Data-Centric Computing Frontiers: A Survey On Processing-In-Memory

Patrick Siegl
TU Braunschweig
Abteilung Technische
Informatik, E.I.S.
Mühlenpfordtstr. 23
38106 Braunschweig,
Germany
siegl@c3e.cs.tu-bs.de

Rainer Buchty
TU Braunschweig
Abteilung Technische
Informatik, E.I.S.
Mühlenpfordtstr. 23
38106 Braunschweig,
Germany
buchty@c3e.cs.tu-bs.de

Mladen Berekovic
TU Braunschweig
Abteilung Technische
Informatik, E.I.S.
Mühlenpfordtstr. 23
38106 Braunschweig,
Germany
berekovic@c3e.cs.tu-bs.de

ABSTRACT

A major shift from compute-centric to data-centric computing systems can be perceived, as novel *big data* workloads like cognitive computing and machine learning strongly enforce embarrassingly parallel and highly efficient processor architectures. With Moore's law having surrendered, innovative architectural concepts as well as technologies are urgently required, to enable a path for tackling exascale and beyond – even though current computing systems face the inevitable instruction-level parallelism, power, memory, and bandwidth walls.

As part of any computing system, the general perception of memories depicts unreliability, power hungriness and slowness, resulting in a future prospective bottleneck. The latter being an outcome of a pin limitation derived by packaging constraints, an unexploited tremendous row bandwidth is determinable, which off-chip diminishes to a bare minimum. Building upon a shift towards data-centric computing systems, the *near-memory processing* concept seems to be most promising, since power efficiency and computing performance increase by co-locating tasks on bandwidth-rich in-memory processing units, whereas data motion mitigates by the avoidance of entire memory hierarchies. By considering the umbrella of *near-data processing* as the urgent required breakthrough for future computing systems, this survey presents its derivations with a special emphasis on *Processing-In-Memory* (PIM), highlighting historical achievements in technology as well as architecture while depicting its advantages and obstacles.

CCS Concepts

•**Computer systems organization** → **Multiple instruction, multiple data; Single instruction, multiple data; Multicore architectures;** *Other architectures;*

MEMSYS 2016 October 3–6, 2016, Washington, DC, USA
© 2016 ACM. ISBN 978-1-4503-4305-3. . . $15.00
DOI: http://dx.doi.org/10.1145/2989081.2989087

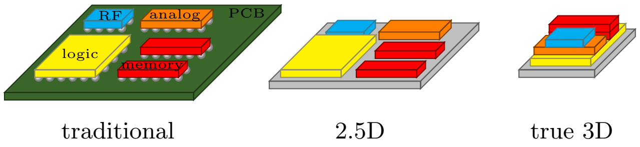
Keywords

memory wall; bandwidth wall; processing-in-memory; near-data processing

1. EVOLUTION IN MEMORY ARCHITECTURES

Ever since Dennard invented the single-transistor DRAM cell (*Field-effect Transistor Memory*) in 1966 (patented 1968) [23], a slow but steady progress occurred regarding the resulting DRAM memories. Memory continually evolved in bandwidth speed and power efficiency, owing to decreased voltages, improved process technology, higher chip- and bus-clock frequencies as well as increased prefetching and data rates. With the occurrence of modern mobile computing devices like smartphones and tablets, novel memory concepts are urgently required, mitigating bandwidth pressure and decreasing latencies as well as power consumption to significantly achieve even smaller form-factors.

An overall trend for memory architectures is determinable, depicting that innovational concepts typically appear in the graphics domain first: concepts like vector processing, parallel pipelines as well as scrambling or interleaving of data across multiple memory chips or banks, accelerating the access rate, and avoiding the *bandwidth wall* were first utilized by graphical experts [105]. While the frame-buffer was still the limiting factor for scaling the graphical processing performance and with techniques like bank-interleaving being no longer sufficient, the first PIM-like architecture *PIXEL-planes* was invented to overcome these constraints [35]. At the same time as Burger et al. highlighted the limitation of memory bandwidth and the resulting restriction for future microprocessors [52], graphical experts like Kugler et al. were already re-evaluting state-of-the-art memory technologies [91, 72]. With bank interleaving as a too costly technique, Kugler et al. underlined the benefits of "logic-embedded memories" like FBRAM [22, 86, 21] and GRAMMY [63] to speed-up parts of the Z-buffer processing [62]. To this, the PIM-like 3D-RAM developed by Mitsubishi and Sun, also known as FBRAM, was already available within some commercial products in the 1990s, e.g. Elite3D (3DRAM) and Sun XVR-1000 (3DRAM-64). Currently, the first commercially available hardware equipped with 3D-stacked dies in the form of high-bandwidth memo-



traditional 2.5D true 3D

Figure 1: Sketch of a traditional packaging with a printed circuit board (PCB), 2.5D integration with silicon interposer, and true 3D stacking.

ries (HBM) is AMD's Fiji graphics chip (Radeon R9 Fury), offering four stacks of HBM, each with a capacity of one Gigabyte [1]. Announced, but not yet obtainable, are the Nvidia Pascal GPU architecture (e.g. Nvidia Tesla P100) [93] and the Altera FPGA Stratix 10MX device [24, 25], which will offer four (the latter even up to eight) HBM2 stacks.

1.1 2.5D integration and 3D stacking

To scale the bandwidth of DRAMs further while increasing their energy efficiency, the techniques *2.5D integration* [28] as well as *3D stacking* [13] are two promising groundbreakers (fig. 1). Both aid the design of future mobile as well as high-performance computing systems by allowing for wider buses and higher clocking of on-chip buses, and by enabling the composition of heterogeneous dies built using disparate process technologies.

Since diverse dies can be integrated on a silicon interposer (2.5D) or stacked on each other (3D), density increases, resulting in a decreased printed circuit board (PCB) space, shorter access latencies due to the on-chip wiring, higher bandwidth, as on-chip buses can be as wide as required, and reduction or even prevention of off-chip accesses [95]. These aspects inevitably lead to an overall improvement in performance and power efficiency. Despite the not yet fully solved thermal issues for 3D stacking [60, 58] as well as higher manufacturing complexity, lower yield and problematic testability for both cases, both techniques manifest a revolutionary step towards future computing systems. Compared to 3D stacking, 2.5D integration permits a lower-risk-affected and manufacturing-cost-effective step [9], even capable of coupling the 3D-stacking technique atop. However, a silicon interposer with through-silicon vias (TSV) between package substrate and the multiple dies is still required. In contrast, 3D is not just a solution for the system-wide scaling challenge of memories, but rather the sole true key-enabler in packaging with low to medium risk involved [114] to accomplish a reduction in system-level power as well as form factor [112]. To remove the silicon interposer burden and therewith TSVs in the 2.5D integration of multiple heterogeneous dies, Intel recently announced their embedded multi-die interconnect bridge (EMIB), enabling a standardized interconnection between the dies and resulting in a normal packaging yield for such configurations [24, 25].

3D stacking itself is a collective term for several vertical stacking techniques, which in the past was accomplished by placing dies staggered, interconnected via the aid of thin wire-bonds. To this, a magnitude of vertical die-connection techniques have been studied and proposed: e.g. package on package (PoP), package in package (PiP), 3D-packaged micro-bumped, face-to-face micro-bumped, contact-less via capacitive or inductive, and die-to-die [87, 20, 106]. Of these, the direct-contact die-to-die technology utilizing TSVs with the appropriate micro-bumps has the simplest structure, hence is the preferred stacking technique for 3D stacking of dies [90].

Memory	**Wide I/O**	**HMC**	**HBM**
JEDEC standard	yes	no	yes
DRAM interface	Wide parallel interface. Signaling is similar to SDRAM. Wide I/O is SDR, WIDE I/O 2 is DDR.	Chip-to-chip SERDES interface.	Wide parallel, multi-channel interface. DDR signaling.
Interface width (bits)	512 (Wide I/O), 256, 512 (Wide I/O 2).	Up to 4 links with up to 16 lanes each (v1.0 - v2.1). Up to 8 links (v1.0, v1.1).	128 per channel, up to 8 independent channels (1024 max).
Maximum bandwidth (GB/s)	Up to 17 (Wide I/O), Up to 68 (Wide I/O 2).	Up to 320 (v1.0, v1.1), Up to 480 (v2.0, v2.1).	Up to 128 (HBM), Up to 256 (HBM2).
System configuration	On top of processor. TSV connection.	Point to point, short reach SERDES. PCB based.	2.5D TSV based silicon interposer (SIP).

Table 1: Comparison of *Wide I/O*, *Hybrid Memory Cube* (HMC) and *High Bandwidth Memory* (HBM) standards [45].

Derived from stacking with TSVs, three competing products are under ongoing development (table 1): these are the *High Bandwidth Memory* (HBM) lead by a consortium consisting of Hynix, AMD, Nvidia, and Samsung [75, 76], the *Hybrid Memory Cube* (HMC) steered by Micron and Intel [99, 50], and Samsung's Wide I/O [39]. HBM itself is a JEDEC standard, offering a wide parallel bus with a maximum bandwidth of up to 256GB/s (HBM2). HMC, in term, constitutes a completely new development, offering a base layer to separate fast logic and dense memory and therefore allows for embedding fast processing units [131]. It furthermore offers a unique novel abstract bus interface and high parallelism via independent so-called vaults, thereby achieving a maximum link bandwidth of up to 480GB/s with a current maximum DRAM data bandwidth of up to 320GB/s [46]. As the HMC's logic layer is flexible, i.e. not bound to Micron's default, any vendor can customize it, resulting in an HMC with Intel-specific interface called Multi-Channel DRAM (MCDRAM)[1] and an Active Memory Cube (AMC) that exploits the logic layer for processing units [92]. In contrast, Wide I/O is designed to offer ultra low-power bandwidth between a System-On-Chip (SoC) and on-top mounted memories, like e.g. in smart-phones. Since stacking has constraints in yield and testing that result in cost, the present foreseen operational scenarios are in high-performance computing (HPC) applications, networking devices [82] and SoCs (in the case of Wide I/O).

1.2 Novel non-volatile memories

A wide field of contemporary but also novel non-volatile memory (NVM) approaches can be determined, which pos-

[1] Intel's Xeon Phi 'Knights Landing' is equipped with MCDRAM.

sess the potential to fill the bandwidth or latency gaps within the memory hierarchy. One already commercially available candidate is NAND flash, which bridges the gap between spinning hardware drives and fast volatile memories with the aid of high write and read bandwidth, high IOPS, and in addition with persistence. In 2015, Intel and Micron announced their phase-change memory (PCM) derivative called 3D Xpoint [15], which shall provide the necessary potential of removing conventional volatile memories as it tries to union the advantages of rapid DRAM and large-capacitive NAND flash. An NVMe SSD sample of Intels 3D Xpoint product called Optane showcased a tremendous average read / write throughput of up to 2GB/s [48]. SanDisk and HP proposed an undefined storage class memory (SCM), which shall provide similar properties [107].

A magnitude of further memories were studied, namely electronic-effects memory (charge-trap), spin-transfer-torque magnetic RAM (STT-MRAM), resistive memory (ReRAM), ferroelectric-gate RAM (FeRAM), nano-mechanical RAM (Nano-Mech), polymer and molecular memories [37, 49, 129]. Each of which has its strengths either in read and/or write bandwidth, latency, storage capacity, cycling capability, lower power (per bit), reliability, or memory-cell factor (cost) [49, 129].

