores in Intel's up-and-coming Larrabee processor

implement the ubiquitous x86 ISA [23], allowing users to use a host of already-existing development tools to port their applications to it. Server products like Tesla [17] with even more compute power are also available.

VI. THE QUEST FOR PROGRAMMABLE AND SPECIALIZED HARDWARE

A wide range of architectures, in addition to GPUs, have beendesigned before to address the problem of providing high performancecomputation efficiently. These solutions maintain or sacrifice programmability to various degrees depending on the domain they target. Fig 3 shows the performance (on the y-axis) and programmability (on the x-axis) expectations from various architecture styles. The numbers next to each of the ovals shows the approximate performance-power ratio offered by each of these solutions.General purpose processors (GPPs) which fall on the lower rightcorner of the figure, are highly programmable solutions but are limitedin terms of the peak performance they can achieve. Further, structureslike instruction decoders and caches that are needed to support programmability consume energy. This results in a very low computational efficiency of about 1 MIPS-per-mW, for example, for the Intel Pentium- M processor. On the other end of the spectrum are Application-specific Integrated Circuits (ASICs). ASICs are custom-designed specifically for a particular problem, without extraneous hardware structures. Thus, ASICs havea high computational density with hardwired control, resulting in high computation efficiency up to 1,000 to 10,000 times more than that of GPPs. The space between these two extremes is populated by different solutions that have varying degrees of programmability. Application specific instruction-set processors (ASIPs) are processors with custom extensions for a particular application or applicationdomain. They can be quite efficient when running the applications for which they are designed, and they are also capable of running any other application, though with reduced efficiency. Examples include processors from Tensilica and ARC, transport triggered architectures [3] and custom-fit processors [9]. Domain loop accelerators are designed to execute computation intensive loops present in media and signal processing domains. Their design is close to that of VLIW processors, but with a much higher number of function units, and consequently, a higher peak performance. Very long instruction words in a control memory direct all FUs every cycle. However, domain loop accelerators (LAs) have less flexibility than GPPs because only highly computationally-intensive loops map well to them. Some examples of architectures in this design space are RSVP [1] and CGRAs [14]. Coarse-grain adaptable architectures have coarser-grain building blocks compared to FPGAs, but, like FPGAs, still maintain bit-level reconfigurability. The coarser reconfiguration granularity improves the computation efficiency of these solutions. However, non-standard tools are needed to map computations onto them and their success has been limited to the multimedia domain. PipeRench [10], RaPiD [6] are some examples of coarse-grain adaptable architectures.

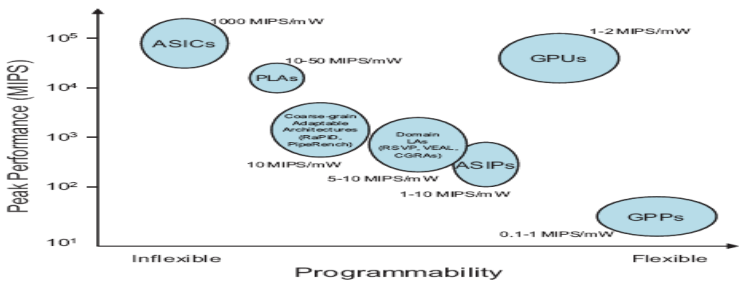

Fig. 3.Comparison of peak performance, power efficiency, and programmability of different architecture design styles.

1. **Programmable Loop Accelerators:** The programmable solutions shown in Figure 1 are all "universally"programmable, allowing any loop to be mapped on to them, although atvarying degrees of efficiency. There is a wide gap between the efficiency that can be achieved by ASICs and the efficiency that can be achieved by these programmable solutions. There are, for example, instances where there is a narrow requirement of flexibility. Using any of these above solutions is overkill for these instances as these solutions sacrifice too much efficiency for the needed flexibility. Further, most of the middlegroundsolutions listed above do not offer any support for fast floatingpoint computation, which limits the number of applications that they can be used for. This work advocates an open area in the design space where a non-trivial amount of programmability is provided in terms of intraprocessor communication, functionality and storage, but the application and domain-specific design, as a whole, resembles an ASIC more than a processor. The design point is labeled Programmable Loop Accelerator, or PLA (not to be confused with programmable logic array).

