Performance Modeling of Matrix Multiplication on 3D Memory Integrated FPGA

Shreyas G. Singapura, Anand Panangadan and Viktor K. Prasanna
Ming Hsieh Department of Electrical Engineering
University of Southern California,
Los Angeles, CA 90089, USA
singapur@usc.edu, anandvp@usc.edu, prasanna@usc.edu

***Abstract*—Recent advances in three dimensional integrated circuits have enabled vertical stacks of memory to be integrated with an FPGA layer. Such architectures enable high bandwidth and low latency access to memory which is beneficial for memory-intensive applications. We build a performance model of a representative 3D Memory Integrated FPGA architecture for matrix multiplication. We derive the peak performance of the algorithm on this model in terms of throughput and energy efficiency. We evaluate the effect of different architecture parameters on performance and identify the critical bottlenecks. The parameters include the configuration of memory layers, vaults, and Through Silicon Vias (TSVs). Our analysis indicates that memory is one of the major consumers of energy on such an architecture. We model memory activation scheduling on vaults for this application and show that it improves energy efficiency by $1.83\times$ while maintaining a throughput of 200 GOPS/s. The 3D Memory Integrated FPGA model achieves a peak performance of 93 GOPS/J for a matrix of size $16K \times 16K$. We also compare the peak performance of a 2D architecture with that of the 3D architecture and observe a marginal improvement in both throughput and energy efficiency. Our analysis indicates that the bottleneck is the FPGA which dominates the total computation time and energy consumption. In addition to matrix multiplication, which requires $O(m^3)$ amount of computation work to be done, we also analyzed the class of applications which require $O(m^2)$ work. In particular, for matrix transposition we found out that the improvement is of the order $3\times$ for energy consumption and $7\times$ in runtime. This indicates that the computation cost of the application must match the memory access time in order to exploit the large bandwidth of 3D memory.**

I. Introduction

While the concept of 3D Integrated Circuits (3D ICs) have been studied extensively, recent advances in technology are resulting in useable off-the-shelf products [1]. 3D ICs are built by stacking multiple dies and connecting them through vertical interconnects such as Through Silicon Vias (TSVs). In particular, logic and memory layers can be vertically stacked to create 3D Memory Integrated Architectures (3D MIA). These have the potential of addressing the memory wall issue by providing enormous bandwidth and enabling Processing Near Memory applications.

3D MIAs enable much more complex memory access patterns. For instance, 3D memory enables parallelism at a fine-grained level with row accesses and at a coarse-grained level with selective activation of banks and vaults. The throughput of an application on a 3D MIA is determined by how its data layout and access patterns exploit the structure and organization of the architecture. The energy consumption of the architecture is also dependent on the mode of operation of individual banks and vaults. These can be controlled individually in a 3D memory and is affected by application data layout and access patterns. Thus, application performance metrics of throughput and energy efficiency are inter-related in a 3D MIA. Careful modeling of these characteristics can enable an application to fully exploit the capabilities of a 3D MIA and trade-off between different performance metrics.

We develop performance models for matrix multiplication on 3D MIA to explore the design space of the architecture with the goal of optimizing performance metrics of throughput and energy efficiency. In this work, we consider the case of FPGA as the logic layer in a 3D MIA to form a 3D Memory Integrated FPGA (denoted as 3D MI-FPGA). 3D MI-FPGA consists of three components: memory, interconnect layer, and FPGA. We model the various factors affecting throughput such as data access time, data transfer time, and computation time. Similarly, we model the overall energy consumption for each component of the architecture. We build upon our earlier work of parametrization of an abstract 3D MI-FPGA architecture [2] to develop the performance model for matrix multiplication. We adapt the memory activation scheduling technique to the case of 3D memory. Specifically, we extend the optimization to include the number of vaults as an additional parameter. We evaluate the effect of architecture parameters on throughput and energy efficiency. These include parameters of interconnect layer and memory layer in the 3D architecture. In doing so, we identify critical performance bottlenecks. The main contributions of this paper are as follows:

- Develop parameterized performance model for matrix multiplication on 3D MI-FPGA.
- Extending Memory Activation Scheduling to 3D memory by incorporating the number of vaults to increase energy efficiency.
- Evaluation of the matrix multiplication algorithm on a 3D MI-FPGA and derivation of the peak performance in terms of throughput and energy efficiency.
- Analysis of the effects of different architecture parameters

This material is based in part upon work supported by the National Science Foundation under Grant Number ACI-1339756

on the target metrics and identifying bottlenecks.
- Performance comparison of matrix multiplication on 3D MI-FPGA with a 2D architecture.

The rest of the paper is organized as follows. Related work is presented in Section II. In Section III, we briefly describe the algorithm used for matrix multiplication. We also define metrics of evaluation in this section. The model of 3D MI-FPGA used for analysis in the paper is explained in Section IV. We develop the performance model for matrix multiplication on 3D MI-FPGA in Section V. Section VII presents the results of our analysis from Section V and we conclude in Section IX.

II. Related Work

One focus area of research has been on exploiting the shorter interconnects and delay aspects of 3D ICs. One such approach is developing specific functional blocks for 3D ICs composed of a vertical stack of logic layers. [3]–[7] explores the porting of functional units such as instruction schedulers, register files, FFT and TCAM to 3D ICs. In [8], [9], an array of image sensors are designed for 3D ICs. These works look into the mapping of specific blocks onto logic-on-logic 3D ICs and not 3D MIAs. Also, energy efficiency is not the key focus in these works.

In the field of 3D MIA and 3D memories, the focus has been to exploit the bandwidth and smaller delay compared to 2D architectures. There have been research focusing on developing optimizations for this metric. In [10], the authors consider latency as the performance metric and compare various layouts in the 3D MIA design space for a Network-on-Chip application. [11] considers power consumption in a 3D MIA with a general purpose processor as the logic layer; the architecture is also optimized for a Network-on-Chip application. In [12], a 3D MIA with one layer each of logic and memory is built to demonstrate the memory bandwidth and power benefits. The results are presented for several applications including matrix multiplication, median filtering, and search algorithms. In their work, the logic layer is a multi-core general purpose processor.