Not yet realized, but the most notable non-volatile memory, could be the memristor. In 1971, L. Chua prepared indices that a forth circuit element, namely the memristor, has to exist besides capacitor, inductor, and resistor [14]. A memristor would be a revolution for memory architectures as it would be capable of permanently storing bits based on resistance instead of capacity. Since 2015, researchers doubt that a memristor will ever be feasible [124].

2. THE WALLS

In 1992, Elliott et al. depicted the tremendous amount of bandwidth within single memory chips, which is obtained by the sense amps [29]. As the sense amps feed the *row buffer*, Kogge highlights in 1994 and 1995 that the huge amount of bandwidth available at the *row buffer* is wasted in excess of 99% as soon as it leaves the memory chip [65]. He associates the tremendous bandwidth loss to packaging and technology constraints, leading to more and more caches within the processor, wasting silicon, die size, and power, just to recover a tiny fraction of the discarded bandwidth [70]. Still, in contrast to the subsequently explained off-chip *memory wall* and *bandwidth wall*, a simplified massive in-chip memory performance scaling can not be anticipated for the future as the memory access latency improvement of regular DRAM blocks is only at a ratio of 7% (fig. 2) [43, 44, 66].

2.1 The memory wall

The technological basis of processors are fast circuits capable of running at high frequencies, while memories rely on dense, much slower circuits, offering a large amount of storage capacity for an comparatively inexpensive price.

In 1995, Wulf et al. depicted a steadily increasing performance gap during technological advancements of processors and memories [128]. Even though, according to Wulf et al. at the time of their study, both technologies gained exponentially more performance in the past and will so in the future, the processors exponent is a magnitude larger. To illustrate their observation, Wulf et al. made the case for ideal caches where cache hit rates are above 99%, while the technological growth in memory as well as processor speeds keep evolving at the same perceived rate. Even with this ideal cache scenario, they derived that the gap between processor and memory performance will dramatically aggravate within the next 5 to 10 years and, as a result, they conclude that future computing systems are unambiguously dominated by the technological progress in memory performance. This they referred to as the *memory wall*.

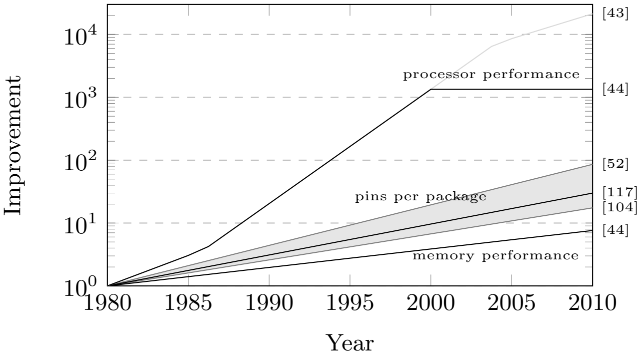


Figure 2: The *memory wall.* Improvements in pins per package as well as processor and memory performance between 1980 and 2010 [43, 117].

As Wulf et al. did not foresee any simple memory technology which scales linearly to the performance advancements in processor technology, they raised a multitude of aspects in how to overcome or at least mitigate the effects of the *memory wall*. The most noticeable and novel aspect for software developers and compiler tools is the need for dealing with non-uniform memory access latencies as a new *status quo* for future computing systems. Considering 3D-stacked memories, this aspect gains significant importance as these offer multiple more locations compared to planar memories, where access latency overheads can occur, e.g. link-controller, row-buffer, crossbar, vault-controller, vaults, bank-controller, banks, policies like closed or open-page policy, scheduling policies, address mapping, resource conflicts, etc..

2.2 The bandwidth wall

Whereas Wulf et al. solely pointed out the symptoms of the widening performance gap between processor and memory technology, Burger et al. investigated into the underlying problem formerly described by Elliott et al. and Kogge a year later (1996), i.e. the off-chip bandwidth between the processor package pins and the memory-package pins [52]. Burger et al. claimed that latency-tolerance techniques, which mitigate the direct memory access latencies, will reveal pin-bandwidth constraints in the future, resulting in a fundamental obstacle for computing systems to gain more processing performance.

While a magnitude of aspects underpin the hypothesis of Burger et al., the most significant are: as processor architectures with a novel technology can issue more loads, the pressure on the memory subsystem significantly increases. Even though latency-tolerant techniques can mitigate such loads via the aid of locality, in the case of success, the processor is capable of creating loads even faster, worsening the band-

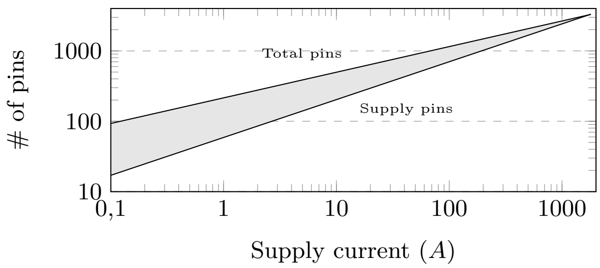


Figure 3: Growth of total pins vs. growth of supply pins in relation to the supply current, forming a pin gap [117].

width situation. If more data and instructions are required for a new processor, more cache-lines will have to be moved from the memory through the processors cache-hierarchy. Each cache-line bears the possibility of being underutilized and to potentially replace more important cache-lines, leading to a significant data-transfer overhead and demanding for an even higher bandwidth. Multiple threads and cores within the same package aggravate this bandwidth issue.

Although bandwidth per pin steadily increases, Burger et al. anticipate slowly growing rates in pin counts (1996: 16% per year [52]; 2005: 10% per year [104]; 2011: pins double every six years ≈ 12% per year [117]) in contrast to the tremendously increasing amount of transistors per chip (doubling every 18 months ≈ 60% per year) predicted by Moore [88]. This leads to processing performance gain as the main cause of their so called *bandwidth wall*, besides minor causes like memory buses (protocol overhead), memory interfaces, packaging cost, power constraints, and cooling requirements (fig. 2) [52, 104]. Considering the access-time growth of 7% per year, as predicted by Patterson et al. [44], the total pin count growth does not seem worrying. However, in 2011 Stanley-Marbell et al. highlighted that both *memory wall* and *power wall* are likewise constricted by pin scaling as the supply-pin growth is massive compared to the moderate developments in total pins (fig. 3) [117]. This could be seen as a *pin wall*.

In contrast to Wulf et al., Burger et al. foresaw a trade-off between access latency and bandwidth to mitigate the *bandwidth wall*. This is also backed by Patterson, who prefers bandwidth over latency, as processor designers can potentially extract more performance from bandwidth [97].

Considering the end of Moore's law as stated by Waldrop [125], Hennessy and Patterson [44] (fig. 2), and even underpinned by Intel with their recent three-phase "process, architecture, and optimization" model (in contrast to its past two-phase "tick-tock" model) [47], the 18- to 24-month rule of doubling transistors per chip does not hold any longer. This leads to a potential mitigation of the *bandwidth wall* and a relief for the pin-count growth from a single-chip perspective. Still, as multiple processing chips can be interconnected to a non-uniform memory access (NUMA) architecture, bandwidth will remain a constraint even after the end of Moore's law, demanding for novel computing concepts to rise processing performance [114].

2.3 The power wall

Based on process-technology scaling, growing transistor density, and raising clock speed / frequency, processor architectures faced the *power wall*, as power dissipation (Watts per cm^2) increased dramatically due to higher power leakage, steering towards thermal and hence cooling constraints, resulting in the potential of melting the computer chips [100].

The last processor neglecting the issues of power and thermal constraints was the Intel Pentium 4 with its NetBurst architecture, which utilized a deep single-thread processing pipeline and very high clock frequencies to achieve high computing performance. Consequently, computer architects are forced to pursue other directions, also avoiding the *ILP wall*, which led to the development of new technologies and to multi- and many-core processor designs by leveraging parallelism [10]. Although such simplistic addition of cores escalates power, cost and die area constraint dramatically. Hence, the new trend steers towards heterogeneity and acceleration by application-specific IP.

Whereas processor architectures are typically associated with the *power wall*, memories participate massively as well [10]. Especially the memory-refresh cycles due to leakage, buffer chips, SerDes links, channels, and data over-fetch contribute to the power budget, tightening the *power wall*.

In the context of the exascale computing initiative $(1,000$ PetaFlops), the *energy wall* is a often referring synonym to the *power wall* with the difference that the *energy wall* is a limitation that DARPA introduced in a report for an exascale supercomputing machine in the year 2008 [64]. The main objective of DARPA's report, led by Kogge et al., was an investigation into a potential exascale machine built with the circumstances of a maximal power budget of 20MW and finished by the year 2020. As a result, these conditions demand for 50pJ/Flop, which is not even close to the current offering (current: 190, foreseen future: between 180 and 420) [67]. Closing such a gap demands for drastic innovations, as even the extrapolation of lightweight systems considers 2GW as a minimum [10].

Summarizing the effects of the *memory wall*, the *power wall* and the *ILP wall*, Patterson concluded that the deriving *brick wall* needs to be seen as the new *conventional wisdom in computer architecture* [98].

3. TRADITIONAL COMPUTER ARCHITECTURE SCALING

This chapter provides a brief summary on how computer architects try to deal with the *memory wall* and the *bandwidth wall*. Both are a result of the diverse technology scaling paradigms and on how the developed techniques affect bus, memory (hierarchy), and the processor architecture. All in all, these techniques are trade-offs between area consumption, technical and physical feasibility, design complexity, thermal and power dissipation, packaging and fabrication requirements, production costs and vendor acceptance (e.g. in the case of memory-access protocols), and potential reliability aspects.

According to Rogers et al., industrial vendors only increased the frequency of the off-chip bus and number of bus channels in the past, which is not suitable any more as frequency is limited by the *power wall*, while the increase in channels is bound by the printed circuit board (PCB) area (layers), cost, and thermal issues [104]. Traditionally, memory-hierarchy deepening and the Harvard architecture are exploited. The first utilizes several levels of SRAM or embedded DRAM (eDRAM) to cache data (e.g. up to level

4: IBM Centaur memory-buffer chip [118] or Intel Haswell & Broadwell eDRAM [40]), while the latter separates instructions and data at the first cache-level (sometimes even up to the second).

Regarding the processor, multiple cores (especially simple ones), a magnitude of hardware threads (e.g. in graphical processing units, Crays general-purpose MTA & XMT machines [71], ...), large vector registers (to hold more data [92] and use less instructions due to *single instruction, multiple data* (SIMD)), (many) regular registers (in contrast to Thoziyoor's register-less architecture [121]), specific prefetching and speculative instructions in combination with speculative execution [52] are being utilized to gain performance by either neglecting latency and bandwidth constraints with multiple hardware contexts (like e.g. SPARCs register windows [126]) or by temporary data caching. Intelligent load scheduling [52], out-of-order execution by aid of the re-order buffer (ROB), and register scoreboarding are also common techniques for hiding latency, as highlighted by Saulsbury et al. [108].