TABLE 1.Medical application characteristics

Benchmark	#instrs	%FP	Data Req'd $\frac{bytes}{instr}$
MRI.FH	38	42	0.95
MRI.Q	34	35	1.06
CT.segment	26	42	1.38
CT.laplace	20	30	1.20
CT.gauss	22	32	1.09

VII. TARGETING MEDICAL APPLICATIONS

Medical imaging devices are generally large, bulky and expensivemachines that have very limited portability and consume large amountsof power. There is an increasing focus on reducing the power ofthese medical imaging devices [20]. In order to address this issue,this work focuses on principle components of Computed Tomography

(CT) and Magnetic Resonance Imaging (MRI) image processing andreconstruction. A CT scan involves capturing a composite image from a series of X-Ray images taken from various angles around a subject. It produces a very large amount of data that can be manipulated using a variety of techniques to best arrive at a diagnosis. Oftentimes, this involves separating different layers of the captured image based on their radio densities. A common way of accomplishing this is by using a well known image-processing algorithm known as "image segmentation". In essence, image segmentation allows one to partition a given image into multiple regions based on any of a number of different criteria such as edges, colors, textures, etc. Being able to partition images in this manner allows for studying of isolated sections of the image rather than of all the information that was captured.The segmentation algorithm used in this work has three main floating-point- intensive components: Graph segmenting (CT.segment), Laplacian filtering (CT. Laplace) and Gaussian convolution (CT. gauss).

aplacian filtering highlights portions of the image that exhibit a rapid change of intensity and is used in the segmentation algorithm for edge detection. Gaussian convolution is used to smooth textures in an image to allow for better partitioning of the image into different regions. An MRI scan, instead of using X-Rays, uses a strong magnetic and radio frequency fields to align, and alter the alignment of, hydrogen atoms in the body. The hydrogen atoms then produce a rotating magnetic field that can be detected by the MRI scanner and converted to an image. The main computational component of reconstructing an MRI image is calculating the value of two different vectors, known here as MRI.FH and MRI.Q, respectively (explained in more detail in [13], [24]). Table I shows some characteristics of the benchmarks in consideration. All of these benchmarks are floating-point-intensive and require large amounts of data for the computation they perform, especially when compared to the 0.15 bytes/instruction supported by the GTX 280 GPU mentioned earlier.

The main loops in these benchmarks are "do-all" loops - there are no inter-iteration dependences. Prior work in this field has predominantly focused on using commercial products to accelerate medical imaging. For instance, in [11], the authors port "large-scale, biomedical image analysis" applications to multi-core CPUs and GPUs, and compare different implementation strategies with each other. In [21], the authors study image registration and segmentation and accelerate those applications by using CUDA on a GPU. In [24], the authors use both the hardware parallelism and the special function units available on an NVIDIA GPU to dramatically improve the performance of an advanced MRI reconstruction algorithm.

There are several other such examples of novel work in this field. In contrast with such research, this work focuses on designing a new, highly efficient, microarchitecture and architecture with the specific hardware requirements of medical imaging in consideration.

VIII. PUMA

PUMA, ***P***arallel ***micro***-architecture for ***M***edical ***A***pplications, is a tiled architecture as shown in Fig 4. It is specifically designed to maximize power-efficiency when executing medical imaging applications while still retaining full programmability. Each tile in PUMA is an instance of a specialized PLA - a generalized loop accelerator. The PLA tiles are connected to their neighboring tiles and to the external interface through a high-bandwidth mesh of on-chip routers.

