A 3D-Stacked Logic-in-Memory Accelerator for Application-Specific Data Intensive Computing

Qiuling Zhu, Berkin Akin, H. Ekin Sumbul, James C. Hoe, Larry Pileggi, Franz Franchetti
Dept. of Electrical and Comp. Eng., Carnegie Mellon University, Pittsburgh, PA, USA
Email: {qiulingz,bakin,hsumbul,jhoe,pileggi,franzf}@ece.cmu.edu

***Abstract*—This paper introduces a 3D-stacked logic-in-memory (LiM) system that integrates the 3D die-stacked DRAM architecture with the application-specific LiM IC to accelerate important data-intensive computing. The system comprises a fine-grained rank-level 3D die-stacked DRAM device and extra LiM layers implementing logic-enhanced SRAM blocks that are dedicated to a particular application. Through silicon vias (TSVs) are used for vertical interconnections providing the required bandwidth to support the high performance LiM computing. We performed a comprehensive 3D DRAM design space exploration and exploit the efficient architectures to accelerate the computing that can balance the performance and power. Our experiments demonstrate orders of magnitude of performance and power efficiency improvements compared with the traditional multi-threaded software implementation on modern CPU.**

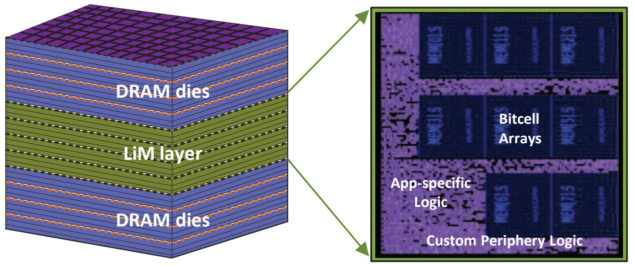
***Index Terms*—3D-Stacked DRAM; TSV; Logic-in-Memory; Sparse Matrix Matrix Multiplication; FFT**

I. Introduction

Emerging 3D die-stacked DRAM technology is one of the most promising solutions to address the well-known memory wall problem of the high-performance computing systems [17], [30], [13]. It is a technology that enables heterogeneous logic dies stacking within one DRAM package and allows the vertical communication with the through-silicon via (TSV) interconnections among the stacked chips [21], [29]. To fully utilize the stacked logic die as well as the huge internal bandwidth, many researchers have proposed to implement additional memory hierarchies, or more aggressively, multi-core processors on the logic layer [20], [28], [31], [22], [14].

In this paper, we extend the 3D DRAM technology to accelerate application-specific data intensive computing that have notoriously inefficient memory access patterns. To achieve that, we customize the logic layer to be highly specialized and particularly optimized for a specific application. More importantly, facilitated by the sub-20nm regular pattern construct based IC design [24], we enhance the functionality of the logic layer by tightly integrating the application-specific logic with the embedded SRAM blocks, resulting in 3D-stacked logic-in-memory accelerating layers (i.e., LiM-layer) (See Fig. 1).

The proposed 3D-stacked LiM system can be used to accelerate both dense and sparse data-intensive computing. For demonstration purpose, we choose dense 2D fast Fourier transform (FFT) used in synthetic aperture radar (SAR) imaging [23] and generalized sparse matrix- sparse matrix multiplication (or SpGEMM) that is a core primitive for many graph algorithms [16]. Both problems are memory-inefficient and challenge current architectures. More specifically, FFT requires multiple passes through data with low arithmetic density and strided memory access patterns, while SpGEMM suffers from poor locality and low ratio of flops to memory access due to its sparse and irregular data patterns [7]. We address these problems via a 3D-stacked DRAM which offers high bandwidth and low latency, and a stacked LiM layer that is customized to specific applications through fine-grain integration of logic and memory. We also change the algorithms to match the DRAM topology well. Eventually it allows us to exploit the application knowledge to fully utilize the TSV and the LiM layer silicon estate, and optimize the system to achieve the best performance at the lowest possible cost.



(a) 3D LiM Stack: DRAM layers and LiM layers are stacked vertically and communicate with TSVs

(b) LiM Layer: memory and logic cells are tightly integrated and customized for a particular application

Fig. 1. Illustration of the 3D-stacked Logic-in-Memory Computing System.

The co-optimization of the algorithm, architecture and LiM hardware gives rise to a huge design tradeoff space. We exploit the CACTI-3DD tool to model the proposed 3D DRAM architecture and to perform a comprehensive design space exploration [10]. Optimal 3D DRAM architecture configurations are identified to efficiently accelerate the selected applications. In this paper, we also propose a TSV saturation memory scheduling strategy to further enhance the sustained memory bandwidth. Furthermore, we optimize the application algorithms as well as their data structure and memory access patterns in order to fully leverage the underlying hardware capabilities. We also developed the LiM hardware synthesis framework for fast design evaluations. Our experimental results demonstrate orders of magnitude of performance and power efficiency improvements compared with the Intel MKL Sparse BLAS Routines implemented on modern CPU.

II. LiM Accelerated 3D-DRAM Architecture

In this section we will introduce the overall 3D-stacked LiM architecture and its various design options and configurations.