A wide field of processor architectures were studied with respect to increasing the number of in-flight instructions, diminishing the influence by memory, and claiming to overcome the *memory wall* [17]. Outcomes are the slice-and-data buffer (SDB; miss-dependent instructions are moved from ROB to SDB, not occupying a reservation station or ROB entries) [51], the Pseudo-ROB, the slow-lane instruction queue (SLIQ; integral component of a kilo-instruction microprocessor; utilizes a queue for instructions which depend on and wait for a L2 cache miss) [16], hardware (multi-) check-pointing for more increased speculative execution [17], and very deep load/store queues [92].

Already Rogers et al. concluded that processor techniques (in their case smaller cores) provide just a moderate boost for performance scaling, even though they are commonly used in computing systems, since they do not directly tackle the issues of the *bandwidth wall* [104]. To this, Roger et al. categorized direct, indirect, and dual memory enhancing techniques.

Indirect techniques are e.g. (lockup-free) caches or scratchpads like eDRAMs or SRAMs in combination with *direct memory access* (DMA) units, controlled by a compiler or software developer [53]. These do not actively solve the *bandwidth wall* issue. In the past, Burger et al. proposed a link compression between processor and memory (direct) to place and interconnect both together on the same silicon interposer (indirect), or – if possible – to integrate both onto the same die or in a 3D-stacked fashion (indirect) [52]. Stanley-Marbell et al. underlined these aspects and added silicon-photonics (direct) as well as on-chip voltage regulators[2] (indirect), wherein the latter is also capable of diminishing the supply-pin constraints [117]. Furthermore, Rogers et al. suggested indirect procedures like cache compression, integration of a DRAM cache onto the same die, a 3D-stacked cache (in contrast to Burger et al.), and unused-data filtering (cache prediction / prefetching) via e.g. the aid of sectored caches, that only retain useful data within cache lines [104].

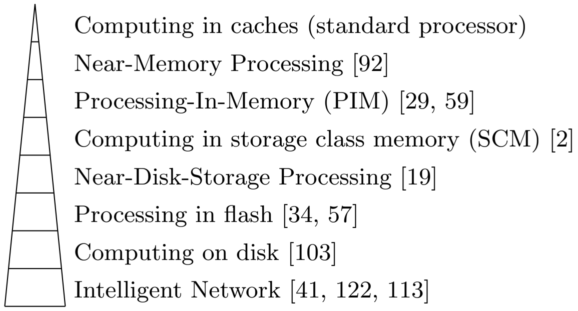

Computing in caches (standard processor)
Near-Memory Processing [92]
Processing-In-Memory (PIM) [29, 59]
Computing in storage class memory (SCM) [2]
Near-Disk-Storage Processing [19]
Processing in flash [34, 57]
Computing on disk [103]
Intelligent Network [41, 122, 113]

Figure 4: Storage hierarchy enhanced with *near-data processing* (NDP) capacity.

4. TAXONOMY OF NEAR-DATA PROCESSING (NDP)

Despite the techniques stated in section 3 still being crucial until today, bridging *memory wall* gap seems to be infeasible as the main issue of pin limitation and its derived constrained bandwidth is improperly addressed. Novel processing concepts are required to bring computing closer to the location where data resides or travels. This avoids the otherwise resulting small, potentially multiplexed buses with their lower frequencies as well as large memory hierarchies involving a massive amount of data motion (from storage to processor registers).

In 1996 Saulsbury et al. questioned the architectural orientation towards processor-centric enhancements [108], while one year later Kaxiras et al. demonstrated the capability of reducing "external traffic, by bringing computation to data rather than data to computation", utilizing so called *parcels* [56]. Shifting tasks during runtime to the location where to-be-processed data resides was not only adapted by DIVA [56], but also by the architectures HTMT and MIND/Gilgamesh. These employed either specific *parcel* hardware support or so-called location-aware memory operations [11, 121]. Recently, Kogge et al. announced a novel machine titled *Emu 1*, which builds upon *near-data processing* paradigms and adapts lightweight threads capable of being transferred to other nodes using a dedicated manager engine, claiming to be a memory-centric architecture design [85]. Whereas the concept of assigning a threads context to the location where the to-be-processed data resides seems to be promising for future data-centric systems, the sole consideration of the context transfer (register state) is insufficient: it neglects the overhead in moving the contexts instructions (also being data) to the memory of the to-be-utilized processing unit. As a result, a trade-off in moving either instructions including the threads context or to-be-processed data is required.

In contrast to simple co-location of single tasks to specific to-be-processed data, Riedel et al. highlighted through their active hard-drive machine in 2001 that novel concepts are required, as traditional techniques (section 3) were producing a physical gap between compute units and data [103]. Recently, IBM moved on with this philosophical shift from processor-centric to data-centric computing via the aid of their data-centric systems (DCS), trying to employ specialized processing units throughout the entire memory hierarchy [27, 57]. According to Agerwala, data motion is greatly expensive, so that computation should be placed wherever

[2] E.g.: Intel Haswell features fully integrated voltage regulators (FIVR) [12].

it runs best [2]. These efforts in distributing heterogeneous computing units to any location where data is stored can be summarized as a shift from hierarchical (static) memory to hierarchical processing or hierarchical active-data storage (fig. 4).

A generic term for such a kind of computing philosophy is called *near-data processing* (NDP). NDP gained a lot of momentum not only on grounds of IBM's efforts, but also due to the occurrence of the *big data* marketing concept [36] and the directive towards data-centric architectures [101]. To this, the recent technological developments around 3D packaging initiated a Processing-In-Memory[3] research hype owing to the ability of embedding custom logic into 3D-stacked memories (section 5). Based on NDP's philosophy a few commercial products exist, most notable the IBM Netezza platform [19], IBM BlueGene/Q with its network collective (which is enhanced with floating-point capability) [41] as well as IBM Hybrid Scalable Solid State Storage (HS4) [57, 34]. Similar to the floating-point-enhanced collectives in BlueGene/Q, Mellanox announced their NDP-inspired Switch-IB-2 product in 2015, which is based on the *Scalable Hierarchical Aggregation Protocol* (SHArP) and includes transparent processing capabilities within the switch, which allows for one-the-flight processing of data within the network [113, 122]. With the acquirement of EZchip and as of its general-purpose network switches and adapters [89], Mellanox seems to become a key enabler for processing-in-network as well as near-disk-storage processing.

Summarizing the NDP philosophy illustrates that a magnitude of operational scenarios can be foreseen (fig. 4), as there is no limitation to the storage / systems hierarchy [2]. NDP architectures start questioning von Neumann's separation, leading inevitably to novel architectures (e.g. [26]) and massive changes to current programming paradigms. Ranganathan even suggested replacing DRAM with non-volatile memory (NVM) altogether as the latter offers significantly more storage including a decreased power consumption [101], leading to systems capable of processing in Flash. In particular, Ranganathan depicts a 3D-stacked combination of PCM and memristor memory to induce a more fundamental architectural change in contrast to current data-intensive, flash-based computing systems. By considering novel announcements like Seagate's NVMe-based solid-state drive [110], offering a throughput of 10 GB/s that is already close to single-channel mid-range (DDR3-1333) and low-end (DDR4-1600) DRAM modules in conjunction with a dedicated – until now transparent, vendor-locked – processor, the replacement and the NDP concept of computing in flash, potentially future host memory, even seems to become feasible.

Since the most notable NDP class of so-called PIM architectures try to overcome or at least mitigate the *memory wall*, enabling a path to tackle some of the exascale computing barriers, the succeeding distinct chapter is explicitly devoted to their benefits and disadvantages (section 5).

5. PROCESSING-IN-MEMORY (PIM) AND NEAR-MEMORY PROCESSING

One of the most auspicious scenarios covered by the NDP context seems to be *processing-in-memory* (PIM) as well as *near-memory processing* (NMP) architectures. Both attempt to close or at least diminish the *memory wall*, the *bandwidth wall*, and the *power wall* gaps by moving processing elements into or near the memory subsystem. The approaches aim at achieving high efficiency and high bandwidth in conjunction with short latencies between processing elements and memory banks.

Various attempts of integrating processing capacity at different locations within planar or stacked memory chips were studied, highlighting the significant advantages, e.g.:

- between the sense amplifiers and the column decoder [29, 59]
- behind the column decoder [38, 4]
- memory embedded into the pipeline of a general-purpose processor (GPP) [121]
- integrated in front of a bus [65, 115, 95] or crossbar switch [92]
- distributed into each vault of a 3D-stack [3]
- embedded as processing dies between memory dies [132]

Considering a memory subsystem as a compound of multiple memory chips, each with several memory banks (and in the case of 3D-stacking a certain amount of vaults), typical PIM architectures leverage parallelism by utilizing multi- and many-core techniques as well as application-specific architectures to increase energy efficiency, overall performance, and throughput by a magnitude while bypassing the *power wall* and the *instruction-level parallelism (ILP) wall* constraints. The techniques utilized are depicted in section 3. This is accomplished by a trade-off, as space area is typically limited and thermal as well as reliability issues are of major concern (section 6).

A clear distinction between PIMs and other closely related architectures is difficult, as the class of processing arrays in particular (e.g. 1973: Reddaways DAP [102], 2012: Adapteva Epiphany [94], 2015: REX Computing Neo [42]) does share a multitude of PIM properties, while PIMs even adapt to processing arrays in some studies [38]. However, PIMs could be acknowledged as a passive acceleration architecture, offering a regular and transparent storage for data in the default mode, while being capable of executing arbitrary instructions for processing the data on behalf of the main processors.

In history, PIM architectures repeatedly became of high interest in the research domain, especially when novel memory technological aspects or walls (section 2.1) appeared. To this, section 5.1 provides an overview of the historical background of PIM and *near-memory processing* architectural developments.

As a brief remark, depending on the author and year of publishing, PIM architectures are titled differently even though the main idea remains the same. In the beginning, when simple logic was embedded in the then novel DRAM technology, PIMs were mainly termed based on the technical aspect, e.g. "Logic-in-Memory" (driven by Minnick, Stone and Kautz) [119] and sometimes "Merged Logic DRAM" (MLD) [54] or even "logic-embedded memories" [72]. Starting within the 1990s, PIMs obtained a more psychological, i.e. cognitive title, followed by "Intelligent RAM" (IRAM, 1996) [18, 54], "Active Memory Model"

[3] Many researchers put PIMs on the same level as NDP, even though PIMs are just a subset of NDP (fig. 4).

(AMM, 1997) [68] (2007: "Active Memory Operations" [31], 2012: "Active memory controller" [32], 2015: "Active Memory Cube" (AMC) [92]) and ending with "smart memories" (1999, 2000) [30, 79].

5.1 Historical background of PIM and its potential

The separate arrangement of processor and volatile memory conceived by von Neumann 70 years ago with his draft of the EDVAC computer [123] aided not only the development of current computer architectures but also the technological scaling: processors could be optimized for performance with large transistors, while memory could be improved towards massive storage capacity with dense, slow transistors [108]. Although this architectural benefit has been successful until today, the concept of PIM question this separation [119] by integrating both onto the same die or into the same stack once more.

2D-integrated Processing-In-Memory

Research on bringing computing elements closer to memory dates back to the 1960s and has since reappeared several times (fig. 5). Whilst Dennard invented the single-transistor DRAM cell [23], Minnick (1964: [83]), Kautz (1969: [55]), and Stone described processing cellular logic capable of being interconnected to massively parallel arrays and allowing to process incoming data on the flight [84]. While their illustrated cellular logic-in-memory (CLIM) arrays were always a special-purpose construction and thus hard-wired to their functionality, Stone proposed a general-purpose logic in memory processor in 1970 [119]. One could argue that these CLIM arrays were no real PIM architectures as they were limited to their application scenario, even though these nowadays primitive architectures established the fundamental idea of combining processing logic and memory storage.

