Acceleration of Microwave Imaging Algorithms for Breast Cancer Detection via High-Level Synthesis

Acceleration of Microwave Imaging Algorithms for Breast Cancer Detection via High-Level Synthesis

Daniele Jahier Pagliari
Dipartimento di Automatica e Informatica
Politecnico di Torino, Turin, Italy
Email: daniele.jahier@polito.it

Mario R. Casu
Dip. Elettronica e Telecomunicazioni
Politecnico di Torino, Turin, Italy
Email: mario.casu@polito.it

Luca P. Carloni
Department of Computer Science
Columbia University, New York, USA
Email: luca@cs.columbia.edu

***Abstract*—We present the system-level design of two accelerators for two microwave imaging algorithms for breast cancer detection. The accelerators were designed in SystemC and optimized via High-Level Synthesis (HLS). The two algorithms stress the capabilities of commercial HLS tools in different ways: the first is communication-bound and requires careful pipelining of communication and computation; the second is computation-bound and requires the implementation of mathematical functions that are not properly supported by HLS tools. Still, in the span of four months we were able to design and validate about one hundred alternative implementations, targeting a Zynq SoC platform. Furthermore, we were pleased to obtain results that are superior to a previous RTL implementation, which confirms the remarkable progress of HLS tools.**

I. Introduction

The adoption of new medical imaging techniques and the enhancement of existing ones are limited by the lack of cost-efficient and power-efficient computational platforms. Various promising algorithms require a computational capacity difficult to enclose in a medical equipment with a constrained form-factor or a limited power budget. Hence, researchers are investigating the design of specialized *accelerators* to obtain power-efficient, high-performance implementations of innovative algorithms for various medical applications.

An important application domain is Microwave Imaging (MI) for breast cancer detection, one of the most promising alternatives to overcome the limitations of X-Ray mammography. The processing involved in MI is too heavy for a pure software implementation, especially for high resolution images. Fig. 1 illustrates a possible alternative solution, in the form of an embedded architecture in which a CPU offloads the critical parts of the computation to a specialized accelerator.

The design and optimization of accelerators can benefit from recent advances in High-Level Synthesis (HLS), which allow the efficient production of synthesizable RTL implementations (in Verilog/VHDL format) from design specifications given in a high-level programming language (e.g. C/C++, SystemC) [1]. State-of-the-art HLS tools provide a set of knobs to explore a multi-objective design space by synthesizing many alternative microarchitectures, in search for those that are Pareto-optimal in terms of cost vs. performance tradeoffs [2]. Often, an experienced user of these tools can synthesize a microarchitecture that outperforms manually-designed RTL.

We examine two alternative methods for breast cancer MI and use HLS to accelerate the corresponding imaging algorithms, which we specified in SystemC. These algorithms stress HLS tools in different ways. The first, called *MIST Beamforming*, executes a set of fairly simple operations on a massive amount of input data. The second, *MUSIC-Inspired (MUSIC-I)*, processes a relatively smaller amount of data but requires significantly more complex operations. Consequently, the performance of the corresponding accelerators are likely communication-bound for MIST Beamforming and computation-bound for MUSIC-I. For both algorithms, we complete a comprehensive Design-Space Exploration (DSE) by varying parallelism, pipelining, resource sharing, mapping of the memory elements and, in the case of MUSIC-I, implementation of complex mathematical operators. To leverage the potential of the accelerators, we aim at microarchitectures that optimize the pipeline between I/O and computation. We synthesize our designs for FPGA technology and validate them on a Xilinx Zynq board, which is the emulation platform closest to a final ASIC SoC implementation.

This work is partially supported by the NSF (A#: 1219001), by C-FAR (C#: 2013-MA-2384), one of the six SRC STARnet centers, and by the Italian MIUR (project MICENEA).

