

Résumé

Les architectures d'aujourd'hui sont basées sur le modèle de von Neumann qui place au centre l'exécution des instructions. Ces architectures font face à de fortes limitations dans le contexte du *big data*. En effet, le mur mémoire est un phénomène lié à l'écart grandissant de performances entre les processeurs et les mémoires depuis les années 80. Pour atténuer cet écart, une hiérarchie de caches a été mise en place mais elle a en contrepartie largement augmentée la consommation énergétique sans être adaptée pour les grands jeux de données modernes. Non seulement ces architectures ont du mal avec une masse de données toujours croissantes à cause de leur haute consommation énergétique et leur faible débit, elles ne peuvent plus uniquement se baser sur les avancées technologiques pour s'améliorer. Ceci appelle à un changement de paradigme vers des architectures *data* centrées où le traitement de quantités de données massives en parallèle est le principe de base.

De nouvelles mémoires non volatiles promettent du stockage haute densité et peuvent intégrer du calcul en mémoire. L'intérêt de calculer en mémoire est d'opérer là où se trouve la donnée, ou tout du moins le plus proche possible, pour supprimer les allées et venues permanentes entre la mémoire et les cœurs de calcul. Les solutions existantes utilisent du calcul analogique très efficace mais prompt au bruit et avec une flexibilité limitée. Quand les données doivent être réécrites en mémoire, l'endurance de ces mémoires non volatiles n'est pas discutée. Nous concevons un emballage numérique qui étend les fonctionnalités mémoire avec du calcul vectoriel et développons une plateforme de simulation pour faire de l'exploration architecturale. Notre circuit, bien nommé C-SRAM, peut être intégré avec la plupart des technologies mémoire et est équipé de sa propre mémoire SRAM. Nous démontrons qu'effectuer le calcul au sommet de la hiérarchie mémoire, c'est à dire proche du stockage permanent, permet une réduction de la consommation énergétique d'un facteur 17.4 et une accélération du traitement en moyenne d'un facteur 12.9 comparé à un traitement avec un cœur SIMD. Grâce à la mémoire tampon intégrée, l'endurance de la mémoire non volatile n'est pas impactée et de fait, l'espérance de vie du système s'en trouve augmentée par rapport à d'autres solutions de calcul en mémoire.

Mots clés : calcul en mémoire, mémoire non volatile, architecture des systèmes, mémoire de classe de stockage, mur mémoire, mur énergétique, goulot d'étranglement de von Neumann

Abstract

Today computing centric von Neumann architectures face strong limitations in the data-intensive context of numerous applications. The key limitation is the memory wall due to increased performance gap between processors and memories. To mitigate this gap, cache hierarchy was introduced but it largely increased energy consumption while not being adapted for modern big datasets. Not only those architectures struggle with big datasets due to their high energy consumption and slow bandwidth, they can no longer be improved through technological advances such as node scaling. This calls for a paradigm shift to data centric architecture where treating massive amounts of data in a parallel fashion is the core principle.

New emerging Non-Volatile Memories (NVM) promise high density data storage and can easily integrate In-Memory Computing (IMC). IMC purposes is to compute where the data is or the closest to, to suppress back and forth data movements from the memory to the cores. Existing solutions use analog computing that has high efficiency but limited flexibility. When data needs to be written back after computation, endurance of NVM is often not discussed. We design a digital wrapper that extends memory functionality with vector computing capabilities and develop a simulation platform for architecture exploration. Our digital wrapper, aka C-SRAM, can be integrated with most memory technologies and comes with its own small SRAM buffer. We demonstrate that computing at the top of the memory hierarchy, i.e. close to the permanent storage, grants in average $17.4\times$ energy reduction and $12.9\times$ speed-up versus SIMD baseline. Thanks to SRAM buffer, NVM's endurance is not impaired and thereby extends system lifetime compared to other IMC solutions.

Keywords: memory wall, energy wall, von Neumann bottleneck, in-memory computing, non volatile memories, system architecture, storage class memory

Remerciements

Censored for review



Contents

Résumé	i
Abstract	ii
Remerciements	iii
Contents	iv
List of Figures	vii
List of Tables	x
List of acronyms	xi
Glossary	xv
1 Introduction & Contextualisation	2
1.1 The end of technology advancement	3
1.1.1 Physical limits	3
1.1.2 Architecture improvements	5
1.1.3 Socioeconomic impacts	8
1.2 Memory technologies	8
1.2.1 Main memory technologies	9
1.2.2 Emerging non volatile memories	14
1.3 A new computing paradigm	22
1.3.1 Big Data	22
1.3.2 Proposed solution : memory computing	23
1.4 Conclusion	26
2 State-of-the-Art	28
2.1 Classification of different topologies/Nomenclature/Taxonomy	29
2.2 Memory computing	30
2.2.1 SRAM	30
2.2.2 DRAM	33
2.2.3 NVM and SCM	34
2.2.4 Other works	39
2.3 Conclusion	40
3 CSRAM Design	44

3.1	Motivations for a digital wrapper using standard design flow	45
3.2	General design	48
3.2.1	Specification	48
3.2.2	ALU design	51
3.2.3	Pipeline design	52
3.3	Obtained results	54
3.3.1	Workflow	54
3.3.2	Simulation Results	55
4	Simulation platform & Tools	60
4.1	Used benchmarks	61
4.1.1	Linear benchmarks	61
4.1.2	Quadratic benchmarks	62
4.1.3	Cubic benchmarks and real application	63
4.2	Existing platforms	64
4.2.1	Analytic model	65
4.2.2	Hardware counters	66
4.2.3	Simulation platforms	67
4.3	Hardware model tools	70
4.3.1	NVSim	70
4.3.2	DRAM	72
4.3.3	C-SRAM	74
4.4	Platform	76
4.4.1	Software interface for benchmarks	77
4.4.2	First version with hard coherency	78
4.4.3	Improved version with soft coherency and real disk accesses	80
4.4.4	Caches and DRAM validation	82
5	IMC/NMC Computing Architecture	87
5.1	Reference SIMD 512-bit architecture	89
5.2	Computing at the top level in the memory hierarchy	89
5.2.1	Scenario NVM 1 : Independent C-SRAM between DRAM and NVM	90
5.2.2	Scenario NVM 2	98
5.2.3	Scenario NVM 1 with page transfer	100
5.2.4	Impact of the reduction loop	100
5.3	Computing near DRAM	102
5.3.1	Scenario DRAM 1 : Independent C-SRAM between DRAM and L3 cache	102
5.3.2	Scenario DRAM 2: DRAM row buffer	105
5.4	Conclusion	108
	Conclusion	111
	Perspectives and future works	117
	Bibliography	118

List of Figures

1.1	ITRS roadmap as of 2020	3
1.2	Transistor leakage and gate length evolution	4
1.3	Predicted scaling cost in 2010 for 2018	4
1.4	Evolution of CPU frequency	5
1.5	Power density evolution	5
1.6	Frequency scaling of Intel Xeon core	6
1.7	CPU and memory performance trends	6
1.8	Instruction energy breakthrough	8
1.9	Cost of chips and investment needed	9
1.10	SRAM bitcell circuit diagram	10
1.11	DRAM bitcell circuit diagram	11
1.12	Example of a DRAM addressing scheme	12
1.13	Memory hierarchy	13
1.14	Die photographs	15
1.15	Different RRAM resistance probability distribution	16
1.16	Circuit diagrams of 3 different bitcell types	17
1.17	Different types of RRAM bitcell	18
1.18	Different types of PCM bitcell	19
1.19	Different types of MRAM bitcell	20
1.20	Quantity of data created per year	23
1.21	Internal versus external memory bandwidth	24
1.22	Memory computing research interest in Google Scholar (from [55]). Search term are not precise enough as they include some result from neurology (<i>background noise</i>).	25
2.1	Taxonomy	29
2.2	Non standard SRAM bitcells used to implement IMC	31
2.3	NVM lifetime for different endurances	42
3.1	Proposed design methodology	46
3.2	Different way of layouting the memories in the C-SRAM	47
3.3	Potential of the double-pump technique combined with our digital wrapper for better pipeline efficiency	48
3.4	Scalar, vector and scalar/vector computing architectures	49
3.5	Defined ISA for 32-bit system	51
3.6	Base implementation of our digital wrapper	52
3.7	C-SRAM module hierarchy for 1RW memory	54

3.8	Design workflow used for C-SRAM digital wrapper	55
3.9	Area and power ratio for different kind of memories (lower is better) . .	56
3.10	Energy vs delay for the MAC instruction	57
3.11	Throughput of different SRAM memories using double-pump technique	57
3.12	Efficiency vs throughput of our solution and state-of-the-art works . .	58
3.13	Different place and routed floorplans	59
4.1	Neural network core functions distribution	62
4.2	Darknet callgraph for image classification	64
4.3	Simplified view of an Intel Skylake core memory system	65
4.4	Cache access either sequential or parallel	71
4.5	C-SRAM tiling energy vs timing access	75
4.6	Extended ISA for 64-bit system. Upper 64 bits are address bits and lower 64 are data bits.	77
4.7	Our work normalized against hardware counters for cache events . . .	84
4.8	DRAM tools timing ( left y-axis) and energy ( right y-axis) es- timation (geometric mean of all benchmarks). Timing is normalized against benchmark execution time and energy is normalized against VAMPIRE.	85
4.9	Computed power reported by different tools using our benchmark suite	85
4.10	Simulation methodology: main steps	86
5.1	Different integration possibility of the C-SRAM within the memory hier- archy	88
5.2	Reference architecture and memories parameters	89
5.3	Energy reduction and speedup for linear benchmarks normalized against SIMD 512-bit reference (higher is better)	90
5.4	Energy reduction and speedup for linear benchmark with high SCM access rate normalized against SIMD 512-bit reference (higher is better)	91
5.5	Energy reduction and speedup for quadratic benchmarks normalized against SIMD 512-bit reference (higher is better)	92
5.6	Relative <i>atax</i> (to SIMD Reference) Energy and Timing distribution for different size and a vector width of 128 bytes (lower is better)	93
5.7	Energy reduction and speedup for cubic benchmarks normalized against SIMD 512-bit reference (higher is better)	94
5.8	Relative <i>gemm</i> (to SIMD Reference) Energy and Timing distribution for different total C-SRAM size and a vector width of 4 kB (lower is better) .	95
5.9	Energy and timing distribution for a C-SRAM of 512 kB and vector size of 512 B normalized to SIMD 512-bit reference architecture (lower is better)	96
5.10	Caches energy and timing distribution for a C-SRAM of 512 kB and vector size of 512 B normalized to SIMD 512-bit reference architecture (lower is better)	97
5.11	NVM memory access for a C-SRAM of 512 kB and vector size of 512 B normalized to SIMD 512-bit reference architecture (lower is better) . .	98

5.12	Best, worst and average (◆) of <i>all</i> cases for both energy reduction (■) and speedup (■) normalized against SIMD reference	98
5.13	Detailed memory hierarchy for NVM row buffer	99
5.14	Energy reduction and speedup of NVM row buffer scenario normalized against SIMD reference (top row), against independent C-SRAM (bottom row) and averaged for one vector size through all total sizes (higher is better)	100
5.15	Energy reduction and speedup of page transfer scenario normalized against SIMD reference (top row), against independent C-SRAM (bottom row) and averaged for one total size through all vector sizes (higher is better)	101
5.16	Energy reduction and speedup when performing reduction loop inside C-SRAM compared to the independent C-SRAM scenario and averaged for one vector size through all total sizes (higher is better)	102
5.17	Energy reduction and speedup for linear benchmarks normalized against SIMD 512-bit reference (higher is better)	103
5.18	Energy reduction and speedup for quadratic benchmarks normalized against SIMD 512-bit reference (higher is better)	104
5.19	Energy reduction and speedup for cubic benchmarks normalized against SIMD 512-bit reference (higher is better)	104
5.22	Best, worst and average (◆) of <i>all</i> cases for both energy reduction (■) and speedup (■) normalized against SIMD reference	106
5.20	Energy and timing distribution for a C-SRAM of 512 kB and vector size of 512 B normalized to SIMD 512-bit reference architecture (lower is better)	107
5.21	Energy reduction and speedup when performing reduction loop inside C-SRAM normalized against independent C-SRAM at DRAM level and averaged for one total size through all vector sizes (higher is better) . .	107
5.23	Minimum, maximum and average for each benchmark and tested scenario	110