1. **Background:** Fig. 5 shows the hardware schema for the single-function loop accelerator [7], [5]. The LA is designed to efficiently execute a modulo scheduled loop [19] in hardware. The lengths of the schedule, and the corresponding run-time of the loop, are determined by the *initiationinterval* (II) - the number of cycles between the beginnings of successive iterations of the loop. Thus, a lower II corresponds to a shorter schedule and increased performance. The modulo schedule contains a *kernel* that repeats every II cycles and may include operations from multiple loop iterations. The LA is designed to exploit the high degree of parallelism available in modulo scheduled loops with a large number of function units (FUs).Each FU performs a specific set of functions that is tailored for the particular loop. Each FU writes to a dedicated shift register file (SRF); in each cycle, the contents of the registers shift downwards to the next register. Point-to-point wires from the registers to FU inputs allow data transfer from producers directly to consumers. Multiple registers may be connected to each FU input; a multiplexer (MUX) is used to select the appropriate one. Since the operations executing in a modulo scheduled loop are periodic, the selector for this MUX is essentially a modulo counter. In addition, a central register file (CRF) holds static live-in register values that cannot be stored in the SRFs. The schema described is a template that is customized for the particular loop being accelerated. The number, types, and widths of the FUs, the widths and depths of the SRFs, and the connections from the SRFs to the FUs are all determined from the loop. During synthesis, the loop is first modulo scheduled to meet a given performance requirement, and then the details of the LA datapath are determined from the communication patterns in the scheduled loop.The control path for the single-function LA consists of a finite state machine with II states corresponding to each of time slots in the kernel of the modulo schedule. In each state, control signals direct the execution of FUs (for FUs capable of multiple operations) and control the MUXes at the FU inputs. Finally, a Verilog HDL realization of the accelerator is generated by emitting modules with pre-defined behavioral Verilog descriptions that correspond to the datapath elements. A simulation environment is used to verify that the Verilog properly implements the loop. Finally, gatelevel synthesis, placement, routing, power analysis and post-synthesis verification are performed on the design.

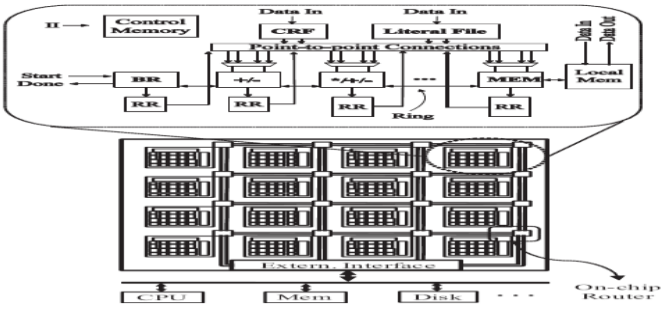

Fig. 4. PUMA. Each tile comprises of a programmable loop accelerator (template pictured) and the control and data memories required for its operation. On-chip routers transfer data between each tile and the external interface.

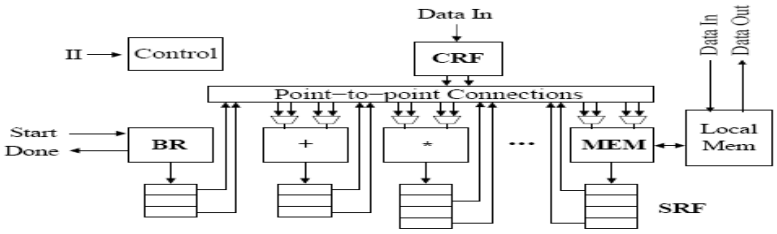

Fig. 5.Template for single-function loop accelerator.