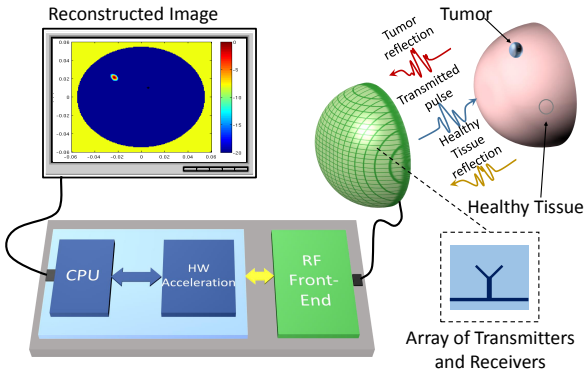

Fig. 1: Microwave Imaging breast-cancer detection system.

II. MI Breast-Cancer Detection

We focus on two linear-scattering methods which irradiate the breast with Ultra Wide Band (UWB) microwave pulses, using multiple (or a single moving) transceiver. The corresponding reflections are sampled and processed to produce an energy map in which tumors are highlighted, thanks to the high dielectric contrast between tumors and healthy tissue in the microwave spectrum. Here we provide a succinct description of the two methods; more details can be found in the cited references.

MIST Beamforming. In Microwave Imaging via Space-Time (MIST) Beamforming [3], the processing synthetically *focuses* the reflections scattered by a point of a volume, i.e. a *voxel*. Sampled reflections are pre-processed to eliminate the clutter due to scattering at the air-skin interface. Then, in the actual beamforming step, they are delayed and filtered to align the scattering contributions of the considered voxel on different antennas, while rejecting noise and reflections due to other locations. Filtering also compensates path-length dependent

Listing 1: MIST Beamforming computational kernel.

```
for vox = 1 : NVOX % loop over voxels
  energy = 0;
  for i = 0 : LH  % loop over samples
    z = 0;
    for ant = 1 : NANT % loop over antennas
      for l = 1 : L % loop over filter weights
        idx = i + l + (NH - L);
        if idx > NA % windowing and alignment
          z = z + ( x(ant, idx - d(vox,ant)) *
              w(vox,ant,l));
        end
      end
    end
    energy = energy + (z * z);
  end
  en(vox) = energy;
end
```

Subtask	Runtime (s)	Subtask	Runtime (s)
FFT	0.013	Computation of F	17.861
Declutter	0.725	Computation of P	0.027
Computation of Eigenvectors	0.007		

TABLE I: Profiling of MUSIC-I steps.

dispersion and attenuation and isolates the frequency band of interest. Multiple windowing steps isolate the samples where the scattered UWB pulse is expected to be found. Finally, signals from different antennas are combined and the voxel energy is computed by squaring and accumulating the samples over the window span. This process is repeated after changing alignment and filtering coefficients, in order to *steer* the beamformer and scan all voxels. Alignment "delays" and filter weights are determined voxel by voxel, and depend on the characteristics of the antennas and of the propagating medium. Listing 1 shows a MATLAB implementation of the beamforming kernel. Matrix x contains the scattered field samples, d and w store alignment delays and filter weights, and en stores the energy values map. A discussion on the determination of the parameters in capital letters can be found in the literature [3], [4].

MUSIC-Inspired Reconstruction. A recently proposed MUltiple SIgnal Classification (MUSIC)-Inspired algorithm has been proved robust when characterization of the UWB antennas and of the heterogeneity in the breast tissue is not possible [5]. While Beamforming works in time domain, MUSIC-I processes the samples in frequency domain via FFT. The N_S frequency-domain samples are organized in a matrix whose columns correspond to N_A antennas. In this case, clutter removal is obtained with a subspace-projection method. Then, the reflected energy map is generated as shown in Listing 2. First, the eigenvectors of the correlation matrix $\mathbf{R} = S_d^n \cdot (S_d^n)^H$ are computed, where S_d^n is the n-th row of the $N_A \times N_S$ decluttered samples matrix S_d. Then, for each voxel position, the dominant eigenvector of $\mathbf{R}$ is multiplied with an array of Green functions, which account for the propagation between the trial voxel and each transceiver, using the Hermitian inner product. The processing is repeated over multiple frequencies (i.e. on all rows of S_d) to reduce image artifacts due to the rank deficiency of $\mathbf{R}$. A detection function P_f combines the results at different frequencies to obtain a peak only for real scattering sources in the breast (i.e. tumors), while rejecting artifacts. In Listing 2, nx and ny represent the image size in pixels, $x(nx)$ and $y(ny)$ store the horizontal and vertical coordinates of the voxels, xo and yo contain the positions of the transceivers, and k_b is the wave number. Matrix P stores the detection function.