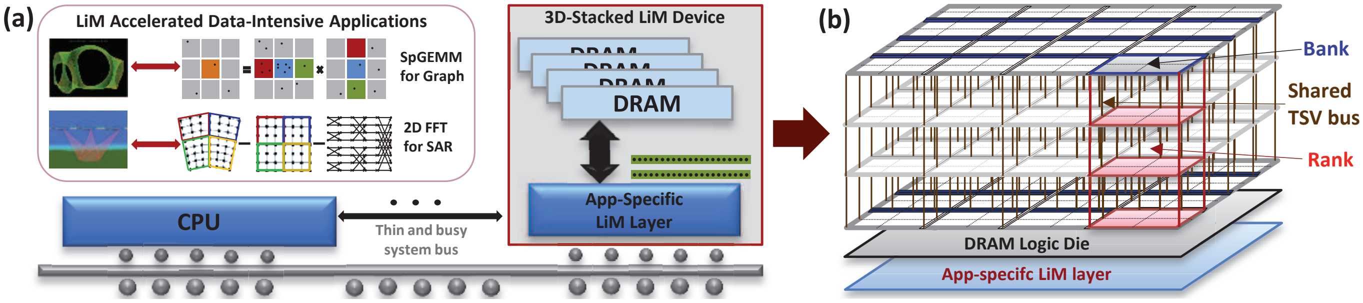


Fig. 2. Mapping Algorithm to Hardware: (a) Overall Architecture (b) Fine-grained 3D Die-stacked DRAM

A. Overall Architecture

Fig. 2 (a) illustrates the overall system architecture. The whole 3D-stacked LiM device implements a standard DRAM interface so it can be transparently used instead of an usual DDR DIMM that is accessed by CPU. The LiM layer is designed to process the data-intensive but logic-simple parts of a data-intensive problem in the most efficient way. This is possible as the dense, short and fast TSV bus is able to transfer a whole DRAM row buffer in a few clock cycles without bandwidth and I/O pin count concerns, and the LiM is designed to operate on DRAM row-buffer size data chunks and orchestrate the DRAM reads and writes. After the LiM processing, the processed data is transferred to the CPU for high-level interpretation which are less memory-bound, greatly alleviating the system bus traffic. The in-memory processing nature of this approach, along with the dedicated hardware acceleration, can deliver orders of magnitude of performance improvements and energy savings.

B. 3D-DRAM Architecture Modeling

To fully utilize the huge internal bandwidth that the TSV can offer, we exploit the fine-grained rank-level 3D die-stacked DRAM main memory architecture, which re-partitions the DRAM arrays and re-architects the DRAM dies by allowing individual memory banks to be stacked in a 3D fashion [10]. Besides, such fine-grained 3D die-stacked DRAM also has a separate logic layer to implement the complicated peripheral circuitry [20], [3], [13]. The goal is to enable bank-level parallelism, which can eliminate the I/O limitation of a conventional DRAM where all banks in a rank share a common bus.

3D-stacked DRAM design space. As shown in Fig. 2 (b), a fine-grained die-stacked DRAM has N_{stack} DRAM dies stacked vertically and each die implements N_{bank} of DDR3 DRAM bank, and each bank has its own N_{io}-bit data TSV I/O. Every N_{stack} stacked banks form a 3D vertical rank. Therefore, the overall system is composed of N_{bank} of vertical ranks. All the banks in a 3D vertical rank share a single TSV bus, which can largely relax the TSV pitch constraints [5].

We use the CACTI-3DD [10], an architecture-level 3D die-stacked off-chip DRAM modeling framework to model the 3D DRAM architecture. Besides the architectural parameters introduced above, the TSV geometry is also a critical 3D feature that needs careful modeling to evaluate its impacts on the timing, area, and power overhead [20], [29]. We use "via-first" TSVs that are fabricated before the front-end of line (FEOL) device fabrication processing [15] and the geometry parameters are from ITRS projections [6]. In our experiments we model two different TSV pitch sizes using the parameter of *TSV_projection*, that is, ITRS aggressive TSV pitch (*TSV_projection* = 0) and industrial conservative TSV pitch (*TSV_projection* = 1). Besides, the configurations of a 3D-stacked DRAM design also include the overall *DRAM_capacity* and *page_size*, and *technology_node*. Different choices of parameters have different impacts on the system. From an architect's perspective, the main attributes of interest of a 3D-stacked DRAM include the sustained memory bandwidth, the DRAM area efficiency and its active power. In order to find the optimal architectures that can be used to efficiently accelerate our application-specific LiM designs, we perform a comprehensive design space exploration of the systems with respect to area, power and timing, the detail of which will be described in Section IV-A.

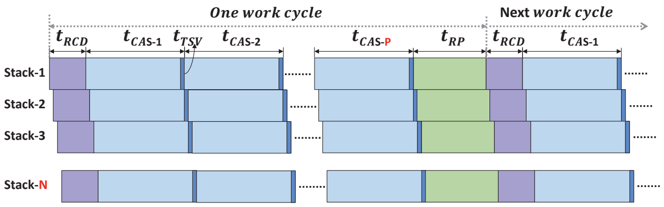


Fig. 3. Timing Diagram of the 3D-stacked DRAM Access.

Bandwidth-enhanced DRAM access. Before going into the details of the design space exploration, we first introduce a stack-staggered DRAM scheduling strategy that can significantly enhance the sustained memory bandwidth. To better illustrate this approach, in Table 1 we present several example 3D DRAM systems. The first row of the table presents a baseline architecture which has 4 stacked die counts ($N_{\text{stack}} = 4$), 8 number of banks per die ($N_{\text{bank}} = 8$) and 512 TSV-based data I/O per bank ($N_{\text{io}} = 512$). And in the next two rows, we show three improved architectures by doubling one parameter while leaving the other two parameters unchanged. For each design, we show the corresponding timing specifications and the resulting achievable bandwidth ($BW1$). We see from the table that the access latency is dominated by the row to column command delay (t_{RCD}) to get the data ready at the sense amplifier, as well as the column access strobe latency (t_{CAS}), which is the time to move data between the sense amplifier and TSV bus [26], while the t_{TSV} itself for vertical TSV data transfer is fairly small. This indicates that the vertical TSV bus stays idle for most of the time waiting for the data, limiting the bandwidth. In [28], the authors proposed to decrease t_{CAS} by further folding the subarrays of one bank to reduce the wire lengths between the sense amplifiers and the TSV bus. However, such bank-level die stacking requires breaking the structure within a bank which will further deteriorate the DRAM density and area efficiency [10]. On the other hand, we observed that the increase of the N_{stack} does not actually increase the bandwidth. The increases in N_{bank} and N_{io} contribute to the DRAM bandwidth, but at the cost of area overhead (See the AE (*Area Efficiency*) in Table 1).

