Algorithm/Hardware Co-optimized SAR Image Reconstruction with 3D-stacked Logic in Memory

Fazle Sadi, Berkin Akin, Doru T. Popovici, James C. Hoe, Larry Pileggi and Franz Franchetti
Department of Electrical and Computer Engineering
Carnegie Mellon University, Pittsburgh, PA, USA
Email: {fsadi,bakin,dpopovic,jhoe,pileggi,franzf}@andrew.cmu.edu

***Abstract*—Real-time system level implementations of complex Synthetic Aperture Radar (SAR) image reconstruction algorithms have always been challenging due to their data intensive characteristics. In this paper, we propose a basis vector transform based novel algorithm to alleviate the data intensity and a 3D-stacked logic in memory based hardware accelerator as the implementation platform. Experimental results indicate that this proposed algorithm/hardware co-optimized system can achieve an accuracy of 91 dB PSNR compared to a reference algorithm implemented in Matlab and energy efficiency of 72 GFLOPS/W for a 8k×8k SAR image reconstruction.**

I. Introduction

Synthetic Aperture Radar (SAR) is a technique for constructing a high resolution image of a remote target by sending repeated radio frequency pulses and receiving the reflections containing the phase shift information of the pulses. The collected phase shift information constitutes a set of frequency domain data distributed on a polar/curvilinear grid. Advanced signal processing algorithms are applied afterwards to construct the target image on a rectangular grid. Among the SAR image reconstruction algorithms, Polar Format Algorithm (PFA) is among most mature and widely used algorithms [1], [2]. PFA mainly accomplishes reconstruction through a non-uniform 2D Fourier transform performed in two steps, namely re-gridding followed by a regular 2D Fast Fourier Transform (FFT). The re-gridding constructs a frequency domain data set distributed on a rectangular grid through re-mapping and interpolation. This re-gridding operation is known to be the most data intensive, and hence expensive, part of SAR image reconstruction [3], [4]. An overview of the entire process is depicted in Figure 1.

Despite the well developed signal acquisition sensors and mature reconstruction algorithms, high performance implementation of SAR algorithm on Hardware (HW) is still challenging. The reason is that the reconstruction algorithms are extremely data intensive and SAR applications require real-time computation. More importantly, energy efficiency is of prime concern for SAR applications in small form factors like for unmanned aerial vehicles (UAVs).

With the advent of parallel processing through multi-cores, a number of state of the art architectures have used for efficient implementations of SAR. In [5], a SAR image reconstruction algorithm has been implemented on a eight core DSP processor (C6678-Shanon from Texas Instruments Inc.). The authors report an energy efficiency of 12.8 GFLOPS/W for a single precision implementation. In [4], Intel Quad Core CPUs are used along with automatically generated program which provides efficient parallelization, vectorization and memory hierarchy tuning. The reported performance in this work is 0.3 GFLOPS/W. IBM's Cell Broadband Engine (CBE) utilizing high-bandwidth XDR main memory has been exploited for efficient SAR image reconstruction in [3]. Using eight synergistic processing elements, this platform achieves a performance of 23.8 Mpixel/s. However, the authors of this work have not reported any estimation of energy efficiency.

In this work, we approach the SAR image reconstruction as an algorithm/HW co-optimization problem. We propose a novel basis vector transform based algorithm for re-gridding of the polar data. This algorithm is highly optimized to reduce memory accesses. In addition, as the implementation platform we propose a Logic in Memory (LiM) enhanced 3D DRAM stacked HW accelerator which enables both high bandwidth and energy efficiency. We propose an offload accelerator design, as shown in Figure 2, where the accelerator resides on the main memory side rather than the conventional CPU side (e.g. OMAP processor [6]). The advantage of this offload design is that the accelerator has access to the entire off-chip main memory which is essential for data intense applications. The computing unit in this proposed system is an application specific LiM layer which tightly integrates logic for the proposed SAR algorithm and embedded memory blocks. This LiM layer is stacked between 3D DRAM dies where communication among the layers are done through Through Silicon Vias (TSVs) [7], [8]. These TSVs enable high data bandwidth to match the high computation power achieved by the optimized algorithm and LiM topology. For the efficient implementation of the 2D inverse FFT, we have used FFT designs automatically generated by Spiral [9]. End-to-end RTL level simulation of our algorithm/HW co-optimized system reports an accuracy of 91dB PSNR and energy efficiency of 72 GFLOPS/W for the SAR image reconstruction at 32 nm technology node.