Roughly ten years later, one of PIMs strengths, i.e. the massive amount of bandwidth, let Fuchs et al. developed their PIXEL-planes architecture, leveraging the bandwidth bottleneck in the graphical processing units (GPU) framebuffer by integrating processing elements for each pixel into the DRAM memory [35]. While PIXEL-planes offered a non-general purpose architecture, Elliott et al. were able to visualize the value of PIMs with a massively parallel GPP another 10 years later [29]. Both architectures can be seen as the first true PIMs, as both offer typical PIM characteristics like short latencies in conjunction with high bandwidth, energy efficiency, etc..

Despite that all of the aforementioned architectures were a construct of processor and memory embedded within a single chip, none of them were actually called PIM. The term PIM was initially mentioned by PIM-researcher Kogge et al. in conjunction with their EXECUBE PIM as "processor in memory" in 1995 [70, 65]. The common form of "Processing-In-Memory" as known today can be led back to the Terasys PIM of Gokhale et al. [38].

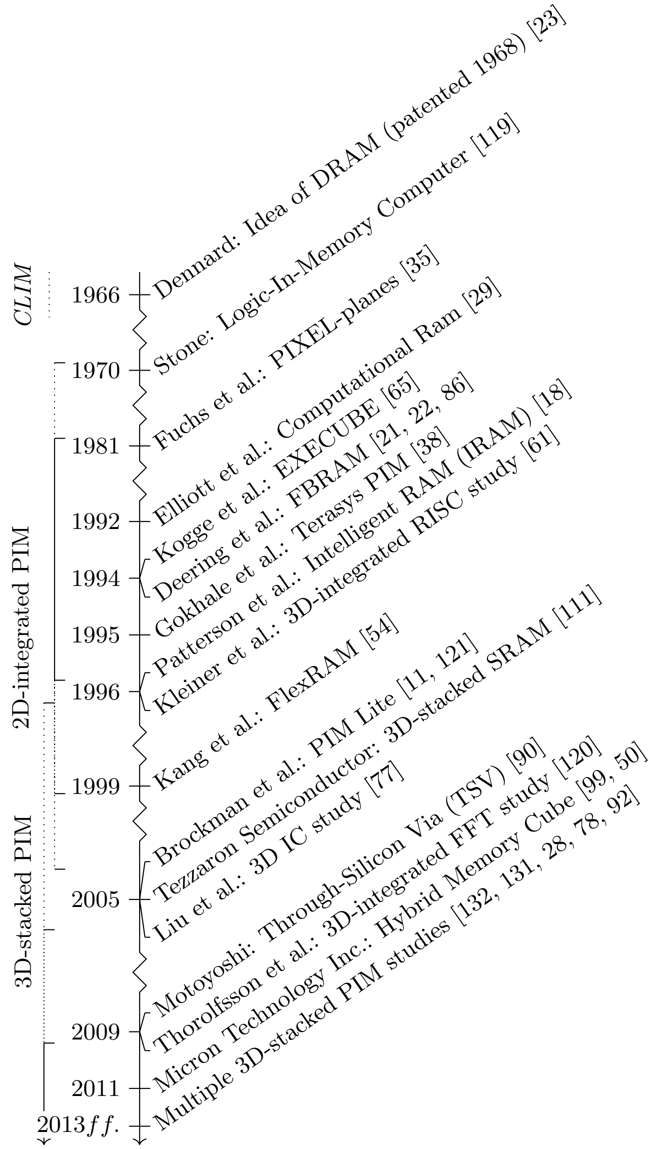

CLIM
2D-integrated PIM
3D-stacked PIM
1966
1970
1981
1992
1994
1995
1996
1999
2005
2009
2011
2013ff.
Dennard: Idea of DRAM (patented 1968) [23]
Stone: Logic-In-Memory Computer [119]
Fuchs et al.: PIXEL-planes [35]
Elliott et al.: Computational Ram [29]
Kogge et al.: EXECUBE [65]
Deering et al.: FBRAM [21, 22, 86]
Gokhale et al.: Terasys PIM [38]
Patterson et al.: Intelligent RAM (IRAM) [18]
Kleiner et al.: 3D-integrated RISC study [61]
Kang et al.: FlexRAM [54]
Brockman et al.: PIM Lite [11, 121]
Tezzaron Semiconductor: 3D-stacked SRAM [111]
Liu et al.: 3D IC study [77]
Motoyoshi: Through-Silicon Via (TSV) [90]
Thorolfsson et al.: 3D-integrated FFT study [120]
Micron Technology Inc.: Hybrid Memory Cube [99, 50]
Multiple 3D-stacked PIM studies [132, 131, 28, 78, 92]

Figure 5: Historical developments in 3D-stacked integrated circuit (IC) and PIM research.

In 1996, Patterson et al. presented their results on a single-core MIPS integrated into memory that mitigates both the programming burden and the complex memory partitioning [18]. This approach differed from other studies that included massively parallel processing elements sitting either between the sense amplifiers and the column decoder [38, 29] or behind the column decoder [70]. Vector Intelligent RAM (VIRAM) developed by Patterson et al. appeared at the peak of the 2D-integrated PIM research, highlighting the significant importance of custom specific architectures (in their case SIMD vectorization) [95], but not addressing the issues of slow compute transistors due to (modified) DRAM technology with larger logic cells [119, 29, 38, 108]. It furthermore lacked upgradeability of memory storage or compute capacity, and lacked any business case. Although the research-peak in 2D-integrated PIM studies had passed, two further studies appeared: in 1999, FlexRAM was introduced by Kang et al., which achieved Processing-In-Memory at several stages, i.e. in parallel with compute elements at each memory bank and in sequential with a single processor in each memory chip [54]. Six years later, Brockman et al. integrated memories directly into the processing pipeline of a regular processor [11, 121].

Besides the already known constraints of 2D-integrated PIM research, one further potential cause of this focus shift could be the advent of easily extractable massive parallel processing performance from cheap, widely available and power-efficient GPUs, as presented by Larsen et al. in 2001 [74]. In summary, Saulsbury et al. and Patterson et al. concluded that the main reasons for the 2D-integrated PIMs failure were cost derived by memory yield, more metal layers, and the introduction of logic into an industry optimized for Bytes/wafer. Further obstacles resulted from the DRAM technology being used to combine processor and memory, leading to an overall slow computation [108, 95, 96]. Further reasons were depicted regarding difficult programmability, limited memory for instructions and data, increased testing time, and the lack of a real business model [96].

Nevertheless, a 2D-integrated PIM study by Dlugosch et al. appeared in 2014, depicting DRAM-embedded automata engines, enabling significant performance gain in processing regular expressions [26]. To this, Intel started to embed DRAM and the processor onto the same silicon interposer with their Haswell architecture [40].

Summarizing the efforts made by early 2D-integrated PIM research shows that:

+ embedding logic and memory onto the same die is functionally feasible, enabling very wide bus interfaces.

+ high bandwidth in conjunction with low latency can be achieved.

+ if a specialized architecture is utilized, tremendous speed-up can be achievable.

+ PIM can typically be utilized as regular DRAM if a main/host processor exists, or can be run stand-alone (possibly attached to external memory).

– easy PIM programmability is key for wide adaptation, depending on how the memory-view is supposed to look like and how the PIM is intended to be used (5.2). This was not given, as memory vendors do typically not need to take care about it [96].

– utilizing DRAM technology for processing units results in slow logic, whereas utilizing (fast) logic technology leads to little memory storage, high leakage, and increased cost per bit.

– a business case is required as memory vendors need a low selling point: Bits / area & layers / $ counts. Regular integration into memory is not competitive to regular memories.

– no path exists to upgrade either processing or memory.

3D-stacked Processing-In-Memory

During the peak-period of 2D-integrated PIM research (fig. 5), Kleiner et al. achieved stacking of memory on top of L1 and L2 caches, which in term were stacked ontop a RISC processor themselves [61]. A multitude of very small vertical vias ($\varnothing \sim$2-5μm) acted as interconnection between these adjacent dies. With the aid of a wider bus, their *near-memory processing* architecture achieved an improved and accelerated memory hierarchy, which allowed for superior system performance by circumventing pin and PCB wiring constraints. With a stronger core, Liu et al. repeated the study of Kleiner et al. nine years later, highlighting that on-chip main memory including an optimized cache hierarchy achieves a near-perfect performance [77].

While Liu et al. did not consider thermal influences, Kleiner et al. concluded that 3D-stacking does not come with thermal issues, which is in contrast to current research publications [60, 58]. In the same year in which Liu et al. presented their study, Tezzaron Semiconductors unveiled their 3D-stacked SRAM product [111]. Although the main technological foundation of 3D interconnection techniques namely TSVs (despite wire bonding, package on package, package in package, 2.5D integration, ... [20, 106]) were already production-ready in 2009 [90], i.e. thermal stress, speed, yield, size, and coupling of TSVs were manageable since then, 3D-stacked *near-memory processing* research was not from interest within the years 2004 until 2011 according to Motoyoshi. The fully 3D-integrated FFT study by Thorolfsson et al. was the only exception during this time, which achieved a significant wire-length and energy reduction [120].

In conclusion, the situation changed due to several aspects:

1. Micron announced their Hybrid Memory Cube (HMC) in 2011, which separates logic from memory, levering the burden of 2D-integrated PIMs by placement of any fast logic into the logic layer, whereas memory storage obtained its dedicated dies stacked above the logic die [99, 50].

2. The second aspect involves the field of data-intensive *big data* workloads, which try to use in-memory computing in the sense that data stays in the host's memory instead of passing it back and forth from disks or even tapes [27, 132]. In the past, such a scenario was not possible as host memories were too tiny. Current systems, however, offer up to one Terabyte of main memory [118], whereas researchers even propose non-volatile memories to form very large host memories [101].

3. Until today, the unresolved power-consumption concerns and derived need for efficient and energy-aware computing both resulting from the *power wall* require novel ideas: current enhancements to GPP architectures will fall short in the future when tackling the exascale barrier [69]. Viable solutions for approaching exascale involve a reduction of data motion by performing computation where data resides [80], which is directly addressed by the PIM / *near-memory processing* concept and the specialization of processing units. Such is already pursued by the industry with dedicated accelerators like the Intel *many integrated core* (MIC) architecture, *general-purpose computing on graphics processing units* (GPGPU) or *field programmable gate arrays* (FPGA).

4. Last, programming languages like OpenMP (target directive [8]) have evolved and OpenACC was invented, offering seamless utilization of dedicated accelerators. Adopting these languages to *near-memory processing* units avoids the programming burden faced by 2D-integrated PIMs.