As t_{RCD} and t_{CAS} are for intra-layer DRAM operations within a single DRAM layer that do not involve inter-layer TSV data transfer, they allow us to schedule the 3D DRAM access in such a way that these intra-layer operations are overlapped to reduce the TSV bus idle time, thus improve the throughput. To achieve this, we stagger the activation of successive stacked memory banks by a time step of t_{TSV} successively in a *work_cycle*, such that the t_{RCD} and t_{CAS} of N_{stack} banks in a single vertical rank are time-shared while the shared TSV bus is kept busy as much as possible (see Fig. 3). Here *work_cycle* is defined as the time to move the data from all the active DRAM bank row buffers to the LiM layer, which is determined by the ratio (P) between the DRAM row buffer size and the I/O counts per bank (N_{io}). At the end of each *work_cycle*, it takes interval t_{RP} to precharge the DRAM array and get ready for another row access (next *work_cycle*). As shown in Table 1, $BW2$ is the achieved memory bandwidth with such staggered scheduling while $BW1$ is the original 3D DRAM memory that is limited by the intra-layer operation latencies. We see that by careful scheduling the memory access we can improve the memory bandwidth by 4 to 8 times without any hardware overhead. Eventually, the increase of the N_{stack} also becomes an important factor to contribute to the system bandwidth. As we can see from Table 1, all the three improved architectures offer more than 256 GB/s memory bandwidth ($BW2$) while only consuming less than 15 Watts of power.

TABLE I
3D-DIE STACKED DRAM EXAMPLES

8 Gb memory with 8192b page size; technology node: 32nm										
N_{stack}	N_{bank}	N_{io}	t_{RCD}	t_{CAS}	t_{TSV}	t_{RP}	AE	BW1	BW2	Power
–	–	–	(ns)	(ns)	(ns)	(ns)	–	(GB/s)	(GB/s)	(Watts)
4	8	512	7.9	12.7	0.67	16.8	0.517	40.4	144.1	6.2
8	8	512	8.3	12.5	1.91	9.6	0.434	40.8	299.7	13.2
4	16	512	7.2	10.6	0.67	8.4	0.438	96.7	354.3	13.1
4	8	1024	7.9	12.1	0.67	16.8	0.445	84.4	269.2	10.5

III. 3D LIM ACCELERATED DATA INTENSIVE APPLICATIONS

To match the high bandwidth that TSV offers, it is necessary to perform equivalently high performance computing on the LiM layer. Moreover, to facilitate the proposed TSV-saturated DRAM access scheduling, it requires regular and coarse-grained row-wise memory access patterns. For demonstration purpose we use two applications to represent both sparse and dense data-intensive computing respectively, that is, SpGEMM for use in graph algorithms [16] and 2D FFT for use in SAR image reconstruction [23], [32]. Next we will introduce briefly the two problems, their algorithm choices as well as the data manipulation and hardware mapping strategies.

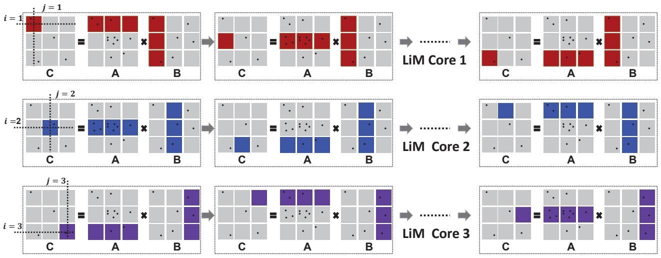


Fig. 4. Parallel SpGEMM with Tiled Data Layout.

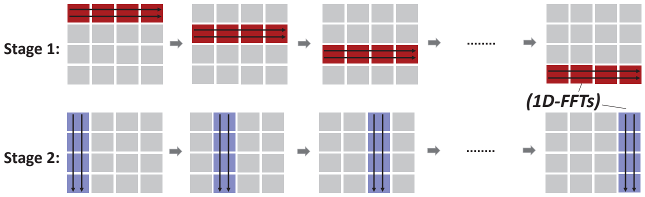


Fig. 5. 2D-FFT with Tiled Data Layout.

SpGEMM. Graphs are fundamental data structures used in many data analysis problems (e.g. WWW graph, social networks) [16]. These problems are important enough to support architecture investments to surpass the traditional computing which has reached the limit for increasing performance without an increase in power [25], [19]. It is widely accepted to exploit the duality between sparse matrix and graph to solve graph algorithms [7]. However, the development of sparse matrix algorithms poses numerous challenges due to their sparse and irregular data structures and the low ratio of flops to memory access. In this paper we focus on the acceleration of the generalized sparse matrix-matrix multiplication (SpGEMM) with the proposed 3D stacked LiM architecture. To adapt to the row-wise 3D DRAM access, we use a 2-dimensional (2D) block data decomposition of the matrices based on the SRUMMA algorithm [18]. As shown in Fig. 4, the matrix A, matrix B, and resulting matrix C are tiled into small blocks which are mapped to the DRAM rows. Each resulting block $C(i, j)$ is computed from the i^{th} block row of the matrix A and j^{th} block column of the matrix B. To enable the parallel processing, we implement multiple identical SpGEMM LiM cores and let each of them work for one column of resulting block C. As illustrated in the Fig. 4, different LiM cores start to compute the blocks on different columns simultaneously by staggering one block with each other and then continue the computation in the sequential order in a column. Such block-staggered parallel processing mechanism guarantees that each processor operates on different source blocks without conflicts.