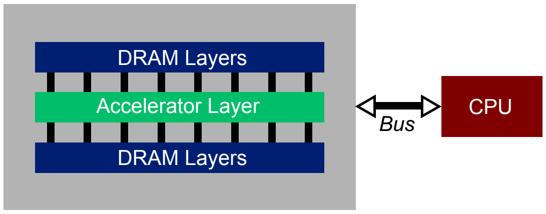


Fig. 2: 3D-stacked offload accelerator.

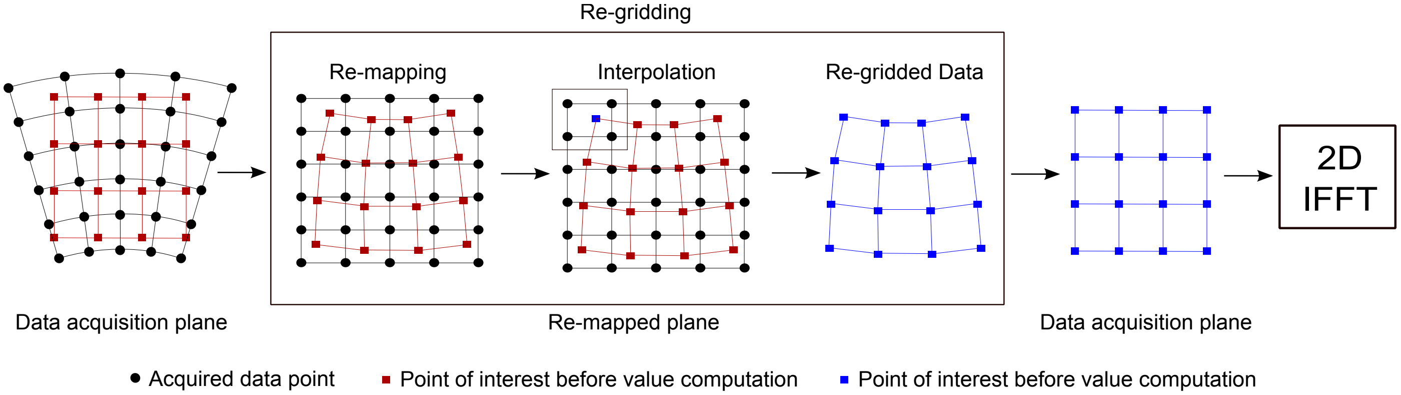


Fig. 1: Different stages of SAR image reconstruction algorithm.

II. Logic in Memory

LiM is a topology where fine-grained interspersion of logic and memory is exploited to build a energy efficient and high internal bandwidth system. This topology has become possible in deeply scaled technology nodes due to the sub-20nm regular pattern construct based IC design [10], [11]. LiM yields tremendous benefits at the system level mainly due to two attributes—proximity and flexibility. The proximity enables high data bandwidth, whereas the flexibility provides more freedom to optimize the algorithm. However, to fully harness the benefits, a design automation framework for LiM HW synthesis is necessary which is developed in [12]. Our work utilizes this framework for simulation. However, the description of these design tools is beyond the scope of this paper.

III. Proposed Re-gridding Method

The re-gridding for SAR image reconstruction is implemented in two steps, namely re-mapping and interpolation. The re-mapping stage converts the polar/curvilinear grid input data into a rectangular grid data set. As a result, the points of interest which are originally in rectangular grid becomes warped after re-mapping as shown in Figure 1. However, this warped grid in the re-mapped plane represents the expected rectangular grid data points in the data acquisition plan required for SAR image reconstruction with uniform 2D inverse FFT.

We propose a basis vector transform based re-mapping algorithm. At first we present the ideal algorithm where no optimization is applied. Later we extend the ideal algorithm to an optimized version which is significantly less data intensive and more energy efficient.

A. Ideal Re-mapping Algorithm

In the data acquisition plane, the polar/curvilinear grid is constructed with trapezoidal grid blocks. For example, in Figure 3, the trapezoid $mnok$ in data acquisition plane constructs the grid block. To compute the value of any point of interest (g_x, g_y) by applying bilinear interpolation, we would need to compute the distances $w_{1\to 4}$ by the Pythagorean theorem four times. This requires square-root operation which is very expensive in HW. More importantly, for each corner points of the trapezoid, three memory accesses are needed (two for the coordinate position and one for the data value). Thus the entire operation becomes both computation and memory intensive.