Listing 2: MUSIC-I computational kernel.

```
inv_P = 1;
for f = 1 : Nfreq % loop over frequencies
  % correlation matrix
  R = Sd(:,:,f).' * conj(Sd(:,:,f));
  % eigenvectors/eigenvalues computation
  [V,D]=eig(R);
  [max_val, max_idx] = max(abs(diag(D)));
  for u = 1 : nx % loop over image rows
    for v = 1 : ny % loop over image columns
      % green function computation
      Wn = (exp(-j*kb(f) * sqrt((x(u)-xo).^2 +
           (y(v)-yo).^2)).').^2;
      % inner product and norm
      F(u,v,f)= norm((Wn / norm(Wn))' * V(:,max_idx));
    end
  end
  % product over different frequencies
  inv_P = inv_P.*(1-F(:,:,f).^2);
end
P = 1/inv_P;
```

III. Hardware Architecture

Profiling. Previous work on the hardware acceleration of MIST showed that filter weights and alignment delays can be computed offline once the system has been characterized and that the air-skin clutter removal step can be implemented sufficiently fast in software [6], [4]. Consequently, we focused our effort on accelerating the beamforming step (Listing 1). To identify the critical parts of MUSIC-I, we profiled a software execution divided in five steps: FFT, declutter, eigenvectors extraction, Hermitian product (matrix F in Listing 2), and detection function computation (matrix P). We evaluated the execution time for a 200×200 2D image reconstructed using 18 antennas and 20 frequencies. Table I reports the results obtained on an Intel Core i5 - 3337U processor (2.7 GHz, 3-MB L3, 1600-MHz FSB, 4-GB DRAM, Linux kernel 3.13.0-32.), showing that the computation of F (lines 8-15 of Listing 2) is the obvious target for hardware acceleration. Note that, while the execution of MUSIC-I on a small image completes in few tens of seconds, a larger image or a 3D image made of several 2D slices, would require minutes for each patient's breast, a time not acceptable for a clinical scenario.

Microarchitecture. Figs. 2 and 3 show high-level views of the accelerators. Through the slave bus interface, the embedded processor connected to the system bus (Fig. 1) accesses the accelerators register bank to write parameters and commands and to read the execution status. Through the master bus interface, accelerators exchange data with the processor via DMA-like transfers on a shared memory. We developed these interfaces using a synthesizable Transaction-Level Modeling (TLM) library that abstracts the bus protocol by means of generic blocking FIFO primitives. TLM simplifies the porting of the designs to different bus (or NoC) architectures. For validation, we mapped this agnostic model to a signal-level implementation of the AMBA AXI standard.

Internally, the two microarchitectures exploit the intrinsic parallelism that both algorithms offer at the voxel level, as evident in Listings 1 and 2: the accelerators are composed of a series of parallel sub-units (blue rectangles in Figs. 2-3) working concurrently on different data. The granularity of the task performed by each sub-block was determined considering the size of the input data set (Table II), part of which is stored in sub-block-private scratchpad memories before processing. In MIST, weights w and delays d have the largest size and are not

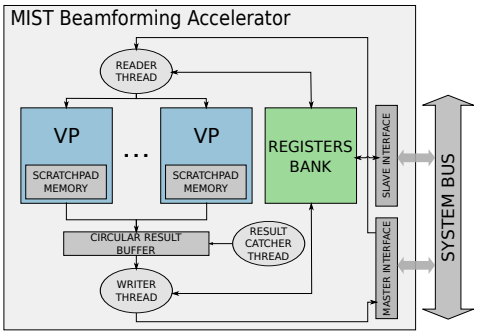

Fig. 2: Accelerator architecture for MIST Beamforming.

