High Performance AXI-4.0 Based Interconnect for Extensible Smart Memory Cubes

Erfan Azarkhish,
Davide Rossi, and Igor Loi
DEI, University of Bologna, Bologna, Italy
Emails: erfan.azarkhish@unibo.it,
davide.rossi@unibo.it, and igor.loi@unibo.it

Luca Benini
ITET, Swiss Federal Institute of Technology,
Zurich, Switzerland,
DEI, University of Bologna, Bologna, Italy
Email: lbenini@iis.ee.ethz.ch

***Abstract*—The recent technological breakthrough represented by the Hybrid Memory Cube is on its way to improve bandwidth, power consumption, and density. This is while heterogeneous 3D integration has provided another opportunity for revisiting near memory computation to fill the gap between the processors and memories even further. In this paper, we take the first step towards a "Smart Memory Cube (SMC)", a fully backward compatible and modular extension to the standard HMC, supporting near memory computation on its Logic Base (LoB), through a high performance interconnect designed for this purpose. The main feature of SMC is the high bandwidth, low latency, and AXI-4.0 compatible interconnect, designed to serve the huge bandwidth demand by HMC's serial links, and to provide extra bandwidth to a processor-in-memory (PIM) embedded in the Logic Base (LoB). Our results obtained from cycle accurate simulation demonstrate that this interconnect can easily meet the demands of current and future projections of HMC (Up to 87GB/s READ bandwidth with 4 serial links and 16 memory vaults, and 175GB/s with 8 serial links and 32 memory vaults, for injected random traffic). Moreover, the interference between the PIM traffic and the main links was found to be negligible with execution time increase of less than 5%, and average memory access time increase of less than 15% when 56GB/s bandwidth is requested by the main links and 15GB/s bandwidth is delivered to the PIM port. Moreover, preliminary logic synthesis with Synopsys Design Compiler confirms that our interconnect is implementable and realistic.**

I. Introduction

The speed and bandwidth disparity between processors and memory, also known as "the memory wall", has been a problem for the last thirty years [1]. Therefore, many researchers, since the early nineties [2], have looked into the possibility to migrate some part of computation closer to the memory systems. Unfortunately, the "processing in memory" research efforts in the late nineties and the first decade of the new millennium (See [2][3][4] for samples) did not lead to successful industrial platforms and products. The main reason for this lack of success was that all these works were assuming that significant amount of logic resources, needed for having processing elements close to the memory arrays, could be integrated on DRAM dies (or vice versa). This could not be achieved economically given the restrictions of DRAM processes (e.g., limited number of metal levels, slow transistors). On the other hand, integration of DRAM in logic processes has achieved some partial success, but it has always been plagued by high cost and low memory density issues [5].

Starting from 2011, this situation started to change with the appearance of heterogeneous 3D integration of logic dies and memory dies based on through-silicon-vias (TSV). TSV technology was brought to commercial maturity by memory manufacturers (DRAM and Flash) to build "memory cubes" made of vertically stacked thinned memory dies which achieve higher capacity in packages with smaller footprint and power compared to traditional multi-chip modules. The last missing piece came in place when an industrial consortium backed by

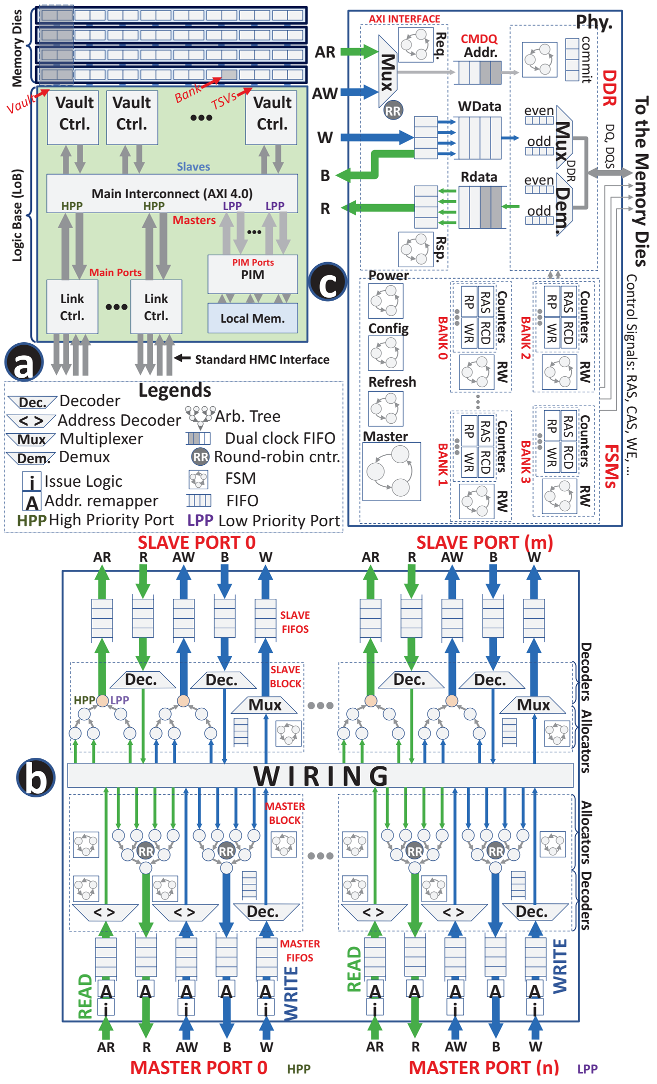

Fig. 1. a) Overview of the Smart Memory Cube, b) proposed AXI-4.0 based interconnect for SMC, c) schematic view of a Vault Controller

several major semiconductor companies introduced the Hybrid Memory Cube (HMC) [6] In the HMC, a memory cube is stacked on top of a logic die. The logic die at the bottom of the hybrid stack provides advanced interface functions between the memory cube on top and the rest of the computing system on the board.

