A Logic-base Interconnect for Supporting Near Memory Computation in the Hybrid Memory Cube

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*—Hybrid Memory Cube (HMC) has promised to improve bandwidth, power consumption, and density for next-generation main memory systems. In addition, 3D integration gives a "second shot" for revisiting near memory computation to fill the gap between the processors and memories. In this paper, we study the concept of "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. We take the first step towards SMC by designing a high bandwidth, low latency, and AXI-4.0 compatible interconnect optimized 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 trace-based cycle accurate simulation demonstrate that this interconnect can easily meet the demands of current and future HMC instances (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). 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. Lastly, a closed-loop co-simulation environment based on gem5 and Modelsim is under development to analyze the feedback effect between software and hardware.**

I. Introduction

The "memory wall problem", or the speed and bandwidth disparity between processors and memory, has been a concern for the last thirty years [1]. Many researchers, since the early nineties [2], have looked into the possibility to migrate some part of computation closer to the memory systems, to solve this issue. 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 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 achieving 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 several major semiconductor companies introduced the Hybrid Memory Cube (HMC) [6].

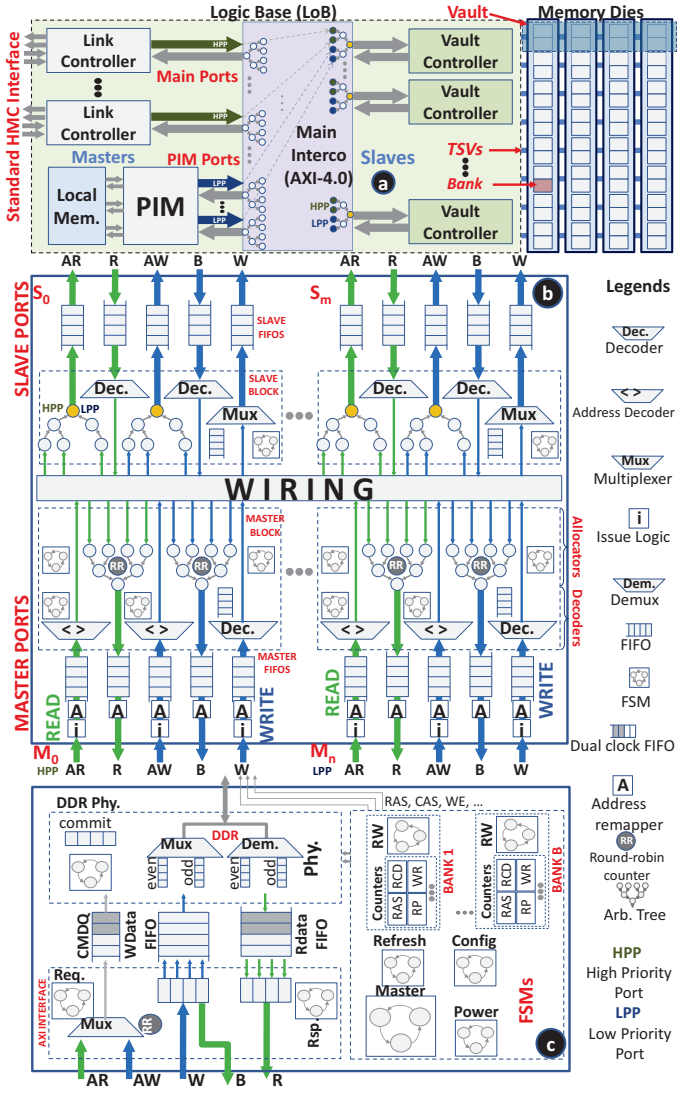

Fig. 1. a) Overview of the Smart Memory Cube, b) proposed AXI 4.0 based logarithmic interconnect for SMC, c) schematic view of a Vault Controller

In this paper, we leverage the recent technology breakthrough represented by the HMC to revisit the possibility of near memory computation 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's IO interface specification, featuring 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 trace-based simulation 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. Lastly, to tackle the feedback-effect problem of trace-driven simulations a closed-loop co-simulation environment based on gem5 and Modelsim has been developed. To speed-up simulation, fast forwarding and dynamic switching techniques of gem5 are utilized.