[13] employs a application specific logic layer for matrix multiplication and focuses on energy efficiency and bandwidth as the metrics. They use logic sandwiched between memory as an accelerator in addition to a CPU. In [14], 3D MI-FPGA is used to demonstrate the improvement in power efficiency as compared to a 2D FPGA; FFT was used as the application.

Power dissipation or the energy consumption in 3D ICs has been studied extensively but the focus has been on the 3D interconnects themselves and not on the entire architecture. In the case of 3D MIA, the focus of our work, the algorithm and the architecture also play an important role in the total energy consumption. The algorithm and architecture together define the access patterns to the memory and these have a significant impact on energy consumption. In this paper, we target the performance of the complete architecture by modeling the effect of each component of a 3D MI-FPGA on the metrics of throughput and energy efficiency.

III. Algorithm and Metrics

A. Algorithm

The matrix multiplication in this paper is performed using block multiplication. We adopt the architecture used in [15]. We focus on large scale matrix multiplication, meaning matrices do not fit on the FPGA. The matrices (A, B, C) are of size $N \times N$ and are divided into blocks of size $m \times m$ and each block of output matrix is calculated using multiplication of N/m blocks. An example is shown in Figure 1 with the corresponding equations.

A1	A2	B1	B2	C1	C2
A3	A4	B3	B4	C3	C4

Fig. 1: Matrices A, B and C divided into blocks

$$C1 = A1 \times B1 + A2 \times B3, \quad C2 = A1 \times B2 + A2 \times B4$$
$$C3 = A3 \times B1 + A4 \times B3, \quad C4 = A3 \times B2 + A4 \times B4$$

We use m processing elements (PE) and each PE consists of a multiplier and adder, local registers and a buffer of size $2m$. Each $m \times m$ matrix is stored in m block RAMs and in each clock cycle, 1 element each of A and B is available. With all PEs operating in parallel, this results in total computation time of $m^2 + 2m$ cycles for $m \times m$ block multiplication. The input matrix A is fed into the PEs on FPGA in row major data layout in m cycles and the B matrix is fed into the PEs in column major data layout. The intermediate results are stored in the buffer of size m in each PE. After the completion of matrix multiplication, m elements from each PE form matrix C and is transferred into the 3D memory. The FPGA architecture contains connections localized between PEs and within a PE. For block matrix multiplication, as the size of the block (m) increases, the total computation time decreases. Therefore, m is chosen to be as large as possible while ensuring that the entire design fits on the FPGA.

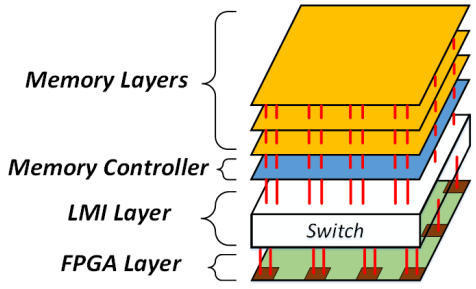
B. Metrics

We evaluate matrix multiplication on 3D MI-FPGA with Throughput and Energy Efficiency as the target metrics.

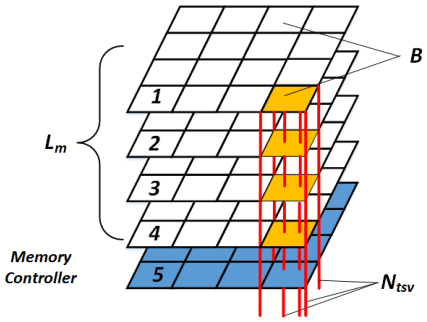
- **Throughput** (**GOPS/s**) is calculated as the total number of useful operations carried out per second. For multiplication of two matrices of size $N \times N$, N^3 additions and multiplications are required. The total time for matrix multiplication on 3D MI-FPGA is an aggregate of the data access time from memory, data transfer time across the interconnects, and computation time on FPGA.
- **Energy Efficiency** is defined as the total number of useful operations per unit of energy consumed and is measured in **GOPS/J**. The energy consumed by memory, FPGA, and the interconnect layer is considered to be the energy consumption of the entire architecture.

IV. 3D MI-FPGA: Model

In order to develop a performance model, the critical parameters affecting performance need to be identified. In [2], we parameterized 3D Memory Integrated Architectures with FPGA as the logic layer and identified the principal parameters for design space exploration. We extend the analysis from our previous work to specify the components and parameters of the architecture used for performance modeling of matrix multiplication.



(a) Components of 3D MI-FPGA



(b) Parameters of Memory Layer

Fig. 2: Architecture of 3D MI-FPGA

The 3D MI-FPGA architecture is illustrated in Figure 2(a). It consists of memory and logic layers stacked vertically interacting through an interconnect layer called Logic-Memory-Interconnect (LMI) layer. There are multiple *layers* of memory (L_m) with each layer divided into *banks*. Group of banks across layers one above the other share a set of interconnects and form a *vault*. There are V vaults in total with each vault having a dedicated memory controller. To keep the analysis simple we limit the number of banks per vault in one layer to one. The banks of a vault contribute to the memory bandwidth of that specific vault denoted as BW_{vault}. The 3D memory architecture and its parameters are depicted in Figure 2(b). The memory controller for each vault of the memory has a set of TSVs (N_{tsv}) to the LMI layer through which it interacts with the logic layer. The LMI layer is modeled as a crossbar switch sandwiched between the memory and logic layers with connections to each of these layers through the TSVs. FPGA is composed of reconfigurable logic, DSP blocks, on-chip memory (Block RAM and Distributed RAM) and memory controllers. Similar to the memory controller in the memory layer, the memory controllers of the FPGA connect to the LMI layer with a set of TSVs. The number of FPGA layers is restricted to one. Powerful FPGAs typically consist of 1 million logic cells, 3K DSP slices, 64 Mb Block RAM and 13 Mb Dist RAM. We model our FPGA layer to be similar in terms of resources. We keep the crossbar switch in the LMI layer symmetric, i.e., the number of inputs is equal to the number of outputs. Therefore, there are equal number of memory controllers on the memory and FPGA layer. Also, the number of TSVs per memory controller is the same in both the memory and FPGA components.