In this paper, we leverage the recent technology breakthrough represented by the HMC to revisit the possibility of near memory processing inside the cube, taking advantage of the heterogeneous 3D stacking technology. We mainly focus on the architectural implications and the required infrastructure inside HMC to support this feature. Therefore, exploiting the high internal bandwidth provided by TSVs we propose a modular and scalable solution, called the "Smart Memory Cube (SMC)". SMC is build upon the existing HMC standard, and is compatible with its interface, with no changes made to the memory dies, and no new die introduced in the stack. In other words, SMC is fully backward compatible with the HMC IO interface specification, and features a high performance and extensible AXI-4.0 based interconnect on its Logic Base (LoB), carefully designed to provide high bandwidth to the external serial links, as well as plenty of extra bandwidth to any generic and AXI-compliant PIM device attached to its extension ports. We have developed a cycle accurate model for the SMC interconnect and its interfaces, and tuned its parameters based on the available data from the literature on HMC. Our results demonstrate that, the proposed interconnect can easily meet the demands of current and future projections of HMC (Up to 87GB/s READ bandwidth with 4 serial links and 16 memory vaults, and 175GB/s with 8 serial links and 32 memory vaults, for injected random traffic). Moreover, the interference between the PIM traffic and the main links was found to be negligible with execution time increase of less than 5%, and average memory access time increase of less than 15% when 56GB/s bandwidth is delivered to the main links and 15GB/s residual bandwidth to the PIM port.

II. Related Works

While the advancement of processor technology has rapidly increased computational capabilities in logic processes, improvements in bandwidth and latency to off-chip memory have not kept up, and in fact DRAM process is drifting further away from the logic process. As a result, an increasing portion of time and power in computing systems is spent on data movement, especially in off-chip memory accesses [1]. A possible solution to the memory wall problem is in-memory processing. Research in this area started more than two decades ago. Computational RAM [2] using SRAMs or DRAMs coupled with processing elements close to the sense amplifiers, and Intelligent-RAM (IRAM) [3] to fill the gap between DRAM and processors, are just a few examples of the efforts in this area. It was shown by [3] that in memory processing can lead to a memory bandwidth and energy efficiency improvement of 50X$\sim$100X and 2X respectively, along with a latency reduction of about 2X. Nevertheless, the effort for PIM dried out soon after 2000's without major commercial adopters, due to several performance, cost, and business model obstacles [4], arising from the incompatibility of DRAM process with logic.

With the recent advancements in process technology and emergence of 3D integration, the interest in near-memory computation has been renewed [7]. [8] introduces a PIM die into the 3D memory stack for accelerating specific FFT and matrix multiplication algorithms. In [9] throughput oriented computing using programmable GPU units for 3D stacked memories has been studied. Automata processor [10] is a very special processor in memory for traversing huge graphs and state machines for pattern matching, security, and computational biology. In DRAMA [11] a reconfigurable accelerator architecture for processing in 3D memory stacks is proposed. Lastly, in [1] logic die of the 3D memory stack has been augmented with PIM functionalities. All these works are not in competition but in synergy with us moving towards the same direction.

The most outstanding examples of 3D-memory stacking as substitutes for traditional DDR devices are the Hybrid Memory Cube (HMC) [12] and the High Bandwidth Memory (HBM) [13], with HMC offering higher flexibility by abstracting away the details of DRAM control, and providing a high-level communication mechanism over serial links. Therefore we believe that HMC is the best target for near memory computation. Focusing on the location of the PIM device in the HMC, it can be either integrated with the existing logic [1][14] or DRAM dies [15], or it can be added as a new die in the stack [8]. Introduction of a new layer to the stack would require redesign and a complete reanalysis of the 3D stack structure and the power distribution networks, affecting manufacturing yield of the stack, as well. Moreover, placing the PIM devices on the memory dies still suffers from the incompatibility of logic and DRAM processes [4] and the functionality and visibility of the PIM device to the address space will become extremely limited. On the other hand, placing the PIM device on the logic die, specifically behind the main interconnect in the HMC (See Fig.1a), could lead to a modular and scalable solution, with a global visibility of the whole memory space, exploiting the large bandwidth provided by TSVs, and without any concerns about the DRAM devices. Besides, this solution is the least intrusive one to the standard HMC architecture, as it does not make any change to the 3D stack or the DRAM dies. Lastly, the range of functionalities covered by the PIM devices has been studied thoroughly in [16], and it has been concluded that fixed-function PIMs without any support for memory management, multi-threading, and synchronization, are the best starting point for the evolution of the smart memories. In this paper, we won't focus on the implementation of the PIM device but on the design of an LoB interconnect that is flexible enough to host a wide range of PIM architectures, to be studied in future works.