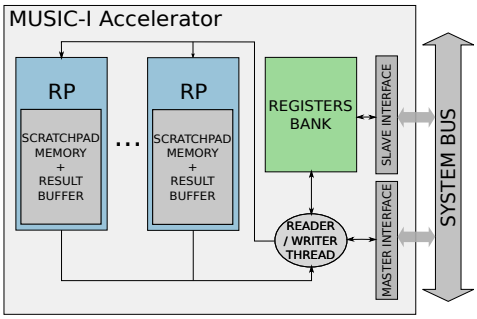

Fig. 3: Accelerator architecture for MUSIC-I.

MIST Beamforming			
Input	**N. elements**	**Input**	**N. elements**
$x[N_S * N_A]$	4608	$d[N_{VOX} * N_A]$	720000
$w[N_{VOX} * N_A * L]$	39600000		
MUSIC-I			
Input	**N. elements**	**Input**	**N. elements**
$xo[N_A]$	18	$y[N_Y]$	200
$yo[N_A]$	18	$kb[2 * N_{FREQ}]$	40
$x[N_X]$	200	$V[2 * N_A * N_{FREQ}]$	720

TABLE II: Accelerators input requirements for a system with 18 antennas, 256 samples/channel, 20 frequencies, 200x200 image size. Factor 2 in MUSIC-I is for complex quantities.

shared among voxels, whereas samples x have a smaller size and are shared. This calls for a natural voxel-level partitioning among parallel elements, hence named *Voxel Processors (VPs)*. To avoid conflicts, each VP receives a copy of x at the beginning of computation. Weights and delays are distributed cyclically to VPs when a new voxel energy must be computed.

In MUSIC-I, instead, the smaller size of the data set and the higher data sharing in the computation of the elements of matrix F (for a single frequency) call for a coarser granularity. Hence, we designed *Row Processors (RPs)* for the computation of each row of F. The larger atomic processing unit reduces the control overhead in the top-level module of the accelerator. On the other hand, an even coarser granularity would have required an excessively large output scratchpad memory for each sub-block. Each RP receives a copy of shared arrays xo, yo, kb, and V, while x and y are suitably partitioned, since every RP operates on separate row ranges.

HLS tools do not permit advanced memory partitioning on a sequential specification (e.g. an unrolled loop structure) during the scheduling phase. Hence, parallelism has to be specified *explicitly* by instantiating VPs/RPs directly in the SystemC description. The resulting model is *structural* at the top-level, due to the explicit instances of the parallel blocks, but is still *behavioral* at the bottom level, due to the TLM-untimed specification of the internal microarchitecture of each block. The number of VPs/RPs can be defined at synthesis time by setting proper preprocessor constants. Hence, it becomes an additional design-space exploration knob, complementary to those provided by the HLS tool such as pipelining, fine-grain parallelization, and resource sharing.

Computation and I/O are orchestrated by the top-level SystemC module. Upon activating the accelerator, the CPU writes in a register the number of voxels/rows to be processed. The top-level module fetches the input data from memory, distributes the workload among VPs or RPs, stalls the I/O if all processing elements are busy, and at end triggers a single interrupt.

Concurrent Computation and Communication. A VP in MIST requires a large input data set and produces a single output energy value. Instead, a RP in MUSIC-I requires much less data but produces an entire image row. To avoid stalling computation when these data are transferred, a ping-pong buffer can be instantiated in VPs input and RPs output scratchpads. Its insertion is another design-exploration variable to be set at synthesis time. In MIST, to maximize I/O throughput, outputs from each VP are stored in a circular buffer and transferred to the bus by the *writer* control logic when enough elements for a *burst* are ready. Meanwhile, the *reader* thread keeps copying input data from memory to VPs scratchpads via DMA. Internal synchronization avoids data loss or overwriting. The control part of the MUSIC-I accelerator is simpler. After a single input burst at the beginning of the processing of each frequency component of matrix F, all RPs start computing together; there is no advantage in pipelining input retrieval and computation. Hence, a single control thread manages both I/O directions. Also, since the number of pixels in an image row is typically larger than the maximum AXI burst length, there is no need to group outputs before a DMA write. In conclusion, MUSIC-I RPs are more complex than MIST VPs, whereas MIST uses a more advanced communication and synchronization scheme.