V. Performance Modeling

For each component of 3D MI-FPGA, we describe the parameters important for performance modeling. We then map matrix multiplication onto the model of the 3D MI-FPGA and derive the relevant equations necessary to carry out performance analysis in terms of throughput and energy efficiency.

A. Throughput Modeling

3D memory is used to store the input and output matrices of matrix multiplication. We model the memory to be operating at peak bandwidth which determines the data access time t_{access}. This is the time for the data to be available at the memory controller layer. To transfer a block with m^2 elements of 16 bit precision at a bandwidth of BW would require $\frac{16m^2}{BW}$ time. The total bandwidth of the 3D memory is the sum of the bandwidths of vaults which contain the data being accessed. These vaults, among which the data is distributed, are called Occupied vaults (V_{occ}). Since the vaults can operate independently in a 3D memory, the total peak bandwidth of the memory is the product of bandwidth of one vault (BW_{vault}) and the number of occupied vaults. The t_{access} can also be interpreted as the time required to transfer 2 blocks of m elements with 16 bit precision. Therefore word rate is $\frac{2\times16m^2}{t_{access}}$. To simplify the analysis, we model the latency of the LMI layer to be equal to the latency of the TSVs (t_{tsv}) and the crossbar switch is ignored in this analysis. N_{tsv} bits can be transferred in t_{tsv} time from one vault. Similarly, $t_{transfer}$, the data transfer time across the LMI layer can be calculated for $16m^2$ bits with V_{occ} vaults. The FPGA layer is modeled to consist of functional units which perform multiplication and accumulation operations, and on-chip memory in the form of block RAMs which store the intermediate results. The computation time on an FPGA (t_{fpga}) for $m \times m$ multiplication using the algorithm in Section III is $m^2 + 2m$ clock cycles with f_{FPGA} as the frequency of operation of FPGA. Using the above parameters, the overall computation time, $t_{compute}$, for a $m\times m$ multiplication of 16-bit precision is given in Equation 1. A block is written to the 3D memory while reading the next

block.

$$
\begin{aligned}
t_{compute} &= t_{access} + t_{transfer} + t_{fpga} \\
t_{access} &= \frac{2 \times 16m^2}{(BW_{vault})(V_{occ})} \\
t_{transfer} &= \frac{2 \times 16m^2}{(N_{tsv})(V_{occ})} t_{tsv} \\
t_{fpga} &= \frac{m^2 + 2m}{f_{fpga}}
\end{aligned} \tag{1}
$$

B. Energy Modeling

We model the power dissipation in memory to be uniformly distributed across all the vaults and banks. For example, if a memory module with V vaults and B banks dissipates P_{mem} of power, the power dissipated by a single vault is P_{mem}/V and power dissipated per bank is P_{mem}/B. The fine grained parallelism in 3D memory enables each bank or vault to be independently turned on or switched off. This gives us additional control over energy consumption of memory. The memory consumes energy until data has been read/written by the FPGA. This time is equal to $t_{access} + t_{transfer}$. The energy consumption of the FPGA is modeled as the sum of the energy consumption of BRAMs and FPUS. The energy consumption of BRAMs (FPUs) is the product of its power dissipation, P_{bram} (P_{fpu}), and the computation time on FPGA (t_{fpga}). The energy consumption of the LMI layer is modeled as the product of power dissipation by TSVs (P_{tsv}) and the transfer time across TSVs $t_{transfer}$. The TSVs are modeled to consume energy only when data is transferred along them. The total energy consumption of 3D MI-FPGA is then modeled as the sum total of energy consumed by the memory, TSVs, and FPGA. These relations are given in Equation 2.

$$
\begin{aligned}
Total\ energy &= (Memory + TSV + FPGA)\ energy \\
Memory\ energy &= (P_{mem})(t_{access} + t_{transfer}) \\
TSV\ energy &= (P_{tsv})(t_{transfer}) \\
FPGA\ energy &= (P_{bram} + P_{fpu})(t_{fpga})
\end{aligned} \tag{2}
$$

VI. Memory Activation Scheduling

The major consumers of energy in a 3D MI-FPGA are the memory and FPGA. With respect to the energy consumption or power dissipation, the memory can be in active standby (ACT) mode or precharge power down (PRE) mode. Only when the memory is in ACT mode, reads and writes are possible and it dissipates the highest amount of power. When the memory is in PRE mode, the data in the memory cannot be accessed and it dissipates the least amount of power. The process of maintaining the memory in ACT mode only during read/write and keeping it in PRE mode at other times is called as **Memory Activation Scheduling** [15]. We can reduce the energy consumption of the memory by using memory activation scheduling and thereby, improving the energy efficiency. When memory activation scheduling is enabled, the memory is put in PRE mode for the duration of the processing on FPGA, t_{fpga}, and the memory is in ACT mode for $t_{access} + t_{transfer}$.

We refine memory activation scheduling to include number of vaults, a parameter specific to 3D memory. In the case of 3D memory, energy consumption can be further reduced by keeping only the vaults containing the data being accessed (V_{occ}) in ACT mode whereas other vaults are in PRE mode.