One last point to mention is that, in the standard published by HMC consortium [6], the external interface is specified in complete details, nevertheless, the implementation of the Logic Base (LoB), the DRAM dies, and specially the main interconnect inside the LoB have been left open. Our main contribution is to design a high performance and low latency interconnect based on the AXI-4.0 standard to serve as the main interconnect in HMC, while providing additional bandwidth to a generic PIM device attached to it, ensuring that interference on the main traffic is minimum. Next section describes our proposal called the smart memory cube.

III. The Smart Memory Cube (SMC)

Smart Memory Cube (SMC) is an extension to HMC [6] providing the possibility of moving part of the computation inside the cube. Fig.1a illustrates an overview of the proposed underlying architecture for Smart Memory Cube. As shown in this figure, the standard interface and the 3D structure of the HMC has been left untouched, while a processor-in-memory (PIM) component has been added to the Logic Base (LoB) behind the global interconnect. It is important to guarantee that the interference caused by PIM's traffic on the traffic injected from the main links is bounded and negligible. For this purpose we propose an ultra-low latency logarithmic interconnect designed following AXI-4.0 standard, and present an analysis on the role of this interconnect in SMC. Next, our cycle accurate model for the baseline HMC system and its SMC extension are presented.

A. Cycle Accurate Modelling

As shown in Fig.1a, the main interconnect and the vault controllers are two key components in design of the smart memory cube. We have designed our interconnect based on the ultra low-latency "logarithmic interconnect" (originally designed for L1 and L2 contexts [17][18]), and modified it to support high bandwidth communication based on AMBA AXI-4.0 [19], the most widely used standard for on-chip communication in system-on-chips. This standard divides traffic streams into 5 categories and dedicates independent channels to each of them (AR: Address Read, R: Read Data, AW: Address Write, W: Write Data, B: Write Response).

Fig.1b illustrates a high-level schematic view of our interconnect design. When a transaction arrives at the AXI master ports, first the "Issue Logic" decides whether it should be allowed to enter the memory system, limiting the maximum outstanding transactions of each master port to a certain number called *MoT*, and assigning unique tags to the ones allowed into the interconnect. Next, address remapping is performed on the transactions based on the intended addressing scheme, by simply shuffling the address bits. AW and W channels deliver address and data for the WRITE transactions, and after decoding and identifying the destination port (inside the "Master Blocks"), WRITE address will be sent to an arbitration tree through the wirings. Among the master ports and separately on the PIM ports fair round-robin arbitration is performed. However, in the last stage of the network a fixed-priority arbitration scheme ensures higher priority for the main ports (hierarchical arbitration). This way only residual bandwidth is delivered to the PIM. The winner request will have its data delivered through multiplexers in the "Slave Block" to the FIFOs on the slave ports. Since AW and W channels do not have to be synchronized in AXI-4.0 standard, a transaction FIFO inside the "Slave Block" associates pending addresses to data bursts. A similar procedure takes place for READ transactions, except that READ requests do not have any data associated with them, and they are 1 flit long. On the other side of the network (slave side), B and R channels deliver back the response data, and acknowledgement of the WRITE transaction, respectively. The responses will arrive at the intended "Master Block" after a simple decoding, and there, they will wait for arbitration with other responses destined to the same master. Finally, they arrive at the intended master port via R and B channels. The arbitration performed inside this network is single-cycle and fully combinational, and easily meets the requirements for up to 32 masters/slaves [17]

Design of the Vault Controllers follows a very simple DRAM Channel Controller, again with a standard AXI-4.0 interface to connect seamlessly to the main interconnect (See Fig.1c). The first stage in this channel controller performs round-robin arbitration among the AXI AW and AR channels to issue one of them to the Command Queue (CMDQ) in each cycle. In case of WRITE, the burst data is stored in the WData FIFO. A set of Finite State Machines (FSMs) control power up/down, auto-refresh, and configuration of the DRAM devices; and for each memory bank a Read-Write FSM keeps track of the bank state and timings such as t_{RAS}, t_{RCD}, t_{RP}, t_{WR} using a set of counters. Finally, one Master FSM controls the main DRAM bus and all other mentioned FSMs. Design of the vault controllers and signalling at the DRAM bus follow the JESD79F JEDEC standard for DDR SDRAMs [20] in a generic and configurable way. Moreover, different techniques of pipelining and latency hiding have been implemented to reach the highest throughput; and unlike the standard vault controllers in HMC, this model supports both open and closed page policies. AXI Interface returns the READ response and WRITE acknowledge through R and B channels respectively, and since data widths and burst sizes of the AXI interconnect and the DRAM devices do not necessarily match, this module performs burst size conversion, as well. In addition, the clock domains of the AXI interconnect and the vault controllers have been designed independently, therefore, Command Queues (CMDQ) and RData FIFOs have been designed based on asynchronous dual-clock FIFOs [21] to ensure data integrity.

The DRAM device models have been adopted from [22] and extended in terms of number of banks, burst size, data width, etc. Detailed design of the link controllers has been left as a future work, since the main scope of this paper is management of traffic inside the memory cube. One final point is that, standard and flexible design of our main interconnect allows for connecting any AXI-compatible device including processor-in-memory modules. Therefore, a PIM device can be easily integrated in this model by simply attaching the PIM device to one of the low-priority ports (LPP) of the main interconnect(See Fig.1a).