Starting from 2013, these aspects led to a significant increase of 3D-stacked PIM / *near-memory processing* architectural research studies, visualizing the benefit of integrating heterogeneous components into the base logic layer of the

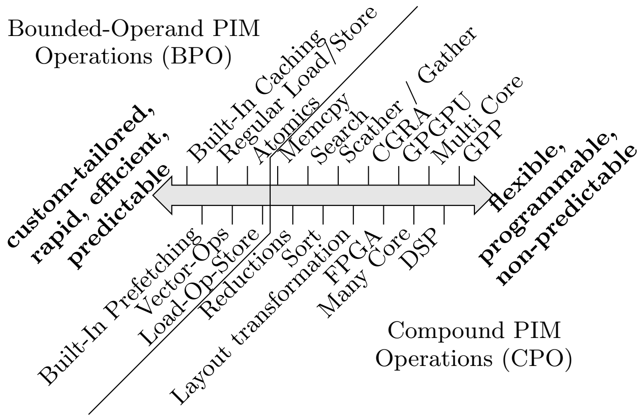


Figure 6: Types of potential PIM / *near-memory processing* architectures [78].

HMC [132, 131, 28, 78, 92]. To this, Aly et al. depicted how future 3D stacks could look like, heterogeneously integrated with multiple materials and diverse components [5]. Based on the outcomes of the past 2D-integrated PIM research, these novel studies mostly avoided the embedding of the compute capacity directly into memory, enabled by stacking of several mixed-technological dies via the aid of TSVs. As a result, in terms of terminology, current research suites better to *near-memory processing* in a sense of computing close to memory in contrast to *Processing-In-Memory*, even though these studies are commonly titled after the latter.

Most of these *near-memory processing* studies concentrate on general-purpose architectures [109] where many lightweight cores [65, 70] are favored over few strong cores [108]. Other research investigates application-specific and custom logic, which is strongly advised by Loh et al. [78]. As a result, performance-optimized caches or even cache-less architectures [65] in conjunction with specialized capabilities like vector-SIMD registers [95] are a typical outcome of these GPP-enhanced PIM studies. Although embedding regular GPP CPUs with the same instruction-set architecture (ISA) is a less radical research perspective and more of an evolutionary approach in fabrication, these PIMs are capable of executing unmodified host code, allowing to slowly migrate a workload from the host to the computing-enhanced memory.

In contrast to the GPP-enhanced PIM studies, more exotic *near-memory processing* architectures build upon GPGPU [130], VLIW [92], coarse-grained reconfigurable architectures (CGRA) [33] or just simple (smart) PIM-enabled instructions [3], applying the lessons learned by early 2D-integrated PIM research studies: These depicted that memory-specialized architectures offer the best possible potential for energy-efficient processing. The GPGPU study is of particular interest for neglecting any latencies while saturating the high bandwidth with a high amount of threads, the CGRA architecture for eliminating the instruction fetch (bandwidth) overhead to memory and decoding of instructions, thus achieving less energy consumption and higher computation speed.

Although a multitude of studies investigate PIM, there is no convention, which type of distinct architectures suits 3D-stacked compounds of memory and compute best, leading to the sole limitations of area consumption, power delivery and dissipation, and cooling or thermal constraints [70] (fig. 6). As concluding remarks, Ranganathan and Balasubramonian et al. highlighted that all old ideas in computer architecture design are new again as of the actual arising PIM / *near-memory processing* research concepts [101, 6]. To this, Loh et al. distinguished between compound (CPO) and bounded-operand PIM operations (BPO), depicting that especially application-specific non-Turing-complete BPO, and highly threaded CPO architectures are the most promising near-memory architectural candidates [78].

In conclusion the on-going 3D-stacked *near-memory processing* research shows:

+ an opportunity for heterogeneously integrated mixed technologies (digital, analog, . . .) and materials based on die stacking with TSV given [77, 6].

+ achievements in unparalleled bandwidth, as a multitude of vias enable an on-chip bus with adjustable width, without any pin constraints due to the on-chip bus frequencies being higher than the reduced capacitances, aided by the reduction of drivers and I/O interfaces [77].

+ a critical path reduction as of lower wire latency, leading to decreased latency [77, 102].

+ higher efficiency and lower system power as of lower latency, higher bandwidth, less off-chip communication, and reduced data motion [95].

+ an opportunity for a more rapid and efficient memory controller, due to the separate logic layer.

+ more available pins, which can be used for more important work than serving loads/stores to/from the host, e.g. integration of networking capabilities.

– that programmability is still a major issue, but the separation into memory and logic layer prevents dependending on a memory vendor (5.2).

– no given upgradeability for either computing or memory.

– that die stacking results either in a dramatic increase in testing or yield constraints (test each die, then stack and test again results in higher yield but lower manufacturing throughput; first stack and then test results in high manufacturing throughput but lower yield) [106]

– no determined distinct PIM architecture.

5.2 Processing-In-Memory usability challenges

NDP and, in particular, PIM architectures constitute a disruptive change to current hardware architecture designs, which in consequence opens up a huge gap in how massive acceleration can be extracted. Despite PIM's enhancements, most of the PIMs are capable of transparently sharing their embedded memory in a regular storage mode. This enables rapid as well as seamless adaption in standard computer architectures, avoiding any data-copying between PIM and host [29]. Levering this, future computing architectures could proceed with a slow transfer from non-PIM activated to partially PIM-enabled (using both, PIMs as well as standard memories [59]) to fully PIM-equipped memories. Since typical accelerators do not share the ISA with the host, the workload transfer could be aided by a similar ISA (or a subset of it). Although this is in contrast to the IRAM study [95], which highlighted that regular cores are not beneficial for PIMs, current studies utilize a multitude of simplified ARM cores to form a massively parallel PIM [81].

As expounded in section 5, PIMs can reside at various locations within the memory architecture, including (but not limited to) integration behind or in front of an in-memory bus or switch, within the banks, vaults, or stacks of memory chips. This leads to diverse memory views, several ways of execution, varying latencies and consequently different requirements regarding programming models [68], leading to a major concern not primarily tackled by any research. PIMs itself can either be executed as a massively-parallel host processor, or more commonly as a fast memory-resident co-processor or accelerator, speeding up workloads provided by a dedicated host processor both in a SIMD or MIMD fashion [65]. The typical PIM execution model favors SIMD, as MIMD implies dedicated instructions for each PIM core, leading to more storage pressure. Furthermore, an active memory management role can be foreseen, wherein the PIM explicitly takes care of moving data to a latency-optimized location for the host. Utilizing a heterogeneous system with PIMs and a dedicated host processor even offers the opportunity to turn the host into idle mode during PIM computation, increasing the overall system efficiency [115].

Independent of a PIM's execution model, excerpts of a global memory management unit (MMU) are required to translate any virtual address to a corresponding physical address within its potentially limited memory view. Alternatively, the PIM just acts on physical addresses whereas the memory view is bound to a segment, page, or even a cacheline as envisaged by Ahn et al. and Reddaway. While PIM hardware becomes simplified, as there is no need for address transforming and boundary checking [102, 3], the processing efficiency could decrease due to potentially required data motion. While the memory view of the latter is massively restricted, a dedicated MMU requires the exchange of entries from host to PIM, complicating the operating system. For the future, it seems that PIMs do require an MMU, since these will offer massively parallel processing units, which in term will demand for a huge amount of data capable to be processed in parallel.

Despite these different memory views and ways of execution, explicit data exchange between the distributed (offchip) PIMs is still required based on multiple memory chips, making it inevitably necessary that the programming model allows for either seamless data motion or seamless task motion between PIMs. To this, the situation of the PIMs memory view is even worse as typical memory controllers exploit the memory scrambling (address multiplexing, interleaving) technique to extract more bandwidth and to decrease latency from the subsystem for the main processor. Typical PIM research studies do not consider this technique. They concentrate on showcasing a high bandwidth for the PIM, but do not consider the hosts's bandwidth loss. Furthermore, coherence and consistency are also a concern for PIM as the main/host processor typically contains the cache.

Considering the research studies, it seems as if there is no limitation to where computational units could reside within the memory (e.g. [54]), leaving an open decision on how a PIM can be exploited by a developer or compiler / run-time. For programming, several concepts can be envisioned, ranging from software libraries that are loaded onto the PIM to predefined methods that are started with the aid of MMIO mapped register calls [38] to intrinsics, manual tuning, or even a novel programming language. Whereas low-level manual tuning is a viable candidate for high-performance computing (HPC) where little applications are running on massively parallel systems, such an approach can not be foreseen for the desktop or even mobile domain where standardized OS and run-time environments (e.g. Android and Java) are utilized on a multitude of similar, but not identical platforms.

In 1970, Stone concluded that processing elements in a DRAM (in his case cache) demand for high-level programming languages [119]. No intrinsical simple programming paradigm capable of utilizing heterogeneous accelerators by offloading tasks existed in the past. Based on the current existence of accelerators like MICs, GPGPUs, FPGAs, . . . , a significant amount of software research tackling efficient workload offloading to boost the processing performance and power efficiency has been undergoing [116]. Besides the languages CUDA and OpenCL that only target accelerators, to date there are at least two programming paradigms, namely OpenMP and OpenACC, and one heterogeneous computing concept of special interest, i.e. AMDs *Heterogeneous System Architecture* (HSA) [73].

For future PIMs, AMDs HSA seems to be most promising as it offers a common virtual memory view, a virtual instruction set capable of being processed on the host processor or the accelerator, and a smart dispatcher. All of it could be applied to PIMs. In conjunction with the analysis of PIM-suitable programming paradigms, hardware-unit evaluation is required regarding the unit's capability of efficiently transferring data between the PIMs memory view and keeping a potential host cache coherent and consistent. This is crucial, as software developers are used to a flat, global memory view without any access restrictions.

With different memory-views and non-uniform accesses, varying memory latencies will be the status quo for PIMs. This is not tackled by any actual programming paradigms, leading to the constraint that either the developer or the run-time system needs to consider these inconsistencies to extract the full bandwidth potential and achieve high processing performance.

As the aforementioned programming paradigms are not deployed yet, current PIM architectures are typically programmed by hand or intrinsics or are accessible via software library calls or just MMIO-mapped registers triggering predefined procedures loaded within a specific PIM. With a clear lack of a seamless integration into current processing systems, these heterogeneous programming paradigms could pave the way for PIM's future.

Taking into consideration that data intense applications like HPC, *big data* workloads and in-memory computing will gain the most performance speed-up as well as computing efficiency out of the *near-data processing* concept, a simplistic distinction is required for a developer to determine if NDP suits a chosen workload, which has not yet been discussed in literature. To this, the authors propose the roofline model of Williams et al., since it visualizes the operational intensity of specific algorithms in relation to a processors memory peak bandwidth and peak single / double precision floating point performance (which can be easily extended to integer operations) (fig. 7) [127]. The lower the operational intensity of

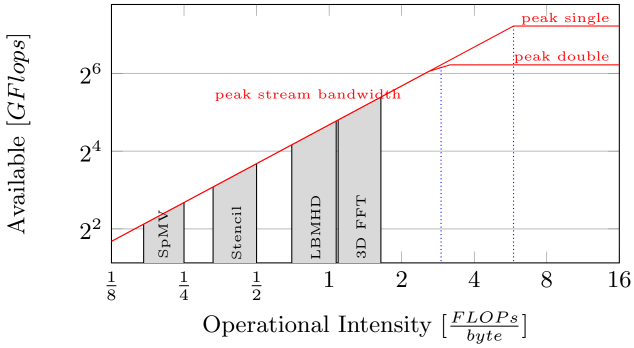


Figure 7: Example roofline model of an Intel Xeon E5405 [127].

an application gets compared to the one given by a specific processor (fig. 7; left of the blue dotted lines), the higher the potential that a NDP architecture surpasses a general-purpose processor in efficiency and computing performance (e.g. LBM and LQCD on the AMC [7]).

