Abstract—Historically, processor performance has increased at a much faster rate than that of main memory and up-coming NoC-based many-core architectures are further tightening the memory bottleneck. 3D integration based on TSV technology may provide a solution, as it enables stacking of multiple memory layers, with orders-of-magnitude increase in memory interface bandwidth, speed and energy efficiency. To fully exploit this potential, the architectural interface to vertically stacked memory must be streamlined. In this paper we present an efficient and flexible distributed memory interface for 3D-stacked DRAM. Our interface ensures ultra-low-latency access to the memory modules on top of each processing element (vertically local memory neighborhoods). Communication to these local modules do not travel through the NoC and takes full advantage of the lower latency of vertical interconnect, thus speeding up significantly the common case. The interface still supports a convenient global address space abstraction with high-latency remote access, due to the slower horizontal interconnect. Experimental results demonstrate significant bandwidth improvement that ranges from 1.44x to 7.40x as compared to the JEDEC standard, with peaks of 4.53GB/s for direct memory access, and 850MB/s for remote access through the NoC.

I. INTRODUCTION

FOR many years DRAM access latencies have not decreased at the same rate as microprocessor cycle times. Hence, relative memory access time (in CPU cycles) have increased from one generation to the next. This unavoidable bottleneck, often called the “memory wall”, has caused the development of complex and expensive memory hierarchies with extensive data replication, and it is now one of the main hurdles to the successful transition from multi-core to many-core computing platforms.

Memory access speed is not the only cause of the memory wall, which is also tied to memory-to-logic interfacing issues. DRAMs currently are developed using high-density NMOS process optimized to create high-quality capacitors and low-leakage transistors. On the other hand, logic chips are manufactured in high-speed CMOS processes optimized for low-leakage transistors. Stacking DRAM directly with silicon bonded with low-latency, high-bandwidth and very dense vertical interconnects [5]. 3D stacking also enables mixing heterogeneous process technologies such as high-speed CMOS with high-density DRAM. Stacking DRAM directly on top of a processor is a natural way to attack the memory bottleneck.

In this paper we consider a 3D-stacked DRAM on a multi-core logic die. On the horizontal logic plane cores are connected through a Network-on-chip [6], [7], which provides bandwidth in a scalable fashion, at the price of non-negligible latency, caused by protocol translation and packetization, as well as network traversal time. In this context, a vertically stacked memory system can be modeled with the abstraction of memory neighborhood: each physical processing element in a large many-core array has fast, large-bandwidth access to a vertical stack of memory banks on top of it. The processor can address vertical stacks on top of other processors, but corresponding memory transactions will have to be transported through the horizontal NoC. This implies a notion of distance: the cost (increased latency and decreased bandwidth) of a memory access sharply increases as we move to memory neighborhoods far away processors.

Figure 1 depicts a high-level view of a 3D integrated architecture and its memory neighborhoods. Note that stacked DRAM memory is still accessed using standard DRAM interfacing protocols. Hence protocol translation between processor memory accesses and DRAM transactions is still needed. The standard approach is to implement protocol translation within a memory controller connected to the NoC via a slave port. However, this is an extremely inefficient way to access a local memory neighborhood, as NoC protocol translation and packetization cost has to be paid at the master and slave interface of the NoC. Hence, we want the common case of accesses to the local memory neighborhood to be as fast as possible.

1Embedded DRAM processes have been developed, but they are characterized by lower-density DRAMs and higher cost, hence they have not succeeded in replacing dual-chip DRAM+Logic solution for mainstream applications [2], which is a major concern for bandwidth-hungry multicore architectures.

In the last few years, three-dimensional die-stacking has received a great deal of attention [3] [4]. 3D stacking enables the construction of circuits using multiple layers of active silicon bonded with low-latency, high-bandwidth and very dense vertical interconnects [5]. 3D stacking also enables mixing heterogeneous process technologies such as high-speed CMOS with high-density DRAM. Stacking DRAM directly on top of a processor is a natural way to attack the memory bottleneck.

In this paper we consider a 3D-stacked DRAM on a multi-core logic die. On the horizontal logic plane cores are connected through a Network-on-chip [6], [7], which provides bandwidth in a scalable fashion, at the price of non-negligible latency, caused by protocol translation and packetization, as well as network traversal time. In this context, a vertically stacked memory system can be modeled with the abstraction of memory neighborhood: each physical processing element in a large many-core array has fast, large-bandwidth access to a vertical stack of memory banks on top of it. The processor can address vertical stacks on top of other processors, but corresponding memory transactions will have to be transported through the horizontal NoC. This implies a notion of distance: the cost (increased latency and decreased bandwidth) of a memory access sharply increases as we move to memory neighborhoods far away processors.

