POLAR FORMAT SYNTHETIC APERTURE RADAR IN ENERGY EFFICIENT APPLICATION-SPECIFIC LOGIC-IN-MEMORY

Qiuling Zhu[⋆], *Christian R. Berger*[†], *Eric L. Turner*[‡], *Larry Pileggi*[⋆], *Franz Franchetti*[⋆]

[⋆]Dept. of Electrical and Comp. Eng., Carnegie Mellon University, Pittsburgh, PA, USA
[†]Wireless System R&D, Marvell Semiconductor, Santa Clara, CA, USA
[‡]Dept. of Electrical Eng. and Comp. Science, University of California Berkeley, Berkeley, CA, USA

ABSTRACT

In this paper we present a local interpolation-based variant of the well-known polar format algorithm used for synthetic aperture radar (SAR) image formation. We develop the algorithm to match the capabilities of the application-specific logic-in-memory processing paradigm, which off-loads lightweight computation directly into the SRAM and DRAM. Our proposed algorithm performs filtering, an image perspective transformation, and a local 2D interpolation and supports partial and low-resolution reconstruction. We implement our customized SAR grid interpolation logic-in-memory hardware in advanced 32nm silicon technology. Our high-level design tools allow to instantiate various optimized design choices to fit image processing and hardware needs of application designers. Our simulation results show that the logic-in-memory approach has the potential to enable substantial improvements in energy efficiency without sacrificing image quality.

Index Terms— Synthetic Aperture Radar, Interpolation, Logic in Memory, Chip Generator

1. INTRODUCTION

The polar format algorithm (PFA) used for image formation in synthetic aperture radar (SAR) is computationally demanding and data-intensive [1, 2]. Its realtime constraints and low-power requirements make it a promising target for advanced power-saving designs. Enabled by latest circuit design improvements [3], the application-specific logic-in-memory paradigm is proposed to move simple computation directly into the memory, and minimize the data movement from memory to the processors (see Fig. 1).

Logic-in-memory builds on the idea of processing in memory [4], however, puts only simple logic instead of actual processing cores right into the memory structures. On an architecture level, the logic-enhanced memories look like normal memories to the CPU, but perform extra (and cheap) operations on the stored data before returning the requested data item to the CPU. Since only limited functionality can be moved into memory, algorithms need to be adapted to match the constraints of the logic-in-memory paradigm.

Design automation is required for handling the increased complexity of memory-logic-mixing hardware accelerators and the intricacies of cutting edge and next-generation silicon technology. Physical implementation of our logic and memory-mixing hardware is enabled by the constructs-based *smart memory compiler* that compiles pattern-compatible logic and memory without the need for pre-designed IP blocks [5]. Further, we build application-specific high-level design tools using the Genesis2 design tool [6, 7]. The combination of these tools allows designers simple design space exploration to optimize their designs for energy budgets, image reconstruction quality, and performance.

Figure 1. Logic-in-Memory computing paradigm.

Contribution. The main contribution of this paper is an algorithm for performing SAR polar format re-gridding interpolation suited for the logic-in-memory paradigm, and to provide the necessary design automation tool chain to implement our proposed algorithm in advanced silicon technology. We combine filtering, geometric transformations, and localized 2D interpolation to provide a virtual rectangular 2D memory address space that is overlaying the polar grid and performs the necessary interpolation on demand. Enabled by this on-demand interpolation our system further provides partial image reconstruction, allowing for reconstructing both low-resolution thumbnails and high-resolution patches.

2. BACKGROUND

Synthetic aperture radar. Synthetic aperture radar is essentially "taking a photo with radar". A radar mounted on a plane sends repeatedly pulses to the scene patch and records the reflections, rotating the antenna to aim at the same scene center for all pulses. The image is formed by computing the inverse fast Fourier transform (FFT) of the recorded data. However, the data is sampled on a polar grid, and the PFA first converts these polar samples into rectangular samples, so that a standard FFT can be applied for image formation. Without this conversion, a computationally infeasible non-uniform Fourier transform would have to be applied [1].

Computationally, the polar-to-rectangular conversion is often done separably (first processing all rows and then all columns of the data), using FFT-based upsampling followed by picking the nearest neighbor to the actual grid points of interest [2]. The reliance on FFTs makes this approach efficient, however, it requires non-local computation due to the well-known FFT data access pattern. An algorithm for logic-in-memory cannot rely on FFTs but requires local computation, as we discuss next. Thus, we need to develop a localized variant of polar-to-rectangular interpolation.

The authors acknowledge the support of the C2S2 Focus Center, one of six research centers funded under the Focus Center Research Program (FCRP), a Semiconductor Research Corporation entity.