2. PUMA Architecture

2.1. ***Base line PLA Design:*** A PLA is generalized loop accelerator,created by modifying the template datapath shown in Figure 5. A generic datapath template for the PLA is illustrated on the right side of Figure 4 The accelerator is designed for a specific loop at a specific throughput, but contains a more general datapath than the single-function LA to allow for different loops to be mapped onto the hardware [8]. These generalizations provide the LA with flexibility in functionality, storage, control and communication. To provide functionality, simple modifications were made to FUs inorder for them to support more operations; adders (both integer andfloating-point) are generalized to adder/subtracter units, left-shift unitsare generalized to left/right rotators, every FU can execute an identityoperation to act like a move instruction, etc. Even load-store units canbe generalized to integer adder/subtracter units if they already had thefunctionality to compute indirect addresses. Further, the input-muxes toFUs are redesigned to allow for operand-swapping as well.The SRFs used in the LA have limited addressability and fixed lifetimesfor variables. To overcome these constraints and provide moregenerality, these SRFs are replaced with rotating-register files (RRs).To improve controllability, the LA's finite state machine is replacedwith a control memory, the contents of which can be changed based onthe loop that needs to be executed. Further, numerical constants whichwere hard-coded in the LA are instead stored in a literal register file.To generalize communication, the PLA has a bus (in addition to thepoint-to-point connections) that connects all the RRs to all the FUs. Toreduce the hardware cost of having a potentially long bus, its width islimited to one operand, but has a predictable latency of one cycle.

$$\text{Maximize:}$$

$$\sum_{i \in T_\alpha} \sum_{j \in T_\beta} C_{ij} \qquad \forall \alpha \forall \beta : \alpha \neq \beta$$

$$\text{Subject to:}$$

$$\begin{array}{lll}
(1) & \sum_{j=0}^{\#FUs} X_{ij} = 5 & i \in [0, \#FUs) \\
(2) & X_{ii} = 1 & i \in [0, \#FUs) \\
(3) & C_{ii} = 1 & i \in [0, \#FUs) \\
(4) & X_{ij} = X_{ji} & i, j \in [0, \#FUs) \\
(5) & C_{ij} = C_{ji} & i, j \in [0, \#FUs) \\
(6) & C_{ij} \leq X_{ij} + I_{ij} & i, j \in [0, \#FUs)
\end{array}$$

Fig. 6.ILP formulation for FU arrangement on the PUMA ring

2.2. ***PUMA PLA:*** The PLA bus is not always a viable solution. Onemain disadvantage with the bus is that it is not a scalable solutionfor larger PLAs with many FUs. Further, the bus only carries a singleoperand per cycle, limiting the amount of programmability available inthe PLA and the sequences of opcodes that can be executed in parallel.To overcome these limitations, the intra-PLA communication fabricin PUMA is changed to a ring. A ring allows for as many operands to betransferred as there are connections to FUs. It does have its limitations,however. The assumed single-cycle latency to transfer data betweentwo arbitrary points in the PLA using the bus is no longer valid, asit takes one cycle to transfer an operand from one ring connection(or ring stop) to another. Also, the longer the distance an operandneeds to travel on the ring, the more FUs that have to execute moveinstructions to propagate the operand along at each ring stop. Theseadded instructions can potentially increase the loop's schedule lengthand reduce the accelerator's performance. In PUMA, the ring architecture actually consists of six rings (three sets of two rings going in opposite directions). The first set of rings has a