Figure 1 depicts a high-level view of a 3D integrated architecture and its memory neighborhoods. Note that stacked DRAM memory is still accessed using standard DRAM interfacing protocols. Hence protocol translation between processor memory accesses and DRAM transactions is still needed. The standard approach is to implement protocol translation within a memory controller connected to the NoC via a slave port. However, this is an extremely inefficient way to access a local memory neighborhood, as NoC protocol translation and packetization cost has to be paid at the master and slave interface of the NoC. Hence, we want the common case of accesses to the local memory neighborhood to be as fast as

![Fig. 1. Target 3D hardware architecture](image-url)
possible, bypassing the NoC interface. For remote memory accesses request, data are routed through the NoC.

The main contribution of this paper is the development of a specialized 3D-DRAM controller featuring a very fast path to the local memory neighborhood and a standard NoC communication facility for accesses to remote memory neighborhoods. The controller is also capable of handling requests coming from remote processors and directing them to the local memory neighborhood, while arbitrating potential conflicts with the local traffic. We also present results on the hardware cost of our memory controller, and an accurate performance analysis for a system composed by four cores, a 2x2 NoC mesh and four 3D DRAM modules. Our results demonstrate that even for such a small and low-congestion system, memory access performance is greatly increased by our distributed, asymmetric memory controller with respect to a standard NoC-interfaced distributed memory controller.

II. RELATED WORK

Interconnect scaling has become one of the most crucial challenges in chip design, and is expected to get worse in the future. 3D integration and Network on Chip design methodologies are expected to overcome many of these challenges. NoCs have been suggested as a scalable communication fabric [8], [9]. 3D integration has been proposed in different ways (e.g. Tezzaron Semiconductor Corporation [3], IMEC, MIT Lincoln Labs, and IBM [4]) providing promising solutions to enable connectivity along the vertical direction.

In recent years, several authors have been exploring the benefits of memory and logic stacking. Significant achievements have been announced in the last few months, confirming the rapidly increasing industrial R&D effort in this area. In [10] a 8GB 3D DDR3 using TSVs to stack 4 DRAM dies is presented. The first die includes both DRAM core, R/W buffers and IO circuitry. Redundant TSVs with check and repair scheme, and power-noise reduction method are also presented. Read and write buses are independent, but row and column address are still multiplexed as in the conventional DRAM. As the author remarks, the DRAM module are simply added on each tiers, therefore this results in increased power and area waste due to duplication of circuit components.

3D bitcells in DRAM are presented in [11]. FaStack memory devices contain up to four layers of stacked DRAM on top of one control and interface layer, greatly increasing memory density up to 300% as compared to current commercial DRAM. Individual bitcell arrays are stacked in a 3D fashion, therefore reducing length of internal buses, wordlines and bitlines, which in turn reduces the access latency of the memory. The I/O capacitance is \(25 fF\), or about 100x less than off-chip memory. Latency is 8-12ns, which is near SRAM speed.

A few prior studies have already started to investigate the potential of 3D-stacked memory. [12] and [13] independently researched the performance impact of placing the main memory system on top of the processor using 3D fabrication technologies. These studies report impressive performance speedups for 3D-stacked memories (92% performance gain in [12] and 65% for [13]).

An early approach in architectural exploration for 3D DRAM stacking on a logic die is presented in [14]. The authors modeled a web server as Chip Multi Processor (CMP): the logic has been placed in the first die, and the memory (DRAM) has been stacked on 4-8 dies on top of the logic. Logic layers include up to 8 cores, wide shared bus and memory controller. The connections between dies are assumed to be provided by Through Silicon Vias (TSVs). Overall power improvements is 2-3x better than multi-core architecture without 3D stacking technology. The memory interface also is modified according to the fact that there is no need to use narrow interfaces (pin-out) and address multiplexing with the familiar two phases CAS/RAS, therefore the logic for latching and multiplexing address/data can be removed (RLDRAM and NetDRAM like). However this solution lacks in scalability, due to the usage of shared bus. In addition, since all the blocks have been modeled at high level and at a very coarse-grained abstraction, the results are not validated with accurate and technology-calibrated models.

In [15] the authors present a 3D stacked memory architecture for CMP. Four DRAM layers stacked on top of logic layer that include 4 Cores (Intel QuadCore Penryn 45nm) and L2 cache. They presented a 3D internal DRAM architecture (based on true-3D memory organization proposed by Tezzaron) to better exploit the possibilities enabled by 3D technologies. Whits these changes up to 280% speedup is achievable over the baseline 2D system with off-chip DRAMs.