B. Calibrating The Model

In the Exascale projection for 2013 [23] for a prototype of HMC manufactured in 2011, 4 Memory dies and 1 Logic Base (LoB) have been reported, with 16 independent memory vaults each consisting of 2 DRAM banks per memory die. With a bank size of 4MB a total memory of 512MB is provided in this configuration. Each vault is expected to deliver a bandwidth of 10GB/s to the LoB, which aggregates to a maximum of 160GB/s. On the other side, four serial links consisting of 16+16 differential lanes for READ and WRITE are advocated. With a lane bit-rate of 10Gb/s [23] a total off-chip bandwidth of 160GB/s for WRITE and READ can be delivered. In the first paper on HMC [12] 32 data TSVs were reported per each vault with a double data rate (DDR) data transfer mechanism, this requires a clock frequency of 1.25GHz to deliver 10GB/s [24]. Unlike existing DDR memories, HMC utilizes Closed-Page policy and its DRAM devices have been redesigned to have shorter rows (256 Bytes matching the maximum burst size of serial links, rather than 8-16KB in a typical DDR3 device) [12], focusing on High Performance Computing (HPC) and server workloads which typically exhibit little or no locality. The reduced row length helps save power by alleviating the over-fetch problem, however, reduces the row buffer hit probability, which makes open page mode impractical. In addition, open page policy exhibits additional overheads for little locality workloads, due to delaying the precharge between accesses to different rows [24][14] (See section IV.A for experiments). It is worth mentioning that, in the HMC projected for 2015, the vault bandwidth and TSV structure are not expected to change [23]. Instead the number of vaults will increase to 32 and the serial links will double to 8, delivering a total bandwidth of 320GB/s. Moreover, HMC specification V2.0 has been released in 2014 which is not available to public yet. The main focus of our experiments will be on the current and silicon demonstrated HMC parameters [12], nevertheless, in one experiment we will show that our model can be easily scaled to 32 memory vaults, and 8 serial links delivering the required bandwidth.

The specifications of the DRAM devices utilized in HMC are proprietary. However, to the best of our knowledge [25] contains the most comprehensive set of parameters that currently published for HMC: $\{t_{RP}$=13.75ns, t_{tCCD}=5ns, t_{RCD}=13.75ns, t_{CL}=13.75ns, t_{WR}=15ns, t_{RAS}=27.5ns$\}$. Moreover, we assume that the DRAM devices have the same clock frequency ($t_{CK} = 0.8ns$) as the TSVs, while the interconnect and the rest of the cube work in different clock domains. Besides, we assume the internal Data Width of the DRAM devices to be 32 bits, matching the TSV count. These assumptions simplify the 3D interface allowing for direct connection between TSVs and the DRAM devices. Similar approach has been taken in [24]. HMC supports different types of address mapping through its Address Mapping Mode Register. The default address mapping in HMC is low-interleaved, optimized to achieve highest memory-level parallelism [6]. In our model, there is a possibility for modifying the address interleaving scheme, through "Address Remapper" modules illustrated in Fig.1b. The baseline address mapping of HMC is [RC-BA-VA-OF] (VA: vault address, BA: bank address inside the vault, RC: row and column addresses, and OF: offset bits of the address). Assuming that transaction splitting is not possible in the HMC [6], a full transaction is always directed to one bank, therefore, OF bits are always in the leas significant position. This results in 6 possible permutations for address mapping, which are investigated in Section IV.

A preliminary logic synthesis confirmed that our AXI-based interconnect can easily meet a frequency of 1GHz for mentioned parameters (See section IV). While a simple calculation reveals that with this clock frequency and a flit size of 128-bits, a total of 128GB/s bandwidth can be delivered to 4 master ports (which is below the intended limit). To workaround this issue three straightforward solutions are available. First, increase in the clock frequency of the interconnect, which may increase power consumption severely. Second, doubling the number of master ports in the main interconnect and connecting each Link Controller to two of them. This solution is not suitable either, because it will increase the depth of the interconnect, and arbitration latency will increase, as well. Last, is to increase the flit-size of the main interconnect from 128 to 256 bits. This solution has none of the shortcomings of the previous methods, and can achieve an aggregate bandwidth of 256GB/s at the master side. Moreover, size conversion from 128-bit to 256-bits flits can be done easily in the Link Controllers where serial data is being converted to parallel. The only issue which should be considered is the increase in the area of the interconnect, which will be studied in Section IV. Throughout the whole experiments flit-size has been assumed to be 256, unless otherwise stated. With this assumption, the maximum burst size inside the interconnect will be reduced to 8. Next section presents the experimental results.

IV. Experimental Results

Our baseline HMC model has been described using SystemVerilog HDL in the cycle-accurate and fully synthesizable level. ModelSim has been utilized for simulation, and preliminary logic synthesis has been performed using Synopsys Design Compiler using STMicroelectronics Bulk CMOS-28nm Low Power technology library. Area of each vault controller was found to be $0.62mm^2$, and the AXI interconnect less than $0.2mm^2$. Summing to a total of about $10.1mm^2$ which is much less than the DRAM area reported for HMC 2011 ($68mm^2$) [12]. Two types of traffic have been exploited for performance analysis: Synthetic random traces generated in ModelSim, and memory access traces gathered in Gem5 [26] running a full-system simulation of eight x86 CPUs with Linux 2.6.22.9 kernel executing PARSEC V2.1 benchmark suite [27]. As previously observed in [24], we noticed that PARSEC benchmarks cannot utilize the full bandwidth provided by HMC. To increase the bandwidth demand of the traces, we utilized trace time compression, and also increased the transaction size of the traces from 64Bytes to 256Bytes (equal to the row buffer size of HMC). This is representative of future systems with much more bandwidth requirements and larger cache lines (a common trend).

