Energy Efficient Application-Specific Logic-in-Memory for Interpolation in Synthetic Aperture Radar *

Qiuling Zhu, Eric L. Turner, Christian R. Berger, Kaushik Vaidyanathan, Larry Pileggi, Franz Franchetti
qiulingz@ece.cmu.edu, elturner@andrew.cmu.edu
Electrical and Computer Engineering, Carnegie Mellon University

Introduction

In the conventional von Neumann model where computing systems are physically and logically split between memory and CPUs, "memory wall" and "power wall" are well known bottlenecks that have severely limited the energy efficiency of applications. While running today's memory intensive applications, modern computers spend most of their time and energy moving data rather than for computation.

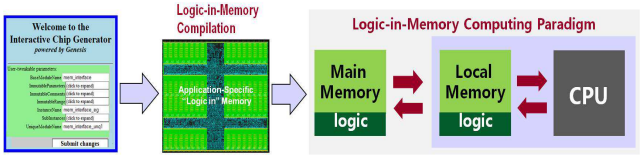


Figure 1: Application-Specific Logic-in-Memory

Application Specific Logic-in-Memory. To enable large savings of energy for important computational tasks, we propose a novel computing paradigm—*Application Specific Logic-in-Memory*—by blurring the distinction between memory and processing logic. As shown in Fig. 1, appropriate application-specific logic is tightly integrated into the on-chip "dumb" memory to enable localized computation. Compared with the well known "Process in Memory" (PIM) [1], the key is to be application-specific, which benefits from algorithm and problem-level knowledge to optimize the embedded logic and memory to a level that is impossible with traditional processor designs. The proposed logic-in-memory blocks act like special-purpose caches or scratch-pads. For the design architect they appear as an ordinary memory block, but internally store and compute data efficiently.

Interpolation Memory. The new memory-centric computational paradigms require that computational logic is simple and amenable enough to be easily embedded into memory arrays for customized localized computation (basic boolean operations, fixed-point additions, shifts, multiplier-less constant multiplications, etc.). While many applications might benefit from this methodology, we initially focus on the "interpolation memory" modules, a "logic in memory" unit that combines the seed table and simple arithmetic logic to efficiently evaluate functions [4]. Previous investigations of 1D interpolation memory have demonstrated very promising results in memory-intensive signal processing applications, and we are convinced that the extension of this study can be widely used in many high performance embedded applications.

*This work was supported by DARPA/SRC FCRP C2S2.

Case Study: Grid Interpolation in SAR. In Synthetic Aperture Radar (SAR), the commonly used Polar Format Algorithm (PFA) is computationally intensive. The majority of the processing time and power is consumed by a grid interpolation involving a resampling of the radar reflectivity function from a curvilinear grid, the polar annulus, to a rectangular grid, the Cartesian grid array [6]. The study shows that this "re-gridding" can be achieved by a simple interpolation algorithm; e.g., local bilinear or bicubic interpolation. The PFA can then be accelerated by using the logic-in-memory computing paradigm and its associated design tools. Simulation results also show that the proposed localized interpolation technique yields comparable error as conventional more computation-intensive interpolation algorithms.

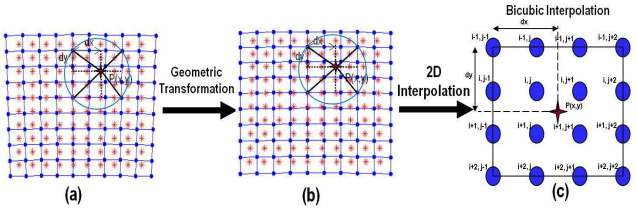
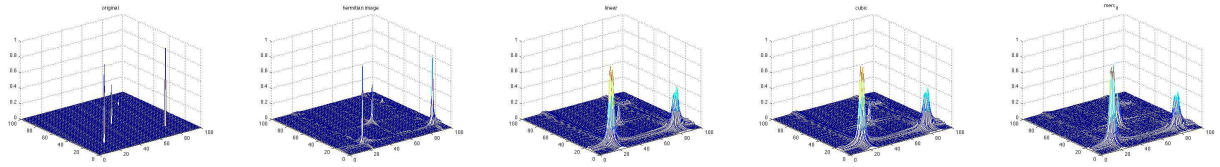


Figure 2: Localized Polar-to-Rectangular Grid Interpolation

Application-Specific Logic-in-Memory for SAR

The proposed polar format interpolation implementation is based on image geometric approximations, 2D surface interpolation, as well as several advanced automatic design methodologies. A high-level visual representation of the process with bicubic interpolation is shown in Fig. 2.

Coordinate Conversion by Geometric Transformation. The localized interpolation is to interpolate the grid points $P(x, y)$ in Cartesian space from their neighbors in the polar annulus, which can be implemented with geometric perspective transformation followed by a general 2D interpolation. For the first step, we map the polar grid into the rectangular grid in a normalized coordinate system by a four-corner image geometric mapping. At the same time, the grid points in Cartesian space is mapped into the same coordinate system by using the same transformation function. Most of this transformation involves trivial arithmetic logic. Although the division operation is required for perspective transformation, its denominator is a linear function of coordinates of the Cartesian grids. So we can implement it by a simple 2D linear interpolation followed by a multiplication leading to negligible accuracy loss. Overall, the whole geometric transformation logic can be efficiently embedded into the memory array via the logic-in-memory method.