The tiled SpGEMM algorithm can be well mapped to the 3D-stacked LiM architecture. As matrix blocks are mapped to DRAM rows which are accessed in the sequential order, for each LiM core computation, we can access the two source blocks from the carefully scheduled active row buffers via the

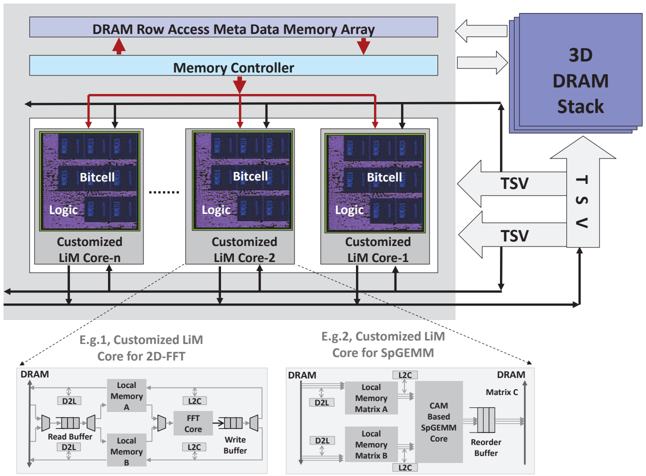


Fig. 6. 3D-Stacked LiM Functional Diagram.

TSV bus. The corresponding LiM core then operates on the two whole matrix blocks of data and computes the resulting block at the highest possible throughput without any pipeline stalls. Besides, the sequential block access order allows an easy-scheduled 3D DRAM row access, which makes the bandwidth-enhanced DRAM scheduling approach possible. It also preserves good data locality which minimizes the DRAM row miss and saves energy.

2D FFT. Next we focus on large size 2D-FFT which is a dense computation used in SAR imaging [23], [4]. Image sizes used in SAR image reconstruction are usually very large, hence requires large-size 2D-FFT computation. Large size 2D-FFT, where the whole dataset cannot be held in local SRAM, is performed as stages where only some portion of the dataset which fits local SRAM is operated at once, requiring roundtrip data transfers to and from DRAM. Furthermore, traditional algorithms for 2D-FFT have inefficient memory access patterns which puts pressure on memory bandwidth and makes the memory the key aspect of the design [4]. We address this problem via 3D-stacked DRAM which offers high bandwidth and low latency off-chip data transfer.

We utilize the algorithm and architecture proposed in [4] for large size 2D-FFT. Similar to our SpGEMM implementation, the proposed system for 2D-FFT exploits 2D-tiled data mapping in DRAM and uses an efficient algorithm targeted for such data mapping. The overall system reads and writes DRAM row-buffer size tiles during the data transfer (see Fig. 5), which minimizes the DRAM row misses. 2D-FFT computation in the LiM layer and the data transfer from the DRAM layer is overlapped by the help of double-buffering. Hence, TSV bus bandwidth is utilized effectively making the overall system an efficient fit for the 3D-stacked LiM architecture. Furthermore, since 2D-FFT system is generated by an automated design generator, we can easily match the computation throughput to the high TSV bandwidth to create balanced designs. We refer reader to [4] for the details of the 2D-FFT algorithm and architecture.

3D-stacked LiM design. Although the introduced two problems are for dense and sparse computing respectively, both applications are challenged by the well-known *memory wall* problem while implemented on modern computer architectures. In terms of the implementation, both problems are based on the block data decomposition of a 2-dimensional (2D) data array (sparse matrix, or dense image). Moreover, both of their algorithms involve regular block-wise data partition and allocations over scalable parallel processing cores (see Fig. 4 and Fig. 5). Therefore, they can both potentially be accelerated by similar 3D-stack LiM architectures.

Fig. 6 shows the corresponding functional diagram of the 3D-stacked core architecture, which is composed of a meta data memory array, a memory interface and parallel LiM cores, and interfaces with the DRAM stacks through TSV buses. The memory controller is dedicated for the communication among the meta-data memory array, the LiM cores and the 3D-DRAM. And the meta data memory array stores the information which maps the blocks of data to DRAM rows. More specifically, it provides the DRAM address information of each requested blocks as well as the global block information (i.e., the block size, the data format, and the number of nonzero elements in that block). The parallel LiM cores are carefully designed to accelerate a particular application with our previously developed logic-in-memory synthesis framework [34]. Particular knowledge from a given algorithm allows us to optimize the LiM designs to meet the required function, under the performance, area and power requirements. At the bottom of the Fig. 6, we show the structures of LiM core customized for the 2D FFT and SpGEMM respectively [4], [33]. As we can see, both LiM cores involve embedded memory arrays, on-chip buffers, arithmetic units, as well as the control models such as DRAM to Local Memory (D2L) and Local Memory to Core (L2C). However, the size and organization of these memory and logic components are designed in different ways for different applications for efficiency. Take SpGEMM for example, each LiM core is composed of two local memory arrays for the storage of the source matrix block A and B, as well as a SpGEMM core which has arithmetic units tightly integrated with SRAM and content addressable memory (CAM) arrays to compute and assemble the resulting matrix block. The CAM based SpGEMM is designed to match the specific sparse data access pattern, and it is able to process the sparse data in an extremely high throughput to match the TSV bandwidth. The design details are beyond the scope of this paper and can be found in another accompanying work [33].