Logic-in-memory. Advances in chip design methodology allow the tight integration of computational logic and memory cells, based on regular pattern constructs [3]. This tight integration enables robust compilation of both logic and memory [5]. This gives rise to the application-specific logic-in-memory computational paradigm, which moves part of a program's computation directly into the memory but keeps the usual memory interface. It is easy to program, as all computational operations are hidden behind the memory abstraction. It requires application-specific logic to reach the desired energy savings. Thus, it is more specialized than the earlier processor-in-memory idea [4]. The major restriction of logic-in memory is that only localized (nearest neighbor) data access can be implemented efficiently, and that, stride-like data access as required for FFTs is prohibitively expensive.

Interpolation. A well-suited algorithm for logic-in-memory is 1D interpolation using the Newton divided difference formula [8],

$$P_n(x) = P_{n-1}(x) + f[x_0, \ldots, x_n](x - x_0) \cdots (x - x_{n-1}), \quad (1)$$

in which $P_n(x)$ interpolates $f(x)$ at the points in $x_0, \ldots, x_{n-1}$ with a recursive definition of the n_{th}-order divided difference of $f(x)$ as follows:

$$f[x_0, \ldots, x_n] = \frac{f[x_1, \ldots, x_n] - f[x_0, \ldots, x_{n-1}]}{x_n - x_0}. \quad (2)$$

This approach can be cheaply implemented in logic-in-memory, both for integer and floating-point output. This insight is a crucial enabling step for our logic-in-memory SAR variant.

Image perspective transformation. Another algorithm that is suitable for logic-in-memory and becomes a core component of our SAR variant is the perspective transformation. It is used to map each point (u, v) in one quadrilateral to the point (x, y) in another quadrilateral. The forward mapping function is given by

$$[x', y', w'] = [u, v, w] \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}, \quad (3)$$

where $x = x'/w'$, $y = y'/w'$ and the coefficients a_{ij} are determined by establishing the correspondences between four corners in the input and output geometry [9].

Frequency filter. The final algorithm we are using to develop our logic-in-memory SAR variant is filtering. Straight-forward implementation of finite impulse response (FIR) filters becomes too expensive for long tap lengths, but a finely-tuned combination of FIR and cascaded integrator-comb (CIC) filters can be implemented very efficiently in logic-in-memory. This enables us to implement partial image reconstruction for both low-resolution thumbnails as well as high-resolution scene patches in logic-in-memory. We rely on simple Fourier transform identities to translate phase shifts in frequency space to time-domain displacements [10].

3. LOCAL INTERPOLATION BASED SAR

The main idea underlying our approach is to simply use a standard 2D interpolation for polar data to rectangular data reformatting, which has the potential of being efficient in logic-in-memory. The basic idea is shown in Fig. 2(a): we compute the value of the star $P(x)$ (an exemplary output point) by taking the weighted sum of its four neighbors (black dots; original measurements), using their euclidian distance as weights. However, this computation involves operations too complex for logic-in-memory (square root, arcus tangent). We have to decompose this interpolation into a coordinate

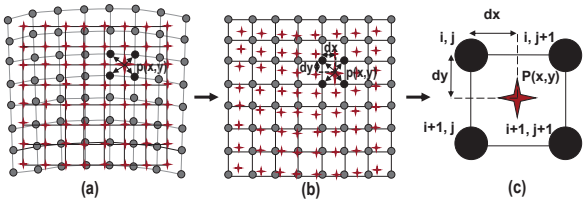

Figure 2. Localized polar-to-rectangular grid interpolation.

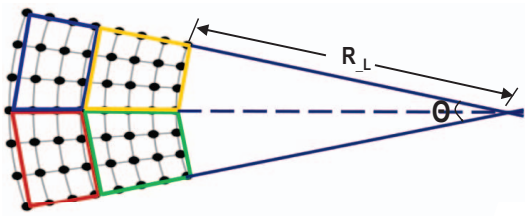

Figure 3. Image tiling for accurate geometric approximation.

transformation and 2D surface interpolation and perform quadrilateral tiling to approximate the curvature by short straight lines.

Coordinate conversion and surface interpolation. Fig. 2(b) shows the first step in implementing the interpolation-based polar formatting: we map the polar anulus (the polar grid on which the SAR data is collected) to a rectangular grid by using the image perspective transformation. This mapping distorts the rectangular destination grid but preserves its distances to the original data points. Then we use standard 2D surface interpolations to calculate the value at the output location from the original data and the interpolation weights derived from the distances in the transformed coordinate system. Fig. 2(c) shows as example the bilinear 2D surface interpolation that requires four neighboring measurements. Both of the transformation and interpolation involve trivial arithmetic logic in logic-in-memory. Although a division is required in perspective transformation, the small dynamic range in the divisor and dividend allows us to replace it by another 2D interpolation.