VII. Results

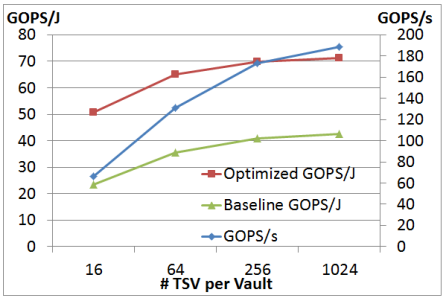
We now summarize the results of our analysis of varying architecture parameters on the performance metrics. In each such case, we evaluate the baseline implementation (without memory activation scheduling) and compare its performance with the optimized implementation (with memory activation scheduling). First, we evaluate the effect of parameters on a matrix of size $16K \times 16K$ with a block size $m = 512$ and fixed point precision (16 bits). We later summarize our results for varying matrix sizes. Our analysis can be easily extended to different precision widths as well. We set the following values for the parameters: N_{tsv} = 16, t_{tsv} = 5 ns, P_{tsv} = 20 mW, V = 4, and L_m = 3. We then vary each of these parameters keeping other parameters constant. This helps in isolating the effect of varying that particular parameter. Other parameters such as bandwidth of a vault [1], power dissipated by FPGA resources [16], and power dissipated by a given size of memory [17] are estimated based on the current state of the technology. The latency and power dissipation along a single connection across the TSVs is set according to the current 3D technology [18].

A. TSVs

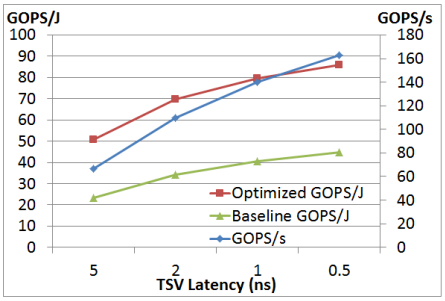
To determine the impact of TSVs on performance, we vary the TSV parameters, such as the number of TSVs per vault, latency of TSVs, and the power dissipation of TSVs.

1) Number of TSVs per Vault (N_{tsv}): We assume that one block each of the matrices A and B can fit into one vault and just one vault is accessed at a time. The banks in this vault are responsible for providing the data bandwidth. The bandwidth of a vault is a constant and the variation in number of TSVs per vault affects just the transfer time across the interconnect layer. Increasing the number of TSVs per vault reduces the transfer time from the memory controller layer to the FPGA layer. Hence, higher the number of TSVs per vault, lower the transfer time and therefore, higher the throughput. As illustrated in Figure 3(a), the throughput increases from 65 GOPS/s for $N_{tsv} = 16$ to 190 GOPS/s for $N_{tsv} = 1024$.

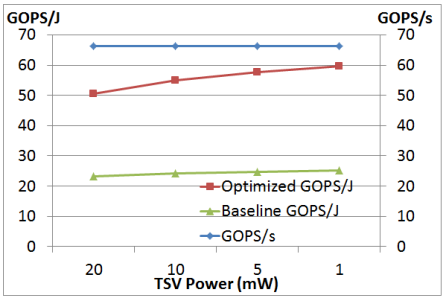
In the case of the baseline implementation, all the vaults are dissipating power even though just one vault contains the data being accessed. The memory consumes energy even while the FPGA is performing computations. Hence, the energy consumption of the memory is large and of the same order as the energy consumed by the FPGA. For matrix multiplication of a given matrix size, the amount of data transferred from the memory layers to the FPGA is constant and independent of the number of TSVs. Therefore, the sum of the active times of all the TSVs remains unchanged. Hence, the amount of energy consumed by the TSVs does not vary with the number of TSVs per vault. Even though the total power dissipated by TSVs is large, they are active for a very short period of time. Therefore, the energy consumption of TSVs is much smaller



(a) Number of TSVs per Vault



(b) Latency of TSVs



(c) Power Dissipation of TSVs

Fig. 3: Effect of varying parameters of TSVs

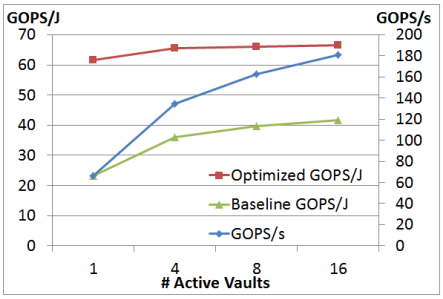
in comparison to memory and FPGA. Figure 3(a) shows the significance of energy consumption by memory which results in the low peak performance of 42 GOPS/J for the baseline implementation.

In terms of throughput, both the baseline and optimized implementations achieve the same performance as the total processing time remains the same. With respect to energy consumption in the optimized implementation, just the vault with the relevant data is in ACT mode whereas the other vaults are in PRE mode. Moreover, while the FPGA is processing data, the entire memory is in PRE mode. This results in significant energy savings with $2.17\times$ increase in energy efficiency. As seen from Figure 3(a), an increase in the number of TSVs per vault has a positive effect on energy efficiency with the performance reaching a peak of 70 GOPS/J. However, as the number of TSVs per vault reaches 1024, the improvement in energy efficiency slows down. This is because it is largely determined by the FPGA which dominates the energy consumption in this case.

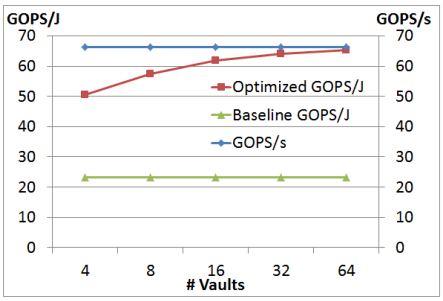
TSVs come in different sizes and vary in latency and power dissipation. Next, we vary the latency of TSVs and the power dissipation per TSV.