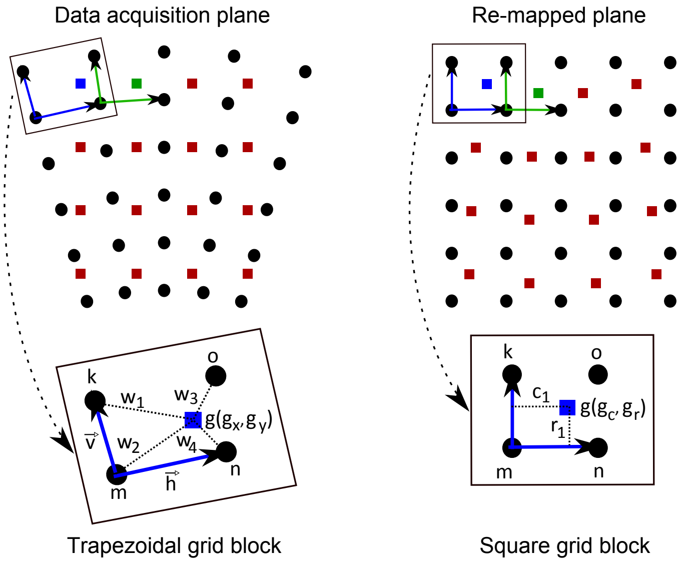


Fig. 3: Data re-mapping through basis vector transform for each grid block.

However, if we convert each trapezoidal grid block into a square grid block, bilinear interpolation would only need the distances c_1 and r_1 which can be computed by simple subtraction as shown in Figure 3. Moreover, as the memory addresses for the data values of the corner points can serve as the coordinate positions, we would only need one memory access (for the data value) to interpolate each point.

To re-map the point of interest (g_x, g_y) to (g_c, g_r), we transform the basis vectors of (g_x, g_y) from the standard Cartesian unit vectors to $\overrightarrow{h}$ and $\overrightarrow{v}$ as shown in Figure 3. This transformation can be done by the formula shown in Equation 1.

$$\begin{bmatrix} g_c \\ g_r \end{bmatrix} = \frac{1}{h_x v_y - v_x h_y} \times \begin{bmatrix} v_y & -v_x \\ -h_y & h_x \end{bmatrix} \times \begin{bmatrix} g_x \\ g_y \end{bmatrix} \qquad (1)$$

Performing such a transformation for all the points of interest for their corresponding trapezoidal block would result in a fully re-mapped rectangular grid with square blocks. Afterwards, interpolation can be done using these square blocks with much less computation compared to the trapezoidal blocks. However, this ideal re-mapping still needs to access memory for the coordinates of three corner points (in this case k, m and n) for each grid block, rendering the process to be data intensive.

B. Optimized Re-mapping Algorithm

For optimization, the entire data set is divided into smaller tiles and the curvilinear grid lines are approximated as straight lines for each individual tile as shown in Figure 4(a). The tile size is selected so that one entire tile data fits in the embedded memory of the LiM layer. The straight line approximation allows us to use one common basis vector for all the points of interest encompassed by a tile. For the tile shown in Figure 4(b), the common basis vector is $\overrightarrow{pq}$. It will be shown in Subsec. VI-A that the straight line approximation incurs very insignificant pixel shift in the resultant image. As can be intuitively understood, the effect of straight line approximation diminishes as the tile size decreases.

To remap any arbitrary point of interest, the other basis vector used is the vector connecting that point and the origin of the data acquisition plane, having a length of the tile size in the range direction [1], [2]. In Figure 4(b), for point (g_x, g_y), the second basis vector would be $\overrightarrow{bd}$.

For processing each tile, the coordinates of the four corner points (p, q, r and s in Figure 4(b)) are passed as parameters to the LiM core. Therefore, computing the common basis vector, $\overrightarrow{pq}$, for the entire tile is straightforward. However, we need the coordinates of points b and d for the second basis vector. As $\overrightarrow{bg}$ passes through the origin of the coordinate we can find (b_x, b_y) with the following formulae.

$$b_x = \frac{p_y(q_x - p_x) - p_x(q_y - p_y)}{(g_y/g_x)(q_x - p_x) - (q_y - p_y)}, \;\; b_y = \frac{g_y}{g_x} b_x$$

To find (d_x, d_y), we first define,

$$\alpha = \frac{b_x}{q_x - p_x} = \frac{b_y}{q_y - p_y}.$$

As $\overrightarrow{ps}$, $\overrightarrow{bd}$ and $\overrightarrow{qr}$ pass through the origin of the data acquisition plane, coordinates of d can be derived from the following equations.

$$d_x = (1-\alpha)s_x + \alpha r_x, \;\; d_y = (1-\alpha)s_y + \alpha r_y$$

After the coordinates of b and d are found in the data acquisition plane, Equation 1 can be used to remap (g_x, g_y) to (g_c, g_r) using $\overrightarrow{bd}$ and $\overrightarrow{pq}$ as the basis vectors. Alternatively, we can use the following equations to remap (g_x, g_y).

$$g_c = \alpha, \;\; g_r = \frac{g_x - b_x}{d_x - b_x} = \frac{g_y - b_y}{d_y - b_y}$$

It should be noticed that in this optimized remapping algorithm, memory accesses for the curvilinear data point coordinates are no longer required. Only the coordinates of the four corner points of each tile in data acquisition plane are needed, which can be passed as parameters to the computational LiM core of the system.