A. *HMC Exploration*

First we adjusted the size of all buffers along with the MoT parameters to achieve the maximum bandwidth requirement of HMC with a reasonable access latency, measured as Average Memory Access Time (AMAT). Through several experiments we realized that only the size of the CMDQ FIFOs in the vault controllers (Fig.1c) and MoT (Fig.1b) affect the delivered bandwidth and latency significantly, while the size of the other FIFOs can be minimized. We found that with CMDQ Size=32 elements, and MoT=44 up to 87GB/s (READ) can be achieved for uniform random traffic. With these numbers, Fig.2a,b illustrate delivered bandwidth and AMAT versus requested bandwidth, in the baseline HMC model (RIGHT), and in one vault only (LEFT). Uniform random READ transactions have been applied to the ports, and AMAT has been measured at the response master ports of the interconnect. In both cases, when the network is not saturated, delivered bandwidth is over 97% of the requested bandwidth (81.2GB/s), matching HMC's intended READ bandwidth (80GB/s). While AMAT is

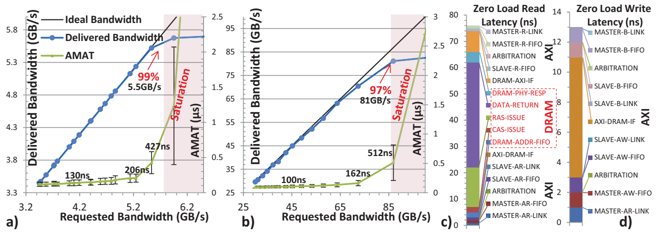

Fig. 2. Delivered bandwidth of a) one vault only and b) the baseline HMC. c) Zero load latency breakdown for READ and WRITE commands.

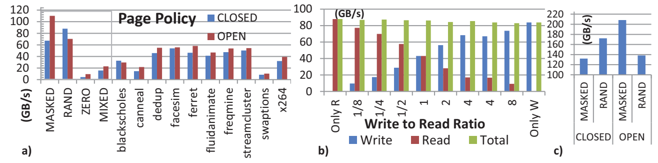

Fig. 3. a) Effect of page policy on delivered bandwidth, b) effect of R/W ratio in RAND traffic on total bandwidth, c) Delivered bandwidth to HMC-2015

bounded and less than 500ns, which is about 7X of the zero load latency (76ns) illustrated in Fig.2c,d.

To demonstrate this fact better, different traces have been applied to the baseline model changing only the page policy, with results plotted in Fig.3.a. RAND is a uniform random traffic with only READ transactions of maximum burst size. MASKED is a random traffic with its lower address bits intentionally masked to zero to achieve the highest hit-rate in open page policy. ZERO is a traffic directed to address 0x00000000, MIXED is a mixture of 8 PARSEC benchmarks, and all PARSEC benchmarks have been time compressed to the highest possible value. Firstly, it can be seen that ZERO in open page mode obtains 9.6GB/s which is 96% of the intended bandwidth for 1 vault. This proves that both the interconnect, and the vault controllers are working at full-bandwidth and are not losing any cycles. Moreover, MASKED traffic in open page mode receives an average bandwidth of 110GB/s for READ transactions. This is the maximum bandwidth which our network is able to deliver. Interestingly, when RAND traffic is applied, closed-page policy operates better than open-page, with a delivered bandwidth beyond requirement of HMC (i.e. 80GB/s for READ). And even in PARSEC benchmarks which have a lot of locality, still open page does not provide superior benefits to cover for its implementation cost. In addition, it can be seen in Fig.3.b that changing READ to WRITE ratio in RAND traffic does not improve the total delivered bandwidth. In the next experiments we will show that this limitation is caused by the DRAM and not the interconnect. Lastly, Fig.3.c illustrates delivered bandwidth to HMC projected for 2015. It can be seen that our model scales easily to the intended limit (160GB/s READ bandwidth for RAND traffic).

Next, we analyse the effect of different architectural parameters on the delivered bandwidth. Fig.4.a illustrates delivered bandwidth when the flit size of the AXI interconnect has been changed from 128 to 512. We can see that, 128-bit flits are not enough (specially for the RAND traffic), while, 512-bits is not necessary in any of the cases. Fig.4.b illustrates the effect of DRAM Bus Width (i.e. number of TSVs per each vault). This plot shows that if TSV manufacturing technology allowed, a bandwidth improvement of about 30% for RAND traffic would be achievable, with 64 data TSVs per each vault. Next, the clock period of the DRAM devices (t_{CK}) and the AXI interconnect have been swept to study their effect on bandwidth. Fig.5a,b illustrate the results. It is observed that decreasing t_{CK} can improve delivered bandwidth, while the clock period of AXI Interconnect does not have much effect on it. This suggests that the main bottleneck of the system is the DRAM and not the AXI interconnect. To further confirm this result we have scaled down all the timing parameters