II. Related Works

As stated earlier, in-memory processing is a possible solution to the memory wall problem. Researches 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 two examples of the efforts in this area. 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]. In [8] throughput oriented computing using programmable GPU units for 3D stacked memories has been studied. In DRAMA [9] 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 [10] and the High Bandwidth Memory (HBM) [11], 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][12] or DRAM dies [13], or it can be added as a new die in the stack [14]. 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. 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 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 [15], 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. Memory-mapped communication with these devices occurs through low-level drivers, and offloading the computation kernels and job dispatching are managed through the programming model [16]. 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. 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 the main interconnect based on the ultra low-latency "logarithmic interconnect" [17] (originally designed for L1/L2 contexts), and modified it to support high bandwidth communication based on AMBA AXI 4.0 [18], the most widely used standard for communication in system-on-chips. It is worth mentioning here that our proposed interconnect can easily meet the demands for 32 masters/slaves, while for cardinalities beyond this number, a hierarchy of pipelined logarithmic trees or a memory-centric network connecting multiple cubes [19] can be adopted. 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.1a 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*. 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 identifying the destination port (inside the "Master Blocks"), WRITE transactions will be sent to the arbitration tree in the intended "Slave Block". There, among the master ports and separately on the PIM ports fair round-robin arbitration is performed [17]. However, in the last stage of the network a fixed-priority arbitration scheme ensures higher priority (HPP) for the main ports and lower priority for the PIM ports (LPP). This hierarchical arbitration guarantees that 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. 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", and there, they will wait for arbitration with other responses destined to the same master. The arbitration performed inside this network is single-cycle, and the whole interconnect has been developed as a synthesizable and parametric RTL model.

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.1b). The first stage in this channel controller is a round-robin arbiter choosing requests from one of the AXI AW and AR channels to issue into the Command Queue (CMDQ). 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 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 the signalling at the DRAM bus follows 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. Since data widths and burst sizes of the AXI interconnect and the DRAM devices do not necessarily match, "AXI Interface" in the vault controllers performs burst size conversion, as well. In addition, the AXI interconnect and the vault controllers work in separate clock domains, therefore, Command Queues (CMDQ) and RData FIFOs have been designed based on asynchronous dual-clock FIFOs to ensure data integrity. The DRAM device models have been adopted from [21], and 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 increasing the number of master ports of the main interconnect and attaching the PIM device to them (See Fig.1a).

B. Calibrating The Model

In the Exascale projections for 2013 [22] for a prototype of HMC manufactured in 2011, 4 Memory dies and 1 Logic Base (LoB) have been reported, with 16 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 lower LoB. This aggregates in total 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 [22] an aggregate off-chip bandwidth of 160GB/s for WRITE and READ can be delivered. In the first paper on HMC [10] 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 [23].

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) [10]. This is because HMC has been mainly designed for 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 [23][12] (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 [22]. 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 specifications [10], nevertheless, in one experiment we will show that our model can be scaled easily 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 [19] 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 [23]. 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.1a. 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 the 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 the most straightforward solution is to increase the flit-size of the main interconnect from 128 to 256 bits. Other alternatives include increasing the clock frequency, which can cause power related issues, and increasing its number of master ports in the interconnect, which can increase arbitration latency. Increasing the flit size effectively improves aggregate bandwidth to 256GB/s at the master side, and in section IV it will be shown that the total area of the interconnect is negligible compared to the total available area in the LoB. 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 for the AXI interconnect, less than $0.2mm^2$. Summing to a total of about $10.1mm^2$,

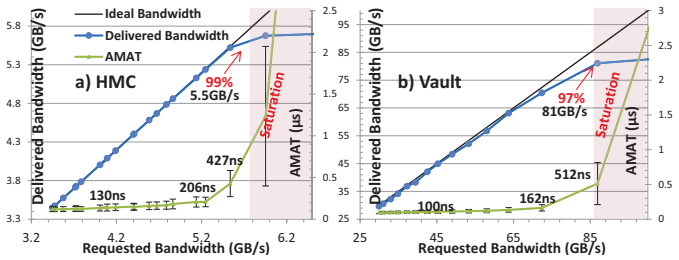

Fig. 2. Delivered bandwidth of a) one vault only and b) the baseline HMC.