2) Latency of TSVs (t_{tsv}): The effect of latency of TSVs can affect the time to transfer the data between memory and FPGA and it also indirectly affects the time for which the memory is in ACT mode. The latency of TSVs influences energy consumption of TSVs as they determine the active time of TSVs. Therefore, the effect of this parameters can be significant on both throughput and energy efficiency. We vary the latency of data transfer across a single TSV from 5 ns to 0.5 ns. The latency of 5 ns for TSVs gives a throughput of just 67 GOPS/s as seen in Figure 3(b). But the decrease in latency of TSVs causes a significant improvement in throughput with the peak performance of 170 GOPS/s for t_{tsv} of 0.5 ns. The energy consumption of memory is also significant due to the low transfer rate across the interconnect layer resulting in a low energy efficiency of 24 GOPS/J (Figure 3(b)). In the case of the optimized implementation, the throughput remains the same as that of baseline implementation. The effect on energy efficiency follows the pattern of the baseline implementation. With 5 ns latency, the energy consumption of TSVs is of the same order as that of the memory and hence the energy efficiency is reduced to a low value of 51 GOPS/J. At a latency of 0.5 ns, the energy efficiency of the optimized implementation reaches a peak of 86 GOPS/J. The lower the transfer time across the LMI layer, the higher the length of time the memory is in PRE mode and hence the energy efficiency increases. As the latency decreases, the impact on performance also decreases and computation time and energy consumption by the FPGA becomes the dominant factor. The above analysis was performed for a specific number of TSV per vault; as the number of TSVs per vault increases, the impact of TSV latency on the target metrics reduces due to the smaller transfer time.

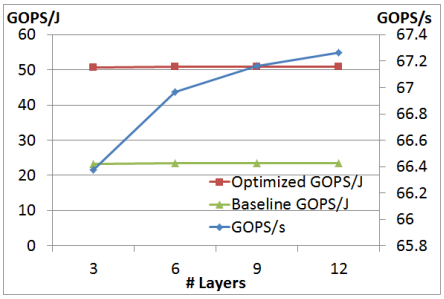
3) Power dissipation of TSVs (P_{tsv}): The power dissipation of TSVs affect the energy consumption of the interconnect layer and hence the overall energy efficiency. However, the transfer time is independent of the power dissipation and so, the throughput remains the same irrespective of changes in power dissipation. We vary the power dissipation per TSV from 20 mW to 1 mW. In the baseline implementation, the effect of power of TSVs and hence the energy consumption due to TSVs is not obvious due to the memory and FPGA energy dominating the total energy consumption. Therefore, as depicted in Figure 3(c), the variation in power from 20 mW to 1 mW results in a minor increase in energy efficiency from 23 GOPS/J to 25 GOPS/J. In the optimized implementation, the effect of P_{tsv} is more evident. With $P_{tsv} = 20$ mW, the energy consumed by the TSVs is comparable to the energy consumed by the memory. With $P_{tsv} = 1$ mW, the energy consumption of TSVs is orders of magnitude lower than the FPGA energy and approximately $6\times$ lower than that of energy consumed by the memory. Even in this case, with memory activation



(a) Number of occupied vaults



(b) Total number of vaults



(c) Number of layers

Fig. 4: Effect of varying number of vaults and layers on performance

scheduling enabled, the energy consumed by memory is lower compared to the FPGA energy. Hence, the effect of decrease in power of TSV results in only a modest increase in energy efficiency from 50.6 GOPS/J to 59.7 GOPS/J as illustrated in Figure 3(c).

B. Vaults

Since the vaults are connected to the FPGA layer through independent TSVs, they can be active at the same time and this results in increased parallelism. We vary the number of occupied vaults, i.e., vaults which contribute to the overall memory bandwidth and the total number of vaults in the 3D memory.

1) Number of Occupied Vaults (V_{occ}): As the vaults do not share the TSVs, the effective bandwidth of the 3D memory is the product of number of occupied vaults and the bandwidth per vault. We now vary the bandwidth available by varying the number of occupied vaults. In this configuration, the total number of vaults is 16 and we vary the number of occupied vaults from 1 to 16. In the baseline implementation, this results in an increased bandwidth and hence lower data transfer time. The effect of increasing the number of active vaults is evident by the large jump in throughput from 66 GOPS/s to 131 GOPS/s when the number of occupied vaults is changed from 1 to 4. This effect diminishes as the number of occupied vaults reaches 16. This is because even though transfer time consistently reduces with the increase in number of active vaults, the overall processing time for matrix multiplication is dominated by its computation time on the FPGA. With the increase in bandwidth, the total computation time for matrix multiplication ($t_{compute}$) reduces and hence, the memory is in ACT mode for a smaller amount of time as compared to the case with lower bandwidth. This results in a significant increase in performance from 23 GOPS/J to 42 GOPS/J as seen in Figure 4(a). As expected, this increase in performance saturates with the number of occupied vaults reaching 16 as the FPGA computation time becomes a major component.

In the optimized implementation, the occupied vaults which contribute towards the total memory bandwidth are in ACT mode and the other vaults are in PRE mode. Once the data transfer from memory to the FPGA has taken place, all the vaults are in PRE mode. The performance in terms of throughput for the optimized implementation is equal to the baseline implementation. The energy efficiency increases with the number of occupied vaults. The reasoning is similar to the baseline implementation. However, unlike the case of the baseline implementation, the increase in number of occupied vaults does not show a large improvement. Figure 4(a) shows a moderate improvement in energy efficiency from 61.7 GOPS/J to 66.5 GOPS/J. This is because memory energy is overshadowed by the energy consumed by FPGA when memory activation scheduling is enabled .

2) Total Number of Vaults (V): For a given size of memory, the number of vaults indirectly controls the memory size of a vault. A higher number of vaults means lower memory size per vault. The bandwidth of a vault is not dependent on its size but on the number of vertical banks which can be simultaneously activated. Since we keep the TSVs per vault constant, the bandwidth of a vault is constant. Figure 4(b) presents the results of varying the number of vaults in the 3D memory. As the number of occupied vaults remains the same, varying the number of vaults does not have any effect on the data transfer time. However, the energy consumption of memory is influenced by this variation because the power dissipation per vault varies with the number of vaults. The higher the number of vaults, the lower the power dissipation per vault. This implies that the throughput remains constant whereas energy efficiency increases with number of vaults for the optimized implementation. This is because as the number of vaults increases, a higher number of vaults can be in PRE mode and this results in lower energy consumption. The overall effect on energy efficiency is not significant. This is because the energy consumption of the FPGA is much larger than that of memory.