6. LESSONS LEARNED

Facing the inevitable *brick walls* in combination with the surrender of Moore's law, new architectural concepts are urgently required to enable a continuation in the scaling of processing system performance.

Since memories have been identified as the main bottleneck, despite tremendous internal (row) bandwidth, novel memory architectures like PCM, STT-MRAM, memristor and many more as well as packaging techniques like 2.5D integration and true 3D stacking were proposed, offering new levels of bandwidth, latency, storage capacity, reliability and energy efficiency. The production-readiness of TSVs paved the way for the packaging with 2.5D integration and 3D stacking, achieving a reduction in the critical path, data movement and power consumption as well as a heterogeneous integration of separated miscellaneous dies like logic and memory, capable of utilizing various types of materials. It furthermore enabled novel *near-data processing* concepts that aid to overcome the pin-limitation derived *bandwidth wall* and *memory wall*, as off-chip communication can be mitigated or even avoided.

In the past, 2D-integrated research depicted the opportunity for *near-data processing*, in particular *Processing-In-Memory*, by integrating compute units at various positions within a memory chip. Back then, this typically suffered from the chosen DRAM technology that lead to insufficient performance. With tight coupling of heterogeneous dies, 3D stacking alleviates some of the past issues and thus enables the *near-memory processing* concept for future data-centric systems.

Even though a wide range of possible architectural computing units can be foreseen to be embedded into the logic layer, research indicates the significant importance of utilizing either application-specific, fixed-function BPO or architectures capable of consuming massive amounts of bandwidth and neglecting latency (e.g. GPGPU, CGRA, . . .). The main feasible technical obstacles are currently thermal constraints: because the bottom layer is envisaged for logic, heating up the overlying memory layers of the dense 3D stack, leading to either a throttling or even a forced pause. To this, 2D-integrated issues like the lack of a programming model, system integration and common memory view between multiple memories and missing upgrade-ability of either compute or storage still need to be addressed by 3D-stacking research, although some of the past issues can be mitigated by novel enhancements in software or hardware design.

To gain wide acceptance within future computing systems, a simplified usability is strongly required for PIMs. Hence, not only programming models but also memory views need to be studied in detail. Past and current PIM research did and does neither address MMU-less PIMs that are bound to cache-lines, pages, or segments nor MMU-enabled PIMs. While the first are capable of addressing any data within a chip or stack, they first need to keep the MMU entries synchronized with the host. The second type, in term, requires transferring data between distributed chips or stacks. Although the (un-)availability of an MMU within the PIM has a tremendous effect on usability already, current memory controllers even utilize memory scrambling / interleaving to gain more bandwidth from the memory subsystem. Typical research does not consider such techniques. While this simplifies the programmability of a PIM it decreases the memory bandwidth. If current PIM research does not address the concerns towards the memory view, future PIM devices will be limited to simple fixed-function near-data operations like the set of atomics already supported by the HMC. While many solutions to resolve the usability concern exist, adapting AMD's HSA heterogeneous concept could be the most promising one.

Whereas the embedding of custom logic into a 3D stack can be seen as an evolutionary step, based on the predecessor of 2D integration, the revolutionary step relies on distributing compute units throughout the entire memory / storage hierarchy. Enhancing memory-/ flash-/ storage-class memory / disk / network with processing capacity is illustrated by research and some available products. These especially highlight possible performance speed-up and higher energy efficiency by transposing either passing data on-the-flight or nearby the data specific location.

Despite the current technical limitations and known suitable application domains, it seems that a massive breakthrough can only be achieved, if vendors face the inevitable *bricks walls* and the surrender of Moore's law directly by considering novel concepts like *near-data processing* or in specific *Processing-In-Memory* for all computing areas.

7. REFERENCES

[1] The road to the amd "fiji" gpu. Taiwan, September 2015.

[2] T. Agerwala. Data centric systems: The next paradigm in computing. Parallel Processing (ICPP), 2014 43rd International Conference on, Sept 2014.

[3] J. Ahn, S. Yoo, O. Mutlu, and K. Choi. Pim-enabled instructions: A low-overhead, locality-aware processing-in-memory architecture. In *Proceedings of the 42Nd Annual International Symposium on Computer Architecture*, ISCA '15, pages 336–348, New York, NY, USA, 2015. ACM.

[4] Y. Aimoto, T. Kimura, Y. Yabe, H. Heiuchi, Y. Nakazawa, M. Motomura, T. Koga, Y. Fujita, M. Hamada, T. Tanigawa, H. Nobusawa, and K. Koyama. A 7.68 gips 3.84 gb/s 1w parallel image processing ram integrating a 16 mb dram and 128 processors. In *Solid-State Circuits Conference, 1996. Digest of Technical Papers. 42nd ISSCC., 1996 IEEE International*, pages 372–373, Feb 1996.

[5] M. M. S. Aly, M. Gao, G. Hills, C.-S. Lee, G. Pitner, M. M. Shulaker, T. F. Wu, M. Asheghi, J. Bokor, F. Franchetti, K. E. Goodson, C. Kozyrakis, I. Markov, K. Olukotun, L. Pileggi, E. Pop, J. Rabaey, C. Re, H. S. P. Wong, and S. Mitra. Energy-efficient abundant-data computing: The n3xt 1,000x. *Computer*, 48(12):24–33, Dec 2015.

[6] R. Balasubramonian, J. Chang, T. Manning, J. H. Moreno, R. Murphy, R. Nair, and S. Swanson. Near-data processing: Insights from a micro-46 workshop. *Micro, IEEE*, 34(4):36–42, July 2014.

[7] P. F. Baumeister, H. Boettiger, J. R. Brunheroto, T. Hater, T. Maurer, A. Nobile, and D. Pleiter. *High Performance Computing: 30th International Conference, ISC High Performance 2015, Frankfurt, Germany, July 12-16, 2015, Proceedings*, chapter Accelerating LBM and LQCD Application Kernels by In-Memory Processing, pages 96–112. Springer International Publishing, 2015.

[8] O. A. R. Board. Openmp application programming interface. Technical report, Nov 2015.

[9] I. Bolsens. 2.5d ics: Just a stepping stone or a long term alternative to 3d? 2011.

[10] P. Bose. The power of communication - trends, challenges (and accounting issues). Discussion as NSF WETI Workshop, Feb 2012.

[11] J. B. Brockman, S. Thoziyoor, S. K. Kuntz, and P. M. Kogge. A low cost, multithreaded processing-in-memory system. In *Proceedings of the 3rd Workshop on Memory Performance Issues: In Conjunction with the 31st International Symposium on Computer Architecture*, WMPI '04, pages 16–22, New York, NY, USA, 2004. ACM.

[12] E. A. Burton, G. Schrom, F. Paillet, J. Douglas, W. J. Lambert, K. Radhakrishnan, and M. J. Hill. Fivr — fully integrated voltage regulators on 4th generation intel® core^{TM} socs. In *Applied Power Electronics Conference and Exposition (APEC), 2014 Twenty-Ninth Annual IEEE*, pages 432–439, March 2014.

[13] D. W. Chang, G. Byun, H. Kim, M. Ahn, S. Ryu, N. S. Kim, and M. Schulte. Reevaluating the latency claims of 3d stacked memories. In *Design Automation Conference (ASP-DAC), 2013 18th Asia and South Pacific*, pages 657–662, Jan 2013.

[14] L. Chua. Memristor-the missing circuit element. *IEEE Transactions on Circuit Theory*, 18(5):507–519, Sep 1971.

[15] T. Coughlin. Crossing the chasm to new solid-state storage architectures. *IEEE Consumer Electronics Magazine*, 5(1):133–142, Jan 2016.

[16] A. Cristal, D. Ortega, J. Llosa, and M. Valero. Out-of-order commit processors. In *Software, IEE Proceedings-*, pages 48–59, Feb 2004.

[17] A. Cristal, O. J. Santana, F. Cazorla, M. Galluzzi, T. Ramirez, M. Pericas, and M. Valero. Kilo-instruction processors: overcoming the memory wall. *Micro, IEEE*, 25(3):48–57, May 2005.

[18] K. Y. David Patterson, Tom Anderson. A case for intelligent dram: Iram. Palo Alto CA., August 1996.

[19] G. Davidson, K. Boyack, R. Zacharski, S. Helmreich, and C. J.R. Data-centric computing with the netezza architecture. Technical report sand2006-3640, Sandia National Laboratories, April 2006.

[20] W. R. Davis, J. Wilson, S. Mick, J. Xu, H. Hua, C. Mineo, A. M. Sule, M. Steer, and P. D. Franzon. Demystifying 3d ics: the pros and cons of going vertical. *Design Test of Computers, IEEE*, 22(6):498–510, Nov 2005.

[21] M. F. Deering, M. G. Lavelle, and S. A. Schlapp. A cached vram for 3d graphics. *HotChips VI*, 1994.

[22] M. F. Deering, S. A. Schlapp, and M. G. Lavelle. Fbram: A new form of memory optimized for 3d graphics. In *Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '94, pages 167–174, New York, NY, USA, 1994. ACM.

[23] R. H. Dennard. Field-effect transistor memory, jun 4 1968. US Patent 3,387,286.

[24] M. Deo. Enabling next-generation platforms using altera's 3d system-in-package technology. Whitepaper, Altera, June 2015.

[25] M. Deo, J. Schulz, and L. Brown. Stratix 10 mx devices solve the memory bandwidth challenge. Whitepaper, Altera, now part of Intel, May 2016.

[26] P. Dlugosch, D. Brown, P. Glendenning, M. Leventhal, and H. Noyes. An efficient and scalable semiconductor architecture for parallel automata processing. *IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS*, 25(12):3088–3098, Dec 2014.

[27] J. Easton. In-memory computing - next generation technologies. November 2013.

[28] R. Egawa, M. Sato, J. Tada, and H. Kobayashi. Vertically integrated processor and memory module design for vector supercomputers. In *3DIC*, pages 1–6, 2013.

[29] D. G. Elliott, W. M. 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*, pages 30–6, May 1992.

[30] D. G. Elliott, M. Stumm, W. M. Snelgrove, C. Cojocaru, and R. McKenzie. Computational ram: implementing processors in memory. *Design Test of Computers, IEEE*, 16(1):32–41, Jan 1999.

[31] Z. Fang, L. Zhang, J. B. Carter, A. Ibrahim, and M. A. Parker. Active memory operations. In *Proceedings of the 21st Annual International Conference on Supercomputing*, ICS '07, pages 232–241, New York, NY, USA, 2007. ACM.

[32] Z. Fang, L. Zhang, J. B. Carter, S. A. McKee, A. Ibrahim, M. A. Parker, and X. Jiang. Active memory controller. *The Journal of Supercomputing*, 62(1):510–549, 2012.

[33] A. Farmahini-Farahani, J. H. Ahn, K. Morrow, and N. S. Kim. Drama: An architecture for accelerated processing near memory. *Computer Architecture Letters*, 14(1):26–29, Jan 2015.