(a) Original Scene (b) Gold Standard (c) Linear Interpolation (d) Cubic Interpolation (e) Upsampling Method

Figure 3: An original point target scene, with the outputs using various interpolation methods

Re-gridding by 2D Surface Interpolation. After coordinate transformation, the previous polar grids now evenly lie on the rectangular grids while the rectangular array becomes a quadrilateral in the same coordinate system. However, the relative distances of two grid arrays do not change. That is, in Fig. 2(a) and (b), grid $P(x, y)$ has the same polar grid neighbors and also the same relative neighbor distances (Δx and Δy). Based on the relative locations between the two grid arrays, we can easily determine the neighbor polar annulus (pulse number, sample number) for the grids in Cartesian space. General 2D surface interpolation techniques are then used to calculate the radar reflectivity function value on Cartesian grids. Fig. 2(c) shows the example of bicubic interpolation. Two-dimensional interpolation is a separable transformation, which is the product of multiple 1D interpolations along orthogonal axes. The bicubic interpolation shown in the figure can be divided into four horizontal 1D interpolations and one vertical 1D interpolation. This property enables us to easily extend 2D interpolation into multiple 1D interpolations, resulting in a computationally efficient algorithm.

Chip Generator. The image formation process requires a series of problem parameters such as the number of grid points (i.e., the size of the grid) and the distances between them. These parameters are determined by SAR image size and geometry as well as radar parameters. Different settings lead to different hardware design. On the other hand, the proposed localized interpolation is a trade-off problem in terms of performance/accuracy/cost. To automatically build various design points and to allow for algorithm-level design optimization, we used the Genesis2 [2] design tool developed by Stanford University to build an application-specific chip generator. We codified all the combinations of SAR problem specifications and design trade-offs into a design template and built an entire family of the SAR image format processing chip designs at once. Detailed information of this design methodology is in [3].

Smart Logic-in-Memory Compiler. To ensure robust and energy-efficient circuit design in sub-22nm, we map memory and logic onto a set of pre-characterized pattern constructs, enabling logic-in-memory [5]. Exploiting the opportunities provided by modern process and physical design tools, applications could be compiled into smart memory module where logic is integrated into the embedded memory. Since 2D interpolation requires one-cycle segmentation-free functional access of a rectangular image window, we create a rectangular-access smart memory with the proposed logic-in-memory compiler and it provides rectangular accessibility at any pixel position. This helps to save a significant area and power overhead of memory peripheral circuits compared with the conventional multi-bank memory design approach. With this new design methodology and associated suite of design tools, design optimization and customization will be enabled at all the levels of abstraction (i.e. architectural, logic, and physical).

Experimental Results

While the proposed implementation uses a localized bilinear or bicubic interpolation, typically a different interpolation scheme is used in existing implementations. Commonly, two 1D interpolation steps are used that upsample the polar grid uniformly and then use a nearest-neighbor approach to find values on the rectangular grid [6]. Since uniform upsampling can be done efficiently using the (fast) Fourier transform, this method is seen as advantageous, although the memory access pattern is not suitable for logic-in-memory. Comparing the output of the common approach with that of the proposed bilinear and bicubic interpolation, we see the distortion caused by these interpolation methods are indistinguishable from one another.

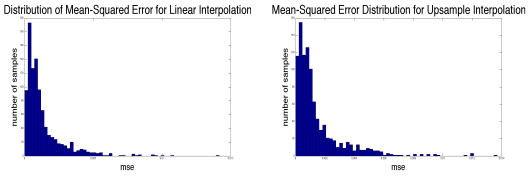


Figure 4: Probability Distribution of Mean-Squared-Error for linear interpolation (left) and upsampling interpolation (right)

For quantitative comparison we simulated a randomized radar scene of point targets and performing the interpolation for each method (see Fig. 3). A reference "gold standard" was included that is based on the non-uniform inverse Fourier transform of the polar samples. The MSE in Fig. 4 is computed relative to the output of this gold standard. This comparison shows that each of the three interpolation methods considered result in the same output image distortion.

References

[1] J. B. Brockman and P. M. Kogge. The case for processing-in-memory. *IEEE Computer*, 1997.

[2] Stanford genesis2 web site. http://genesis2.stanford.edu/mediawiki/index.php.

[3] A. Solomatnikov; A. Firoozshahian; W. Qadeer; O. Shacham; K. Kelley; Z. Asgar; M. Wachs; R. Hameed and M. Horowitz. Chip multi-processor generator. *DAC*, pages 262–263, 2007.

[4] A. S. Noetzel. An interpolating memory unit for function evaluation: Analysis and design. *IEEE TRANS. ON COMPUTERS*, 38(3), 1989.

[5] D. Morris; V. Rovner; L. Pileggi; A. Strojwas; and K. Vaidyanathan. Enabling application-specific integrated circuits on limited pattern constructs. *IEEE Symposium on VLSI Technology*, 2010.

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