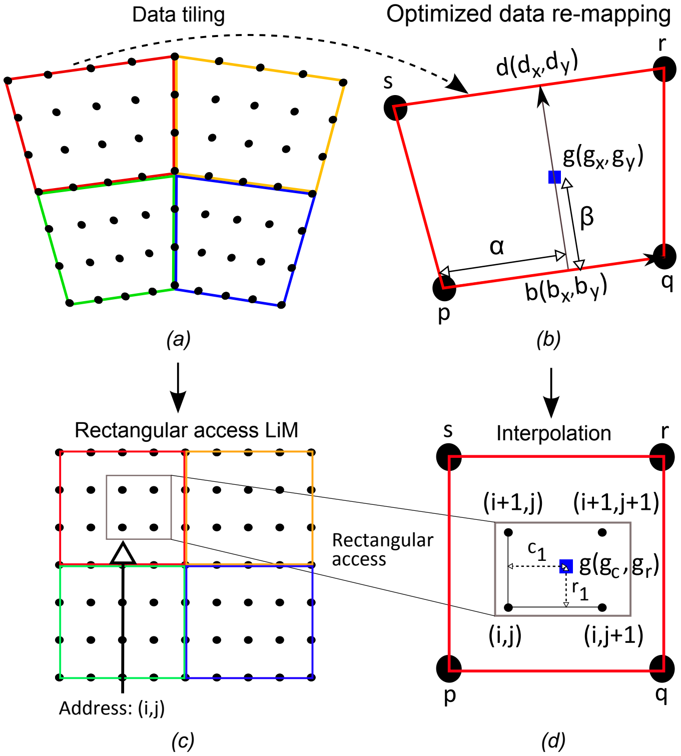


Fig. 4: Data re-mapping through basis vector transform for each grid block.

C. Interpolation with Rectangular Access LiM

For the interpolation in 2D space, multiple memory accesses are generally needed, depending on the order of the interpolation. For example, in bilinear interpolation four data points are needed (two memory accesses), and in bicubic interpolation sixteen data points are needed (four memory accesses). However, using LiM topology we are able to implement a rectangular access memory, which can provide all the required data points in a single memory access. As shown in Figure 4(c, d), for the given address (i, j) all the four points (assuming bilinear interpolation) $(i, j), (i+1, j), (i, j+1)$ and $(i+1, j+1)$ are provided in a single memory access. In addition, we need the distances c_1 and r_1 for the interpolation which can be computed by the following equations.

$$c_1 = g_c \times (\text{number of columns in one tile}) - j$$
$$r_1 = g_r \times (\text{number of rows in one tile}) - i$$

Thus, the LiM based HW specially tuned for 2D interpolation and the optimized remapping algorithm enables an algorithm/HW co-optimized system for high performance and energy efficient regridding operation.

IV. 2D-IFFT and System Integration

After re-gridding, a 2D inverse FFT operation is needed to reconstruct SAR image. For an efficient HW implementation of FFT we use Spiral [9] formula generation and optimization framework. Spiral features block data layout FFTs for large datasets of SAR images to address the DRAM bandwidth utilization [13], [14]. These DRAM-optimized FFT implementations make use of the tiled memory layout by mapping each

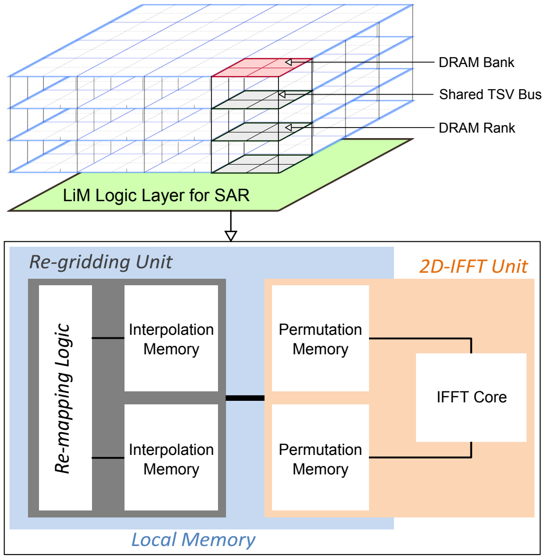


Fig. 5: Integration of re-gridding and 2D-IFFT.

tile to a DRAM row hence minimize the number of row buffer misses.