Geometry approximation and image tiling. Our localized grid interpolation is based on several geometric approximations. Firstly, we approximate the polar annulus by quadrilateral tiles (Fig. 3). Secondly, we assume that the measurement grids are evenly distributed on a rectangular grid after the transformation. These approximations could result in distortions in the resulting reconstructed image. As shown in Fig. 3, accurate approximation is achieved if the radian spatial frequency lower bound (R_L) is large enough (which is true for most SAR applications) and the coherent integration angular interval (Θ) is small enough. Therefore, an effective solution is to tile the image into small parts and perform the geometry approximation on each tile. We tile the output image in the Cartesian grid and find the minimum subset of the polar annulus that contains the corresponding rectangular tile. The resulting distortion is smaller than the intrinsic distortion of perfect SAR image reconstruction.

4. SAR IMAGE PARTIAL RECONSTRUCTION

When reconstructing large data-set problems for small display devices (e.g., handheld devices), partial reconstruction would be preferable to prevent energy waste from processing all pixels and then displaying only a subset. Since our local interpolation-based scheme is reconstructing one pixel at a time in an on-demand fashion, partial reconstruction becomes feasible (see Fig. 4). We support two partial reconstruction modes, as described below.

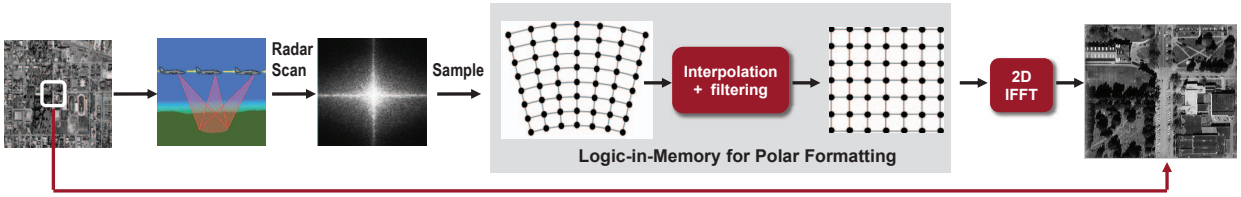

Figure 4. SAR partial image reconstruction.

Low resolution full-size image display. In this scenario, we get a quick overall view of the whole image without the fine-scale details (a thumb nail). This corresponds to multiplying the Fourier space (the original data) with a mask which attenuates the high frequency components. Only data elements that correspond to the low frequency components are interpolated and computations for high frequency components are saved. A much smaller 2D inverse FFT can be used afterwards, saving a substantial amount of operations.

High resolution partial-size image display. As second scenario we reconstruct only a small portion of image (however, at full resolution). This can be seen as multiplication by a mask in the spatial domain, or equivalently, as decimation filtering in the frequency space [10]. Filtering is necessary for image anti-aliasing and the filter decimation factor corresponds to the proportion of the image area to be reconstructed in space. Using Fourier identities we can reconstruct sub-patches of an image at arbitrary position with arbitrary size. In the implementation we rely on the combination of a CIC and short FIR filter for decimation since the CIC filter requires no multiplications and its simple hardware implementation can be easily integrated with the logic-in-memory interpolation, however, accuracy requires us to uses some FIR filtering.

Computational cost. The proposed grid interpolation has economical hardware implementations. For example, 2D bilinear interpolation can be separated into two horizontal and one vertical 1D linear interpolations (or vice versa), and each 1D linear interpolation involves only 2 adders, 1 multiplier and several boolean logic operations. Moreover, these operations are computed locally in the memory and therefore consume much less energy compared with in-CPU computing. For partial reconstruction, the chosen CIC filter only involves 8 adders and 8 storage registers.

5. AUTOMATED DESIGN TOOLS

We use the Genesis2 design tool [7] as the infrastructure to build the SAR polar reformatting logic-in-memory chip generator (see Fig. 5(a)). It enables an application designer to explore the design space to optimize the design by simply varying the parameters and automatically generates the optimized synthesizable logic-in-memory hardware. The physical synthesis of the logic-in-memory hardware is implemented by the pattern construct based "smart memory" compiler [3, 5]. Fig. 5(b) shows the user interface of the SAR polar reformatting logic-in-memory chip generator.