[34] B. G. Fitch, A. Rayshubskiy, M. C. Pitman, T. J. C. Ward, and R. S. Germain. Using the active storage fabrics model to address petascale storage challenges. In *Proceedings of the 4th Annual Workshop on Petascale Data Storage*, PDSW '09, pages 47–54, New York, NY, USA, 2009. ACM.

[35] H. Fuchs and J. Poulton. Pixel-planes: A vlsi-oriented design for a raster graphics engine. In *VLSI-DESIGN*, 81(3), pages 20–28, 1981.

[36] M. Gao, G. Ayers, and C. Kozyrakis. Practical near-data processing for in-memory analytics frameworks. In *Proceedings of the 24th International Conference on Parallel Architectures and Compilation*, PACT '15, New York, NY, USA, 2015. ACM.

[37] A. Gara. The long term impact of codesign. In *High Performance Computing, Networking, Storage and Analysis (SCC), 2012 SC Companion:*, pages 2212–2246, Nov 2012.

[38] M. Gokhale, B. Holmes, and K. Iobst. Processing in memory: The terasys massively parallel pim array. *Computer*, 28(4):23–31, Apr 1995.

[39] M. H. Hajkazemi, M. K. Tavana, and H. Homayoun. Wide i/o or lpddr?: Exploration and analysis of performance, power and temperature trade-offs of emerging dram technologies in embedded mpsocs. In *Computer Design (ICCD), 2015 33rd IEEE International Conference on*, pages 62–69, Oct 2015.

[40] P. Hammarlund, A. J. Martinez, A. A. Bajwa, D. L. Hill, E. Hallnor, H. Jiang, M. Dixon, M. Derr, M. Hunsaker, R. Kumar, R. B. Osborne, R. Rajwar, R. Singhal, R. D'Sa, R. Chappell, S. Kaushik, S. Chennupaty, S. Jourdan, S. Gunther, T. Piazza, and T. Burton. Haswell: The fourth-generation intel core processor. *Micro, IEEE*, 34(2):6–20, Mar 2014.

[41] R. A. Haring, M. Ohmacht, T. W. Fox, M. K. Gschwind, D. L. Satterfield, K. Sugavanam, P. W. Coteus, P. Heidelberger, M. A. Blumrich, R. W. Wisniewski, A. Gara, G. L. T. Chiu, P. A. Boyle, N. H. Chist, and C. Kim. The ibm blue gene/q compute chip. *Micro, IEEE*, 32(2):48–60, March 2012.

[42] N. Hemsoth. The tiny chip that could disrupt exascale computing. The Next Platform, March 2015. http://www.nextplatform.com/2015/03/12/the-little-chip-that-could-disrupt-exascale-computing/.

[43] J. L. Hennessy and D. A. Patterson. *Computer Architecture, Fourth Edition: A Quantitative Approach*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2006.

[44] J. L. Hennessy and D. A. Patterson. *Computer Architecture, Fifth Edition: A Quantitative Approach*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 5th edition, 2011.

[45] J. Hruska. Beyond ddr4: The differences between wide i/o, hbm, and hybrid memory cube. Online, Jan 2015. ExtremTech.

[46] Hybrid Memory Cube Consortium. Hybrid memory cube specification 2.1. Technical report, 2014.

[47] Intel. 2015 annual report. Form 10-k, March 2016.

[48] Intel. Intel developer forum (idf16). Shenzhen, April 2016.

[49] ISSCC. Isscc 2014 trends. Technical report, 2014.

[50] J. Jeddeloh and B. Keeth. Hybrid memory cube new dram architecture increases density and performance. In *VLSI*

Technology (VLSIT), 2012 Symposium on, pages 87–88, June 2012.

[51] K. Jothi, M. Sharafeddine, and H. Akkary. Simultaneous continual flow pipeline architecture. In *Computer Design (ICCD), 2011 IEEE 29th International Conference on*, pages 127–134, Oct 2011.

[52] A. Kagi, J. R. Goodman, and D. Burger. Memory bandwidth limitations of future microprocessors. In *Computer Architecture, 1996 23rd Annual International Symposium on*, pages 78–78, May 1996.

[53] J. Kahle. The cell processor architecture. In *Microarchitecture, 2005. MICRO-38. Proceedings. 38th Annual IEEE/ACM International Symposium on*, pages 3–3, Nov 2005.

[54] Y. Kang, W. Huang, S.-M. Yoo, D. Keen, Z. Ge, V. Lam, P. Pattnaik, and J. Torrellas. Flexram: toward an advanced intelligent memory system. In *Computer Design, 1999. (ICCD '99) International Conference on*, pages 192–201, 1999.

[55] W. H. Kautz. Cellular logic-in-memory arrays. *Computers, IEEE Transactions on*, C-18(8):719–727, Aug 1969.

[56] S. Kaxiras, R. Sugumar, and J. Schwarzmeier. Distributed vector architecture: Beyond a single vector-iram. In *In First Workshop on Mixing Logic and DRAM: Chips that Compute and Remember*, 1997.

[57] C. Keable. Data centric deep computing (dc2), 2012.

[58] M. J. Khurshid and M. Lipasti. Data compression for thermal mitigation in the hybrid memory cube. In *Computer Design (ICCD), 2013 IEEE 31st International Conference on*, pages 185–192, Oct 2013.

[59] Y. Kim, T.-D. Han, S.-D. Kim, and S.-B. Yang. An effective memory-processor integrated architecture for computer vision. In *Parallel Processing, 1997., Proceedings of the 1997 International Conference on*, pages 266–269, Aug 1997.

[60] Y. Kim and Y. H. Song. Analysis of thermal behavior for 3d integration of dram. In *Consumer Electronics (ISCE 2014), The 18th IEEE International Symposium on*, pages 1–2, June 2014.

[61] M. B. Kleiner, S. A. Kuhn, P. Ramm, and W. Weber. Performance improvement of the memory hierarchy of risc-systems by application of 3-d technology. *Components, Packaging, and Manufacturing Technology, Part B: Advanced Packaging, IEEE Transactions on*, 19(4):709–718, Nov 1996.

[62] G. Knittel and A. Schilling. Eliminating the z-buffer bottleneck. In *European Design and Test Conference, 1995. ED TC 1995, Proceedings.*, pages 12–16, Mar 1995.

[63] G. Knittel, A. Schilling, and W. Straßer. *High Performance Computing for Computer Graphics and Visualisation: Proceedings of the International Workshop on High Performance Computing for Computer Graphics and Visualisation, Swansea 3–4 July 1995*, chapter GRAMMY: High Performance Graphics Using Graphics Memories, pages 33–48. Springer London, London, 1996.

[64] P. Kogge. Exascale computing study: Technology challenges in achieving exascale systems. Technical Report TR-2008-13, University of Note Dame, 2008.

[65] P. M. Kogge. Execube-a new architecture for scaleable mpps. In *Parallel Processing, 1994. Vol. 1. ICPP 1994. International Conference on*, volume 1, pages 77–84, Aug 1994.

[66] P. M. Kogge. Exploring the possible past futures of a single part type multi-core pim chip. In *Innovative Architecture for Future Generation High Performance (IWIA), 2010 International Workshop on*, pages 30–38, Jan 2010.

[67] P. M. Kogge. *Updating the Energy Model for Future Exascale Systems*, chapter High Performance Computing: 30th International Conference, ISC High Performance 2015, Frankfurt, Germany, July 12-16, 2015, Proceedings, pages 323–339. Springer International Publishing, Cham, 2015.

[68] P. M. Kogge, J. B. Brockman, T. Sterling, and G. Gao. Processing in memory: Chips to petaflops. In *In Workshop on Mixing Logic and DRAM: Chips that Compute and Remember at ISCA '97*, 1997.

[69] P. M. Kogge, P. La Fratta, and M. Vance. Facing the exascale energy wall. In *Innovative Architecture for Future Generation High Performance (IWIA), 2010 International Workshop on*, pages 51–58, Jan 2010.

[70] P. M. Kogge, T. Sunaga, H. Miyataka, K. Kitamura, and E. Retter. Combined dram and logic chip for massively parallel systems. In *Advanced Research in VLSI, 1995. Proceedings., Sixteenth Conference on*, pages 4–16, Mar 1995.

[71] A. Kopser and D. Vollrath. Overview of the next generation cray xmt. In *53rd Cray User Group meeting, CUG 2011*, Fairbanks, Alaska, 2011.

[72] A. Kugler, G. Knittel, A. G. Schilling, and W. Straßer. High-performance texture mapping architectures. In *Proceedings of the 6th OMI Annual Conference on Embedded Microprocessor Systems*, pages 189–198. IOS Press, sep 1996.

[73] G. Kyriazis. Heterogeneous system architecture: A technical review. Whitepaper, AMD, August 2012.

[74] E. S. Larsen and D. McAllister. Fast matrix multiplies using graphics hardware. In *Proceedings of the 2001 ACM/IEEE Conference on Supercomputing*, SC '01, pages 55–55, New York, NY, USA, 2001. ACM.

[75] D. U. Lee, K. W. Kim, K. W. Kim, H. Kim, J. Y. Kim, Y. J. Park, J. H. Kim, D. S. Kim, H. B. Park, J. W. Shin, and et. al. 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*, pages 432–433, Feb 2014.

[76] D. U. Lee, K. W. Kim, K. W. Kim, K. S. Lee, S. J. Byeon, J. H. Kim, J. H. Cho, J. Lee, and J. H. Chun. A 1.2v 8gb 8-channel 128gb/s high-bandwidth memory (hbm) stacked dram with effective i/o test circuits. *Solid-State Circuits, IEEE Journal of*, 50(1):191–203, Jan 2015.

[77] C. C. Liu, I. Ganusov, M. Burtscher, and S. Tiwari. Bridging the processor-memory performance gap with 3d ic technology. *Design Test of Computers, IEEE*, 22(6):556–564, Nov 2005.

[78] G. Loh, N. Jayasena, M. Oskin, M. Nutter, D. Roberts, M. Meswani, D. P. Zhang, and M. Ignatowski. A processing in memory taxonomy and a case for studying fixed-function pim. In *WoNDP: 1st Workshop on Near-Data Processing in conjunction with the 46th IEEE/ACM International Symposium on Microarchitecture (MICRO-46)*, 2013.

[79] K. Mai, T. Paaske, N. Jayasena, R. Ho, W. J. Dally, and M. Horowitz. Smart memories: a modular reconfigurable architecture. In *Computer Architecture, 2000. Proceedings of the 27th International Symposium on*, pages 161–171, June 2000.

[80] M. Martonosi. Power-aware computing: Then, now, and into the future. 2014.

[81] J. Menon, L. De Carli, V. Thiruvengadam, K. Sankaralingam, and C. Estan. Memory processing units, 2014.

[82] Micron. 2016 analyst conference positioned for success, Feb 2016. http://files.shareholder.com/downloads/ABEA-45YXOQ/1517834575x0x875021/4BEAA02E-BBC2-402C-A51D-B3B2C6B8C3D4/Winter_Analyst_Day_2016.pdf.

[83] R. C. Minnick. A survey of microcellular research. *J. ACM*, 14(2):203–241, apr 1967.

[84] R. C. Minnick, J. Goldberg, M. W. Green, W. H. Kautz, R. A. Short, H. S. Stone, and M. Yoeli. Cellular arrays for logic and storage. Final rept., Stanford Research Institute, Menlo Park, Calif., April 1966.