List of Tables

1.1	Main memories key parameters. Data is from [22, 23].	14
1.2	Non volatile memories parameters	21
3.1	Versions used for design workflow	55
4.1	Benchmarks parameters	63
4.2	Simulation vs Instrumentation	68
4.3	Platforms overview	69
4.4	NVSim's parameter used to design caches	71
4.5	Energy and latency of the selected PCRAM	72
4.6	Comparison of different DRAM simulation tools	74
4.7	Tiling timing and energy factor	75
5.1	Reference architecture memories parameters	89
5.2	Best parameters for energy reduction, speedup and energy-delay product	109

List of acronyms

ADC	Analog Digital Converter. x , 31 , 35 , 38 , 41 , 43 , 45
AI	Artificial Intelligence. x , 7 , 22 , 27 , 32–34 , 61 , 112
ALU	Arithmetic & Logical Unit. x , 25 , 46 , 49 , 51 , 53 , 54 , 59 , 113
API	Application Programming Interface. x , 26 , 94 , <i>Glossary: API</i>
ASIC	Application Specific Integrated Circuit. x , 7
BCAM	Binary Content Addressable Memory. x
BEOL	Back-End Of Line. x
BLAS	Basic Linear Algebra Subprograms. x , 62 , 63 , 94 , 114
BRAM	Block Random Access Memory. x , 7
CAM	Content Addressable Memory. x , 10 , 32
CBRAM	Conductive Bridge Random Access Memory. x , 18
CF	Conductive Filament. x , 18
CIM	Computing In-Memory. x , 29 , 33 , 37
CMOS	Complementary Metal Oxide Semiconductor. x
CNN	Convolutional Neural Network. x , 61 , 63
CPI	Cycles Per Instruction. x
CPU	Central Processing Unit. x , 3 , 5–11 , 13 , 17 , 22 , 24 , 26 , 30–34 , 37 , 39 , 40 , 42 , 50–52 , 61 , 66–69 , 73 , 76 , 79 , 81 , 89 , 93–96 , 100–102 , 105 , 111–117
CSRAM	Computational Static Random Access Memory. x , 88
DBT	Dynamic Binary Translation. x , 68 , 69
DLP	Data Level Parallelism. x , 5 , 6
DRAM	Dynamic Random Access Memory. x , 6 , 7 , 9–18 , 20 , 22 , 24 , 25 , 27 , 28 , 32–35 , 37 , 40 , 51 , 64 , 66 , 67 , 70 , 71 , 73–75 , 79 , 81–86 , 88–90 , 92–100 , 102–106 , 108 , 111 , 112 , 114–117
DSP	Digital Signal Processor. x , 7
DVFS	Dynamic Voltage and Frequency Scaling. x , 5

EDA	Electronic Design Automation. x , 45 , 47
FPGA	Field Programmable Gate Array. x , 7 , 25 , 27 , 40 , 41
FS	Full System. x , 68
FSM	Finite State Machine. x , 46 , 52 , 53
GPU	Graphic Processing Unit. x , 6 , 7 , 24 , 26 , 27 , 33 , 34 , 67 , 80
HBM	High Bandwidth Memory. x , 7 , 11 , 27 , 33 , 111
HDD	Hard Disk Drive. x , 6 , 10 , 12 , 13 , 15 , 17 , 22 , 112
HMC	Hybrid Memory Cube. x , 33 , 111
HP	High Performance. x , 71
HPC	High Performance Computing. x , 8 , 10 , 41 , 42 , 86 , 89 , 116
HRS	High Resistive State. x , 16 , 18–20
IMC	In-Memory Computing. x , 1 , 17 , 22 , 25 , 26 , 28–30 , 32–36 , 38–45 , 47 , 49 , 56 , 61 , 69 , 112 , 113 , 117
IoT	Internet of Things. x , 2 , 23 , 37 , 42 , 49
IP	Intellectual Property. x
IPC	Instructions Per Cycle. x , 7
IR	Intermediate Representation. x , 69
ISA	Instruction Set Architecture. x , 26 , 30 , 39 , 40 , 42 , 50 , 51 , 53 , 54 , 77 , 113 , 115 , 117
LFB	Line Fill Buffer. x , 66
LIM	Logic In Memory. x , 29 , 37
LLC	Last Level Cache. x , 66
LOP	Low Operating Power. x , 72
LRS	Low Resistive State. x , 16 , 18–20
LRU	Least Recently Used. x , 79
LSQ	Load Store Queue. x , 66
LSTP	Low Standby Power. x , 71
MLC	Multi Level Cell. x , 73

MMU	Memory Management Unit. x, 76
MRAM	Magnetic Random Access Memory. x, 18, 20, 21, 38, 70, 112
MTJ	Magnetic Tunnel Junction. x, 20
MVM	Matrix Vector Multiplication. x, 41, 61, 62, 112, 114
NMC	Near-Memory Computing. x, 26, 29, 30, 32, 35, 39, 40, 47, 69, 112
NMP	Near-Memory Processing. x, 29, 39, 40
NVM	Non Volatile Memory. x, 1, 14, 15, 20, 21, 27, 39–43, 47, 48, 59, 68, 70, 71, 73, 75, 87–91, 95–100, 105, 106, 108, 111–114, 116
OoO	Out of Order. x, 5, 69
OS	Operating System. x, 17, 67–69, 77
OTS	Ovonic Threshold Switch. x, 18
OxRAM	Oxide Random Access Memory. x, 18
PCM	Phase Change Memory. x, 9, 18–21, 38, 47, 70–72, 112, 114, 116
PCRAM	Phase Change Random Access Memory. x, 72
PIM	Processing In Memory. x, 29, 33, 34, 36, 39, 40, 112
PMU	Performance Monitoring Unit. x, 65, 66
RAPL	Running Average Power Limit. x, 67
RRAM	Resistive Random Access Memory. x, 9, 17–21, 36–38, 40, 44, 47, 70, 112
RTL	Register Transfer Level. x
SA	Sense Amplifier. x, 9, 11, 16, 25, 31, 35, 37, 41, 112
SCM	Storage Class Memory. x, 1, 15, 20, 22, 28, 39, 62, 64, 79–82, 88–90, 98, 100, 108, 114–117
SE	System Emulation. x, 68
SIMD	Single Instruction Multiple Data. x, 3, 5, 6, 26, 27, 34, 69, 77, 88, 92, 106, 111, 116
SIMT	Single Instruction Multiple Threads. x, 6
SLC	Single Level Cell. x, 35
SRAM	Static Random Access Memory. x, 1, 9–16, 18, 20–22, 25, 28, 30, 32–34, 37, 38, 40–48, 52, 53, 56, 59, 71, 72, 112, 113
SSD	Solid State Drive. x, 12, 13, 15, 17, 20, 35, 39
STT-MRAM	Spin Transfer Torque MRAM. x, 20, 38, 39

TCAM	Ternary Content Addressable Memory. x , 31 , 38
TLP	Thread Level Parallelism. x
TTM	Time To Market. x , 45 , 47

Glossary

API	An Application Programming Interface (API) is a particular set of rules and specifications that a software program can follow to access and make use of the services and resources provided by another particular software program that implements that API. x, 26
NVM	A Non Volatile Memory is a memory. x
Thrashing	Thrashing is a phenomenon where pages are constantly swapped from dram to disk such that the operating system becomes unresponsive. In this thesis, thrashing can also denote the same effect but between Dynamic Random Access Memory (DRAM) and csram.. x, 94, 104, 105
VCD	Value Change Display. A file that contains all the value changes of all the wires in a circuit.. x, 55

Preamble

This thesis is divided in five chapters that will introduce you to the *why* am i doing this thesis: the global context raises issues about computing performances and efficiency that requires an innovating solution (Chapter 1). In-Memory Computing (IMC) is a promising solution compatible with new emerging Non Volatile Memories (NVMs) that also brings new technological improvements in the computer world. We study state-of-the-art in Chapter 2 and show how it misses two key points about NVMs endurance and where to compute in the memory hierarchy. We propose our solution, a digital wrapper around a Static Random Access Memory (SRAM) that we call C-SRAM (Chapter 3). Our C-SRAM can then be tightly coupled to others NVMs or Storage Class Memories (SCMs). To perform an architectural evaluation, we develop a simulation platform fed with technological parameters from state-of-the-art and our own works (Chapter 4). Putting it all together, we show that computing at the top of the memory hierarchy, i.e. close to mass and permanent storage, yields the most gains for both execution time and energy reduction (Chapter 5).

I hope you will enjoy reading this thesis as much as I enjoyed writing this final sentence.

1 Introduction & Contextualisation

Hardware design comes to the end of its golden era where a simple wait of a few months could yield huge improvements for both performance and energy consumption. This was mainly driven by technology scaling and moving to smaller and more advanced nodes. However, as industry reaches the smallest possible node (3 nm), progress can no longer come from technology itself but must come from finer architecture and software designs to better utilize hardware. Famous von Neumann architecture where memories and computing are physically separate and logically distinct units must evolve to face new computing requirements posed by recent rise of big data applications and artificial intelligence. [Internet of Things \(IoT\)](#) devices are also presenting a challenge for energy efficient designs in the wake of societal changes in regard to global warming and energy sobriety.

Contents

1.1	The end of technology advancement	3
1.1.1	Physical limits	3
1.1.2	Architecture improvements	5
1.1.3	Socioeconomic impacts	8
1.2	Memory technologies	8
1.2.1	Main memory technologies	9
1.2.1.1	SRAM	9
1.2.1.2	DRAM	10
1.2.1.3	Hard disk and tapes	12
1.2.1.4	NAND Flash	12
1.2.1.5	Current memory hierarchy	13
1.2.2	Emerging non volatile memories	14
1.2.2.1	RRAM	17
1.2.2.2	PCM	19
1.2.2.3	MRAM	20
1.3	A new computing paradigm	22
1.3.1	Big Data	22
1.3.2	Proposed solution : memory computing	23
1.4	Conclusion	26

This first chapter will give the reader a very wide introduction and contextualisation on semiconductor technology facing the end of a cycle with a halt to miniaturisation and other well known obstacles to densification. The key to improve performance is to add more transistors into circuits. However this is defined by physical limits that have been or are being reached nowadays, including energy and memory wall problems (section 1.1). New technologies that may resolve partially these problems are being introduced, especially for emerging memories (section 1.2). These new memories enable a new computing paradigm to solve the admitted von Neumann bottleneck that is exacerbated by big data applications and the rise of artificial intelligence (section 1.3).

1.1 The end of technology advancement

1.1.1 Physical limits

For years, what has driven the semiconductor industry progress is the technology scaling, i.e. the miniaturisation of the transistor. Reducing the transistor, base unit of all the digital world, by a factor of $\sqrt{2}$ leads to a doubling in the total number of transistors in the same area. Gordon Moore predicted that this doubling would occur every 24 months, later revised down to 18 months. This is known as the Moore's law [1] which held true for almost 50 years (Figure 1.1). More transistors equals more functionalities or more complex ones; as such, we have seen parallel computing emerged during the 2000s with [Single Instruction Multiple Data \(SIMD\)](#) and multicore [Central Processing Units \(CPUs\)](#). However, miniaturisation has limits that cannot be exceeded. It is physically impossible to make a transistor that is smaller than a few atoms and we are already hitting this limit with 3 nm. This means that to answer the growing need for more computing power, semiconductor industry will have to rely on better architectural designs and smarter software models.