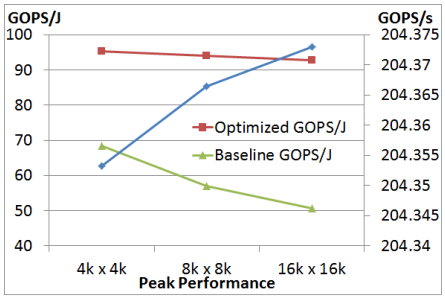
In the case of the baseline implementation, the number of vaults do not affect the energy efficiency as all the vaults are always in ACT mode. The ratio of energy efficiency of the optimized implementation to the baseline implementation consistently increases with the number of vaults. This is because the baseline implementation's energy consumption is independent of the number of vaults, whereas the optimized implementation's energy consumption reduces as the number of vaults are increased.

C. Number of Layers (L_m)

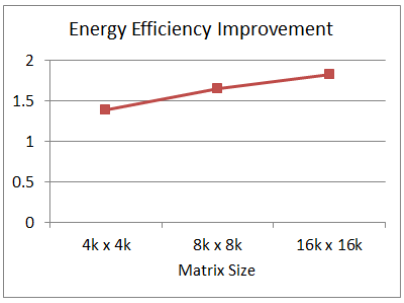
In the above analyses, the matrices occupied one layer each of the 3D memory. Increasing the number of layers effectively increases the number of banks in a vault. This results in more parallelism as more number of banks can be active at the same time. However, this also means more power dissipation. In Figure 4(c), we vary the number of layers from 3 to 12. This also represents the number of banks in a vault. Therefore, as the number of layers increases, the bandwidth available from a vault also increases. Surprisingly, this does not translate into a significant improvement in throughput performance. This is because with the increase in the number of layers, even though the bandwidth increases and the access time reduces, the amount of data to be transferred remains the same. Therefore, the transfer time across the LMI layer, $t_{transfer}$, is unaltered. Hence, the total computation time reduces by a very small margin and, as illustrated in Figure 4(c), the throughput does not show the improvement expected with the increase in number of layers. As the transfer time directly affects the time during which memory is in ACT mode, the energy consumption as well does not show noticeable variation. As a consequence, the energy efficiency also remains approximately the same at 51 GOPS/J. In the baseline implementation, varying the number of layers, as with the optimized implementation, causes an increase in the bandwidth with the transfer time being the bottleneck. Also, since memory is in ACT mode for approximately the same period of time, the energy consumption by memory is significant. Therefore, both the performance metrics, throughput and energy efficiency, do not show notable variation.

D. Different Matrix sizes

We vary the problem size from $4K \times 4K$ to $16K \times 16K$; these are all considered large scale matrices in that they do not fit on the FPGA. In Figure 5(a), we have plotted the peak performance of the baseline and optimized implementations for various matrix sizes. In this analysis, we have chosen the parameter values which represent the futuristic state of the art in 3D technology. We have chosen 32 vaults with 512 TSVs per vault. The data being accessed is distributed among all the vaults. The power dissipation per TSV is 5 mW and latency is 1 ns. As seen from Figure 5(a), the peak performance for $16K \times 16K$ matrix multiplication is 204 GOPS/s and the throughput decreases marginally with the reduction in problem size. The reason is that throughput is defined as ratio of useful operations to $t_{compute}$ (Equation 1).



(a) Peak Performance



(b) Energy Efficiency Improvement

Fig. 5: Peak performance for different matrix sizes

This ratio scales approximately as $\frac{1}{1/N+1/m}$. Thus, the impact of increasing problem size (N) results in only a marginal increase in throughput.

For smaller matrices, the smaller number of computations on the FPGA implies that the energy consumption by the memory also reduces significantly. In the case of the baseline implementation, as energy consumption of memory and FPGA are comparable, the energy efficiency increases with the decrease in problem size. The peak energy efficiency for $16K \times 16K$ problem size in the optimized implementation is 93 GOPS/J, and the effect of problem size on energy efficiency is less pronounced. This is because the FPGA energy dominates the total energy consumption. The FPGA energy scales with the problem size since it is proportional to the total computation time. Thus, the energy efficiency does not improve significantly. Since the improvement in performance of baseline implementation is much stronger than the optimized implementation, the improvement in energy efficiency decreases with problem size as shown in Figure 5(b).

E. Comparison with 2D Architecture

We now compare the peak performance of 3D MI-FPGA with that of a 2D architecture. Our 2D architecture consists of an FPGA connected to an external memory (i.e., the LMI layer in 3D MI-FPGA is replaced by off-chip connections). We assume the memory operates at a frequency 800 MHz with a theoretical peak bandwidth of 15 GB/s. The data transfer time from memory to FPGA for one block of size $m \times m$ with each element of 16 bits precision is $\frac{16m^2}{15\times8}ns$. Therefore, $t_{access} + t_{transfer}$ for the 2D architecture is 69.9

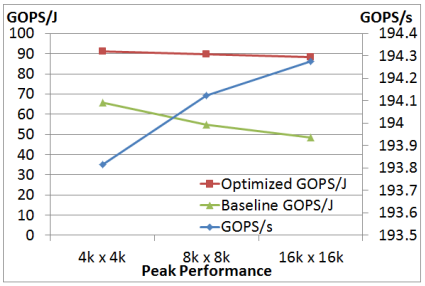


Fig. 6: Peak performance of 2D architecture

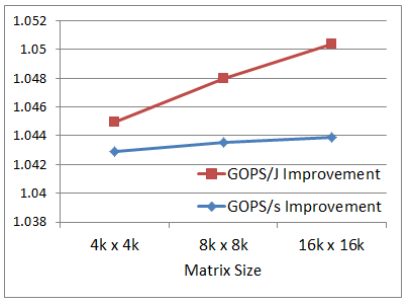


Fig. 7: Comparison of performance between 2D architecture and 3D MI-FPGA architecture