which is much less than the DRAM area reported for HMC 2011 ($68mm^2$) [10]. Two types of traces have been exploited for performance analysis: Synthetic random traffic generated in ModelSim, and traces gathered in Gem5 [24] running a full-system simulation of eight x86 CPUs with Linux 2.6.22.9 kernel executing PARSEC V2.1 benchmark suite [25]. It has been previously observed in [23] that current multi-threaded workloads cannot easily utilize the huge bandwidth provided by HMC, mainly due to the overheads of the cache-coherence mechanisms. To increase the bandwidth demand [23] proposes ideal-caches which always result in a hit. In this work, we have compressed the time-scale of the traces to adjust their bandwidth demand to HMC. A more sophisticated closed-loop traffic generation mechanism composed of ARM64 and GPGPU cores in gem5 simulation environment in under development and is briefly described in Section V.

A. HMC Exploration

First we adjusted the size of all buffers along with the MoT parameter to achieve the maximum bandwidth requirement of HMC with a reasonable access latency. 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. As this figure shows, 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 bounded and less than 500ns, which is about 7X of the zero load latency for READ (Zero load latency: 76ns for READ and 13ns for WRITE).

To demonstrate this fact better, different traffics 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

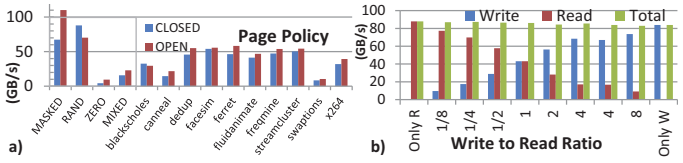

Fig. 3. a) Effect of page policy on delivered bandwidth, b) effect of R/W ratio in RAND traffic on total bandwidth

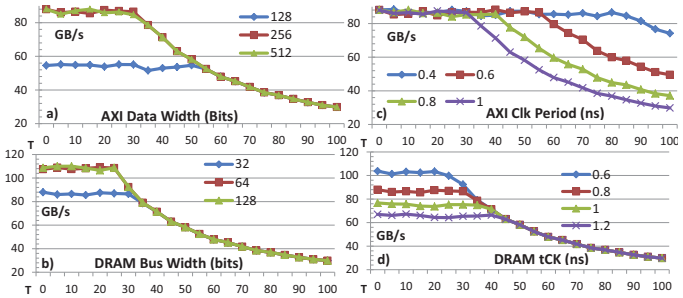

Fig. 4. Effect of a) AXI data width, b) DRAM bus width, c) AXI clock period, and d) DRAM t_{CK} on READ bandwidth. Random traffic with inter-arrival time Random[0,T] has been applied (Closed page policy)

and not the interconnect. Lastly, in a separate experiment on the 2015 projection of HMC, a maximum READ bandwidth of 172GB/s for RAND traffic was obtained (Policy: Closed), which is beyond its requirement (160GB/s READ bandwidth).

Next, we analysed the effect of different architectural parameters on the delivered bandwidth. Fig.4.a illustrates delivered READ bandwidth when the flit size of the AXI interconnect has been changed from 128 to 512. Packet inter-arrival time of the RAND traffic has been changed based on Random[0,T], with T being on the X-axis. This preliminary results shows that, 128-bit flits are not enough, while, 512-bit flits do not seem necessary. 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 20% for random traffic would be achievable with 64 data TSVs per each vault. Lastly, Fig.4.c,d show that decreasing the AXI interconnect cannot improve maximum delivered bandwidth while reducing t_{CK} can. This suggests that the main bottleneck of the system is the DRAM and not the AXI interconnect. To confirm this result we have scaled down all the timing parameters of the DRAM devices from their default values by the scale factors illustrated in Fig.5.a. A set of PARSEC and random traffics have 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.5.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.6 illustrates the delivered bandwidth over time for 4 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 and streamcluster 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

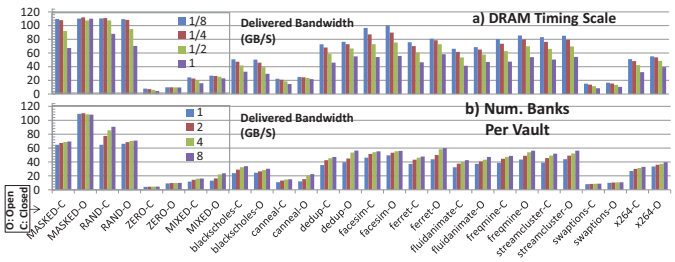

Fig. 5. Effect of a) scaling down the timing parameters of DRAM and b) effect of number of banks per vault on bandwidth