Moving forward to more advanced nodes has also some technical limits that can be seen as side effects for laypeople readers. Smaller transistors are more leaky as the space between different voltage domains is also reduced. On the other hand, it allows to reduce voltage because the threshold voltage is lowered as well. All in one,

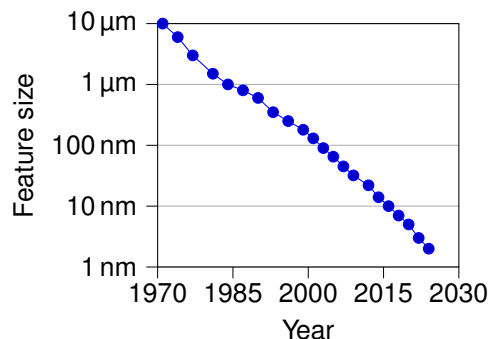


Figure 1.1: ITRS roadmap as of 2020

1 Introduction & Contextualisation – 1.1 The end of technology advancement

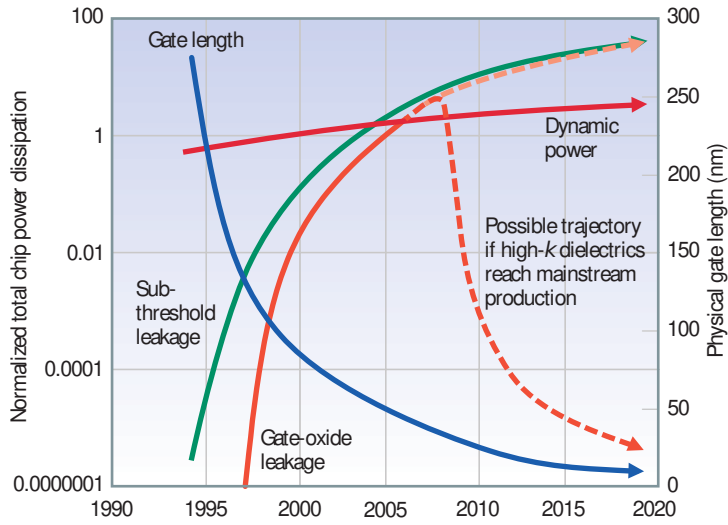


Figure 1.2: Transistor leakage and gate length function of year. From [3]

Process technology	2010	2017	
	40 nm	10 nm, high frequency	10 nm, low voltage
V_{DD} (nominal)	0.9 V	0.75 V	0.65 V
Frequency target	1.6 GHz	2.5 GHz	2 GHz
Double-precision fused-multiply add (DFMA) energy	50 picojoules (pJ)	8.7 pJ	6.5 pJ
64-bit read from an 8-Kbyte static RAM (SRAM)	14 pJ	2.4 pJ	1.8 pJ
Wire energy (per transition)	240 femtojoules (fJ)	150 fJ/bit/mm	115 fJ/bit/mm
Wire energy (256 bits, 10 mm)	310 pJ	200 pJ	150 pJ

Figure 1.3: Predicted scaling cost in 2010 (45 nm) for 2018 (10 nm). From [4]

this leads static power to be dominant in nodes smaller than 90 nm [2] and to increase for each smaller node (Figure 1.2). Moreover, although reducing transistor leads to gain in dynamic power, wire cost is not scaling down with the same tendency. Copper resistivity remains constant and data transit over long wires still stands as the main power sink in every design, especially when memory is on a chip of its own. This is shown in Figure 1.3 dating from 2011 that forecast this difference in power reduction from computing complex operations compared to transmitting that will expand four times between 2010 and 2017.

Another problem linked to miniaturisation is Dennard’s scaling [5]. It states that as transistors shrink, their power density remains constant. This was true from 1974 to approximately 2006. At that point, power density started to increase and it ultimately limited frequency increase. Indeed dynamic power is determined by two main factors which are the voltage and the frequency (Equation 1.1). Voltage is fixed by the technology and cannot go below the threshold voltage plus the line loss. C is the parasitic wire capacity swung at every clock cycle and is also a fixed parameter of the technology. So we can only play on the frequency but as we want the most performance, we tend to push it to the maximum acceptable limits by the design, i.e. the maximum power we can either deliver or dissipate. As power density increased, it soon started to be impossible to rise frequency without damaging the circuit hence a frequency saturation from 2005 as shown in Figure 1.4. These technology problems are physical limits that cannot be broken without a new disrupting technology such as optronic or spintronic that could leverage them. It also led to the famous expression by Herb Sutter: “*The free lunch is over*” [6].

$$P_{\text{dyn}} = CV^2 f \quad (1.1)$$

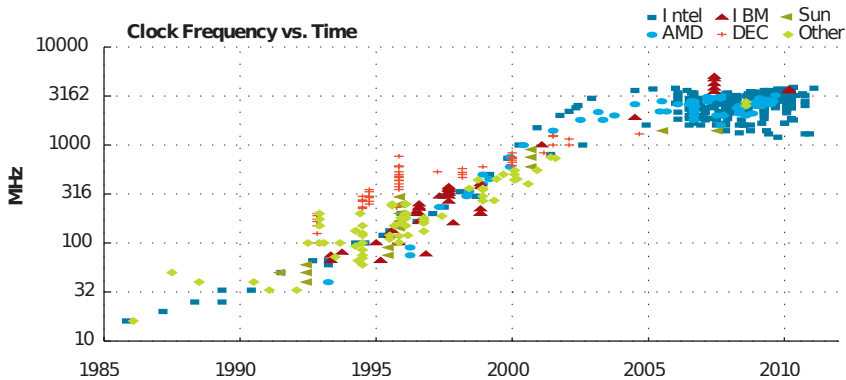


Figure 1.4: Evolution of CPU frequency over years. From [7]

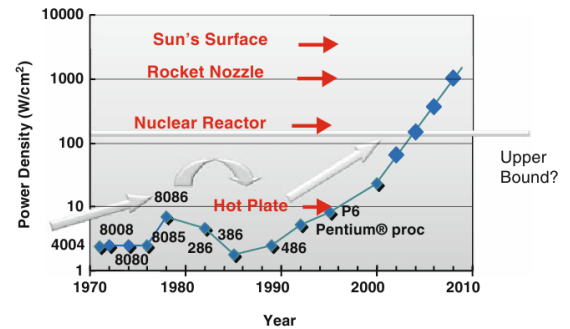


Figure 1.5: Power density evolution. From [2]

1.1.2 Architecture improvements

Industry now faces a double challenge, the impossibility to increase working frequency and the increase in leakage current when moving on to more advanced nodes. To keep performance development in their chips, industries introduced multiple workarounds: **Single Instruction Multiple Data (SIMD)**, multicore and **Out of Order (OoO)**. First, **SIMD CPUs** were developed to treat multiple data in a single instruction using vector larger (128 bits or more) than the base register (32 or 64 bits at that time). **SIMD** exploits intrinsic **Data Level Parallelism (DLP)** in applications. The widest **SIMD** processor supports up to 512 bits vectors. Secondly, multicore designs permit two independent instruction flows to execute concurrently although they share some hardware, especially memories above L2 or L3 and buses. In some recent commercial chips, up to 64 cores can be used in parallel [8]. Third, **OoO CPUs** introduction improved compute unit use and reordering of instructions allowed **CPUs** to mitigate memory timings on independent data paths. With multiple compute units available, processors are said to be superscalar, i.e. capable of executing multiple instructions simultaneously. These three improvements however reached the limit to their computing performances due to power constraint, and insidiously led to the apparition of dark silicon [9]. This happens when complex circuits cannot be fully powered permanently or simply overheat and need to dynamically choose which part to power or to adjust either voltage or frequency using Dynamic Voltage and Frequency Scaling. The latter was adopted by the industry because it allowed more flexibility and less *stuttering* in data streams. An example is given for Intel multicore chips and the use of **SIMD** extensions in Figure 1.6. Dark silicon reveals the low energy efficiency of these designs.

Not only those improvements are not sustainable in the long term, they also put pressure on other system components, typically on the memory system (Figure 1.13). For **SIMD**, memory now has to serve request up to 512 bits instead of scalar data of 32 or 64 bits. Caches are designed to respond swiftly to these requests but when they

1 Introduction & Contextualisation – 1.1 The end of technology advancement

Mode	Base	Turbo Frequency/Active Cores											
		1	2	3	4	5	6	7	8	9	10	11	12
Normal	2,100 MHz	3,000 MHz	3,000 MHz	2,800 MHz	2,800 MHz	2,700 MHz	2,700 MHz	2,700 MHz	2,700 MHz	2,400 MHz	2,400 MHz	2,400 MHz	2,400 MHz
AVX2	1,700 MHz	2,900 MHz	2,900 MHz	2,700 MHz	2,700 MHz	2,400 MHz	2,400 MHz	2,400 MHz	2,400 MHz	2,100 MHz	2,100 MHz	2,100 MHz	2,100 MHz
AVX512	1,100 MHz	1,800 MHz	1,800 MHz	1,600 MHz	1,600 MHz	1,500 MHz	1,500 MHz	1,500 MHz	1,500 MHz	1,400 MHz	1,400 MHz	1,400 MHz	1,400 MHz

Figure 1.6: Frequency scaling of an Intel Xeon Silver 4116, a 12 cores chip, function of active cores and active SIMD extension. From [10]

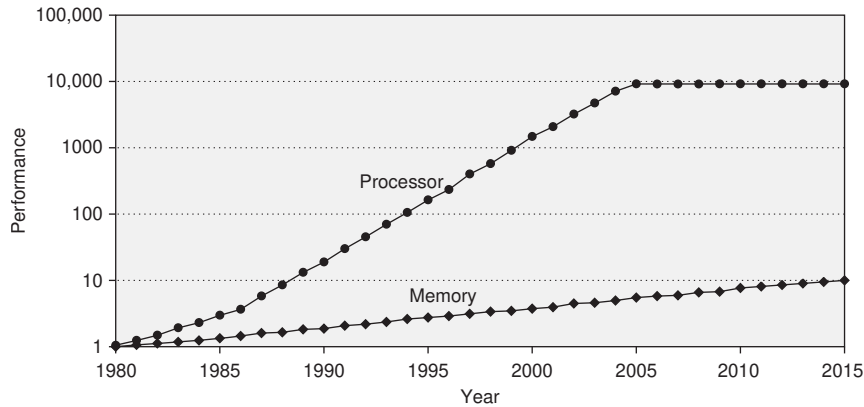


Figure 1.7: CPU and memory performance trends (from [11])

would have to serve only a single data, they now have to load large batch of data which increases their power consumption. As vector CPUs treat batch of data, which is now the size of a cache line, caches experience a high miss rate putting more pressure on the slower DRAM which becomes the von Neumann bottleneck. This is worsened by multicores because each core will request data to DRAM that the L3 cache cannot store due to its limited capacity. So now, DRAM has to deliver data to several cores simultaneously instead of just one. Each core having its own data set, data locality is reduced which also impacts caches and DRAM performance. This is illustrated in Figure 1.7 where performance of CPUs increase faster than which of memories leading to a performance gap between the computing and the memory systems. This is what is called the memory wall, because the memory cannot deliver data fast enough and the CPU just waits doing nothing. Note that above DRAM, Hard Disk Drive (HDD) and Flash disks have long been surpassed and cannot compete in terms of bandwidth with the need of modern CPUs nor of DRAMs.

To keep increasing throughput and energy efficiency, Graphic Processing Units (GPUs) were pushed in. They use Single Instruction Multiple Threads approach, i.e. different threads all executing same instruction on different data with predicates to allow branches and conditional execution to occur. It heavily simplifies the internal design of the processing elements making them more compact so that thousands can be put on a single chip. This benefits to application with heavy DLP such as filtering an image where the same operation is carried on all pixels with conditional code to handle edge cases. GPUs come with their own main memory, nowadays of type High Bandwidth Memory (HBM) with 256 bits IO and high bandwidth. They also have

their own internal caches with 2 levels of cache. Nonetheless, initial data transfer from system main DRAM memory to GPU's memory must still take place before the algorithm runs and data must be sent back once it is done. This back and forth can end up representing more than 90 % of the total execution time depending mainly on the algorithm complexness and the data set size [12]. Overall, GPUs work pretty well on the same regular access patterns as CPUs. They also provide similar program flow with a wide range of complex instructions. Their massive parallelism is used to build some of the Top500 supercomputers [13, 14].

However, both CPUs and GPUs are very generic and can be considered as swiss knives of computing. They do the job but not in a very efficient way except for regular linear access patterns. To improve energy efficiency and throughput, co-processors dedicated to specific tasks were designed, most common one being the Digital Signal Processor for embedded systems with real time constraints. Unfortunately, the need for more, better and greener computing requires flexibility that these extra co-processors do not offer. Field Programmable Gate Arrays (FPGAs) are yet another possible mean to gain extra performance by allowing CPU to turn part of itself into a highly energy efficient application specific accelerator and bridge the gap between flexibility of use and efficient designs. Their programmability combined with their natural energy efficiency makes them suitable candidates for use as co-processors. They come with their own memory in the form of Block Random Access Memory (BRAM) with wide IO to feed their natural data level parallelism. Unfortunately, these BRAMs still need to be filled from another external memory which is often DRAM, but FPGAs do improve energy efficiency. So the main problem of memory wall is still there for initial and final data transfer, just like for GPUs. Another step further is using Application Specific Integrated Circuit, which are fixed designs but with even better energy efficiency and throughput than FPGAs, but once again, the memory wall remains.

All these hardware solutions are to boost classic algorithms performance but there was also the breakthrough of new algorithms in the last decade, mainly Artificial Intelligence (AI) with neural networks. AI is a solution to treat massive amount of data and extract meaningful tendencies but it comes with its own data that are the neurons parameters which can also be counted in billions for some networks. Aforementioned hardware solutions can all improve neural networks performances but all end up hitting the memory wall.