Bus/FU connector (or ring-stop) at every single FU. The second set of rings has a ring-stop at all the odd-numbered FUs, and the third set of rings has a ring-stop at all the even-numbered FUs. This effectivelyconnects an FU/RF pair to its two neighbors and also to its neighbors'neighbors; i.e. every FU can communicate with itself or with otherFUs one or two positions on either side of it on the ring. With thisconfiguration, the number of cycles required to transmit data betweenany two arbitrary FUs is no more than $\lceil \frac{\#FUs}{4} \rceil$, and regardless of theordering of FUs on the ring, every possible producer-consumer pairingcan be executed, provided sufficient time.In order to best maintain generality, we chose to arrange the FUs along the ring to allow maximum connectivity and to distribute the varioustypes of FUs as evenly as possible. This was done by formulatingthe problem as an integer linear program (ILP) as shown in Fig 6.In the objective function, T_ and T_ are two different sets of FUs, each set having all and only the FUs of a particular type. The subscriptsi and j are FU IDs and Cij is a binary variable that is 1 if a connectionexists between FU i and FU j. Essentially, this objective function aimsto maximize the number of connections between different types of FUs, subject to the following constraints: In constraint set (1), Xij is a binary variable that is 1 if FU i is"positioned" adjacent to FU j, implying that they are connected bythe ring. Every FU, therefore, is "positioned" next to 5 other FUs:itself, its two neighbors and the two additional FUs neighboringits neighbors. • Constraint sets (2) and (3) specify that every FU is "positioned"next to and connected to itself.• Constraint sets (4) and (5) specify that all added connections arebidirectional.• In constraint set (6), Iij is a binary number that is 1 if a connectionbetween FU i and FU j has already been inserted by the synthesistool. This constraint enforces the rule that a connection betweenFU i and FU j can only exist if they are either "positioned" nextto each other or are already connected. A 7th set of constraints was initially used which specified thatthere must always be a path between any two FUs with exactly $\lceil \frac{\#FUs}{4} \rceil$ connections between them This constraint was used toprevent insular sets of 5 FUs or sets of 5 FUs connected linearlyrather than in a ring (i.e. without a direct connection between thetwo ends). While this problem might occur in theory, the preexistingconnections put in place by the synthesis system preventit from happening in practice and these constraints were removedto reduce the size of the ILP. Once the optimal solution is obtained, the values of the Xij variables provide a unique ring arrangement.

IX. EXPERIMENTS AND RESULTS

1. Setup: All the PLAs in this work were synthesized for (and run at) afrequency of 450 MHz. The logic synthesis was done using SynopsysDesign Compiler 2006-06 and Synopsys Physical Compiler 2006-06,targeting a 65nm process technology with a nominal supply voltage of0.9 Volts. Energy numbers were obtained using Synopsys PrimeTime-PX 2006-12. For the purposes of this study, we assume that a peakmemory bandwidth of 142 GB/s is available to each PUMA system.This is the same amount of bandwidth afforded to the NVIDIA GTX280 processor.

2. PLA Characteristics: PUMA systems were built using PLAs for each of the five benchmarksin considerations (five systems, each composed entirely ofmultiple tiles of a single type of PLA). Table II shows variouscharacteristics of these accelerators. The "Peak Perf." columns show

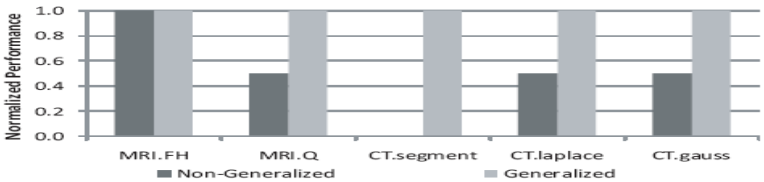

Fig. 7.Normalized performance of benchmarks on LA and PUMA PLA designed for MRI.FH

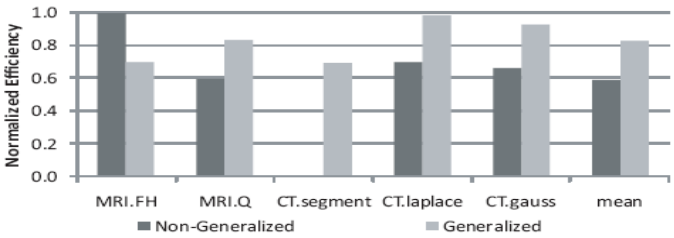

Fig. 8.Normalized $\frac{Performance}{Power}$ efficiency of benchmarks relative to MRI.FH

The throughput when executing floating-point operations and integer operations, respectively, in billions of operations per second. The nextcolumn shows the minimum bandwidth required by each applicationto prevent starvation. Finally, the last column shows the total numberof tiles of each PLA that would be present in a PUMA system. Thenumber of tiles was chosen to prevent data starvation, to make the mostefficient use of the resources available. For example, the number

of tilesin a system with MRI.FH tiles is $\left\lceil \frac{142}{16.2} \right\rceil$ or 9.Fig 8 shows the normalized performance difference betweenthe non-generalized and generalized loop accelerators across various