Mathematical Kernels in MUSIC-I. The math of MUSIC-I is non-trivial from a hardware viewpoint (Listing 2), and most commercial HLS tools do not provide standard implementations of the operations required. Hence, we implemented a SystemC library that maps each operation to one or more synthesizable specifications of known algorithms. When multiple options are available for the same operation, a scheduling directive allows the user to select among them. Thanks to the behavioral TLM-untimed description, further exploration is possible for each algorithm, using standard HLS strategies (e.g. loop unrolling/pipelining, inlining/sharing of the function code, etc). This solution provides more flexibility than devising a "fixed" RTL IP for each operation.

IV. Experimental Results

Setup. We synthesized the SystemC specifications of the two accelerators using a commercial HLS tool. We targeted a Xilinx Artix-7 FPGA technology, in order to validate the designs on a prototype platform based on the Xilinx Zynq XC7Z702 SoC, which is composed of a dual-core ARM Cortex-A9 and the Artix-7 Programmable Logic (PL). The Zynq system bus implements the AXI standard, with slave and master ports accessible from the PL. Hence, we could validate our design from a full system perspective, connecting the accelerated parts with the rest of the algorithms, implemented in software on the Cortex-A9. To complete the HW/SW stack, we wrote a

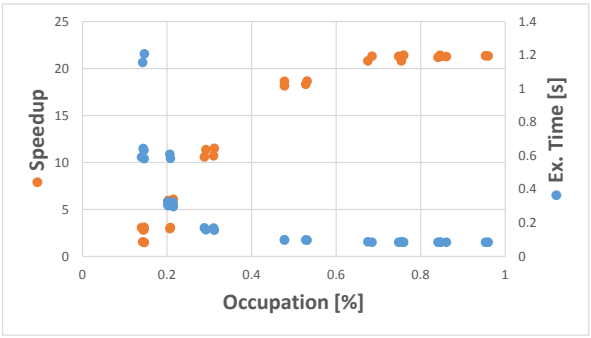

Fig. 4: DSE Results: MIST Beamforming accelerator.

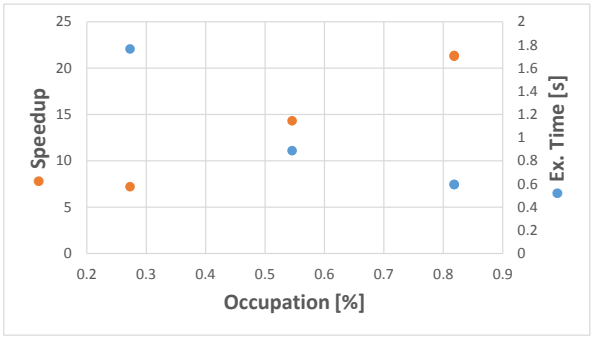

Fig. 5: DSE Results: MUSIC-Inspired accelerator.

device driver for each accelerator on Linux, kernel version 3.13.0. Starting from the HLS-generated RTL code of the two accelerators, we obtained gate-level netlists through logic synthesis using Vivado Design Suite. Due to the different use of PL resources by the two kernels, especially DSP-blocks, MIST and MUSIC-I implementations were synthesized with a clock frequency of 75MHz and 50MHz, respectively.

Design Space Exploration (DSE). For both accelerators, we explored the design space by acting on the HLS-knobs and the custom architectural configurations (e.g. the amount of VP/RP parallelism). *From a single SystemC specification*, we obtained 80 different implementations of MIST and 104 of MUSIC. Fig. 4 and 5 report the Pareto-optimal implementations in the performance vs. cost plane. The cost metric (X axis) is the percentage of occupation of the most critical type of resources on the PL. In most MUSIC-I implementations these resources are DSP-blocks (due to the complex math operations), while in MIST are LUTs. To measure performance (Y axis), we profiled the execution time of the complete reconstruction algorithm on the Zynq, including SW parts and driver overhead. We also evaluated the speedup over the non-accelerated version of the algorithms (running on the Zynq Cortex-A9 at 667 MHz). In these experiments, the MIST Beamforming parameters are N_A=9, L=55, for a 152x176 2D image, while the MUSIC-I parameters are N_A=18, N_{FREQ}=20 for a 200x200 2D image. These results are valid for any set of input samples of the same size because the operations are data independent.