The race for best performances, although a great source of hardware improvements such as branch predictors, prefetchers and so on, induced a rising complexity of CPUs that led to some security flaws [15]. But it also drives for more power and ironically reduces energy efficiency [16]. The industry focused on instruction centric paradigm where everything was done to increase throughput of instructions, measured in Instructions Per Cycle. But when looking at the energy bill of simple instructions (Figure 1.8), we see that this is not very efficient as most of the energy comes from moving the data around. With the introduction of big data and artificial intelligence applications that uses huge batches of data, this calls for a shift to data centric architectures to solve all the two major challenges : the von Neumann

Integer		FP		Memory	
Add		FAdd		Cache (64bit)	
8 bit	0.03pJ	16 bit	0.4pJ	8KB	10pJ
32 bit	0.1pJ	32 bit	0.9pJ	32KB	20pJ
Mult		FMult		1MB	100pJ
8 bit	0.2pJ	16 bit	1.1pJ	DRAM	1.3-2.6nJ
32 bit	3.1pJ	32 bit	3.7pJ		

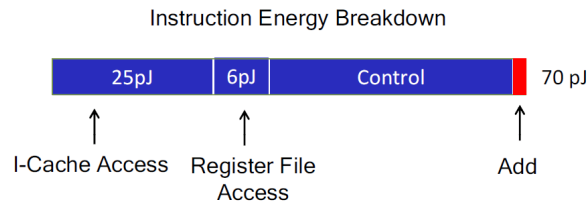


Figure 1.8: Instruction energy breakthrough along with some energy consumption of common instructions and memory accesses. From [17]

bottleneck aka memory wall and the energy wall or dark silicon.

1.1.3 Socioeconomic impacts

This changing paradigm is in accordance with the evergrowing need for greener computing and better energy efficiency in data centres. [High Performance Computing \(HPC\)](#) centres are reaching tens of megawatt of power consumption which is the equivalent of a 20 000 inhabitants city [13]. Another important point is the economic cost of moving to more advanced nodes which grants no more benefits due to the rising cost of state-of-the-art technologies presented in [Figure 1.9](#). As shown in [Figure 1.8](#), the best way to reduce energy consumption is to minimize data movement. This can yield performance improvement per the vector nature of memory computing and by the suppression of costly back and forths data movements. Moreover, it can alleviate the energy wall problem by reducing the performance and energy constraints put on the [CPU](#).

Up to this point, we have presented the global context and the challenges facing the semiconductor industry for the following years: no more possible scaling, no more power and a growing need for more energy efficient computing. These challenges call for either a shift to different technology or to rethink the architecture of systems to better use them. In the following section, we present the standard memory technologies and the emerging memories that appeared in the last decade.

1.2 Memory technologies

Previous section dealt mainly with [CPUs](#) which is the core of computing systems. We have shown that instruction centric architectures faced a soon to come dead end due

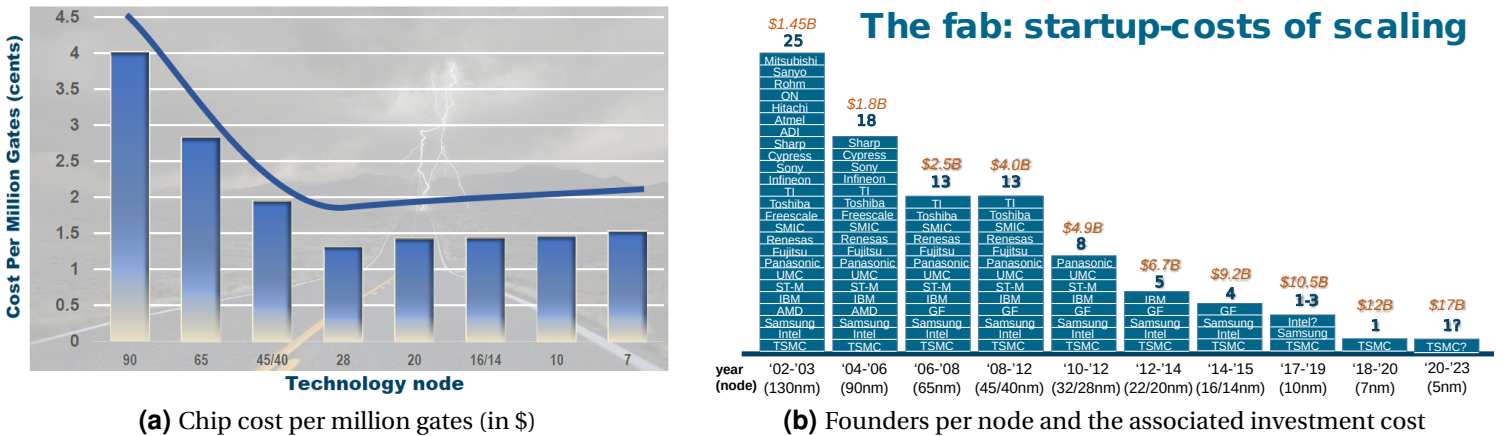


Figure 1.9: Cost of chips and investment needed for the founder. From [18]

to energy and memory walls. This section introduces main memory technologies such as **Static Random Access Memory (SRAM)** and **Dynamic Random Access Memory (DRAM)** but also persistent storage to give the reader a broad range of possibilities and perspectives with their associated limitations which represent a major challenge in the data movement cost. Emerging memory technologies including **Resistive Random Access Memory (RRAM)** or **Phase Change Memory (PCM)** are presented along with their remaining challenges to make them viable economically and offer substantial benefits for system architects over conventional memories.

1.2.1 Main memory technologies

The main memory technologies are the most common ones that can be found in any consumer device. They are the most mature ones and present in the market for decades. However they have some intrinsic design flaws such as high leakage (whether dynamic or static power) or very high latency for non volatiles ones.

1.2.1.1 SRAM

Static Random Access Memory (SRAM) is a fast memory used in almost all existing CPUs dating back to 1964. It provides an extremely fast memory whose working clock frequency is above 1 GHz with virtually infinite endurance. The circuit diagram of a six transistors SRAM bitcell is shown in **Figure 1.10**. It is made up of two head to toe inverters and two access transistors. Read operation is performed by first precharging the bitlines to $\frac{V_{dd}}{2}$, then by activating the two access transistors and using a **Sense Amplifier (SA)** at the bottom of the bitlines to minimize the error margin. Write operation is done similarly by forcing the data on both bitlines which will switch the state of both inverters. However, the inverters are not perfect and leak, so the SRAM bitcell presents a high static power consumption. It is often arranged in large array, up to 8192 wordlines or bitlines which increases the dynamic consumption due to the

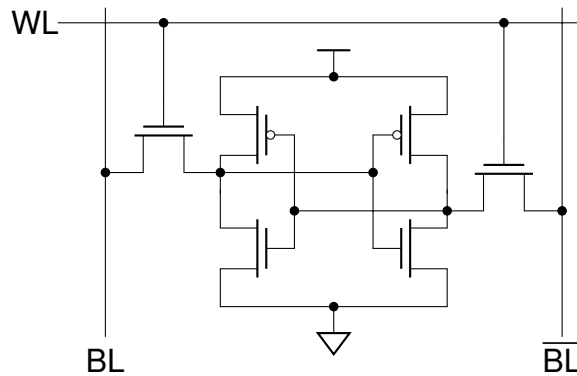


Figure 1.10: SRAM bitcell circuit diagram

large capacitance of the lines. To reduce dynamic switching power, bitlines are often split in local groups with access transistors to commute global bitlines.

Per se, the 6T bitcell is a 1 read-write (1RW) bitcell, which means that it can either be read or written once per cycle. SRAM bitcell have a large diversity as it also exists in 8T up to 16T. These extra transistors allow to add isolation between the bitcell and the bitlines so that read or write to several bitcells on the same bitline can occur concurrently. This is used to add more access port to the memory to make 1R1W, 1R1RW and even 2RW bitcells. Literature also shows that 6R6W bitcell is possible [19]. SRAM bitcell can also be used as Content Addressable Memory memory that is commonly used in routers. Finally, as it is made up of six transistors, it has a very low density that does not allow to have large SRAM memory bigger than a few megabytes. What makes SRAM so interesting is that it is a CMOS circuit that can be incorporated directly in chips design and scales down along with the technology. It is used as cache or scratchpad memory and is often tightly coupled to CPUs as it is the only memory to keep up the pace with high frequency. Other uses include small buffer memory in devices like HDD, Flash drives or anything that needs few amount of memory before transmitting over serial bus or medium that requires serialisation, e.g. radio transmission. Its flexibility allow designers to easily use custom SRAMs with wide IO or even asymmetrical IO (for serialisation for instance) as well as odd row number.

1.2.1.2 DRAM

Dynamic Random Access Memory (DRAM) is the main memory in non embedded systems such as desktops, servers, HPC and even in some embedded systems like autonomous cars. It features an infinite endurance with medium speed (relative to SRAM) while having a very high density. Figure 1.11 shows the circuit diagram of a DRAM bitcell. It is composed of an access transistor and a capacitor to store the data. This capacitor is leaking so it needs to be refreshed periodically, hence the *dynamic* in the name. This leaking along with the refresh operation cause this memory to have a high dynamic power consumption even when the memory is idle. Read is performed by precharging the bitline to $\frac{V_{dd}}{2}$ and then by activating the access transistor. The

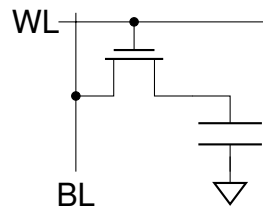


Figure 1.11: DRAM bitcell circuit diagram

capacitor then discharges or charges the bitline and a SA catches the difference. Read is thus destructive as the capacitor shares its charge with the bitline and the original data needs to be restored. Write is simply done by activating the access transistor and pulling the capacitor to the desired voltage (high for 1, low for 0).

To prevent the whole memory from being inaccessible during a refresh, DRAM is organized in ranks subdivided in chips and in banks. Banks are split across several chips for parallelism reason. Each bank is itself partitioned in subarrays which contain the wordlines and bitlines. Wordlines are referred to as logical rows that spans several chips while bitlines are logical columns. Columns are muxed in a similar fashion to SRAM. To read or write, a DRAM row must first be activated, i.e. selected, it is then loaded in the row buffer where read and write take place for faster operation. In particular, burst mode allow several operation to contiguous addresses to happen with a single command and fully benefit from the row buffer. When all operations on the current row are done, either a different row within the same bank can be activated or a row in a different bank is selected. The former bank must first receive a *precharge* command to reset bitlines to $\frac{V_{dd}}{2}$ to minimize leakage before activating a row in another bank. When a row is closed, the row buffer is written back in place to restore data. Refresh affects a whole bank at a time and makes it unavailable until it is finished. The addressing scheme vary from chip to chip but it is mainly column first then bank then row as shown in Figure 1.12. Above ranks are channels which are physical buses and may be shared by several DRAM devices. Addressing schemes can also be interleaved or with some XOR between some bits to increase row-hit rate.

Banks are the physical output and have 8 bits IO. To have a 64 bits IO, 8 banks are disposed in parallel. The complicated rules and state machine to handle DRAM commands and its dynamic nature requires complex designs to ensure correctness. That is why CPUs have a portion of their area reserved for DRAM scheduling (see Figure 1.14b). However, DRAM's high density with its intermediate speed and high bandwidth makes it a suitable choice to fit between long term but slow storage and SRAM's high speed but low capacity. To answer the growing need for bandwidth, manufacturers have developed HBM that uses 3D stacking and have very wide IO (256 bits) compared to DRAM standards. Both are standardised by the JEDEC committee which makes DRAM a somewhat rigid memory format. Due to the complex state machine needed to respect timings and transitions, DRAM has a latency that can greatly vary between 20 ns to more than 400 ns. Finally, DRAM suffer from write disturb. Continuous write to the same row and bitcells by alternating activation, write and precharge leads neigh-

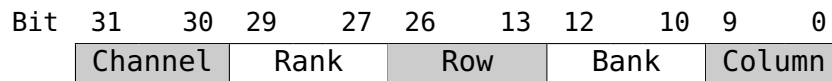


Figure 1.12: Example of a DRAM addressing scheme

bouring cells to be affected and even flipped due to parasitic capacitance between lines. The row hammer attack exploits this vulnerability [20]. Smaller nodes have more parasitic capacitance which augments this risks but also increases the leakage and reduces the stored charge which induces more refresh and more unavailability. As such, DRAM is limited in scaling and faces its own technological challenges.