The overall architecture shown in Figure 5 demonstrates the integration of the SAR re-gridding with the 2D-FFT hardware. The re-gridding and FFT units are implemented in the logic layer of a 3D-stacked DRAM similar to [15], [16]. The 2D-FFT requires double-buffered local memory that performs data permutations and a local FFT core that executes the FFT kernel [13], [16]. We also construct a double-buffered interpolation unit that streams the interpolated rectangular grid data into the 2D-FFT unit. We term the LiM block needed for the interpolation as *Interpolation memory* and the SRAM block needed to permute the local as *Permutation memory*. Tile size of the interpolation and the tile size of the 2D-FFT are matched to a DRAM row to minimize the row buffer misses in the data read/write from/to DRAM layers. Our architecture exploits the parallelism provided by multiple banks/ranks/layers/TSVs by transferring multiple elements in parallel. Further, it overlaps the computation and the data transfer via double-buffering. Finally, tiled memory layout allows reading/writing large contiguous data chunks exploiting the data locality. Thus, the overall architecture constitutes a DRAM-optimized LiM based SAR image reconstruction unit.

V. Experimental Methodology

Given the SAR input data set and hardware parameters, our tool generates the HDL as well as simulation and synthesis scripts. The generated HDL is then synthesized targeting a commercial *32 nm* standard cell library using Synopsis Design Compiler following the standard ASIC synthesis flow. In addition to the standard ASIC synthesis flow, for non-HDL components, we use the following tools: CACTI 6.5 for on-chip RAMs and ROMs [17], McPAT for DRAM memory controllers [18], and DesignWare for single and double precision floating point units [19]. For the 3D-stacked DRAM model we use CACTI-3DD [20]. Finally, for the overall performance estimation, we use a custom performance model calibrated by cycle-accurate simulation. All of the tools are integrated resulting in an automatic push-button end-to-end design generation and exploration tool.

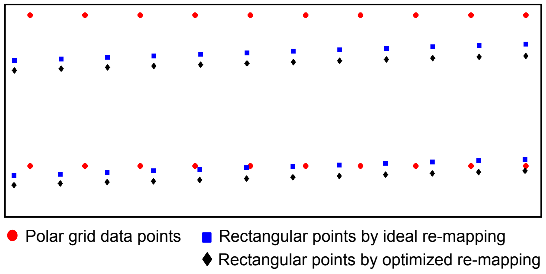


Fig. 8: Effect of curvature approximation.

VI. Experimental Results

We evaluate our 3D-stacked LiM HW accelerator in terms of accuracy, performance and energy efficiency. While accuracy of the reconstructed image is mainly dictated by the re-gridding algorithm, performance and energy efficiency depend on the HW design parameters and resources used.

A. Accuracy

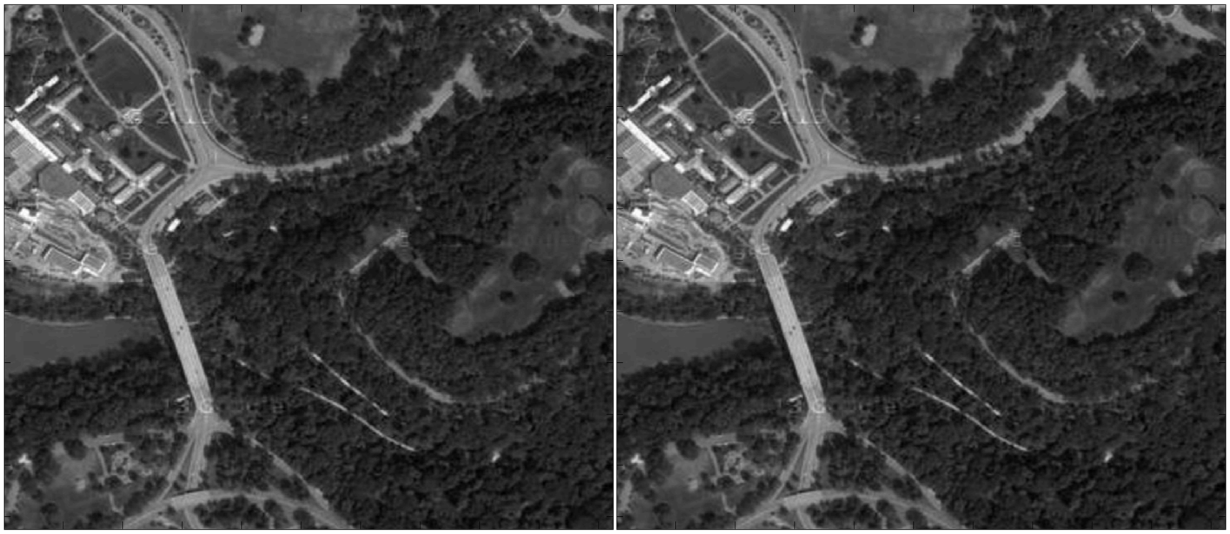
The proposed re-mapping algorithm along with bicubic interpolation using LiM topology is implemented in Verilog. Computation is done in single precision. To measure the accuracy of only the re-gridding technique, a spatial domain polar grid image is constructed from a benchmark image (1024×1024 with 32×32 tile size) and passed through our re-gridding algorithm. An example of the benchmark and the result of proposed re-gridding technique is presented in Figure 6 for which the calculated SNR is 48.4dB and PSNR is 58.8dB.