The image formation process requires a series of problem parameters and each parameter setting leads to a different hardware implementation. In addition, both interpolation and filtering are trade-off problems in terms of performance/accuracy/cost. For example, the transition-region of a non-ideal filter will result in added distortion at the image edge. Therefore, the narrower the transition region the better the edge quality, but the higher the hardware cost.

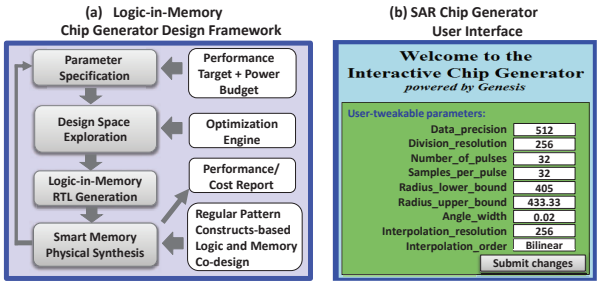

Figure 5. Design automation flow and chip generator.

6. EXPERIMENTAL RESULTS

In this section we evaluate our logic-in-memory based SAR implementation for accuracy, performance and cost. We use our design tool to automatically synthesis the hardware for measurement and also build an architectural model to simulate the algorithm.

Accuracy and hardware cost. First we compare the accuracy of our local interpolation SAR algorithm to the conventional FFT upsampling based approach. We simulate randomized radar scenes of point targets and perform the re-gridding using both methods. Fig. 6(a) shows the mean square error (MSE) distribution for both approaches, which is computed relative to a reference that uses a computationally infeasible non-uniform inverse FFT that has a closed-form solution for point targets. We see that the distortions caused by the two interpolation methods are statistically indistinguishable. In Fig. 6(b) we vary tile numbers and interpolation order. As expected, we see the MSE decreasing for larger tile numbers and higher interpolation order.

Next we evaluate the cost of logic-in-memory. Fig. 6(c) shows that the number of arithmetic operations for the 2D IFFT is decreasing when the decimation factor in partial reconstruction is increased. Fig. 6(d) shows the hardware cost of logic-in-memory blocks on 32nm CMOS; the y axis values are the logic area relative to the data storage memory area. The bottom curve shows the grid interpolation area for the full image reconstruction. For partial reconstruction, the top three curves add in the decimation filter area for three filter design specifications. We see that although the area for partial reconstruction increases slightly with the increase of decimation factor, the y axis values are fairly small for all the design points. Thus, the logic area is negligible compared with memory area for both full and partial reconstruction. Further, to the results in Fig. 6(c) and Fig. 6(d), the decrease in operations through smaller IFFTs is not increasing the hardware cost substantially.

Energy efficiency. To evaluate the energy efficiency of our logic-in-memory SAR implementation, we simulate the whole SAR polar format algorithm in two variants: (1) we run the image reconstruction on a simple processor with a standard SRAM cache, and