1.2.1.3 Hard disk and tapes

HDDs and tapes are the most ancient forms of digital storages that are still in use today. They are also the only form of modern storage to use mechanical parts, i.e. an engine, incorporated for HDDs and external for tapes, to spin the disks or roll up and unroll the tape. As such, they require a vibration free environment to be used safely. Shocks may damage data permanently, especially for HDDs and make the device completely unusable. Both HDD and tape have a high data density, not necessarily in surface but more in volume as disks can be easily stacked and tape film is really thin, around 10 µm. Largest commercial HDD is around 15 TB while tape goes up to almost 500 TB. The main cons of these technologies is obviously their very high latency around 10 ms for HDD while tape can go anywhere between one second to more than a minute depending on how far on the tape the data is. Main use of tape includes long term storage such as archiving or data back up for companies. One of unthought advantage of tape is being offline storage which protects data from online attacks. Endurance of both storages is not really a concern as mechanical parts wear out before it is reached.

1.2.1.4 NAND Flash

NAND Flash memory is the most common type of non volatile memory. It is more recent than SRAM and DRAM but the absence of mechanical parts allowed it to be used in numerous devices thanks to its non volatility. It is used in SD memory cards, USB sticks, smartphones and Solid State Drives (SSDs) for the most common devices. NAND Flash is made up of a single transistor with a floating gate which stores the information by retaining the charge after power down. Read is simply done by sensing the current flowing in the channel whereas write is more complex and requires several steps. First, due to how NAND Flash is built to maximize density, it is organised in blocks that cannot be written word per word but only as a whole. This means that even for changing a single bit, a full block must be written. Moreover, write operation requires the block to be erased before so data must first be read to keep non modified data intact. A block is typically around 512–4 kB.

The advantages of NAND Flash are its non volatility with high shock resistance thanks to no mechanical parts compared to HDDs or tapes. Besides, it has a very

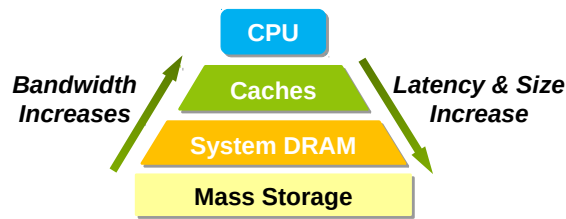


Figure 1.13: Memory hierarchy in a conventional system. In server or cluster, DRAM and mass storage may be distributed or remote.

high density in comparison to [SRAM](#) and [DRAM](#). NAND Flash indeed supports 3D stacking and most recent chips have up to 176 bitcells stacked [21]. This allows to have a virtual footprint of less than the theoretical minimum of $4F^2$. Moreover, each die is also vertically stacked with up to 16 other dies in a standard commercial SSD. This sums up to density superior to 100 Gbit/cm^2 . On the other hand, NAND Flash is quite slow in regard to previous volatile memories. Its read speed is around 1 GB/s but its write speed is 5 times slower around 200 MB/s due to the erase operation. The main disadvantage of NAND Flash is its latency around $10 \mu\text{s}$ for reading which makes it around 100 times slower than [DRAM](#). For writing, latency around $100 \mu\text{s}$ can be expected. These high latencies are due to the high voltage required to operate on the memory array, up to 15 V which takes some time to reach.

When people talk about memory, they often mention capacity, density and bandwidth but they rarely talk about endurance and persistence. Writing in NAND Flash requires high voltage which ends up damaging the cell after many programming cycles. This means that a NAND Flash has its lifetime determined by the write bandwidth and the capacity. To circumvent this problem, industrials added more memory to devices to be used when a block is failing. Commercial devices may have up to 20 % of extra memory. Another technique used is wear levelling. This allows to dynamically remap some blocks onto others to even the number of writes across the device. It also protects against write attacks aiming to destroy data by wearing out [SSD](#) prematurely. To manage wear levelling, NAND Flash devices embed a controller with their own [SRAM](#) memory that also allows to perform some operations on data. Finally, to speedup writes, [SSDs](#) may embed some [DRAM](#) to act as a write buffer but this is only effective if the amount of data is lower than the buffer size. Scalability is also a concern for NAND Flash as the high voltage needed to write the cell constrains the transistor size and limits the downscaling.

1.2.1.5 Current memory hierarchy

On one hand, [CPUs](#) need a working memory to store their temporary data. This memory can be a [SRAM](#) for small microcontrollers or [DRAM](#) for larger processors. On the other hand, a permanent storage is required to store programs and associated data. It is often made with NAND Flash or [HDD](#). As explained in [Section 1.1.2](#), processors have seen numerous architectural improvements to boost their performances. However,

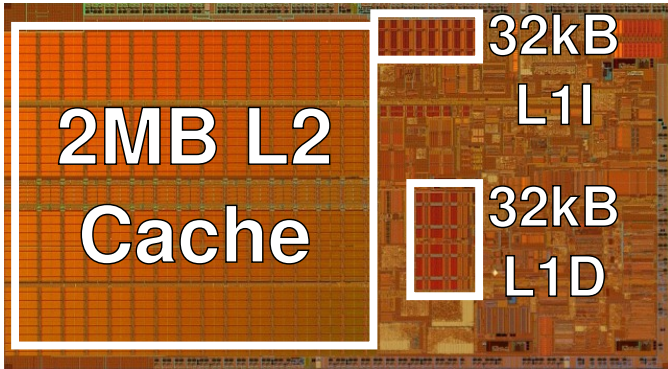
Table 1.1: Main memories key parameters. Data is from [22, 23].

	Price (\$/GiB)	Density	Latency	Bandwidth	Persistence	Largest size
SRAM	5000	120 F ² or 2 Gbit/cm ²	1–5 ns	1 TiB/s	10 μs (Power off)	10–100 MiB
DRAM	20	8 F ² or 25 Gbit/cm ²	20–400 ns	10–100 GiB/s	64 ms	100–1000 GiB
Flash	4	<1 F ² or >100 Gbit/cm ²	1–10 μs	1 GiB/s	10–20 yr	10–100 TiB
HDD	0.1	100 Gbit/cm ²	5–20 ms	100 MiB/s	10–100 yr	10–100 TiB
Tape	0.01	49 Gbit/cm ²	1–100 s	300 MiB/s	100+ yr	100–1000 TiB

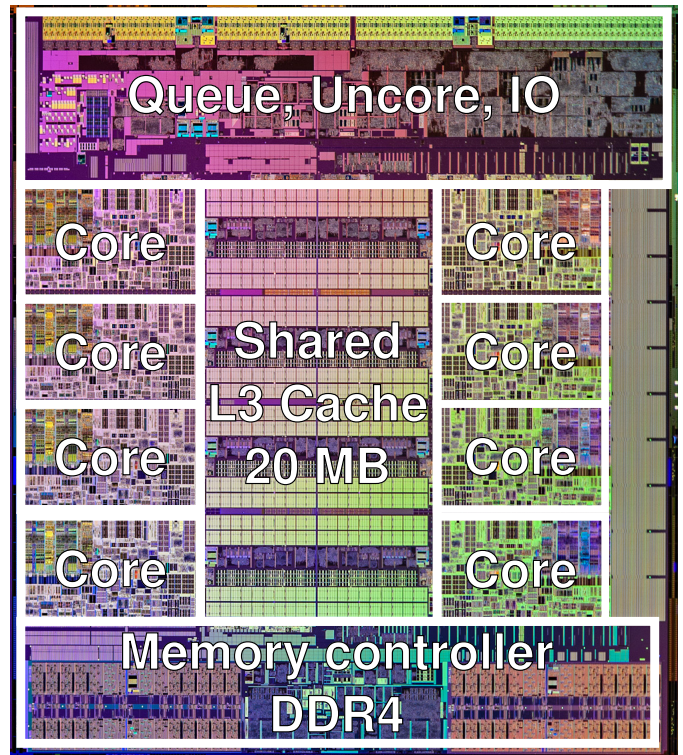
memories did not keep up the pace and, as a result, intermediate memories known as caches were introduced to mitigate timings. If we take a desktop or server CPU, its working frequency is around 3 GHz so it needs a memory to be the fastest possible to not waste clock cycles waiting for data. This is the goal of the L1 cache made in SRAM which is around 16–128 kB and usually have a latency of around 1–3 ns. To be able to serve an instruction and a data at once, there are often two L1 cache, one for instructions and one for data. To bridge the latency and capacity gap with the main DRAM memory, a L2 and a L3 caches were introduced. L2 has a capacity of 128–1024 kB and a latency between 5–10 ns while L3 can be up to more than 50 MB but with a higher latency around 20–50 ns. In rare cases, a L4 made from embedded DRAM can be present. We now have the complete modern memory hierarchy as shown in Figure 1.13. Three caches, one or more external DRAM chips and a, possibly remote, permanent mass storage. All these memories end up eating most of the available area. Figure 1.14a shows a 130 nm Intel Pentium M from 2005 where memory represents more than 60 % of the chip area, so this tendency is already decades old. A more recent processor (Figure 1.14b), a 2015 22 nm Intel Haswell shows a similar area distribution including complex DRAM controller taking the same surface as 2 or 3 cores. That is why new emerging non volatile memories with much better integration and higher density can be of great help. A summary of main memories parameters is presented in Table 1.1.

1.2.2 Emerging non volatile memories

Emerging NVMs are a group of recent (namely 2010 and later) memory technologies that offer promising performances, density and scalability. From an electrical point of view, they all share the same characteristics. In previous memories, such as SRAM, DRAM or NAND Flash, the physical property used to store data is the charge of the bitcell. These charges are maintained through power supply and are gone when the power is shut down (except for NAND Flash). In the case of resistive memories, the physical property used to retain data is the resistive state of the bitcell. This resistance changes depending on the current that flows through the bitcell, but the underlying phenomenon depends on the technology. These emerging NVMs provide a huge benefit compared to SRAM and DRAM especially, because it eliminates the need to have several of current levels of memory in the hierarchy. As such, it would make



(a) A 130 nm Intel Pentium M die



(b) A 22 nm Intel Haswell die. Each core has a 256 kB L2 cache and 2×32 kB L1 caches

Figure 1.14: Die photographs

a big leap forward if it would allow to suppress the L3 cache, the DRAM and also the main storage (either spinning HDD or SSD). The introduction of NVMs would thus potentially replace 3 levels of the memory hierarchy into only one, leveraging huge gains in power consumption, timing (latency and bandwidth), density and silicon area. Another possible use is as Storage Class Memory (SCM), which is a class of intermediate memory between DRAM and NAND Flash, in terms of latency, bandwidth and energy.

The gains in power consumption must however be tempered. As of today, reading these NVMs may be cheaper than reading DRAM, but the write operation can be extremely costly depending on the considered technology. There is no static power compared to SRAM nor dynamic idle power compared to DRAM which make these memories more energy efficient. But if they are to replace SRAM, as of today, it would increase power consumption for this specific use with high bandwidth requirement. The gains in latency are also to be nuanced due to the write asymmetry where the write operation can take up to 10 times longer than the read operation which is problematic in a system point of view. To ensure system responsiveness and guarantee performances in all use cases, write asymmetry still needs to find workarounds. However, as these are non volatiles, it suppresses the refresh operation that can hinder the access

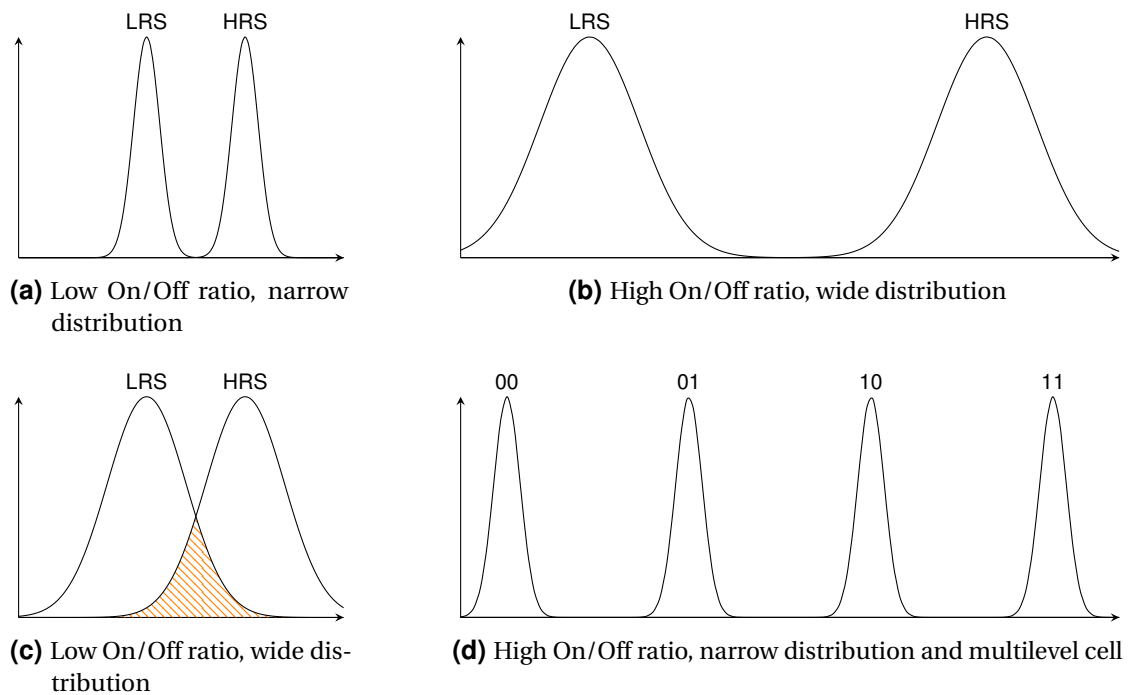



Figure 1.15: Different RRAM resistance probability distribution. Orange hatched  denotes state intersection and should be avoided at all cost.

timing on [DRAM](#). Another issue is that timing operations are usually better than at least [DRAM](#), but not of [SRAM](#).