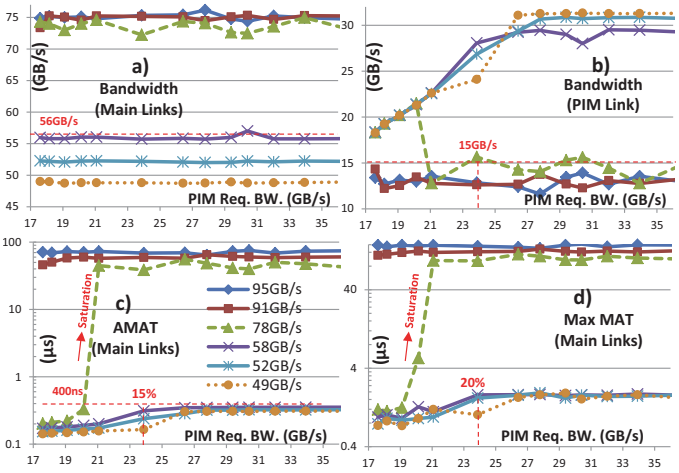

Fig. 7. a) Effect of PIM traffic on the main, b) Delivered bandwidth to PIM as a function of its requested bandwidth, Increase in c) average and d) maximum memory access time caused by PIM.

addressing mechanism (See Fig.1a) is required to tune the performance of the cube based on the running application.

B. Handling PIM Traffic

Assuming a generic PIM device is connected to the main interconnect through one additional link, we aim to find the upper bound on the bandwidth that it can request without disrupting the main traffic. As a worst case scenario, we assume that the bandwidth demand of the PIM and the main links are independent from each other, and increasing one does not decrease the other. Moreover, we assume that the PIM device can process received data instantaneously. On the four main links as well as the PIM link we inject RAND transactions with various bandwidth profiles. Fig.7a,b show total delivered bandwidth on the main and the PIM links, and Fig.7c,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. Since for typical general purpose servers, the traffic on the main links mostly consists of cache refill requests, 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,

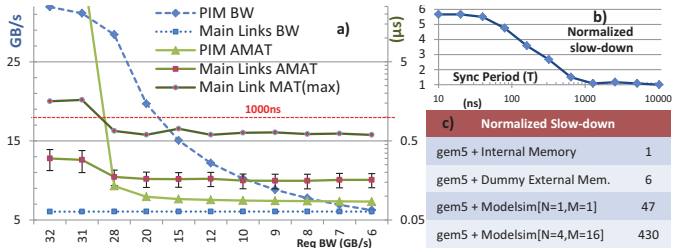

Fig. 8. a) Effect of requested bandwidth by ideal PIM on x264, b) effect of synchronization period (T) on simulation slow down (closed-loop simulation), c) normalized simulation slow-down of different closed-loop configurations

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.8.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 similar bandwidth delivered to the PIM device, and less than 1% increase in total execution time. Next section, discusses about issues faced simulation and proposes solutions to them.