μs for two blocks of data and the corresponding parameter value in 3D MI-FPGA is 2.7 μs. Figure 6 shows the peak performance of the 2D architecture for different problem sizes. We see that even though the 3D memory reduces the transfer time by a large extent, as seen from Figure 7, this does not translate to a similar improvement in throughput. Since the computation time for FPGA (t_{fpga}) is the same in both 2D and 3D architectures, it is the major bottleneck in achieving higher performance. The power dissipation for a given size of 2D memory is the same as that of 3D memory (P_{mem}). The difference is that the 2D memory is divided into banks and achieving peak bandwidth requires that all banks must be active. The parameters equivalent to TSV power is I/O power and we use a reasonable estimate [19] to calculate the I/O energy. With the energy consumed by FPGA given by Equation 2, the peak energy efficiency of the 2D architecture is approximately 91 GOPS/J. From Section VII-D, the peak performance of 3D MI-FPGA is marginally better than that of 2D architecture. Since FPGA energy is the same in both 2D and 3D MI-FPGA architectures and the FPGA energy far exceeds the memory energy when the memory activation scheduling optimization is enabled, there is no significant difference in the energy efficiency.

VIII. Discussion

The time required for data transfer between memory and FPGA is orders of magnitude smaller than the computation time for matrix multiplication on FPGA. This is the reason for only a marginal improvement in matrix multiplication performance on a 3D MI-FPGA as compared to 2D architecture. The large computation time on FPGA also means that the energy consumption of FPGA dominates the total energy as the memory can remain in power down mode for the major portion of time in both 2D and 3D architectures. Therefore, the energy efficiency (GOPS/J) also does not show significant improvement when moving from 2D to 3D architecture. So, the FPGA appears to be the bottleneck in both performance metrics GOPS and GOPS/J.

Matrix multiplication belongs to the class of application which require $O(m^3)$ work done for a given problem of size $m \times m$. The computation time on logic layer should be of the same order as the data transfer time between memory and FPGA, otherwise, the 3D memory will be idle for the major portion of the total processing time. The high data transfer rate of 3D memory is the reason for the significant difference between the data transfer time and computation time on FPGA. Therefore, in this section we look at the general class of applications which require $O(m^2)$ amount of work on FPGA for a data of size m^2. The data is brought from the memory into the FPGA and the computations are carried out on the data which take $O(m^2)$ work. We assume that the computation time can be reduced to $O(m)$ by using m PEs. An example of such an application is matrix transposition. In our architecture, we bring two blocks of size $m \times m$ from the memory and store it in m Block RAMs. We use m PEs and local buffers of size m to transpose the data in the Block RAMs. Once the transpose is completed in m cycles, the data is written back into the memory. Therefore, the only difference between 3D MI-FPGA and the 2D architecture is the data transfer time between memory and FPGA. For a given size of matrix, the 3D memory results in low data transfer time and hence higher throughput whereas the 2D architecture having a lower bandwidth has a lower throughput. This increased bandwidth of the 3D memory also means that the memory has to be in ACT mode for a shorter period of time compared to the 2D memory. This also results in higher energy efficiency in the case of 3D MI-FPGA compared to 2D architecture.

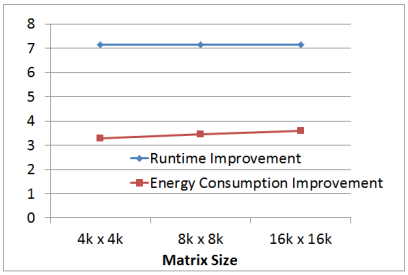


Fig. 8: Matrix Transposition: Relative performance comparison between 2D and 3D architectures

In Figure 8, we compare the peak performance of the optimized 3D MI-FPGA and the optimized 2D architecture of matrix transposition for various matrix sizes. Since matrix transpose involves data movement and no operations, *total runtime* and *energy consumption* are chosen as the performance metrics. We do not consider the baseline implementation as their performance will be inferior to the optimized implementation.

Compared to the matrix multiplication, the improvement in the case of matrix transposition is significant as illustrated in the Figure 8. Relatively, 3D MI-FPGA consumes 3x less energy compared to 2D architecture and the total runtime of 3D MI-FPGA is approximately 7x lower. The reason for the significant improvement is attributed to the fact that the FPGA computation time is of the same order as that of data transfer time. And, as the computation time scales with the data transfer time, the improvement for different problem sizes remains constant. When it comes to the other performance metric energy efficiency, as the problem size increases, the amount of time memory is in ACT mode is higher in the 2D architecture compared to 3D memory. Therefore, there is a marginal improvement in energy consumption as the problem size increases.

IX. Conclusion

We developed a performance model for matrix multiplication on an abstract 3D Memory Integrated FPGA architecture. We evaluated the effects of various architecture parameters on throughput and energy efficiency. With respect to varying TSV parameters (number of TSVs per vault, latency, and power dissipated by TSVs), the computation time and energy consumed by the FPGA proved to be the dominating factor. With respect to varying the number of memory vaults as well, the large energy consumption by the FPGA was the reason for only a minor improvement in energy efficiency. However, while varying the number of memory layers, the transfer time across the LMI layer (TSVs) was the major bottleneck resulting in almost no improvement in both throughput and energy efficiency. Applying the memory activation scheduling optimization to vaults and banks results in a $1.83\times$ improvement in peak energy efficiency. Compared to a 2D architecture, the improvement in performance metrics is negligible for both throughput and energy efficiency as the large computation time on the FPGA and the energy consumed by the FPGA is the major bottleneck. Matrix multiplication requires a computation work of $O(m^3)$ and this resulted in FPGA dominating the energy and processing time. Therefore, we also looked at the general class of applications which require $O(m^2)$ amount of computation work. In this context, the capabilities of 3D memory can be exploited to provide a significant improvement in both the performance metrics. Specifically, in the case of Matrix Transposition, the improvement in performance is of the order $7\times$ for throughput and $3\times$ for energy efficiency. Therefore, in order to take advantage of the 3D memory's enormous bandwidth, the logic layer and the application also plays a crucial role. For future work, we will develop more fine grained performance models for different application kernels as implemented on a 3D MI-FPGA.