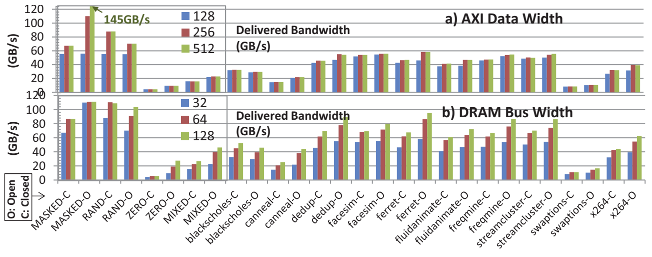

Fig. 4. Effect of a) AXI data width and b) DRAM bus width on bandwidth

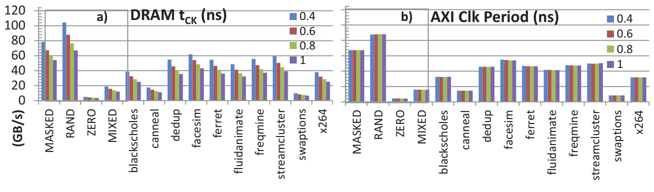

Fig. 5. Effect of a) DRAM t_{CK} and b) AXI Clk period on bandwidth

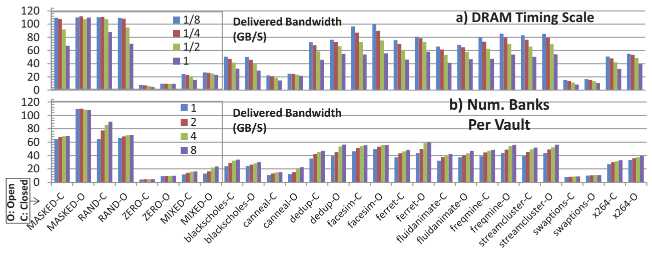

Fig. 6. a) Effect of scaling down the timing parameters of DRAM and b) effect of the number of banks per vault on bandwidth

of the DRAM devices from their default values by the scale factors illustrated in Fig.6.a, where the same traffic set has been applied and the page policy is closed. Interestingly, the delivered bandwidth can be highly improved when DRAM overheads are very small, and close to an ideal SRAM.

Fig.6.b depicts the effect of the number of banks per each vault on the delivered bandwidth. When there is only one bank per each vault, closed-page behaves almost similarly to the open-page, while as this number increases, there is potential for latency hiding specially for the RAND traffic where bank-level parallelism is high. For traffics which exhibit locality, address mapping was found to have an extremely important effect on the delivered bandwidth and total execution time. Fig.7 illustrates the delivered bandwidth over time for 3 most bandwidth critical PARSEC benchmarks using the 6 possible address mapping schemes. The benchmarks are time compressed to the maximum value and there is no empty space among the consecutive transactions (End time of each case has been highlighted in the graph with a vertical line). Interestingly, address mapping scheme affects the benchmarks differently. For x264 the best address mapping is RC-BA-VA-OF (HMC's default mapping), while in canneal RC-VA-BA-OF achieves the highest performance with an execution time reduction of over 2X compared to the default address mapping. And in swaptions BA-RC-VA-OF achieves the highest performance. This suggests that a configurable addressing mechanism (See Fig.1b) is required to tune the performance of the cube based on the running application.

B. Handling PIM Traffic

Assuming a PIM device is connected to the main interconnect through one low priority port, firstly, we aim to find the upper bound on the bandwidth that it can request without disrupting the main traffic. Therefore, we assume that the PIM device can process received data instantaneously, and on the four main links as well as the PIM link RAND transactions are injected with various bandwidth profiles. Fig.8a,b show total

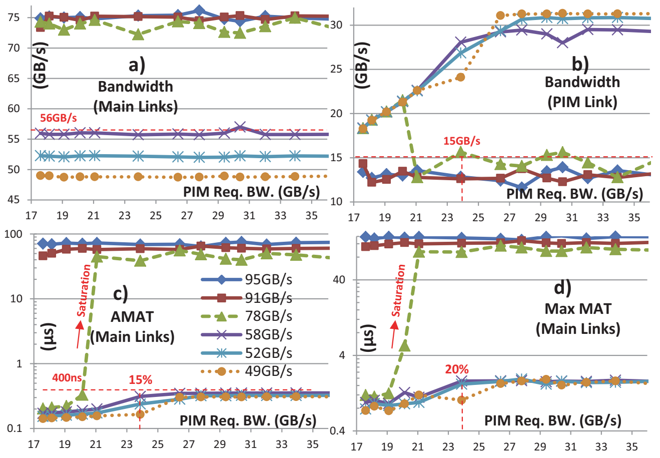

Fig. 8. a) Effect of PIM traffic on the bandwidth of the main links, b) delivered bandwidth to PIM as a function of its requested bandwidth, and increase in c) average and d) maximum memory access time caused by PIM.