To test the accuracy of the overall proposed system, the resultant image is compared to an image reconstructed by a Matlab double precision *gold standard* reference implementation of SAR with FFT-based interpolation. As the frequency domain interpolation is done locally by our system and globally by Matlab's implementation, interpolated data are renormalized before inverse FFT to allow for a fair comparison. Examples of the reconstructed images are shown in Figure 7. Here, the SNR and PSNR of the proposed reconstruction with respect to gold standard is 19.7dB and 91.1dB. The difference between SNR and PSNR can be attributed to slight low-frequency disturbances due to non-conservation of overall energy. The effect is a small location-dependent average energy level mismatch that has a stronger impact on SNR than PSNR, but does not impact further processing of the reconstructed image.

Effect of Curvature Approximation: To explore the effect of the straight line approximation of of the optimized re-mapping algorithm, ideal re-mapping is applied on a 1024×1024 point polar grid. The resultant rectangular grid is then super-imposed on the grid found by optimized re-mapping. The result is shown in Figure 8. The peak shift is less than $1/20$ of a pixel size and, as can be seen, is very insignificant.

B. Performance and Energy Efficiency

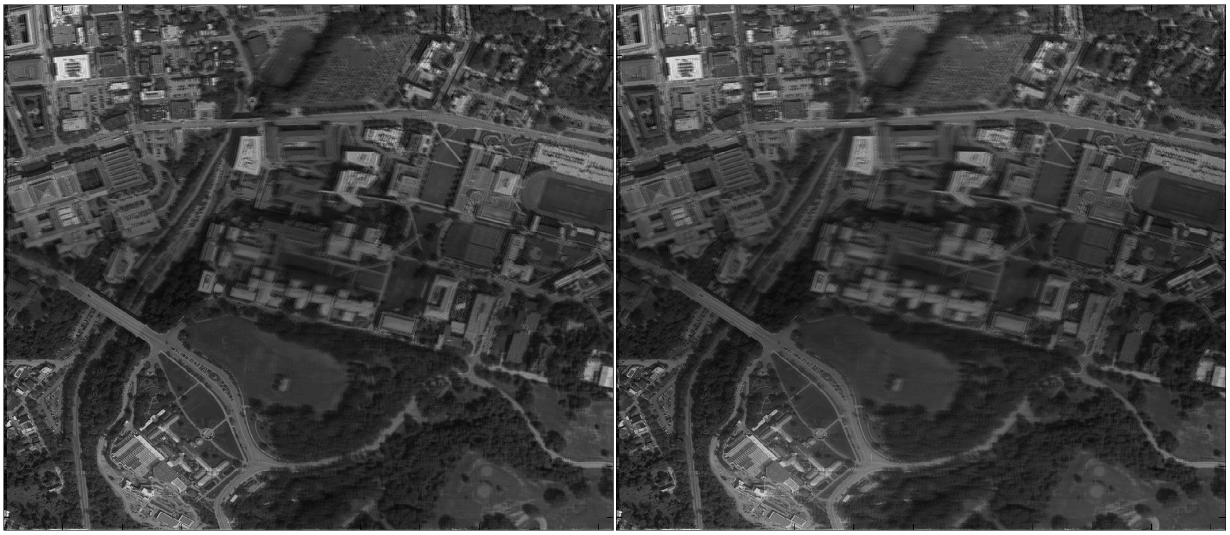
The measurement of performance and energy of SAR reconstruction on our proposed platform leads to a *design*



(a) Benchmark image

(b) RTL result of the proposed re-gridding

Fig. 6: Result of accuracy test of the proposed re-gridding method.