IV. Evaluation and Results

In this section, we will evaluate the design space of the 3D-stacked DRAM architecture and identify the optimal design points. We will also evaluate the performance and energy efficiency improvements of the accelerated applications.

A. 3D DRAM Architecture Design Space Exploration

The modeling of the 3D-stacked DRAM is a large design tradeoff problem that involves a large number of design parameters, the choices of which have significant impacts on

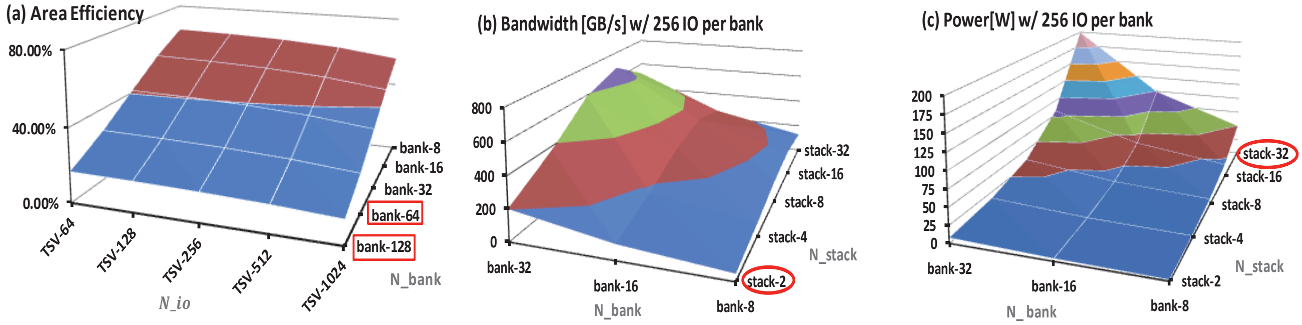


Fig. 7. 3D-DRAM Design Space Exploration for Area, Bandwidth and Power.

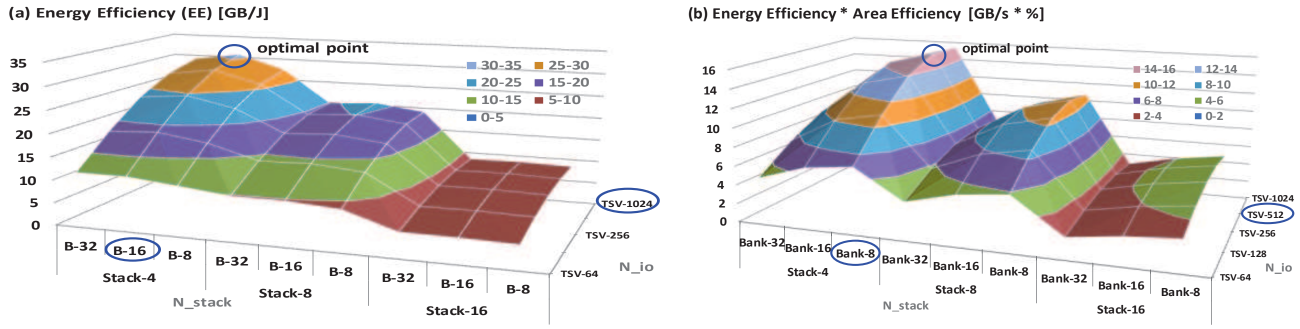


Fig. 8. 3D-DRAM Design Space Exploration for Energy Efficiency.

different design metrics. We use the CACTI-3DD [10] which explore the design space with the goal to identify the optimal design points that can balance performance, power and area.

Area efficiency (AE) is a fundamental metric for a DRAM device which is defined as the ratio of the cell array area to the total die area and it is typically in the range of 45% to 55% for a commodity DRAM [27]. In the proposed 3D DRAM, the increase of both N_{bank} and N_{io} will cause significant area overhead. Fig. 7 (a) plots the AE for sweeping N_{bank} from 8 to 128, and N_{io} from 64 to 1024. We keep the total DRAM capacity of each die fixed as 2 Gbits. We see that when N_{bank} is larger than 32, AE of all designs points are less than 40% regardless the TSV counts. That implies that the fined-grained bank partition cause too much area overhead. Therefore the designs with $N_{\text{bank}} = 64$ or $N_{\text{bank}} = 128$ are excluded and the remaining designs points have the bank size of 256 Mb ($N_{\text{bank}} = 8$), 128 Mb ($N_{\text{bank}} = 16$), and 64 Mb ($N_{\text{bank}} = 32$). We then measure the sustained memory bandwidth and the corresponding power consumption with respect to the remaining N_{bank} and N_{stack}, as shown in Fig. 7 (b) and (c) respectively. We see that the DRAM bandwidth keeps increasing when N_{stack} increases from 2 to 16, after which the bandwidth starts to shrink, indicating that the increased stack height is unable to supply higher bandwidth due to the increased latency. Fig. 7 (c) shows the escalating power consumption with the increasing N_{stack}. Due to both of the latency and power limitations, $N_{\text{stack}} = 32$ is excluded. $N_{\text{stack}} = 2$ is also excluded due to the low bandwidth supplied.