Benchmarks, to illustrate the effects of the modifications made to thebaseline accelerator to increase programmability. Each of the different benchmarks were compiled for the MRI.FH accelerator.The left column for each benchmark shows its normalized performance. The benchmarks MRI.Q, CT. Laplace and CT. gauss suffered a 50% reduction in performance; i.e. their II values had tobe doubled, from 1 to 2, in order for them to execute on the baselineloop-accelerator. The benchmark CT.segment could not be compiledfor the MRI.FH accelerator at all. For each benchmark, the column on the right shows the achieved performance on the generalized accelerator, with the hardware modificationsspecified in section III-B1. As shown, these modifications allowedall the benchmarks to run at full performance, at minimum II.Fig 8shows a graph similar to that in Fig 7, but showsthe normalized efficiency in terms of the accelerator's performance to-power ratio. Due to the increase in overall performance providedby the generalizations, the benchmarks MRI.Q, CT.laplace andCT.gauss had a significant increase in efficiency despite the poweroverhead of the additions. The MRI.FH benchmark, however, whichwould not experience any improved performance from the generalizations loses efficiency due to the increase in the accelerator's power consumption. On average, the generalizations increased the accelerator's efficiency increased by approximately 40%.

3. Commodity GPGPU Comparison: While other architectures may certainly be used for this domain,GPGPUs are the solutions that are currently in use in many medicalimaging applications and, therefore, the most suitable comparison point.For this reason, we assessed the performance and efficiency of fiveNVIDIA GPUs.

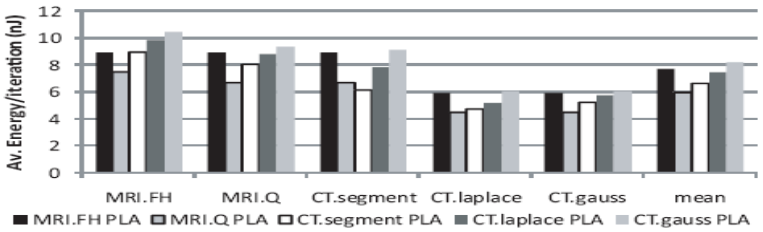

Fig. 9.Average energy consumed (per iteration) by each benchmark while running on PUMA systems designed around different PLAs

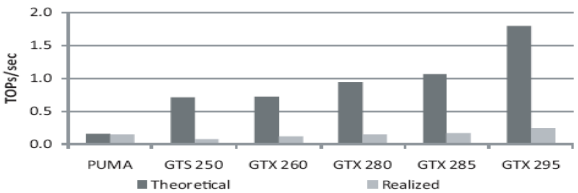

Fig. 10.Achieved performance of the MRI.FH benchmark (in trillions of operations) on the MRI.FH PUMA system and on various NVIDIA GPUs based on the GT200 architecture

Fig9 shows the result of direct performance comparisons betweenan MRI.FH PUMA system and the GPUs in consideration. Thecolumn on the left shows the total compute capability of each of theprocessors. The column on the right shows the realized performancewhile executing the MRI.FH benchmark, accounting for bandwidth restrictions. PUMA achieves a very small fraction of the peak performance offered by the GPUs, between 8.6% of the dual-GPU GTX 295 and 21.8% of the GTS 250. This gap changes dramatically, however, when accounting for the bandwidth-intensive nature of the application in question. PUMA delivers between 63% (on the dual-GPU GTX 295) and 2X the performance (on the GTS 250) of the GPUs.The case for PUMA is further underscored by examining the GPUs'power efficiency, as shown in Fig 10.

This graph shows how many*times* more efficient, in terms of number of operations per Watt, PUMAsystems are relative to the GPUs in consideration. These values rangefrom 20X, for the most complex benchmark running on the mostefficient GPU, to 54X, for the least complex benchmark running onthe least efficient GPU.