As said earlier, the resistance is the physical property used to store data. We call [Low Resistive State \(LRS\)](#) the logical 0 and [High Resistive State \(HRS\)](#) the logical 1. The ratio between [HRS](#) and [LRS](#) is called the On/Off ratio and determines the precision of the [SA](#), the working frequency and if multilevel cells can be used. Unfortunately, contrary to electrical charge, resistance cannot be controlled accurately and follows a normal or log-normal distribution as shown in [Figure 1.15](#). A narrow distribution with a high On/Off ratio is the best case as both state can clearly and easily be distinguished and may even allow multilevel cell ([Figure 1.15d](#)). The worst case is a low ratio with a wide distribution where some [LRS](#) cells might have a higher resistance than some [HRS](#) cells ([Figure 1.15c](#)). In this case, either error correcting code can be used but this requires more space and may fail if the distributions are really bad, or write verify loop to make sure the cells end up in a distinguishable state but this is non deterministic and write may take a long time. Low ratio with narrow distribution ([Figure 1.15a](#)) and high ratio with wide distribution ([Figure 1.15b](#)) are acceptable cases if the distributions do not overlap.

In terms of density, these can reach the theoretical maximum of $4 F^2$, but it depends on the array structure and the access device to the bitcell: none (crossbar structure), transistor (1T1R bitcell) or back-end of line selector (1S1R) as shown in [Figure 1.16](#). 3D technologies can enable even higher density like Flash already offers. As technology

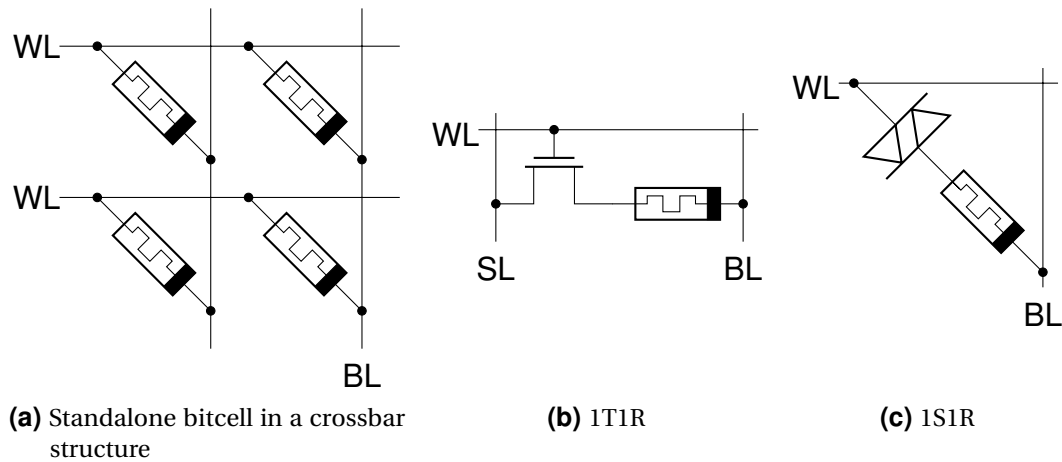


Figure 1.16: Circuit diagrams of 3 different bitcell types

will mature, denser designs will ensue. For silicon area, as we can *theoretically* remove the [DRAM](#) and the mass storage (whether [HDD](#) or [SSD](#)), this removes 2 external chips from the system allowing more compact and efficient systems to be produced. With the advance of [IMC](#) and 3D stacking, we can even dream of a all in one chip where memory and [CPU](#) are tightly coupled [24].

There are still some work to be carried at hardware level including technology and architecture, but on software side as well. New data structures can benefit from the non volatility and [Operating System \(OS\)](#) needs to take it into account. Indeed, non volatility ensures that data remains even after power off, nonetheless this also cause some security threats as data will remain permanently which can include sensitive data such as passwords. [OS](#) must take care of erasing data after deallocation which was easier with [DRAM](#). On the technology side, endurance for all these emerging memory technologies remains a serious concern that prevent any to be used for their purposed introduction. On the other hand, their integration and compatibility with the fabrication process, depending on the material used for some memories, greatly ease their adoption by industry and reduce the need for investment in new fabrication lines.

1.2.2.1 RRAM

[Resistive Random Access Memory \(RRAM\)](#)¹ is the first discovered and manufactured type of emerging non volatile memory dating back to the 1960s but it only attracted attention in the 2000s when it was made with back-end of line compatible materials. Although in its general form [RRAM](#) embraces all resistive memories including [PCM](#) and [MRAM](#), we discuss in this section only about [Oxide Random Access Memory \(OxRAM\)](#) and [Conductive Bridge Random Access Memory \(CBRAM\)](#). In the literature,

¹ ↑RRAM® is a registered trademark in Japan and EU until 20/02/2023 [26]. ReRAM is also encountered in the literature.

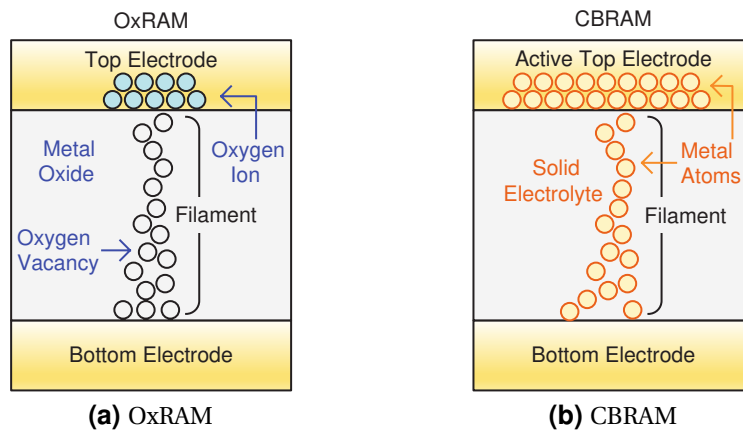


Figure 1.17: Different types of RRAM bitcell. Taken from [25]

RRAM are sometimes referred as filamentous RRAM. Indeed, these technologies rely on a **Conductive Filament (CF)** inside an insulating material. **OxRAM** depends on oxygen vacancies as filament while **CBRAM** uses metal ions (Figure 1.17). *Set* operation is performed by applying a positive voltage across the device and *reset* requires a negative voltage. This means that the selector cannot be a one way device such as a diode and also slightly complicates write drivers to be reversible. Access device can thus be a single transistor, an Ovonic Threshold Switch or none at all in a crossbar array structure (Figure 1.16a).

It is often made from HfO_2 which is a high- k dielectric (highly insulating) used in transistor to make smaller grids and thus RRAM is easily integrated in current fabrication lines. With a crossbar array structure, it should be the most dense on-chip memory available, excluding 3D stacking technologies. It is aimed to replace potentially SRAM in higher level cache, typically L3 [27] while L2 and L1 are expected to remain with fast SRAM memory. Nonetheless, there are still challenges to reach these goals with serious reserves on endurance and variability within an array.

First of all, RRAM requires higher voltage and current than conventional SRAM to form and reset the CF. It requires bigger transistor to drive enough current (up to $100\ \mu\text{A}$). Higher current also means it is harder to shrink the pitch of the metal between lines due to IR-drop effect. Cycling between forming and resetting the CF ends up damaging the cell with micro cracks or migrating material (oxygen or metal) cemented up to the point where the cell is stuck in either LRS or HRS. Current technologies have an estimated endurance between 10 million to a billion cycles [28, 29] which is way too low for caches memory or even DRAM where the write bandwidth can be over a billion writes per second. Wear levelling techniques must be used to mitigate these bandwidth and equalize the wearing out on all the bitcells which slows down the working frequency of RRAM.

RRAM has a medium On/Off ratio often combined with wide distribution (intermediate between Figure 1.15b and Figure 1.15c) which makes reading slower to ensure the state of the bitcell. Another problem is the drift associated with the repeated

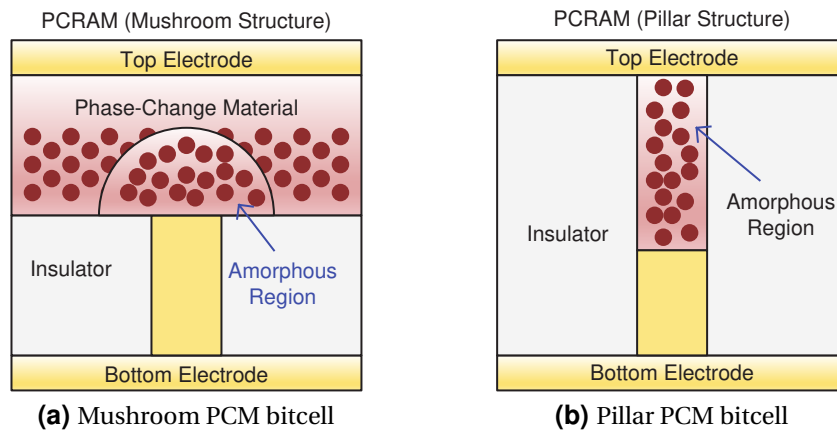


Figure 1.18: Different types of PCM bitcell. Taken from [25]

read/write cycle. Both **LRS** and **HRS** distributions will shift independently for each cell meaning that some cells will have a worsened ratio while others will see it improves across the lifetime of the cell. Worsened ratio may overlap distribution rendering the cell useless which can be alleviated with wear levelling to move data to extra bitcells. Some array structure such as 2T2R [30] can be used to lessen low uniformity issues.

Overall, **RRAM** still has a promising future with write currents going down around $1\ \mu\text{A}$, reading and programming times lower than 10 ns and a retention time of at least 10 years. Endurance above 10^{12} cycles have been reported [31] although it is still a little too low for integrated cache memories. Power density due to higher write currents may also be problematic for some power constrained applications. Highest On/Off ratios are between 100 and 1000 which permits 4 level bitcells (2 bits) [32].

1.2.2.2 PCM

Phase Change Memory (PCM) is another type of resistive memory relying on the transition between amorphous and crystalline phase of a material, usually a chalcogenide. These two phases have greatly different electrical resistance which is used to store data. **HRS** corresponds to the amorphous phase, whereas crystalline is **LRS**. The *reset* operation consists of sending a burst current to melt the material and let it cool down to reach the amorphous phase. *Set* is done by sending a smaller current than *reset* and let the material slowly crystallize. *Set* operation is thus seemingly slower than *reset*. Contrary to **RRAM**, current is one way only as it is only used to heat the material so the selector can now be a diode which is slightly more compact than a transistor. Unfortunately, the high temperature needed to melt the material requires high current for a short amount of time which makes writing an high power operation. Current used to be over 1 mA and has now decreased to $250\ \mu\text{A}$ with voltage around 3 V [33] similar to **RRAM**. Moreover, high temperature limits the density to prevent a write to disturb neighbouring cells. Multiple designs coexist such as mushroom or pillar type as shown in Figure 1.18, depending on the materials used.