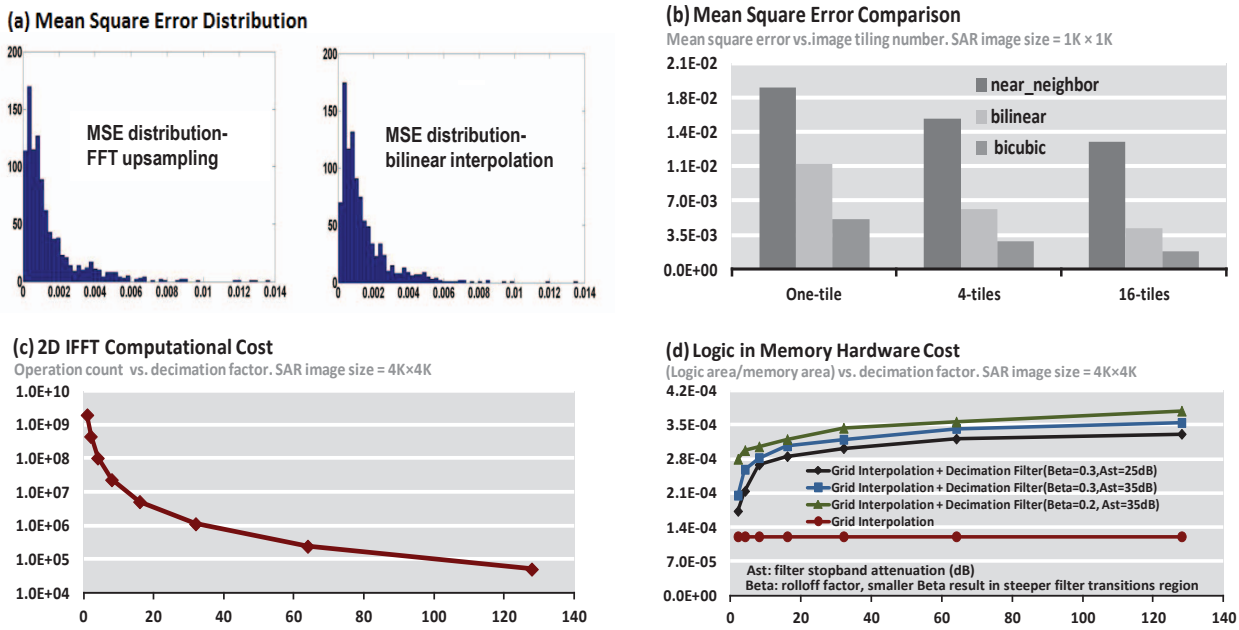

Figure 6. Experimental results.

(2) we replace the cache with our logic-in-memory hardware that performs the interpolation in the memory and run a program reconstructing the image using this memory. We measure the energy consumption using the Wattch simulator, which is an architectural level power simulator based on SimpleScalar [11]. We model the logic-in-memory as direct-mapped on chip memory and scale the memory accessing energy by adding the normalized embedded logic cost from the hardware characterization results. We plot the results for both the conventional and logic-in-memory architecture at different problem sizes from 32 to 512. The results in Fig. 7 show orders of magnitude of energy saving achieved by logic-in-memory especially for large data-size problems.

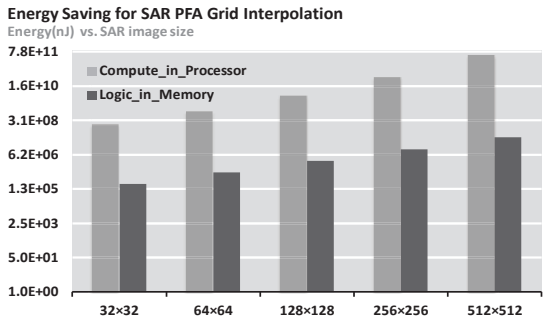

Figure 7. LiM energy consumption savings.

7. CONCLUSION

Advances in integrated circuit design enable the energy-saving logic-in-memory paradigm, which moves part of the computation directly into the memory array. This cutting-edge design methodology requires redesign of well-known algorithms to match its performance characteristics. In this paper we derive a logic-in-memory variant of the polar formatting algorithm used in SAR image formation, and it has equal accuracy as the traditional FFT-based polar formatting algorithm but requires much less energy. Our algorithm further supports partial image reconstruction. We provide the necessary design automation tool chain to enable users to study design trade-offs in the energy and performance space. Our experimental results show substantial energy saving at the same accuracy level.

8. REFERENCES

[1] W. Carrara, R. Goodman, and R. Majewski, *Spotlight Synthetic Aperture Radar: Signal Processing Algorithms*. Artech House, 1995.

[2] D. McFarlin, F. Franchetti, M. Püschel, and J. Moura, "High performance synthetic aperture radar image formation on commodity multicore architectures," *SPIE*, 2009.

[3] D. Morris, V. Rovner, L. Pileggi, A. Strojwas, and K. Vaidyanathan, "Enabling application-specific integrated circuits on limited pattern constructs," *Symp. VLSI Technology*, June 2010.

[4] P. M. Kogge, T. Sunaga, H. Miyataka, K. Kitamura and E. Retter, "Combined DRAM and Logic Chip for Massively Parallel Systems," *Conf. Advanced Research in VLSI.*, 1995.

[5] D. Morris, K. Vaidyanathan, N. Lafferty, K. Lai, L. Liebmann, and L. Pileggi, "Design of embedded memory and logic based on pattern constructs," *Symp. VLSI Technology*, June 2011.

[6] O. Shacham, O. Azizi. et.al., "Rethinking digital design: Why design must change," *IEEE Micro*, vol. 30, no. 6, pp. 9–24, 2010.

[7] O. Shacham, "Chip multiprocessor generator: automatic generation of custom and heterogeneous compute platforms," *PhD Thesis*, Stanford 2011.

[8] A. S. Noetzel, "An interpolating memory unit for function evaluation: Analysis and design," *IEEE Trans. Computers*, vol. 38, no. 3, pp. 377–384, 1989.

[9] G. Wolberg, *Digital Image Warping (Systems)*. IEEE Computer Society Press, 1990.

[10] R. Lyons, *Understanding Digital Signal Processing*. Prentice Hall, 2004.

[11] D. Brooks, and V. Tiwari, "Wattch: a framework for architectural-level power analysis and optimizations," *Proc.ISCA*, 2000.