X. CONCLUSION

We have seen that the application of the theory of optimal control of PDEs to image processing problems is a fruitful field of research. Besides promising result, even for easy models like the linear heat equation, new interesting mathematical problems arise, like the treatment of non-quadratic penaltyterms. For future research, better adapted PDEs (like the anisotropic diffusion equations) could be investigated.The PUMA architecture is a power-efficient accelerator system designedspecifically for efficient medical image reconstruction. It consistsof tiles of programmable loop accelerators - ASICs with added hardwareto support general-purpose computing - designed around the computation requirements of the image reconstruction domain. As applications in this domain are normally executed on very high-performance GPGPUs, the latest NVIDIA GPU architecture was used to gauge the performance and efficiency of PUMA. The results are very encouraging – PUMA achieves up to 2 times the performance of a modern GPU architecture and has up to 54 times the power efficiency.

REFERENCES

[1]. K. Bredies, D. A. Lorenz, and P. Mass. Equivalence of a generalized conditional gradient method and the method of surrogate functionals.Preprint of the DFG Priority Program 1114? University of Bremen, 2005.
[2]. Cohen. Numerical Analysis of Wavelet Methods. Elsevier ScienceB.V., 2003.
[3]. Daubechies, M. Defrise, and C. De Mol. An iterative thres holding algorithm for linear inverse problems with a sparsity constraint. Com-munications in Pure and Applied Mathematics, 57(11):1413–1457, 2004.
[4]. Ekeland and R. Temam. Convex analysis and variational problems. North-Holland, Amsterdam, 1976.
[5]. L. Florack and A. Kuijper.The topological structure of Scale-Spaceimages. Journal of Mathematical Imaging and Vision, 12:65–79, 2000.
[6]. M. Frazier, B. Jawerth, and G. Weiss. Little wood-Paley theory and the study of function spaces. Number 79 in Regional Conference Series inMathematics.American Mathematical Society, 1991.
[7]. P. Heinlein, J. Drexl, and W. Schneider. Integrated wavelets for enhancement of micro calcifications in digital mammography. IEEE Trans-actions on Medical Imaging, 22(3):402–413, March 2003.
[8]. J.-L. Lions. Optimal Control of Systems Governed by Partial Differential Equations. Springer, 1971.
[9]. Y. Meyer. Wavelets and Operators, volume 37 of Cambridge Studies inAdvanced Mathematics.Cambridge University Press, 1992.
[10]. P. Perona and J. Malik.Scale-space and edge detection usinganisotropic diffusion. IEEE Transactions on Pattern Analysis and Ma-chine Intelligence, 12(7):629–639, 1990.
[11]. L. I. Rudin, S. J. Osher, and E. Fatemi. Nonlinear total variation basednoise removal algorithms. Physical D, 60:259–268, 1992.
[12]. H. Triebel. Theory of Function Spaces II. Monographs in Mathematics. Birkh¨ auser, 1992.
[13]. Weickert. Anisotropic diffusion in image processing. Teubner, Stuttgart, 1998.
[14]. P. Witkin. Scale-space filtering.In Proceedings of the International Joint Conference on Artificial Intelligence, pages 1019–1021, 1983.
[15]. S. Ciricescu et al. The reconfigurable streaming vector processor (RSVP).In *Proc. of the 36th Annual International Symposium on Microarchitecture*,pages 141–150, 2003.
[16]. CNET. The Gizmo Report: NVIDIA's GeForce GTX 280 GPU – introduction, 2008. http://news.cnet.com/8301-13512_3-9969234-23.html.
[17]. H. Corporal. TTAs: Missing the ILP complexity wall. *Journal of SystemArchitecture*, 45(1):949–973, 1999.