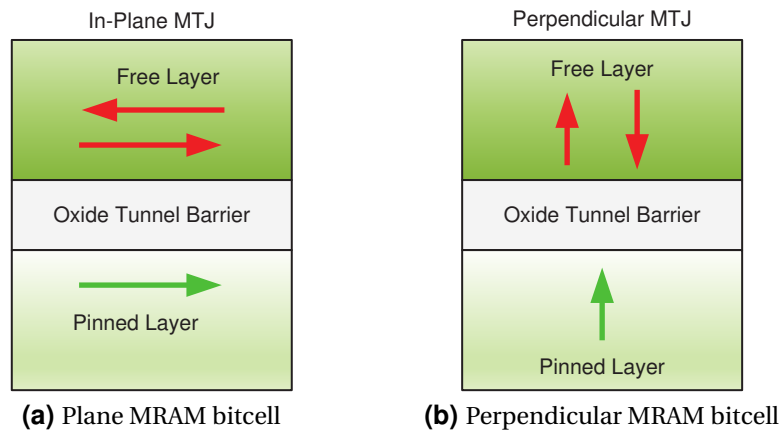


Figure 1.19: Different types of MRAM bitcell. Taken from [25]

Similarly to [RRAM](#), [PCM](#) is subject to endurance issues that are even worse due to thermal expansion. It either creates voids in the cell until it gets stuck open or, due to melting repeatedly, have material migrating and forming a permanent conductive wire. Current technology has endurance between 1 million and a billion cycles [33–35], which is better than most recent SLC NAND Flash. This is enough to replace the former in fast permanent storage as [SSD](#). Although heating and cooling down the material takes time, it is still faster than Flash with write timings of 100 ns and even less reported [33]. Given its better performance compared to Flash, [PCM](#) was the first [NVM](#) sold in consumer electronics by Micron and Intel under the Optane brand name with their 3D XPoint technology. Its On/Off ratio is also way better than [RRAM](#) up to 10^4 allowing multilevel cells to be used with 3 and even 4 bits [36]. Indeed, precise control of current and timing during pulse gives highly repetitive resistance output in contrast to [RRAM](#) which has very wide resistance distribution.

In perspective, [PCM](#) is planned to replace [DRAM](#) [34, 35, 37–39] in computing systems if its endurance is high enough. Otherwise, its use as [SCM](#) has already began with Intel and Micron 3D XPoint technology. Read timings are in the tens of nanosecond and write in the hundreds of nanoseconds. Write power is of concern due to high drive current which makes it the most energy consuming memory to write a bit with 10 pJ/bit whereas [RRAM](#) is around 100 fJ/bit and [SRAM](#) is even lower. Density is limited due to thermal constraint but this is partially circumvented with 3D stacking. Retention time is above 10 years thanks to the material stability in both phases. However, resistance drift due to thermal dilatation and cycling can be a problem in the long term for multilevel cells.

1.2.2.3 MRAM

[Magnetic Random Access Memory \(MRAM\)](#), and more specifically [Spin Transfer Torque MRAM \(STT-MRAM\)](#) is yet another kind of resistive memory using the magnetic orientation of a Magnetic Tunnel Junction to store data. A free layer that can take

Table 1.2: NVMs parameters. Data collected from [25, 28, 33, 43–45]

	Cell Size	Multibit	Read Time	Write Time	Write Energy (/bit)	Endurance
RRAM	4–12 F ²	2	~10 ns	10–50 ns	0.1–10 pJ	10 ⁶ –10 ¹²
PCM	4–30 F ²	4	10–60 ns	20–150 ns	10–500 pJ	10 ⁷ –10 ¹⁰
MRAM	6–50 F ²	2	2–35 ns	3–50 ns	0.01–1 pJ	10 ¹² –10 ¹⁵

two different orientations and a fixed reference layer separated by an oxide barrier make up the bitcell (Figure 1.19). If the layers have the same direction, the cell is in a LRS and if they have opposite direction, then it is a HRS. *Set* is performed by sending a current pulse in the wanted orientation and *reset* is done by reverting this current pulse. Just like RRAM, write drivers must thus be reversible and this constrains the device selector as well. Contrary to PCM, it does not require a lot of power to switch state with current in the range of 100 μ A and write timing inferior to 10 ns [40, 41]. Endurance is the best advantage of MRAMs as it does not suffer from any thermal dilatation or high current going through the cell. Estimated cell endurance are over 10¹² cycles [25]. Voltage to operate the cell is also lower compared to previous memories and within 1.5 V, there also reducing constraints on transistor size.

Unfortunately, magnetic materials required for the fabrication process are not compatible with conventional CMOS technology which is still a problem to be solved. Magnetic nature of the bitcell requires the storage to not be in a magnetic environment that may disturb cells' data which can limit the use for some applications. Heat is also a limitation to the use of this memory in non controlled environment as it largely reduces the data retention time. Finally, MRAM yields a low On/Off ratio with a narrow distribution which makes multilevel cell harder to achieve, but not impossible [42]. 3D stacking is still a work in progress [40] and should help improve the relatively low density compared to RRAM or PCM [43]. Small nodes may also be problematic due to magnetic field interference between cells. As such, MRAM is planned to be a medium density memory compared to RRAM but its high speed and low power makes it a suitable candidate to fully replace SRAM in cache memories thanks to its high endurance [41, 43].

Emerging memory technologies have intermediate energy and timing between either SRAM or DRAM and Flash. Non volatility removes static and dynamic power consumption in the bitcell array which greatly improves energy efficiency. They can replace several memories in the system, mainly DRAM as it is the most power consuming one as well as the higher level cache. Non volatility also allow Flash replacement and better system integration. Another possibility is their integration as SCM in between DRAM and NAND Flash. However, they have limited endurance that is too low to consider a full replacement as of today. Having a permanent storage tightly coupled to CPU will cut down power loss over transmission line instead of having multiple chip connected on a system bus. A summary of their characteristic is given in Table 1.2.

1.3 A new computing paradigm

Now that we have introduced old and emerging memory technologies, we need to explain why we need to revisit the standard architecture model to fit the new needs of the industry. As we have seen, current memory technology are not really scalable with permanent storage being done with spinning HDDs and tapes which both have very high latency, low bandwidth and use mechanical parts that are more prone to failure. Although both have seen tremendous improvement for data density, their high latency and low throughput make them unsuitable for future high demanding uses. We first introduce the rising demand for high throughput data treatment (*big data*) with the use of AI, then we discuss the challenges this trend faces and raises and finally, we introduce a proposed solution that is IMC.

1.3.1 Big Data

Since 1980, data storage has substantially increased and doubles every 40 months [46] which is an exponential growth as shown in Figure 1.20. This trend is still valid as of today but what changed is how this information is treated. What started with high density information, mostly sensor data, is now a sea of low density information which we need to extract the valuable droplets from:

- In finance, data now comes from stock variations but also analysis of political discourses, behavioral analysis, press text analysis to predict the most accurately how the stock will evolve;
- In social networks, text analysis, photo recognition, graph analysis and behavioral analysis all require huge amount of data (and tracking);
- In informatics, database search, insertion and deletion are recurring operation that are lengthy on terabytes dataset;
- In science, it includes: meteorology with the multiplication of sensor data and higher precision with models containing billions of nodes; biology with DNA analysis and pattern matching for protein modifications research; astrophysics where telescopes' data are harvested faster than they are treated leaving huge untreated databases even after telescopes retirement; subatomic physics such as particle accelerator that can generate terabytes of data in a second; medicine that has very wide input dataset to look for correlation between lifestyles and diseases; etc.

All of these are the consequences of the shift from industrial society to information society where data is the new colorless gold. With the IoT, it will be further exacerbated, but thankfully only 2 % of data is stored [47]. These new data as listed previously are of great volume, vary largely in quality and type, are generated quite rapidly and thus demand a fast treatment.

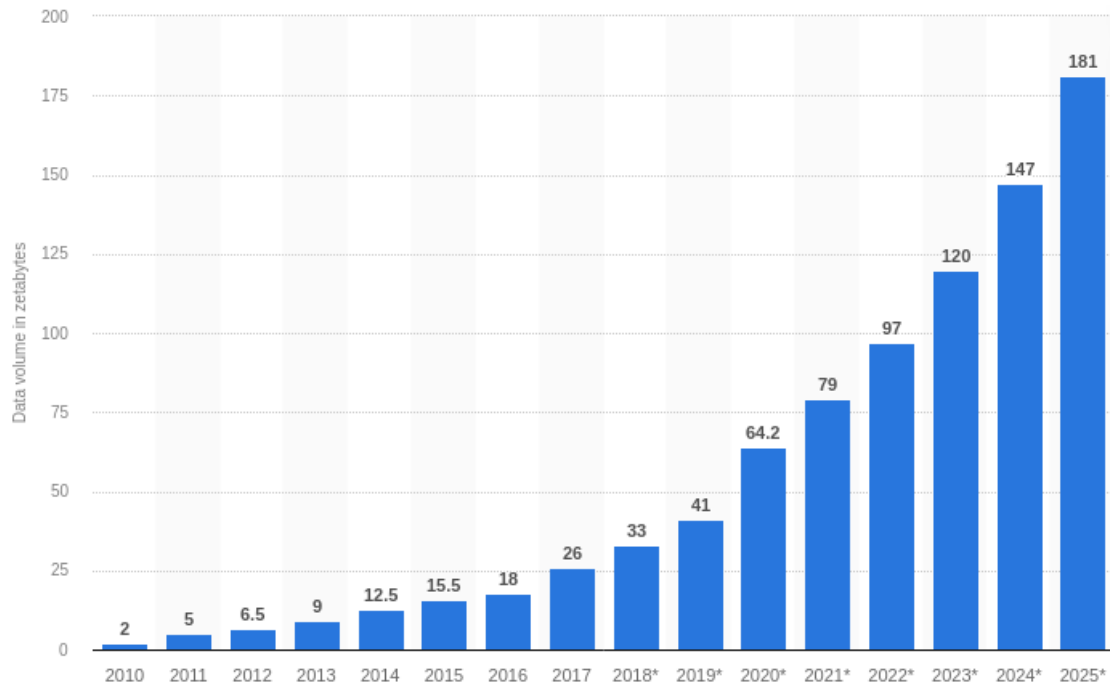


Figure 1.20: Quantity of data created per year. Only 2 % is stored, rest is treated then thrown away. From [47]

Not only data quantity increases as well as its diversity, algorithms also evolved with more complex access patterns. Graph processing with bunny hopping from node to node are not predictable in their patterns and prevent any form of caching relying on spatial locality. These irregular access patterns increase stress on memory systems. Improvements in image processing led to stride access patterns where only part of data is used in a regular way. But memory must still serve the full data to accommodate caches leading to underutilisation of the ideal bandwidth. Combining this with stencils application such as convolution used in filtering where data is accessed in subpart of the total also decreases temporal locality with nowadays high resolution pictures. All in all, we face an absurd amount of data whose algorithms needed to manage them have complex access patterns reducing the effectiveness of cache techniques. This forces data to be read and evicted from caches multiple times increasing the energy and timing cost.

1.3.2 Proposed solution : memory computing

In this data intensive world, we have to treat data in an energy efficient way and with high throughput. Standard computer architecture is compute centric rather than data centric; it is built in order to execute the maximum number of operations in the shortest amount of time but that does not equate to faster data treatment due to memory hierarchy and its intrinsic latencies and limited bandwidth. Data centric architecture is all about treating a massive amount of data in a parallel fashion while

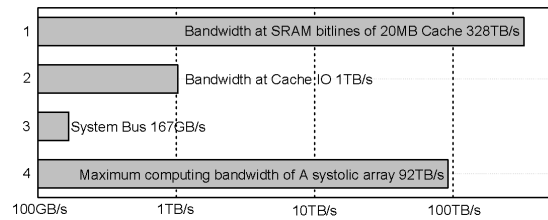


Figure 1.21: Internal versus external memory bandwidth. From [48]

compute centric is about controlling the program path and needs the data to be brought through a complex memory hierarchy designed to hinder the slow memory timings.

The only foreseeable solution is to work at the bottleneck, i.e. the memory. As it is the bottleneck, we cannot treat data faster than its external bandwidth and thus this renders all cache levels and lower memories in the hierarchy obsolete. The data should be handled where it resides, at the topmost level of hierarchy that is the permanent storage or eventually the DRAM. A possible parallel is remote working rather than office work where humans are data to be managed. By doing so, we remove the morning and evening commute representing data transfer with its bottlenecks (roads, railways, etc.), its latencies and its energy cost (electrical, gas, etc.). Another solution is also to compute data where it is produced (local consumption) rather than to send it to an external compute unit, whether it is local to the system or a remote server. But this is not always applicable.

Why memory computing is a viable and promising solution? First, it removes most of data transfers between the memory and the CPU or GPU. This yields timing and power improvements by removing costly intermediate memories such as cache and also slacken power constraint on the CPU (see dark silicon and heating problems in section 1.1). Second, it takes advantage of the much higher internal bandwidth of the memory, sometimes 100× faster than the external one (Figure 1.21). Not only this can incidentally increase throughput but also reduce the average time of algorithms execution. Third, it uses the full internal width of memory lines which is ranging from 100 up to 1000 larger data width, depending on the considered memory technology.