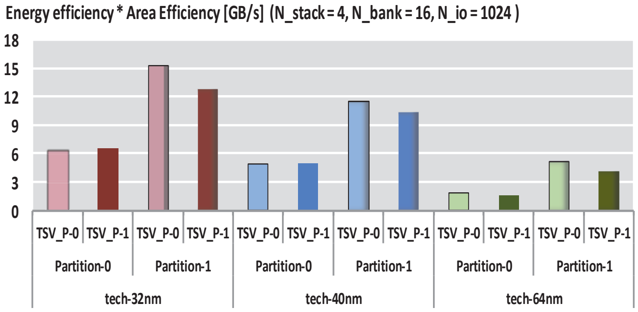
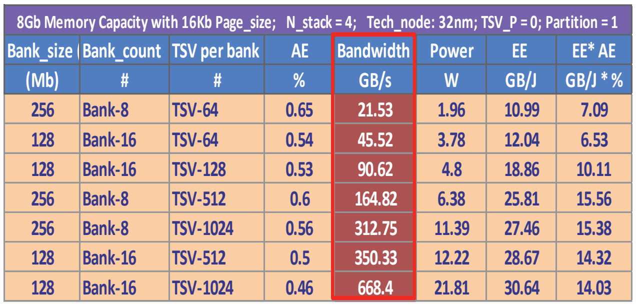


Fig. 9. More 3D-DRAM Design Space Exploration.



8Gb Memory Capacity with 16Kb Page_size; N_stack = 4; Tech_node: 32nm; TSV_P = 0; Partition = 1							
Bank_size (	Bank_count	TSV per bank	AE	Bandwidth	Power	EE	EE* AE
(Mb)	#	#	%	GB/s	W	GB/J	GB/J * %
256	Bank-8	TSV-64	0.65	21.53	1.96	10.99	7.09
128	Bank-16	TSV-64	0.54	45.52	3.78	12.04	6.53
128	Bank-16	TSV-128	0.53	90.62	4.8	18.86	10.11
256	Bank-8	TSV-512	0.6	164.82	6.38	25.81	15.56
256	Bank-8	TSV-1024	0.56	312.75	11.39	27.46	15.38
128	Bank-16	TSV-512	0.5	350.33	12.22	28.67	14.32
128	Bank-16	TSV-1024	0.46	668.4	21.81	30.64	14.03

Fig. 10. Potential Optimal 3D DRAM Design Points.

To identify the optimal design points, it requires more comprehensive optimization criteria to quantify the design space. We use *energy efficiency* (EE = bandwidth/power) as well as the product of the *energy efficiency* and *area efficiency* (EE×AE) and plot them with respect to the narrowed down N_{stack}, N_{bank} and N_{io} in Fig. 8 (a) and (b), respectively. From the figures it is straightforward to identify the optimal design points in the remaining design space. As we highlighted in the blue circles, the optimal design points are ($N_{\text{stack}} = 4$, $N_{\text{bank}} = 16$, $N_{\text{io}} = 1024$) for an optimized EE and ($N_{\text{stack}} = 4$, $N_{\text{bank}} = 8$, $N_{\text{io}} = 512$) for an optimized EE×AE.

We continue to evaluate the impacts of other design options. Besides *technology_node* and *TSV_projection* that we

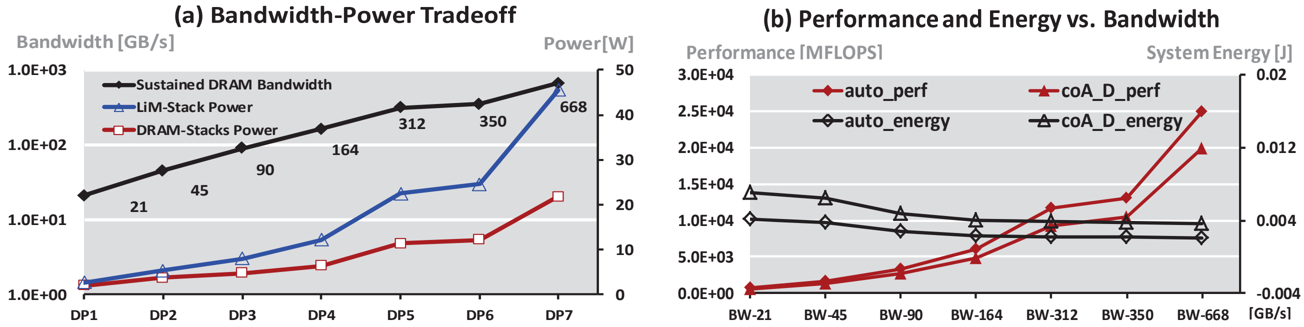


Fig. 11. Impact of 3D DRAM Bandwidth on SpGEMM Performance, Power and Energy.

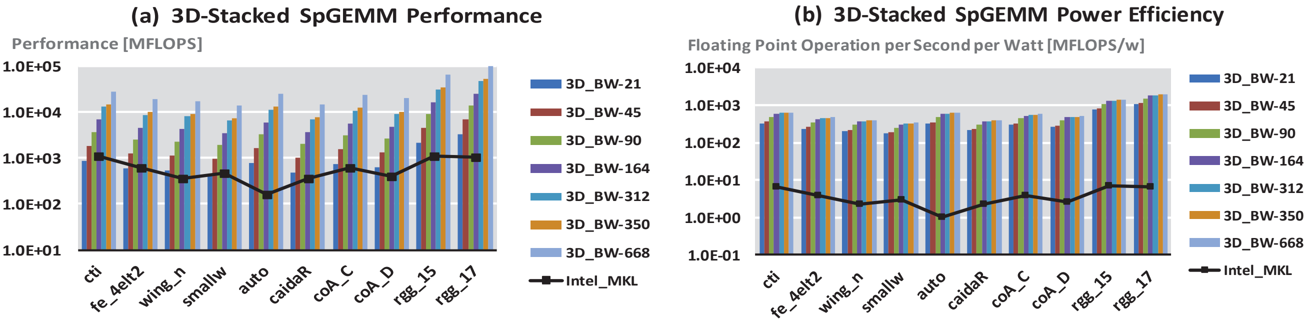


Fig. 12. 3D-Stacked SpGEMM Performance and Power Efficiency Evaluation.