[18]. W. Dally et al. Merrimac: Supercomputing with streams. In *Proceedings ofthe 2003 ACM/IEEE conference on Supercomputing*, pages 35–42, 2003.
[19]. G. Dasika, S. Das, K. Fan, S. Mahlke, and D. Bull. DVFS in loopaccelerators using BLADES. In *Proc. of the 45th Design AutomationConference*, pages 894–897, June 2008.
[20]. C. Ebeling et al. mapping applications to the Rapid configurable architecture. In *Proc. of the 5th IEEE Symposium on Field Programmable CustomComputing Machines*, pages 106–115, Apr. 1997.
[21]. Fan et al. Cost sensitive modulo scheduling in a loop acceleratorsynthesis system. In *Proc. of the 38th Annual International Symposiumon Microarchitecture*, pages 219–230, Nov. 2005.
[22]. Fan et al. Modulo scheduling for highly customized data paths to increase hardware reusability. In *Proc. of the 2008 International Symposium on CodeGeneration and Optimization*, pages 124–133, Apr. 2008.
[23]. A. Fisher et al. Custom-fit processors: Letting applications definearchitectures. In *Proc. of the 29th Annual International Symposium onMicroarchitecture*, pages 324–335, Dec. 1996.
[24]. S. Goldstein et al. PipeRench: A coprocessor for streaming multimediaacceleration. In *Proc. of the 26th Annual International Symposium onComputer Architecture*, pages 28–39, June 1999.
[25]. T. D. Hartley, U. Catalyurek, A. Ruiz, F. Igual, R. Mayo, and M. Ujaldon. Biomedical image analysis on a cooperative cluster of GPUs and multicores.In *Proc. of the 2008 International Conference on Supercomputing*, pages15–25, 2008.
[26]. G. Lu et al. The MorphoSys parallel reconfigurable system. In *Proc.Ofthe 5th International Euro-Par Conference*, pages 727–734, 1999.
[27]. Mahesri et al. Tradeoffs in designing accelerator architectures forvisual computing. In *Proc. of the 41st Annual International Symposiumon Microarchitecture*, pages 164–175, Nov. 2008.
[28]. Mei et al. exploiting loop-level parallelism on coarse-grained reconfigurablearchitectures using modulo scheduling. In *Proc. of the 2003 Design, Automation and Test in Europe*, pages 296–301, Mar. 2003.
[29]. Nvidia. *CUDA Programming Guide*, June 2007.http://developer.download.nvidia.com/compute/cuda.
[30]. NVIDIA. GeForce GTX 280, 2008. http://www.nvidia.com/object/ product geforcegtx 280 us.html.
[31]. NVIDIA. NVIDIA Tesla S1070, 2008.http://www.nvidia.com/object/product tesla s1070 us.html.
[32]. Nvidia. Cuda Zone, 2009.http://www.nvidia.com/object/cuda home.html.
[33]. B. R. Rau. Iterative modulo scheduling: An algorithm for softwarepipelining loops. In *Proc. of the 27th Annual International Symposiumon Microarchitecture*, pages 63–74, Nov. 1994.
[34]. T. Review. Cheap, Portable MRI, 2006.http://www.technologyreview.com/computing/17499/?a=f.
[35]. Ruiz, M. Ujaldon, L. Cooper, and K. Huang.Non-rigid registration forlarge sets of microscopic images on graphics processors.*Springer Journalof Signal Processing*, May 2008.
[36]. Sankaralingam et al. Exploiting ILP, TLP, and DLP using polymorphismin the TRIPS architecture. In *Proc. of the 30th Annual InternationalSymposium on Computer Architecture*, pages 422–433, June 2003.
[37]. Seiler et al. Larrabee: a many-core x86 architecture for visual computing.*ACM Transactions on Graphics*, 27(3):1–15, 2008.
[38]. S. S. Stone et al. accelerating advanced MRI reconstructions on GPUs. In*2008 Symposium on Computing Frontiers*, pages 261–272, 2008.
[39]. Taylor et al. Evaluation of the Raw microprocessor: An exposed wire-delay architecture for ILP and streams. In *Proc. of the 31st AnnualInternational Symposium on Computer Architecture*, pages 2–13, June 2004.