(a) Reconstruction by benchmark Matlab algorithm

(b) Reconstruction by proposed overall system

Fig. 7: Accuracy comparison between Matlab's golden benchmark algorithm and proposed system.

space rather than a single efficient design. Figure 9 shows the design space of an $8k \times 8k$ size SAR image reconstruction with bicubic interpolation using a 3D-stacked DRAM configuration given in Table I. Here, N_{stack}, N_{bank} and N_{TSV} are the number of DRAM dies stacked, banks per die and TSVs per bank. The overall system configuration and the parameters for the design space exploration are given in Table II. Single precision accuracy is considered for all computations.

TABLE I: 3D-stacked DRAM configuration.

Configuration $N_{\text{stack}}/N_{\text{bank}}/N_{\text{TSV}}/Row$(Kb)/Tech(nm)	tCL-tRCD-tRP-tTSV (ns)	Max BW (GB/s)
4 / 8 / 512 / 8 / 32	12.2-7.89-16.8-0.68	337.2

TABLE II: Design space exploration parameters for the system.

System configuration		Design space parameters
8Gbit, 4-layer DRAM	1-layer logic	FFT radix: 2 cpx words
8 banks/layer	512 TSVs/bank	Streaming Width: 2 → 16 cpx words
Row buffer = 1KB	Max BW = 335GB/s	Tile size: 0.125x → 2x row-buffer
		Frequency: 0.4 → 2 GHz

Finding the most suitable system configuration given the task/platform constraints establish an optimization problem. Naively choosing the highest performance or the lowest power consumption design point is not sufficient to get the most efficient system. Therefore, our design automation framework selects the best design point in terms of the power efficiency

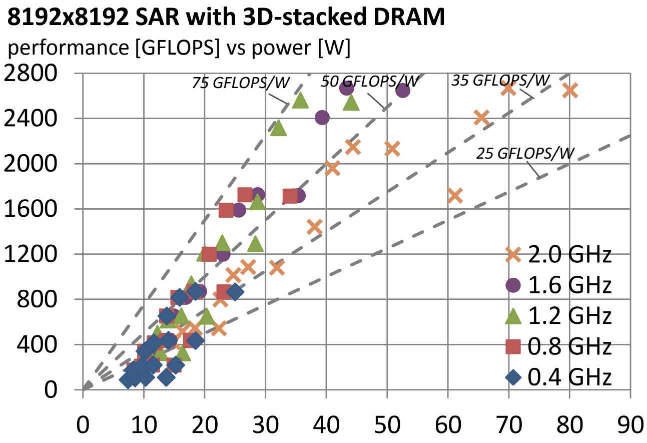


Fig. 9: Design space exploration for performance and power evaluation.

for a given task/platform. As can be seen from Figure 9, our proposed system can reach 72 GFLOPS/W, which outperforms the SAR energy efficiencies in state of the art architectures. Furthermore, to explore the effect of image size on performance and energy efficiency, in Table III the power (W), performance (GFLOPS) and power efficiency (GFLOPS/W) numbers of the selected best designs for the given configurations provided.

TABLE III: Power, performance and energy efficiency of different images sizes.

Image Size (cpx words)	Total Power (W)	Performance (GFLOPS)	Energy Efficiency (GFLOPS/W)
$2^9 \times 2^9$	28.64	1820.8	63.6
$2^{10} \times 2^{10}$	29.08	1985.4	68.3
$2^{11} \times 2^{11}$	32.16	2310.1	71.8
$2^{12} \times 2^{12}$	33.58	2446.9	72.9
$2^{13} \times 2^{13}$	32.14	2318.2	72.1

VII. Conclusion

In this paper we demonstrated a real-time SAR implementation platform finely tuned for a novel basis vector transform based re-gridding algorithm. The presented system exploits LiM topology along with 3D-stacking technology and comprehensively outperforms the SAR implementations on traditional architectures in terms of accuracy, performance and energy efficiency. The optimized algorithm reduces the data intensity while the LiM 3D-stacking topology enables high bandwidth and power efficiency. To the best of our knowledge, this is the first such co-optimized system proposed for SAR image reconstruction. Moreover, this work shows that by leveraging the state of the art IC technologies and topologies algorithm/HW co-optimization approach can break through the difficulties that many modern day data intensive applications pose.

Acknowledgment

The work was sponsored by Defense Advanced Research Projects Agency (DARPA) under agreement No. HR0011-13-2-0007. The content, views and conclusions presented in this document do not necessarily reflect the position or the policy of DARPA or the U.S. Government. No official endorsement should be inferred.