The Pareto curve of MIST (Fig. 4) is much denser in points than the MUSIC-I one (Fig. 5). The reason is that VPs are simpler and smaller than RPs and, therefore, it is possible to allocate from 1 to 16 of them without exceeding the FPGA resources. Instead, all MUSIC-I implementations with more than two RPs exceed the available DSP-blocks and LUTs. Moreover, since the FPGA cost metric considers a single resource type, most solutions were discarded as Pareto-dominated: e.g. in MUSIC-I, for a given number of RPs, solutions without ping-pong buffers were pruned because they use the same number of critical resources (LUTs/DSPs) but execute more slowly. By using HLS, we quickly identified and discarded the dominated implementations, a task that would have required multiple iterations with RTL design. The maximum achieved speedup is similar for the two accelerators, close to 25x. Table III summarizes the DSE results reporting the minimum and maximum values of the main metrics over the whole set of implementations obtained from a single SystemC specification. The performance and area tradeoffs vary by more than one order of magnitude, thus offering a rich set of options for the design reuse of each accelerator across different systems.

	Exec. Time (ms) (HW/SW Speedup)		Occupation (%)	
Algorithm	**Min**	**Max**	**Min**	**Max**
MIST Beamforming	84.43 (21.42)	1207.63 (1.49)	13.9	95.9
MUSIC-I	595.07 (21.38)	1765.92 (7.20)	27.3	100

TABLE III: Range of implementation results.

Speedup versus RTL Design. Finally, we evaluated the effectiveness of the HLS methodology against the RTL MIST Beamforming accelerator described in [6]. We compared the results of the two methods after synthesis, for the same Artix-7 FPGA technology. We obtained that for similar occupation metrics the design synthesized with RTL has an execution time of $148ms$ while the HLS design takes $48ms$, i.e. 3X faster.

V. Conclusions

We presented the system-level design of two accelerators for microwave imaging algorithms for breast-cancer detection. Thanks to the configurable degree of parallelism, from a single SystemC specification for each accelerator we obtain a variety of solutions in terms of cost vs. performance tradeoff. Our work shows the potential that HLS offers in terms of design-productivity gains: while the design and testing of a single RTL implementation of MIST required three months [6], in about four months one designer with no prior knowledge of the two algorithms was able to specify, synthesize, and evaluate 80 (MIST) and 104 (MUSIC-I) alternative implementations.

References

[1] G. Martin and G. Smith, "High-level synthesis: Past, present, and future," *IEEE Design & Test*, vol. 26, no. 4, pp. 18–25, 2009.

[2] H.-Y. Liu, M. Petracca, and L. P. Carloni, "Compositional system-level design exploration with planning of high-level synthesis," in *Proc. of DATE*, 2012, pp. 641–646.

[3] X. Li *et al.*, "An overview of ultra-wideband microwave imaging via space-time beamforming for early-stage breast-cancer detection," vol. 47, no. 1, pp. 19–34, Feb 2005.

[4] F. Colonna *et al.*, "Hardware acceleration of beamforming in a UWB imaging unit for breast cancer detection," *VLSI Design*, vol. 2013, p. 11, 2013.

[5] G. Ruvio *et al.*, "Comparison of noncoherent linear breast cancer detection algorithms applied to a 2-d numerical model," *IEEE Antennas and Wireless Propagation Letters*, vol. 12, pp. 853–856, 2013.

[6] M. R. Casu *et al.*, "UWB microwave imaging for breast cancer detection: Many-core, GPU, or FPGA?" *ACM Trans. Embed. Comput. Syst.*, vol. 13, no. 3s, pp. 109:1–109:22, Mar. 2014.