V. Simulation Challenges

While cycle accurate modelling provides highest possible accuracy, some issues need to be addressed to ensure that the promising results obtained in the previous sections would remain valid in a full-system context. First, trace-based traffic generation neglects the feedback effect between the processors and the memory system, and may lead to incorrect results in analysis of the whole system. Random and trace-based traffic are suitable for fast and easy design space exploration and studying the effect of the low level architectural parameters. However, for full system simulation other alternatives should be adopted. For this purpose, a closed loop simulation wrapper has been developed which allows data communication between gem5 and Modelsim through Linux pipes and periodically synchronizes time and data between them (with period T). This method can reflect the dynamic effects of the operating system, APIs, drivers, cache coherence, and consistency protocols on the overall performance. Yet two important issues should be considered: Periodic synchronization introduces an error in memory access time (MAT). Since transactions must cross the gem5-Modelsim interface twice, the minimum reported MAT becomes $2 \times T$. Moreover, reducing T to increase accuracy has a negative effect on simulation time. Fig.8.b illustrates this effect where T has been swept from 10ns to 10μS. A slow down of over 5X is observable when T is very small. This situation is nevertheless tolerable in comparison with the slow-down caused by Modelsim itself, degrading gem5's performance down to cycle accurate simulation. Fig.8.c compares simulation slowdown in four cases. First, baseline gem5 simulation in timing-accurate mode with a "SimpleMemory" model. Second, a dummy external memory connected to gem5 using the developed co-simulation wrapper and synchronization period has been set to 10ns. Third, a cycle-accurate memory with 1 AXI port and 1 vault connected to gem5, and last, the baseline cycle accurate HMC model connected to gem5. It can be seen that the slow down caused by the synchronization method is about 6X, while Modelsim running the baseline HMC configuration slows down the simulation by another 70X. To bypass this issue, we rely on gem5's check-pointing, restoring, and fast-forwarding abilities, to skip over the less-interesting time periods and only simulate the region-of-interest accurately. For this purpose we utilize gem5's internal memory model for keeping data and functional behaviour, and Modelsim only for correction of gem5's timing. This way, whenever accuracy is required, simulation mode is switched from "atomic" to timing accurate, and simple memory model of gem5 switches to the

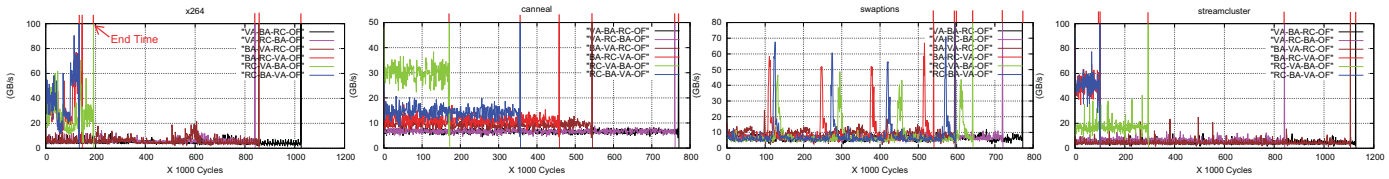

Fig. 6. Effect of address mapping on delivered bandwidth to PARSEC benchmarks, measured in epochs of 1000ns

cycle-accurate model. We are currently in the process of setting up a complete system configuration including multi-core CPU and GP-GPU, with memory-intensive benchmarks to assess the performance of our interconnect design in a full-system context.

VI. Conclusions and Future Works

In this paper, we presented the concept of 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 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. Preliminary logic synthesis confirms that our proposed models are implementable and realistic. Moreover, since CPU traffic does not utilize HMC's bandwidth completely [23], we plan to use gem5-gpu [26] which integrates GPGPU-Sim [27] with gem5. gem5-gpu allows for seamless closed-loop co-simulation with our models and is under active development. We are also looking into development of higher level models calibrated based on cycle-accurate results to improve simulation time. The goal is to use statistical models to reflect timings of HMC for very long simulations, and switch to the cycle accurate model whenever error is beyond a certain threshold.

Acknowledgement

This work was supported, in parts, by EU FP7 ERC Project MULTITHERMAN (GA n. 291125); YINS RTD project (no. 20NA21_150939), evaluated by the Swiss NSF and funded by Nano-Tera.ch with Swiss Confederation financing; and Samsung Electronics Company.

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] 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.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

[14] 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.

[15] 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.

[16] D. P. Z. M. I. M. Chu, N. Jayasena, "High-level programming model abstractions for processing in memory," 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] *AMBA AXI Protocol Specification V2.0*, ARM Holdings plc Std., 2010.

[19] 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.

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

[21] 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.

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

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

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

[25] 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.

[26] J. Power, J. Hestness, M. Orr *et al.*, "gem5-gpu: A heterogeneous CPU-GPU simulator," *Computer Architecture Letters*, vol. PP, no. 99, pp. 1–1, 2014.

[27] A. Bakhoda, G. Yuan, W. Fung *et al.*, "Analyzing CUDA workloads using a detailed GPU simulator," in *Performance Analysis of Systems and Software, 2009. ISPASS 2009. IEEE International Symposium on*, April 2009, pp. 163–174.