Physical connectivity to DRAM is only a facet of the problem. Memory controllers manage the architectural and circuit level interface between processors and DRAM. State-of-the art memory controllers [16] are still designed for narrow off-chip interfaces. They are quite complex component and they deploy many complex features to maximize exploitable interface bandwidth. The front end includes a multiprotocol arbitration interface and I/O queues with reordering capabilities, to improve power consumption and access latency.

The idea of distributing memory controllers to manage multiple DRAM channels is starting to make inroads, as the number of processors per chip increases. In [17] a novel scheme for memory controller placement in many-core CMP is presented. Memory controllers are attached on 2D mesh NoC nodes, therefore achievable bandwidth and numbers of on-chip memory controllers are constrained due to pin limitations. To our knowledge our work is the first to present a distributed memory controller architecture for vertically stacked memory.

III. 3D DRAM MEMORY INTERFACE

In this section we discuss the changes in the DRAM interface when moving from 2D to 3D domain. Figure 2 illustrates the functional block diagram for a conventional 512MB DDR SDRAM. As shown, the data bus is both bidirectional and narrow. The former implies three-state buffers both in the DRAM and in the memory controller, while the latter implies low throughput. Moreover, since conventional DRAM is packaged separately from the processor and accessed through IO pad pins and wires on a PCB, synchronization and noise are the main concern. To tackle synchronization issues, DRAM is equipped with data strobe circuitry and DLL, since the propagation delay between memory controller and memory module is unknown at design time. Signal integrity is an important issue that limits to reach higher frequencies on the IOs. To tackle it, the output impedance of the drivers must be matched with the characteristic impedance of the line.

In the 3D context, as mentioned in Section I, the limitation on IOs count is no longer an issue. Owning to the low capacitance of the connection achieved by stacking a memory die on
top of a processor die, power consumption is reduced about 24\(\mu\)W per pin compared to 30-40 mW per pin for DDR [18]. Area cost is well limited, and since TSV process allows fine pitch interconnects, up to 10K TSVs can be fitted in a space of 1\(mm^2\) [19], compared to 2K ultrafine microbumps [20](die to package), and only 11 IO’s microbumps (BGA 300\(\mu\)m pitch).

As shown in Figure 2 b, theoretically, the data bus width can range from a single word, to the half width of the row. Moreover, there is no need to have a single bidirectional bus, thus write and read data are routed on separated buses. This implies that output drivers are simple buffers (three-state buffers are 3 times bigger in the same condition load) and the driven load is simply the capacitance of a TSV. The parasitic capacitance and resistance for 3D vias are negligible compared to off-chip interconnects, which are 3 order of magnitude bigger than the typical on-chip interconnect based on TSVs [19] [14].

The increasing in data bus width results in the explosion of the achievable bandwidth. Moreover the number of columns for each row is going to decrease, therefore the column address decoder becomes smaller and faster, and the CAS latency decreases. On the other hand this solution requires additional buffering resources to load the data before the movement into the prefetch buffer. Figure 3 shows the timing diagram for a write request in two scenarios: on top, the JEDEC compliant model and on the bottom the 3D compliant model. During the first phase, the row and bank are activated. In the second step the column is selected while the data is written in the internal write buffer at rate of 2 word per cycle (DDR), and then committed in the prefetch buffer. Finally if the row must be deactivated, the precharge command writes back the data in the bank row, and then bank is ready for a new request in a different row. The read request is similar, but during the second step, there is latency of several clock cycle before the data reaches the IO. This latency is called CAS Latency (CL) and it depends on the complexity of the column address decoder. This parameter can be tuned during the setup stage, accordingly with the target frequency. In the 3D version, Activate and Precharge show the same latency since they depend on the internal memory structure, but the R/W phase is shorter due to improvement on the IO interface and lean column decoder architecture.

In our 3D implementation, the CL decreases with the number of columns. Roughly speaking, in the conventional DRAM depicted in Figure 2 a the column address decoder manages 2048 columns while in our implementation (Figure 2 b) the columns are 256.

In the following subsection we present the main blocks of the proposed architecture.
A. Processing Element Interface

We supposed a simple processing element interface with independent buses for read and write and with a Stall/GO flow control policy. We also assumed that the width of these buses are 256 bit (8 words) and that all the memory requests come from L1 cache misses. Since our bus provides up to 8 words, during the read or write requests, the processing element provides a data mask to inform the network interface or the DRAM controller that only few words must be read/written. We also suppose that the processing element generates outstanding transactions, therefore we developed both the memory controller and the NoC interfaces keeping in mind these important features.