delivered bandwidth on the main links and to the PIM device. While Fig.8c,d illustrate the average and maximum Memory Access Time (MAT) on the main links, respectively. All plots have been characterized based on the amount of requested bandwidth on the main links (from 49GB/s to 95GB/s), and the X-axis shows requested bandwidth by the PIM device. Firstly, when 56GB/s or less bandwidth is requested on the main links, the PIM device can request up to 31GB/s without pushing the system into saturation, and without observing any drop in the bandwidth of the main links. Moreover, Average Memory Access Time (AMAT) is below 400ns (5X of zero load AMAT). However, to keep the increase in maximum MAT within 20% and AMAT within 15%, a bandwidth of less than 15GB/s should be requested on the PIM device. This is because in typical general purpose servers, the traffic on the main links mostly consists of cache refill requests, and they are very sensitive to latency. Therefore not only AMAT but also the worst-case MAT should be small. While, a less conservative bandwidth partitioning could be speculated for accelerator-dominated traffics (e.g. GPGPU) which are much less latency-sensitive. To demonstrate this fact better, instead of random traffic, real traces of the high bandwidth demanding x264 benchmark with no time compression are applied on the main port, and interference caused by random traffic from PIM has been plotted in Fig.9.a. This plot shows that the PIM device can request up 28GB/s (90% of its theoretical max.) without affecting the AMAT of x264 transactions by over 10% percent. This result persists even with a 2X trace time compression with a similar bandwidth delivered to the PIM device, and less than 1% increase in total execution time.

Lastly, we study the effect of double-buffering on PIM's performance, assuming that an ideal PIM is working on one buffer while the other buffer is being fetched from the memory by a Direct Memory Access (DMA) module. Again, x264 with no time compression is played on the main links, and the goal is to find the best buffer size to incur minimal latency increase to the main traffic. It can be seen in Fig.9.b that for a buffer size up to 4KB (which provides a bandwidth of 18GB/s to the PIM) the maximum latency of the main links increases only by a factor of 11% compared to the case with no PIM.

V. Conclusions and Future Works

In this paper, we presented the smart memory cube, a fully backward compatible and modular extension to the standard HMC, supporting near memory computation on its Logic Base (LoB), through a high performance AXI-compatible interconnect designed for this purpose. Cycle accurate simulation demonstrated that, the proposed interconnect can easily

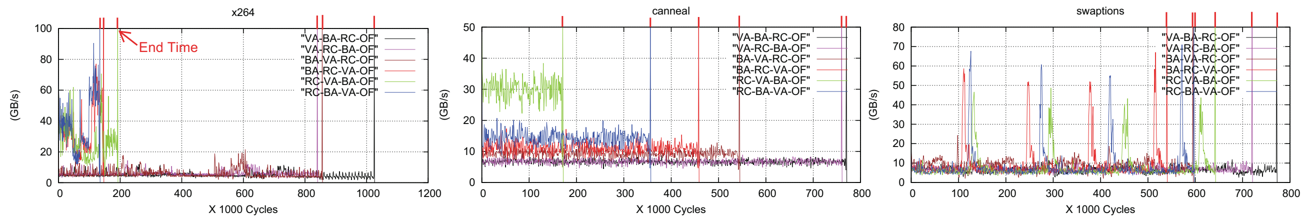

Fig. 7. Effect of address mapping on delivered bandwidth to PARSEC benchmarks, measured in epochs of 1000ns

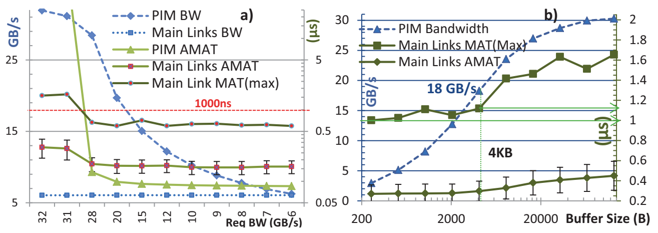

Fig. 9. a) Effect of requested bandwidth by ideal PIM on x264 b) effect of buffer size of double-buffering PIM on x264

meet the demands of current and future projections of HMC (Up to 87GB/s READ bandwidth with 4 serial links and 16 memory vaults, and 175GB/s with 8 serial links and 32 memory vaults, for injected random traffic). Moreover, the interference between the PIM traffic and the main links was found to be negligible with execution time increase of less than 5%, and average memory access time increase of less than 15% when 56GB/s bandwidth is requested by the main links and 15GB/s bandwidth is delivered to the PIM port. Moreover, preliminary logic synthesis confirms that our proposed models are implementable and realistic. Our current research is directed towards two directions: 1. Cycle accurate design of the PIM device and physical implementation of the LoB for power/area/temperature explorations. 2. Development of the required drivers and programming model to study the effect of the whole software stack on different parameters.

Acknowledgment

This work was supported in part by the European Union (EU) through the FP7 Programme, European Research Council, within the Multitherman Project under Grant 291125, and in part through the YINS Project, financed with a grant from the Swiss Nano-Tera.ch initiative, and evaluated by the Swiss National Science Foundation.

References

[1] D. P. Zhang, N. Jayasena, A. Lyashevsky *et al.*, "A new perspective on processing-in-memory architecture design," in *Proceedings of the ACM SIGPLAN Workshop on Memory Systems Performance and Correctness*, ser. MSPC '13. New York, NY, USA: ACM, 2013, pp. 7:1–7:3.

[2] D. Elliott, W. Snelgrove, and M. Stumm, "Computational RAM: A memory-SIMD hybrid and its application to DSP," in *Custom Integrated Circuits Conference, 1992., Proceedings of the IEEE 1992*, May 1992, pp. 30.6.1–30.6.4.

[3] D. Patterson, T. Anderson, N. Cardwell *et al.*, "A case for intelligent RAM," *Micro, IEEE*, vol. 17, no. 2, pp. 34–44, Mar 1997.