[85] M. Minutoli, S. Kuntz, A. Tumeo, and P. Kogge. Implementing radix sort on emu 1. In *In the 3rd Workshop on Near-Data Processing (WoNDP)*, Waikiki, Hawaii, 2015.

[86] Mitsubishi, Electronic Device Group. 3d-ram: Frame buffer memory for high-performance 3d graphics. Data book, Mitsubishi, 1996.

[87] N. Miura, Y. Koizumi, E. Sasaki, Y. Take, H. Matsutani, T. Kuroda, H. Amano, R. Sakamoto, M. Namiki, K. Usami, M. Kondo, and H. Nakamura. A scalable 3d heterogeneous multi-core processor with inductive-coupling thruchip interface. In *Cool Chips XVI (COOL Chips), 2013 IEEE*, pages 1–3, April 2013.

[88] G. E. Moore. Cramming more components onto integrated circuits. *Proceedings of the IEEE*, 86(1):82–85, Jan 1998.

[89] T. P. Morgan. Putting mroe brains in the network frees up compute. The Next Platform, June 2016. http://www.nextplatform.com/2016/06/08/putting-brains-network-frees-compute/.

[90] M. Motoyoshi. Through-silicon via (tsv). *Proceedings of the IEEE*, 97(1):43–48, Jan 2009.

[91] C. Muller-Schloer, F. Geerinckx, and B. Stanford-Smit, editors. *Embedded Microprocessor Systems.* IOS Press, Amsterdam, The Netherlands, 1st edition, 1996.

[92] R. Nair, S. F. Antao, C. Bertolli, P. Bose, J. R. Brunheroto,

T. Chen, C. Cher, C. H. A. Costa, J. Doi, C. Evangelinos, B. M. Fleischer, T. W. Fox, D. S. Gallo, L. Grinberg, J. A. Gunnels, A. C. Jacob, P. Jacob, H. M. Jacobson, T. Karkhanis, C. Kim, J. H. Moreno, J. K. O'Brien, M. Ohmacht, Y. Park, D. A. Prener, B. S. Rosenburg, K. D. Ryu, O. Sallenave, M. J. Serrano, P. D. M. Siegl, K. Sugavanam, and Z. Sura. Active memory cube: A processing-in-memory architecture for exascale systems. *IBM Journal of Research and Development*, 59(2/3):17–1, March 2015.

[93] Nvidia. Nvidia tesla p100 - the most advanced datacenter accelerator ever built featuring pascal gp100, the world's fastest gpu. Whitepaper, 2016.

[94] A. Olofsson, T. Nordström, and Z. Ul-Abdin. Kickstarting high-performance energy-efficient manycore architectures with epiphany. In *Signals, Systems and Computers, 2014 48th Asilomar Conference on*, pages 1719–1726, Nov 2014.

[95] D. Patterson, T. Anderson, N. Cardwell, R. Fromm, K. Keeton, C. Kozyrakis, R. Thomas, and K. Yelick. Intelligent ram (iram): chips that remember and compute. In *Solid-State Circuits Conference, 1997. Digest of Technical Papers. 43rd ISSCC., 1997 IEEE International*, pages 224–225, Feb 1997.

[96] D. Patterson, K. Asanovic, A. Brown, R. Fromm, J. Golbus, B. Gribstad, K. Keeton, C. Kozyrakis, D. Martin, S. Perissakis, R. Thomas, N. Treuhaft, and K. Yelick. 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*, pages 2–7, Oct 1997.

[97] D. A. Patterson. Latency lags bandwith. *Commun. ACM*, 47(10):71–75, oct 2004.

[98] D. A. Patterson. Future of computer architecture. Berkeley EECS Annual Research Symposium (BEARS), Feb 2006.

[99] J. T. Pawlowski. Hybrid memory cube (hmc). HOT CHIPS 23, August 2011.

[100] F. J. Pollack. New microarchitecture challenges in the coming generations of cmos process technologies (keynote address)(abstract only). In *Proceedings of the 32Nd Annual ACM/IEEE International Symposium on Microarchitecture*, MICRO 32, page 2, Washington, DC, USA, 1999. IEEE Computer Society.

[101] P. Ranganathan. From microprocessors to nanostores: Rethinking data-centric systems. *Computer*, 44(1):39–48, Jan 2011.

[102] S. F. Reddaway. Dap - a distributed array processor. *SIGARCH Comput. Archit. News*, 2(4):61–65, dec 1973.

[103] E. Riedel, C. Faloutsos, G. A. Gibson, and D. Nagle. Active disks for large-scale data processing. *Computer*, 34(6):68–74, jun 2001.

[104] B. M. Rogers, A. Krishna, G. B. Bell, K. Vu, X. Jiang, and Y. Solihin. Scaling the bandwidth wall: Challenges in and avenues for cmp scaling. *SIGARCH Comput. Archit. News*, 37(3):371–382, jun 2009.

[105] D. F. Rogers and R. Earnshaw. *State of the Art in Computer Graphics: Visualization and Modeling.* Springer Publishing Company, Incorporated, 1st edition, 2012.

[106] K. Sakuma, P. Andry, K. Sueoka, R. Horton, S. Wright, Y. Oyama, B. Webb, C. Patel, B. Dang, C. Tsang, E. Sprogis, R. Polastre, and J. Knickerbocker. Die cavity integration technology for through-silicon-vias stacking. San Diego, CA, Sept 2008.

[107] SanDisk. Sandisk and hp launch partnership to create memory-driven computing solutions, Oct 2015. https://www.sandisk.com/about/media-center/press-releases/2015/sandisk-and-hp-launch-partnership.

[108] A. Saulsbury, F. Pong, and A. Nowatzyk. Missing the memory wall: The case for processor/memory integration. In *Proceedings of the 23rd Annual International Symposium on Computer Architecture*, ISCA '96, pages 90–101, New York, NY, USA, 1996. ACM.

[109] M. Scrbak, M. Islam, K. Kavi, M. Ignatowski, and N. Jayasena. Processing-in-memory: Exploring the design space. In L. M. P. Pinho, W. Karl, A. Cohen, and U. Brinkschulte, editors, *Architecture of Computing Systems – ARCS 2015*, volume 9017, chapter Lecture Notes in Computer Science, pages 43–54. Springer International Publishing, 2015.

[110] Seagate. Seagate demonstrates fastest-ever ssd flash drive. Press Release, March 2016. http://www.seagate.com/de/de/about-seagate/news/seagate-demonstrates-fastest-ever-ssd-flash-drive-pr/.

[111] T. Semiconductor. Tezzaron unveils 3d sram, January 2005.

[112] I. T. R. F. Semiconductors. Itrs 2.0 system integration whitepaper. Technical report, Dec 2014.

[113] G. Shainer. Intelligent networks: A new co-processor emerges. The Next Platform, March 2016. http://www.nextplatform.com/2016/03/02/intelligent-networks-a-new-co-processor-emerges/.

[114] J. M. Shalf and R. Leland. Computing beyond moore's law. *Computer*, 48(12):14–23, Dec 2015.

[115] T. Shimizu, J. Korematu, M. Satou, H. Kondo, S. Iwata, K. Sawai, and et. al. A multimedia 32b risc microprocessor with 16mb dram. In *Solid-State Circuits Conference, 1996. Digest of Technical Papers. 42nd ISSCC., 1996 IEEE International*, pages 216–217, Feb 1996.

[116] P. Siegl, R. Buchty, and M. Berekovic. Revealing potential performance improvements by utilizing hybrid work-sharing for resource-intensive seismic applications. In *Proceedings of the 2015 23rd Euromicro International Conference on Parallel, Distributed, and Network-Based Processing.* IEEE Computer Society, Mar 2015.

[117] P. Stanley-Marbell, V. C. Cabezas, and R. P. Luijten. Pinned to the walls - impact of packaging and application properties on the memory and power walls. In *Low Power Electronics and Design (ISLPED) 2011 International Symposium on*, pages 51–56, Aug 2011.

[118] W. J. Starke, J. Stuecheli, D. M. Daly, J. S. Dodson, F. Auernhammer, P. M. Sagmeister, G. L. Guthrie, C. F. Marino, M. Siegel, and B. Blaner. The cache and memory subsystems of the ibm power8 processor. *IBM Journal of Research and Development*, 59(1):3–1, Jan 2015.

[119] H. S. Stone. A logic-in-memory computer. *Computers, IEEE Transactions on*, C-19(1):73–78, Jan 1970.

[120] T. Thorolfsson, N. Moezzi-Madani, and P. D. Franzon. A low power 3d integrated fft engine using hypercube memory division. In *ISLPED*, ISLPED '09, pages 231–236, New York, NY, USA, 2009. ACM.

[121] S. Thoziyoor, J. Brockman, and D. Rinzler. Pim lite: A multithreaded processor-in-memory prototype. In *Proceedings of the 15th ACM Great Lakes Symposium on VLSI*, GLSVLSI '05, pages 64–69, New York, NY, USA, 2005. ACM.

[122] T. Trader. Mellanox touts arrival of intelligent interconnect. HPCwire, November 2015. http://www.hpcwire.com/2015/11/16/mellanox-touts-arrival-of-intelligent-interconnect/.

[123] J. von Neumann. First draft of a report on the edvac. *Annals of the History of Computing, IEEE*, 15(4):27–75, 1993.

[124] S. Vongehr and X. Meng. The missing memristor has not been found. In *Nature Scientific Reports*, volume 5. Macmillan Publishers Limited, 2015.

[125] M. M. WALDROP. More than moore. *NATURE*, 530:144–147, feb 2016.

[126] D. L. Weaver and T. Germond. The sparc architecture manual, version 9. Technical report, SPARC International, Inc., San Jose, California, 1994.

[127] S. Williams, A. Waterman, and D. Patterson. Roofline: An insightful visual performance model for multicore architectures. *Commun. ACM*, 52(4):65–76, apr 2009.

[128] W. A. Wulf and S. A. McKee. Hitting the memory wall: Implications of the obvious. *SIGARCH Comput. Archit. News*, 23(1):20–24, mar 1995.

[129] Y. Xie. Future memory and interconnect technologies. In *Design, Automation Test in Europe Conference Exhibition (DATE), 2013*, pages 964–969, March 2013.

[130] D. Zhang, N. Jayasena, A. Lyashevsky, J. L. Greathouse, L. Xu, and M. Ignatowski. Top-pim: Throughput-oriented programmable processing in memory. In *Proceedings of the 23rd International Symposium on High-performance Parallel and Distributed Computing*, HPDC '14, pages 85–98, New York, NY, USA, 2014. ACM.

[131] D. P. Zhang, N. Jayasena, A. Lyashevsky, J. Greathouse, M. Meswani, M. Nutter, and M. Ignatowski. A new perspective on processing-in-memory architecture design. In *Proceedings of the ACM SIGPLAN Workshop on Memory Systems Performance and Correctness*, MSPC '13, pages 7–1, New York, NY, USA, 2013. ACM.

[132] Q. Zhu, B. Akin, H. E. Sumbul, F. Sadi, J. C. Hoe, L. Pileggi, and F. Franchetti. A 3d-stacked logic-in-memory accelerator for application-specific data intensive computing. In *3DIC*, pages 1–7, Oct 2013.