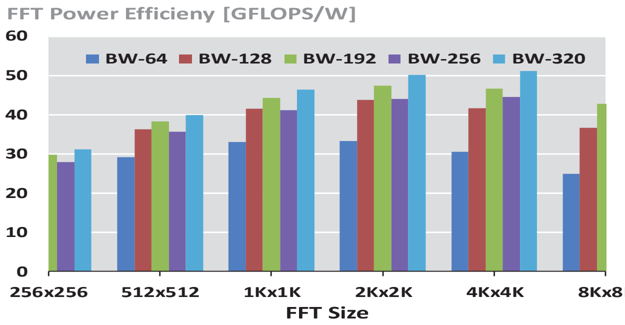


Fig. 13. Power efficiency for 2D-FFT (single precision).

introduced in Section II-B, we use the parameter *Partition* to specify two different DRAM partition strategies. Besides the introduced fine-grained rank-level die-stacking (*Partition* = 1), we also simulated the coarse-grained rank-level die-stacking DRAM (*Partition* = 0) where TSVs is only used as inter-rank interconnects while the rank still consists of the planar dies as in a 2D design [10]. From Fig. 9, we clearly see the advantages of the advanced technology node, the fine-grained partition strategies and the smaller TSV pitch. Based on all the exploration results, in Fig. 10 we summarize seven potential optimal 3D DRAM architectures that we will use to accelerate the applications and they are ordered with the bandwidth values. As we can see, all the selected designs have more than 45% area efficiencies and more than 10 GB/J energy efficiency, and the offered bandwidth varied from 21 GB/s to 668 GB/s at 2 to 22 Watts of power consumption.

B. Application Evaluation

We build the cycle-accurate HDL model of the blocked SpGEMM LiM and simulate it on the selected optimal 3D DRAM architecture model. The LiM layer power numbers are based on HDL synthesis using a commercial 32nm library. We first present the bandwidth-power tradeoff in Fig. 11 (a), and it shows that not only the DRAM power but also the LiM layer power consumption increase with the increasing memory bandwidth, as it requires more active computational resources on the LiM layer in order to consume the high throughput data. In Fig. 11 (b) we simulate SpGEMM for two benchmark matrices and plot their performance and energy consumption over the seven selected DRAM architectures. As expected, the achieved SpGEMM performance keeps increasing on architectures with higher memory bandwidths. But we know from Fig. 11 (a) that the corresponding power consumption will increase as well. Interestingly, increase in the performance overshadows the power consumption increase, as the total energy consumption, which is the product of power and latency, slightly decreases with higher memory bandwidth.

For comparison purpose, we run Intel Math Kernel Library (MKL) Sparse Basic Linear Algebra Subprograms (BLAS) Routines on Intel Xeon machines [1]. We then use the Sniper multi-core simulator to model the processor simulating the same SpGEMM and estimate the processor power [8]. We use Intel Pin tool to generate the memory trace statistics and use a modified USIMM DRAM simulator together with Micron power calculator for the power estimation of the planar DRAM system [12], [9], [2]. Fig. 12 (a) and (b) present the comparison of performance and power efficiency of the two systems for a wide variety of benchmark matrices which are collected from the University of Florida sparse matrix collection [11]. We see that the performance of the proposed SpGEMM implementation varies from 1 GFLOPS (FLoating-point Operations Per Second) to 100 GFLOPS for different memory bandwidth configurations. And for high-bandwidth implementations, it can achieve more than one order of magnitude of performance improvement as well as more than two orders of magnitude of power efficiency improvements compared with the Intel MKL Sparse BLAS Routines implemented on Intel Xeon machines.

To evaluate the impact on dense computing, we similarly simulated the performance of the 2D-FFT using a custom system performance model backed up by cycle-accurate HDL simulation. We demonstrate simulated systems in Fig. 13, where DRAM bandwidth is ranging from 64GB/s to 320GB/s and 2D-FFT size is ranging from 256×256 to $8K \times 8K$. We picked the optimal design parameters (frequency, parallelism degree) that saturate the given bandwidth and problem size. Results show that again the proposed system can achieve one to two orders of magnitude power efficiency improvements compared to optimized FPGA, GPU and CPU implementations which achieve only a few GFLOPS/W [4]. Also as we expected, the dense computing delivers much higher power efficiency than the sparse one on the same computing system.

V. Conclusion

This paper presents a TSV based 3D computing system that stacks a 3D DRAM device with high performance LiM chips to accelerate data intensive problems that are limited by the "memory wall" bottleneck of the modern processor architectures. The novelty lies in an application-specific 3D-stacked DRAM architecture which offers high bandwidth and low latency data transfer via TSV, and a stacked LiM layer that is customized to the particular problem through a fine-grain integration of logic and SRAM. In addition, we revised the application algorithms to match the underlying hardware and developed the necessary modeling and design framework tools for fast design evaluations. The resulting system is a transparent, energy efficient device for accelerating notoriously memory-bound problems. This paper demonstrates that recent cutting-edge IC design advances create opportunities to build an extremely energy, power and performance-efficient computing platform to accelerate data intensive computing.

ACKNOWLEDGEMENT

This work was supported in part by the Intelligence Advanced Research Program Agency and Space and Naval Warfare Systems Center Pacific under Contract No. N66001-12-C-2008, Program Manager Dennis Polla. The work was also 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] Intel. math kernel library. [online] http://developer.intel.com/software/products/mkl/.

[2] Micron system power calculator. [online] http://www.micron.com/products/support/power-calc.