B. NoC

Each processing element is connected to a 4 way custom crossbar, that allows to route incoming traffic both through the NoC (master and slave NoC port) for global request and directly to the memory controller for local accesses. Similarly, memory responses are managed in a comparable way. The custom crossbar includes a programmable built-in arbiter which supports 3 different priority policies: fixed priority, round robin and TDMA. The arbiter captures the requests and generates the grant signals both for the requests and for the responses, in order to reconfigure the crossbar in the right way.

A translation protocol layer is needed to attach the NoC to the crossbar, since the Network on chip is not capable of transferring data bigger than one word. Global write requests are serialized as burst access with the OCP protocol, and then fed to the Network Interface (NI) initiator. Packets are collected by the NI target, and data are sent to the deserializer. Once that all the burstiness data have been collected, the deserializer forward the request for exclusive access to the memory controller (through the destination crossbar). For read operations, the request first must reach the memory controller, and then read data are serialized - deserialized through the NoC and finally delivered to the related processing element.

C. 3D DDR Controller

The memory controller handles one of the toughest problems in complex multicore design: sustain the on-chip data bandwidth requirements of a high performance communication infrastructure. Starting from this idea, we developed a flexible memory controller that can be tuned to achieve the desired performance. The memory controller is composed by three main blocks:

- the front end includes the buffering stage for the incoming and outgoing PE signals and queues for asynchronous synchronization. The minimum depth for these buffers is 4 in case of asynchronous interface, and 2 in case of synchronous system.
- the back end is composed by the DRAM command encoder and the Status Register (SR) that tracks open banks/rows and also includes counters for DRAM refresh.
- the physical interface, generates the data strobes for write operation (through a DLL or delay chain), forwards the data to the DRAM at double data rate and collects the incoming chunks of the read data.

Figure 6 shows the architecture of the memory controller at block level. Request and response channels are independent and the implemented flow control policy is the STALL/GO. The command encoder takes the command/address from the request buffer and generates the proper signals to drive the DRAM, with the JEDEC standard or 3D interface compatibility. Since the propagation time along the TSVs is negligible as compared to delays on 2D routing, the role of the physical interface is merely to align data strobe ad data for safe data sampling in the memory controller buffers.

V. EXPERIMENTAL RESULTS

This section provides the experimental results for the proposed memory interface and the bandwidth/latency improvement that we achieved. We first quantify the cost for remote requests across the NoC. Then we quantify the local/remote latency cost of the whole system when sweeping the frequency in two corner case: with and without memory contentions on a synchronous and asynchronous system. Finally we present the area and power cost and the impact on the whole system.
A. Timing Analysis

In a remote access, packets travel within the NoC until they reach the destination (memory controller), therefore the NoC crossing latency is strictly correlated on the topology parameters: flit width, burst length, number of hops and number of repeaters. This latency (not includes the memory access time) can be expressed in an analytic form as follow:

\[
\text{NoC Lat}_{w} = N_H + N_R + BL \cdot F_P(fw) + F_H(fw) + 2
\]

(1)

\[
\text{NoC Lat}_r = 2 \cdot N_H + 2 \cdot N_R + BL \cdot F_P(fw) + 2 \cdot F_H(fw) + 4
\]

(2)

\(N_H\) and \(N_R\) mean respectively the number of NoC hops and the number of repeaters between source and destination. \(BL\) stands for burst length (number of words to write or read), and \(F_P(fw)\) and \(F_H(fw)\) represent respectively the number of flits needed to encode respectively payload and header which are a function of the flit width of the NoC. The constant additive terms (2 and 4) represent the latency due serializer and deserializer pipeline stages. In equation 2 there is a round-trip path since the request first must reach the memory controller and then data must come back to the processing element. For that reason the read latency shows double latency for several terms of the Formula.