[4] D. Patterson, K. Asanovic, A. Brown *et al.*, "Intelligent RAM (IRAM): the industrial setting, applications, and architectures," in *Computer Design: VLSI in Computers and Processors, 1997. ICCD '97. Proceedings., 1997 IEEE International Conference on*, Oct 1997, pp. 2–7.

[5] F. Hamzaoglu, U. Arslan, N. Bisnik *et al.*, "13.1 a 1Gb 2GHz embedded DRAM in 22nm tri-gate CMOS technology," in *Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2014 IEEE International*, Feb 2014, pp. 230–231.

[6] *Hybrid Memory Cube Specification 1.1*, Hybrid Memory Cube Consortium Std., 2014.

[7] R. Balasubramonian, J. Chang, T. Manning *et al.*, "Near-data processing: Insights from a MICRO-46 workshop," *Micro, IEEE*, vol. 34, no. 4, pp. 36–42, July 2014.

[8] Q. Zhu, B. Akin, H. Sumbul *et al.*, "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.

[9] D. Zhang, N. Jayasena, A. Lyashevsky *et al.*, "TOP-PIM: Throughput-oriented programmable processing in memory," in *Proceedings of the 23rd International Symposium on High-performance Parallel and Distributed Computing*, ser. HPDC '14. New York, NY, USA: ACM, 2014, pp. 85–98.

[10] P. Dlugosch, D. Brown, P. Glendenning *et al.*, "An efficient and scalable semiconductor architecture for parallel automata processing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 99, no. PrePrints, p. 1, 2014.

[11] A. Farmahini-Farahani, J. Ahn, K. Compton, and N. Kim, "DRAMA: An architecture for accelerated processing near memory," *Computer Architecture Letters*, vol. PP, no. 99, pp. 1–1, 2014.

[12] J. Jeddeloh and B. Keeth, "Hybrid memory cube new DRAM architecture increases density and performance," in *VLSI Technology (VLSIT), 2012 Symposium on*, June 2012, pp. 87–88.

[13] D. U. Lee, K. W. Kim, K. W. Kim *et al.*, "25.2 A 1.2V 8Gb 8-channel 128GB/s high-bandwidth memory (HBM) stacked DRAM with effective microbump i/o test methods using 29nm process and TSV," in *Solid-State Circuits Conference Digest of Technical Papers (ISSCC), 2014 IEEE International*, Feb 2014, pp. 432–433.

[14] S. H. Pugsley, J. Jestes, R. Balasubramonian *et al.*, "Comparing implementations of near-data computing with in-memory mapreduce workloads," *Micro, IEEE*, vol. 34, no. 4, pp. 44–52, July 2014.

[15] Y. Kang, W. Huang, S.-M. Yoo *et al.*, "FlexRAM: Toward an advanced intelligent memory system," in *Computer Design (ICCD), 2012 IEEE 30th International Conference on*, Sept 2012, pp. 5–14.

[16] G. H. Loh, N. Jayasena, M. Oskin *et al.*, "A processing in memory taxonomy and a case for studying fixed-function pim," in *Workshop on Near-Data Processing (WoNDP)*, Dec 2013.

[17] E. Azarkhish, I. Loi, and L. Benini, "A case for three-dimensional stacking of tightly coupled data memories over multi-core clusters using low-latency interconnects," *Computers Digital Techniques, IET*, vol. 7, no. 5, pp. 191–199, September 2013.

[18] E. Azarkhish, D. Rossi, I. Loi, and L. Benini, "A modular shared L2 memory design for 3-D integration," *Very Large Scale Integration (VLSI) Systems, IEEE Transactions on*, vol. PP, no. 99, pp. 1–1, 2014.

[19] *AMBA AXI Protocol Specification V2.0*, ARM Holdings plc Std., 2010.

[20] *Double Data Rate (DDR) SDRAM*, JEDEC Std. JESD79F, 2005.

[21] G.Ramesh, V. Kumar, and K. Reddy, "Asynchronous FIFO design with gray code pointer for high speed AMBA AHB compliant memory controller," *IOSR Journal of VLSI and Signal Processing (IOSR-JVSP)*, vol. 1, pp. 32–37, Nov 2012.

[22] C. Weis, I. Loi *et al.*, "An energy efficient DRAM subsystem for 3D integrated SoCs," in *Design, Automation Test in Europe Conference Exhibition (DATE), 2012*, March 2012, pp. 1138–1141.

[23] P. M. Kogge and D. R. Resnick, "Yearly update: Exascale projections for 2013," Sandia National Laboratoris, Tech. Rep. SAND2013-9229, Oct. 2013.

[24] P. Rosenfeld, "Performance exploration of the hybrid memory cube," Ph.D. dissertation, univ. of Maryland, 2014.

[25] G. Kim, J. Kim, J. H. Ahn, and J. Kim, "Memory-centric system interconnect design with hybrid memory cubes," in *Parallel Architectures and Compilation Techniques (PACT), 2013 22nd International Conference on*, Sept 2013, pp. 145–155.

[26] N. Binkert *et al.*, "The gem5 simulator," *SIGARCH Comput. Archit. News*, vol. 39, no. 2, pp. 1–7, Aug. 2011.

[27] C. Bienia and K. Li, "PARSEC 2.0: A new benchmark suite for chip-multiprocessors," in *Proceedings of the 5th Annual Workshop on Modeling, Benchmarking and Simulation*, June 2009.