References

[1] Hybrid Memory Cube Consortium. Hybrid Memory Cube Specification. http://hybridmemorycube.org/files/SiteDownloads/HMC_Specification%201_0.pdf.

[2] Shreyas G. Singapura, Anand Panangadan, and Viktor K Prasanna. Towards Performance Modeling of 3D Memory Integrated FPGA Architectures. In *To Appear in Applied Reconfigurable Computing*. 2015.

[3] Balaji Vaidyanathan, Wei-Lun Hung, Feng Wang, Yuan Xie, Vijaykrishnan Narayanan, and Mary Jane Irwin. Architecting Microprocessor Components in 3D Design Space. In *VLSI Design, 2007. Held jointly with 6th International Conference on Embedded Systems., 20th International Conference on*, pages 103–108. IEEE, 2007.

[4] Kiran Puttaswamy and Gabriel H Loh. The Impact of 3-Dimensional Integration on the Design of Arithmetic Units. In *Circuits and Systems, 2006. ISCAS 2006. Proceedings. 2006 IEEE International Symposium on*, pages 4–pp. IEEE, 2006.

[5] Kiran Puttaswamy and Gabriel H Loh. Implementing Register Files for High-Performance Microprocessors in a Die-Stacked (3D) Technology. In *Emerging VLSI Technologies and Architectures, 2006. IEEE Computer Society Annual Symposium on*, pages 6–pp. IEEE, 2006.

[6] Kiran Puttaswamy and Gabriel H Loh. Dynamic Instruction Schedulers in a 3-dimensional Integration Technology. In *Proceedings of the 16th ACM Great Lakes symposium on VLSI*, pages 153–158. ACM, 2006.

[7] William Rhett Davis, Eun Chu Oh, Ambarish M Sule, and Paul D Franzon. Application Exploration for 3-D Integrated Circuits: TCAM, FIFO, and FFT Case Studies. *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, 17(4):496–506, 2009.

[8] Sam Kavusi, Kunal Ghosh, and Abbas El Gamal. *Architectures for High Dynamic Range, High Speed Image Sensor Readout Circuits*. Springer, 2008.

[9] Negin Golshani, Jaber Derakhshandeh, Ryoichi Ishihara, CIM Beenakker, Michael Robertson, and Thomas Morrison. Monolithic 3D Integration of SRAM and Image Sensor Using Two Layers of Single Grain Silicon. In *3D Systems Integration Conference (3DIC), 2010 IEEE International*, pages 1–4. IEEE, 2010.

[10] Awet Yemane Weldezion, Zhonghai Lu, Roshan Weerasekera, and Hannu Tenhunen. 3-D Memory Organization and Performance Analysis for Multi-processor Network-On-Chip Architecture. In *3D System Integration, 2009. 3DIC 2009. IEEE International Conference on*, pages 1–7. IEEE, 2009.

[11] Xiao Yu, Li Li, Yuang Zhang, Hongbing Pan, and Shuzhuan He. Performance and Power Consumption Analysis of Memory Efficient 3D Network-on-Chip Architecture. In *Control and Automation (ICCA), 2013 10th IEEE International Conference on*, pages 340–344. IEEE, 2013.

[12] Sung Kyu Lim. 3D-MAPS: 3D Massively Parallel Processor with Stacked Memory. In *Design for High Performance, Low Power, and Reliable 3D Integrated Circuits*, pages 537–560. Springer, 2013.

[13] Qiuling Zhu, Berkin Akin, H Ekin Sumbul, Fazle Sadi, James C Hoe, Larry Pileggi, and Franz Franchetti. A 3D-Stacked Logic-in-Memory Accelerator for Application-Specific Data Intensive Computing. In *3D Systems Integration Conference (3DIC), 2013 IEEE International*, pages 1–7. IEEE, 2013.

[14] Peter Gadfort, Aravind Dasu, Ali Akoglu, Yoon Kah Leow, and Michael Fritze. A Power Efficient Reconfigurable System-in-Stack: 3D Integration of Accelerators, FPGAs, and DRAM. In *System-on-Chip Conference (SOCC), 2014 27th IEEE International*, pages 11–16. IEEE, 2014.

[15] Kiran Kumar Matam, Hoang Le, and Viktor K Prasanna. Evaluating Energy Efficiency of Floating Point Matrix Multiplication on FPGAs. In *High Performance Extreme Computing Conference (HPEC), 2013 IEEE*, pages 1–6. IEEE, 2013.

[16] Xilinx. XPower Analyzer. http://www.xilinx.com/products/design_tools/logic_design/verification/xpower_an.htm.

[17] Micron. System Power Caclulator. http://www.micron.com/~/media/documents/products/power-calculator/ddr3_power_calc.xlsm?la=en.

[18] Joohee Kim, Jonghyun Cho, Jun So Pak, Taigon Song, Joungho Kim, Hyungdong Lee, Junho Lee, and Kunwoo Park. I/O Power Estimation and Analysis of High-speed Channels in Through-Silicon Via (TSV)-based 3D IC. In *Electrical Performance of Electronic Packaging and Systems (EPEPS), 2010 IEEE 19th Conference on*, pages 41–44. IEEE, 2010.

[19] Xilinx. High-Performance, Lower-Power Memory Interfaces with UltraScale Architecture FPGAs. http://www.xilinx.com/support/documentation/white_papers/wp454-ultrascale-memory.pdf.