[3] Tezzaron semiconductors. octopus 8-port dram for die-stack applications. data sheet. [online] http://www.tezzaron.com/memory/datasheets/.

[4] B. Akin, P. Milder, F. Franchetti, and J. Hoe. Memory bandwidth efficient two-dimensional fast fourier transform algorithm and implementation for large problem sizes. In *FCCM*, pages 188–191.

[5] R. Anigundi, H. Sun, J. Lu, K. Rose, and T. Zhang. Architecture design exploration of three-dimensional (3d) integrated dram. *ISQED*, pages 86 – 90, 2009.

[6] S. I. Association. International technology roadmap for semiconductors.

[7] A. Buluc and J. Gilbert. Challenges and advances in parallel sparse matrix-matrix multiplication. *ICPP*, pages 503 – 510, 2008.

[8] T. Carlson, W. Heirman, and L. Eeckhout. Sniper: Exploring the level of abstraction for scalable and accurate parallel multi-core simulations. *SC*, pages 1 – 12, 2011.

[9] N. Chatterjee. Usimm: the utah simulated memory module. *University of Utah and Intel Corp*, 2012.

[10] K. Chen, S. Li, , and et.al. Cacti-3dd: Architecture-level modeling for 3d die-stacked dram main memory. *DATE*, pages 33 – 38, 2012.

[11] T. A. Davis and Y. Hu. The university of florida sparse matrix collection. *ACM Transactions on Mathematical Software*, 38(1):1 – 25, 2011. [online] http://www.cise.ufl.edu/research/sparse/matrices.

[12] C. Fischer and C. McCurdy. Using pin as a memory reference generator for multiprocessor simulation. *SIGARCH Computer Architecture News*, pages 39–44, 2005.

[13] J. Jeddeloh and B. Keeth. Hybrid memory cube new dram architecture increases density and performance. *VLSIT*, pages 87 – 88, 2012.

[14] U. Kang, H. Chung, and et.al. 8 gb 3-d ddr3 dram using through-silicon-via technology. *JSSC*, pages 111 – 119, 2009.

[15] G. Katti, M. Stucchi, K. de Meyer, and W. Dehaene. Electrical modeling and characterization of through silicon via for three-dimensional ics. *IEEE Trans. on Electron Devices*, page 256262, 2010.

[16] J. Kepner and J. Gilbert. *Graph algorithms in the language of linear algebra*. Society for Industrial and Applied Mathematics, 2011.

[17] J. Kim, C. Oh, H. Lee, D. Lee, and et.al. A 1.2v 12.8gb/s 2gb mobile wide-i/o dram with 4128 i/os using tsv-based stacking. *ISSCC*, 2011.

[18] M. Krishnan and J. Nieplocha. Srumma: a matrix multiplication algorithm suitable for clusters and scalable shared memory systems. *Proceedings of Parallel and Distributed Processing Symposium*, 2004.

[19] C. Lin, Z. Zhang, and et.al. Design space exploration for sparse matrix-matrix multiplication on fpgas. *FPT*, pages 369–372, 2010.

[20] G. Loh. 3d-stacked memory architectures for multi-core processors. *ISCA*, pages 453 – 464, 2008.

[21] G. Loi, B. Agrawal, and et.al. A thermally-aware performance analysis of vertically integrated (3-d) processor-memory hierarchy. *DAC*, pages 991 – 996, 2006.

[22] A. Maashri, G. Sun, X. Dong, V. Narayanan, and Y. Xie. 3d gpu architecture using cache stacking: Performance, cost, power and thermal analysis. *ICCD*, pages 254 – 259, 2009.

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

[24] D. Morris, K. Vaidyanathan, and et.al. Design of embedded memory and logic based on pattern constructs,. *Symp. VLSI Technology*, June 2011.

[25] J. Stevenson, A. Firoozshahian, and et.al. Sparse matrix-vector multiply on the hicamp architecture. *ICS*, pages 195–204, 2012.

[26] D. T. Wang. Modern dram memory systems: Performance analysis and scheduling algorithm,. *PhD Thesis,University of Maryland*, 2005.

[27] C. Weis, N. Wehn, L. Igor, and L. Benini. Design space exploration for 3d-stacked drams. *DATE*, pages 1 – 6, 2011.

[28] D. Woo, N. Seong, and H. Lee. Heterogeneous die stacking of sram row cache and 3d dram: An empirical design evaluation. *MWSCAS*, pages 1 – 4, 2011.

[29] D. Woo, N. Seong, D. Lewis, and H. Lee. An optimized 3d stacked memory architecture by exploiting excessive, high density tsv bandwidth. *HPCA*, pages 1 – 12, 2010.

[30] Q. Wu, R. Ken, J. Lu, and T. Zhang. Impacts of though-dram vias in 3d processor-dram integrated systems. *3DIC*, pages 1 – 6, 2009.

[31] T. Zhang, K. Wang, and et.al. A 3d soc design for h.264 application with on-chip dram stacking. *3DIC*, pages 1 – 6, 2010.

[32] Q. Zhu, C. R. Bergery, E. L. Turnerz, L. Pileggi, and F. Franchetti. Polar format synthetic aperture radar in energy efficient application-specific logic-in-memory. *ICASSP*, pages 1557 – 1560, 2012.

[33] Q. Zhu, T. Graf, H. E. Sumbul, L. Pileggi, and F. Franchetti. Accelerating sparse matrix-matrix multiplication with 3d-stacked logic-in-memory hardware. *HPEC*, 2013.

[34] Q. Zhu, K. Vaidyanathan, L. Pileggi, and F. Franchetti. Design automation framework for application-specific logic-in-memory blocks. *ASAP*, pages 125 – 132, 2012.