References

[1] W. G. Carrara, R. S. Goodman, and R. M. Majewski, *Spotlight Synthetic Aperture Radar: Signal Processing Algorithms*, ser. Artech House signal processing library.

[2] C. V. J. Jr., D. E. Wahl, P. H. Eichel, D. C. Ghiglia, and P. A. Thompson, *Spotlight-Mode Synthetic Aperture Radar: A Signal Processing Approach*.

[3] J. A. Rudin, "Implementation of polar format sar image formation on the ibm cell broadband engine," in *High Performance Embedded Computing (HPEC)*.

[4] D. S. McFarlin, F. Franchetti, M. Puschel, and J. M. Moura, "High performance synthetic aperture radar image formation on commodity multicore architectures," in *SPIE Proceedings Vol. 7337*, 2009.

[5] D. Wang and M. Ali, "Synthetic aperture radar on low power multi-core digital signal processor," in *IEEE Conference on High Performance Extreme Computing (HPEC)*, 2012, pp. 1–6.

[6] "OMAP applications processor," http://www.ti.com/lsds/ti/omap-applications-processors/overview.page.

[7] G. H. Loh, "3D-Stacked memory architectures for multi-core processors," in *35th International Symposium on Computer Architecture (ISCA)*, June 2008, pp. 453–464.

[8] D. H. Woo, N. H. Seong, and H.-H. Lee, "Heterogeneous die stacking of SRAM row cache and 3-D DRAM: An empirical design evaluation," in *Circuits and Systems (MWSCAS), 2011 IEEE 54th International Midwest Symposium on*, Aug 2011, pp. 1–4.

[9] M. Püschel, J. M. F. Moura, J. Johnson, D. Padua, M. Veloso, B. Singer, J. Xiong, F. Franchetti, A. Gacic, Y. Voronenko, K. Chen, R. W. Johnson, and N. Rizzolo, "SPIRAL: Code generation for DSP transforms," *Proc. of IEEE, special issue on "Program Generation, Optimization, and Adaptation"*, vol. 93, no. 2, pp. 232–275, 2005.

[10] D. Morris, K. Vaidyanathan, N. Lafferty, K. Lai, L. Liebmann, and L. Pileggi, "Design of embedded memory and logic based on pattern constructs," in *Symposium on VLSI Technology (VLSIT)*, 2011, pp. 104–105.

[11] D. Morris, V. Rovner, L. Pileggi, A. Strojwas, and K. Vaidyanathan, "Enabling application-specific integrated circuits on limited pattern constructs," in *Symposium on VLSI Technology (VLSIT)*, 2010, pp. 139–140.

[12] Q. Zhu, K. Vaidyanathan, O. Shachamy, M. Horowitzy, L. Pileggi, and F. Franchetti, "Design automation framework for application-specific logic-in-memory blocks," in *IEEE 23rd International Conference on Application-Specific Systems, Architectures and Processors (ASAP)*, July 2012, pp. 125–132.

[13] B. Akin, P. A. Milder, F. Franchetti, and J. C. Hoe, "Memory bandwidth efficient two-dimensional fast Fourier transform algorithm and implementation for large problem sizes," in *Proc. of the 20th IEEE Symp. on Field-Programmable Custom Computing Machines (FCCM)*, 2012, pp. 188–191.

[14] B. Akin, F. Franchetti, and J. C. Hoe, "FFTs with near-optimal memory access through block data layouts," in *Proc. IEEE Intl. Conf. Acoustics Speech and Signal Processing (ICASSP)*, 2014.

[15] Q. Zhu, B. Akin, H. Sumbul, F. Sadi, J. Hoe, L. Pileggi, and F. Franchetti, "A 3d-stacked logic-in-memory accelerator for application-specific data intensive computing," in *3D Systems Integration Conference (3DIC), 2013 IEEE International*, Oct 2013, pp. 1–7.

[16] B. Akin, F. Franchetti, and J. C. Hoe, "Understanding the design space of DRAM-optimized FFT hardware accelerators," in *Proc. of IEEE Int. Conference on Application-Specific Systems, Architectures and Processors (ASAP)*, 2014.

[17] "CACTI 6.5, HP labs," http://www.hpl.hp.com/research/cacti/.

[18] "McPAT 1.0, HP labs," http://www.hpl.hp.com/research/mcpat/.

[19] "DesignWare library, Synopsys," http://www.synopsys.com/dw.

[20] K. Chen, S. Li, N. Muralimanohar, J.-H. Ahn, J. Brockman, and N. Jouppi, "CACTI-3DD: Architecture-level modeling for 3D die-stacked DRAM main memory," in *Design, Automation Test in Europe (DATE)*, 2012, pp. 33–38.