In case of low traffic injection (no memory conflicts) the overall latency for a remote request can be expressed as the sum of two terms: the NoC crossing latency and the memory access time, that includes the latency for buffering and the access RAS/CAS sequence. Latency for local request is simply the memory access time. Starting from the assumptions that we discussed in section III and IV, we simulated several configurations, sweeping both the memory and the NoC clock frequency (synchronous system). Figure 7 shows the latency trend for a synchronous memory system (without conflicts) in four cases: best and worst case for both local and remote accesses. The best case means that all the requests are issued within the same open rows (this features is denoted as “locality”), therefore, there is no need to precharge and activate a different row whenever a command is issued. On the other hand, the worst case is given by accessing on random rows, that results in a sequence of activations and precharges. It is clear that when there is no data locality the latencies increases quickly. To improve the system efficiency we increase the clock frequency of the whole system (as shown in Figure 7) from 133 MHz to 1 GHz. The remote requests gain benefits from the overclocking, since the NoC latency is proportional to the clock speed. The memory access time on the other hand is locked on the timing parameters of the DRAM, just overclocking influences only the rate at which the data is read/written in the DRAM prefetch buffer. Activation and precharge latencies remain unchanged. We achieving a speed-up (as compared to the JEDEC standard) that ranges from 7.6\(\times\) for local write request (with data locality) to 3.3\(\times\) for local read request (without data locality).

Moreover, we swept the memory system clock frequency, keeping fixed the NoC frequency at 1GHz. Experimental results are shown in Figure 8. The speed-up ranges from 6.4\(\times\) for local write request (with data locality), to 2.3\(\times\) for remote request (with data locality). As shown in the figure, the latencies for remote accesses saturate for frequencies above 333MHz (NoC is dominant), while local requests latencies scale well in the whole frequency span.

The memory system bandwidth for the synchronous memory system ranges from 4.53GB/s (7.6\(\times\) better than the JEDEC DDR compliant case) to 848MB/s (which leads an improvement of 3.3\(\times\)). The bandwidth improvement in the asynchronous case ranges from 4.53GB/s (5\(\times\) better than the JEDEC DDR compliant case) to 478MB/s. (1.6\(\times\)).

Finally we evaluated the bandwidth performance when the system is under high traffic injection, on the same target memory. As discussed in section IV, the arbiter supports three priority policies: fixed priority, round robin and TDMA. We simulated the system under different configuration and we extracted, the relative bandwidth achievable for both the local and remote accesses. Figure 9 shows the relative bandwidth for remote accesses, expressed in %. Round robin arbitration shows the best performance, while the TDMA is a little worse. Fixed priority, with priority set to the local accesses show the worst bandwidth result as depicted in the Figure.

B. Physical Analysis

We synthesized the whole asynchronous platform (Mesh 2x2 NoC with 4 Memory controllers and 4 custom interfaces for low-latency local accesses) with the TSMC 65nm technology library (general purpose process). The front-end flow (Multi Vth) has been performed with Synopsys Design Compiler in topographical mode, while the back end with Cadence SoC Encounter.

Figure 10 (a) shows the silicon and power cost for the proposed asynchronous system. The total silicon cost is about
127K equivalent gates (NAND2) and includes the cost of NoC, data serializer and deserializer, custom crossbar-arbiter and memory controllers (PEs and DRAMs are not included in this chart). The custom crossbars (CC) require only the 5% of the total area (each is around 1.25%) while the expensive components are the memory controller, which consumes about 43% of the total area (each is around 11%). This high cost is mainly due to front end buffering (wide buses) and requires about 40% of the total area. The four TSV channels, each made by 89 TSVs (two independent buses for read and write) consumes about 8% of the silicon cost.

Figure 10 (b) illustrates the relative power cost of the proposed asynchronous architecture. The total power consumption is 33mW at 1GHz for both NoC and DRAM controllers. The custom crossbar shows a negligible power consumption (about 1.25% each) while the NoC is the most power hungry block 51% due the large amount of switching activity. On the other hand, each memory controller consume only 2.5% since the clock is gated due low switching activity on the front end side. Similarly, the power consumption on the TSV drivers is around 1.25% for each link.

VI. CONCLUSIONS AND FUTURE WORK

In this work, we have presented a specialized 3D-DRAM controller featuring a very fast path to the local memory neighborhood and a standard NoC communication facility for accesses to remote memory neighborhoods. We have shown hardware cost of our memory controller, and an accurate performance analysis for a system composed by four cores, a 2x2 NoC mesh and four 3D DRAM modules. Our results demonstrate that even for such a small and low-congestion system, memory access performance is greatly increased by our distributed, asymmetric memory controller with respect to a standard NoC-interfaced distributed memory controller. We have demonstrate significant bandwidth improvement that ranges from 1.44x to 7.40x as compared to the JEDEC standard, with peaks of 4.53GB/s for direct memory access, and 850MB/s for remote access through the NoC. Finally we have shown feasibility, and we quantified silicon and power cost for both the DRAM controller and the low latency interface for local accesses.

Research on 3D memory stacking is just now beginning, and much work remains to be done. Among the areas requiring more attention, we plan on focusing on reordering buffer that increases the memory bandwidth therefore increasing the overall energy efficiency.