How does memory computing work? Multiples classes of solutions have been proposed in the state-of-the-art and we can split them in roughly two groups: analog computing in the bitcell array or digital computing after the SAs. The first group makes extended use of basic electrical rules to compute logical operation such as NAND, OR or XOR. It rely on the array interconnection between bitlines but is also heavily technology dependent. More complex operations are performed through multiple successive logical operations or with some additions to the periphery circuits. Second group adds digital computing units in the periphery but may also use some analog pre-computing performed in the array. Digital additions allow more complex operations such as arithmetic ones (ADD, MUL, etc.) to be computed in place in single cycle. For a more complete view on IMC techniques, refer to Chapter 2.

Note that memory computing is almost as old as digital computing itself. The

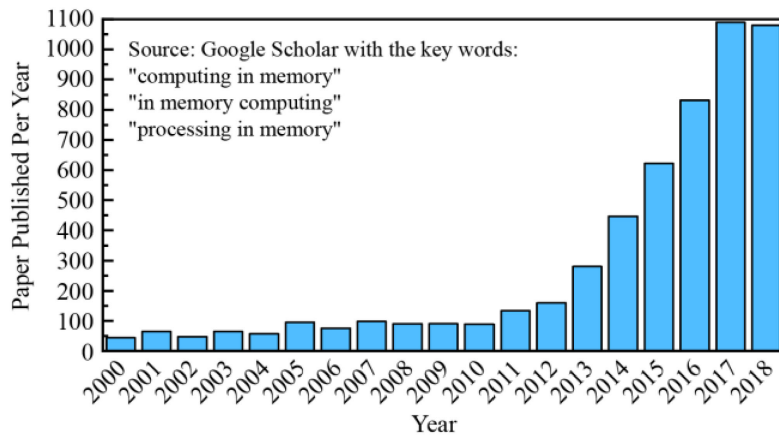


Figure 1.22: Memory computing research interest in Google Scholar (from [55]). Search term are not precise enough as they include some result from neurology (*background noise*).

very first paper that can be connected to **IMC** dates back to 1969 and proposes to interleave memory with logic units to compute basic logic functions [49]. It can be considered as a common ancestor to both **IMC** and **FPGAs**. A similar proposition is found in [50] from 1970 that introduces compute caches with *search*, *add* and *scale* operations. However, the first chip implementation leaps 20 years forward with a 8 kB **SRAM** prototype reaching 1.7 GOPS on a discrete cosine transform application used in video compression [51]. Those previous papers were not introducing solution to the memory wall as they do not mention it, nonetheless the memory wall problem has been known for more than 25 years [52]. Already in 1994, the memory was the limiting factor, the von Neumann bottleneck, and the authors only state that scientists and engineers should think outside the box without providing any hint or direction. A first attempt was designed with the Terasys array [53] with 1-bit Arithmetic & Logical Unit incorporated into **SRAM** memory chip for each column. It supports basic logic operations with distributed computing approach and their own high level language. It reached speedup between $5\times$ to $50\times$ versus single CRAY core. Another nail in the coffin is a paper by D. Patterson et al. in 1997 that reports many programs spend most of their time waiting for the memory [54]. It proposes an intelligent RAM that is vector computing tightly coupled to **DRAM** and foresees up to $4.5\times$ better energy efficiency compared to conventional architecture.

Since 2010, there has been a new surge of papers on **IMC** or Near-Memory Computing indicating a real interest in this solution in both academics and industry ([Figure 1.22](#)). Some questions are still open in the community, concerning mainly more software points than hardware. The programming of these new **IMC** devices and their associated **Instruction Set Architecture (ISA)** is for now not debated with everyone using homemade **ISAs**. RISC-V may be a starting point but it is a scalar **ISA** rather than a vector one (see [Chapter 3](#) for our own). How the instructions are sent to the device or how it is synchronized with the rest of the system is also an unanswered question (see [Chapter 4](#) for our approach). Coherency issues are almost non-existent in the literature although they do exist as in every unified memory system. Overall it is how it should be integrated in a real system with a software Application Programming Interface (API) that also has hardware implications.

1.4 Conclusion

We have shown that the global context in the semiconductor industry is close to reach a dead end. Moore's law is coming to an end in the next few years after more than 50 years of continuous growth ([Figure 1.1](#)). Dennard's law has been broken since 2005 which led to dark silicon, i.e. part of circuit that cannot be powered due to power constraints and heat dissipation issues. Frequency scaling has also reached a plateau around 2005, there also putting a cap on performance for single cores ([Figure 1.4](#)). Architectural improvements were the key to performance increase from 2005 to nowadays with the introduction of multicores **CPU**. Wider **SIMD** extension also provided some boosts to treat vectors of data which is particularly useful for the more demanding applications of today such as neural networks. To speed these applications even more and to improve their energy efficiency, general purpose **GPUs** were introduced with thousands of processing elements executing the same instructions simultaneously. **FPGAs** are now common in commercial datacenter but it is more a marketing strategy than a definitive solution. Application specific accelerators were also designed such as Google's Tensor Processing Unit (TPU). Finally, the rising costs of more advanced nodes make new chips more expensive per million gates. The technological investments required to fabricate the sub 10 nm nodes are astronomical and leave very few foundries choice ([Figure 1.9](#)) which also increases prices. All in all, semiconductor industry is at the end of an era after pushing everything it could to the extreme, both technologically (smaller nodes, better materials) and architecturally (**SIMD**, multicores, etc.).

All of the previous points were only tackling the problem at the compute or logic level but never at the memory level while the memory is the limiting factor ([Figure 1.7](#)) either in bandwidth (memory wall or von Neumann bottleneck) and in efficiency as data transfer is costly. Recent new memory technologies such as **HBM** and **HBM2** partly reduced the gap between **GPU** and memory performance, but this gap still remains. In summary, we do not need to change how we treat the data but *where* we treat it. Thus, we need a shift from compute centric to data centric architectures

where data is the focus point rather than the processing element, especially in the era of big data and AI. This shift should improve energy efficiency in the context of better energy efficient systems to meet the requirements of the Paris Agreements, while still increasing performance with a limited energy budget.

Memory computing is a promising solution to these problems. It satisfies the energy efficiency part by removing useless caches and memories in the system. It offers better performance in vector computing by profiting from the internal memory bandwidth. Emerging NVM technologies offer promising performances such as better read/write energies and faster timings. Their compatibility with CMOS process grants better integration and higher density than DRAM. Non volatility paves the way to a unified circuit containing memory for permanent storage and computing, all in a single chip using 3D integration technologies. This is already envisioned in the literature as the next leap forward [24].

Bibliography

- [1] Gordon E Moore. “Cramming more components onto integrated circuits”. In: 38.8 (1965), p. 4 (cit. on p. 3).
- [2] Jan Rabaey. *Low Power Design Essentials*. Integrated Circuits and Systems. Boston, MA: Springer US, 2009. ISBN: 978-0-387-71712-8. DOI: [10.1007/978-0-387-71713-5](https://doi.org/10.1007/978-0-387-71713-5). URL: <http://link.springer.com/10.1007/978-0-387-71713-5> (visited on 2021-08-30) (cit. on pp. 4, 5).
- [3] Nam Sung Kim et al. “Leakage current: Moore’s law meets static power”. In: *Computer* 36.12 (2003-12), pp. 68–75. ISSN: 0018-9162. DOI: [10.1109/MC.2003.1250885](https://doi.org/10.1109/MC.2003.1250885). URL: <http://ieeexplore.ieee.org/document/1250885/> (visited on 2022-05-31) (cit. on p. 4).
- [4] Stephen W. Keckler et al. “GPUs and the Future of Parallel Computing”. In: *IEEE Micro* 31.5 (2011-09), pp. 7–17. ISSN: 0272-1732. DOI: [10.1109/MM.2011.89](https://doi.org/10.1109/MM.2011.89). URL: <http://ieeexplore.ieee.org/document/6045685/> (visited on 2022-05-25) (cit. on p. 4).
- [5] R.H. Dennard et al. “Design of ion-implanted MOSFET’s with very small physical dimensions”. In: *IEEE Journal of Solid-State Circuits* 9.5 (1974-10), pp. 256–268. ISSN: 1558-173X. DOI: [10.1109/JSSC.1974.1050511](https://doi.org/10.1109/JSSC.1974.1050511) (cit. on p. 4).
- [6] Anja Bog. “The Free Lunch Is Over_ A Fundamental Turn Toward Concurrency in Software”. In: (), p. 8 (cit. on p. 4).
- [7] Andrew Danowitz et al. “CPU DB: recording microprocessor history”. In: *Communications of the ACM* 55.4 (2012-04-01), pp. 55–63. ISSN: 0001-0782. DOI: [10.1145/2133806.2133822](https://doi.org/10.1145/2133806.2133822). URL: <https://doi.org/10.1145/2133806.2133822> (visited on 2022-05-25) (cit. on p. 5).
- [8] *AMD Ryzen™ Threadripper™ 3990X Processor*. URL: <https://www.amd.com/en/products/cpu/amd-ryzen-threadripper-3990x> (visited on 2022-07-13) (cit. on p. 5).
- [9] H. Esmailzadeh et al. “Dark silicon and the end of multicore scaling”. In: *2011 38th Annual International Symposium on Computer Architecture (ISCA)*. 2011-06, pp. 365–376 (cit. on p. 5).
- [10] *Xeon Silver 4116 - Intel - WikiChip*. URL: https://en.wikichip.org/wiki/intel/xeon_silver/4116 (visited on 2022-07-05) (cit. on p. 6).
- [11] John L Hennessy. *Computer Architecture: A Quantitative Approach* (cit. on p. 6).

- [12] Chris Gregg and Kim Hazelwood. “Where is the data? Why you cannot debate CPU vs. GPU performance without the answer”. In: *(IEEE ISPASS) IEEE International Symposium on Performance Analysis of Systems and Software*. 2011-04, pp. 134–144. DOI: [10.1109/ISPASS.2011.5762730](https://doi.org/10.1109/ISPASS.2011.5762730) (cit. on p. 7).
- [13] *June 2022 | TOP500*. URL: <https://www.top500.org/lists/top500/2022/06/> (visited on 2022-07-13) (cit. on pp. 7, 8).
- [14] Bill Dally. “To exascale and Beyond”. In: *Supercomputing*. 2010. URL: https://www.nvidia.com/content/PDF/sc_2010/theater/Dally_SC10.pdf (visited on 2021-11-11) (cit. on p. 7).
- [15] Paul Kocher et al. “Spectre Attacks: Exploiting Speculative Execution”. In: (2018), p. 19 (cit. on p. 7).
- [16] Hadi Esmaeilzadeh et al. “Looking back on the language and hardware revolutions: measured power, performance, and scaling”. In: *Proceedings of the sixteenth international conference on Architectural support for programming languages and operating systems*. ASPLOS XVI. New York, NY, USA: Association for Computing Machinery, 2011-03-05, pp. 319–332. ISBN: 978-1-4503-0266-1. DOI: [10.1145/1950365.1950402](https://doi.org/10.1145/1950365.1950402). URL: <https://doi.org/10.1145/1950365.1950402> (visited on 2022-07-04) (cit. on p. 7).
- [17] M. Horowitz. “1.1 Computing’s energy problem (and what we can do about it)”. In: *2014 IEEE International Solid-State Circuits Conference Digest of Technical Papers (ISSCC)*. 2014-02, pp. 10–14. DOI: [10.1109/ISSCC.2014.6757323](https://doi.org/10.1109/ISSCC.2014.6757323) (cit. on p. 8).
- [18] Valeria Bertacco. “Re-Imagining Scalable System Design”. In: *2018 IFIP/IEEE International Conference on Very Large Scale Integration (VLSI-SoC)*. 2018 IFIP/IEEE International Conference on Very Large Scale Integration (VLSI-SoC). 2018-10, pp. ix–xiii. DOI: [10.1109/VLSI-SoC.2018.8644750](https://doi.org/10.1109/VLSI-SoC.2018.8644750) (cit. on p. 9).
- [19] Hoan Nguyen et al. “A 7NM Double-Pumped 6R6W Register File for Machine Learning Memory”. In: *2018 IEEE Symposium on VLSI Circuits*. 2018 IEEE Symposium on VLSI Circuits. 2018-06, pp. 1–2. DOI: [10.1109/VLSIC.2018.8502393](https://doi.org/10.1109/VLSIC.2018.8502393) (cit. on p. 10).
- [20] Yoongu Kim et al. “Flipping bits in memory without accessing them: An experimental study of DRAM disturbance errors”. In: *2014 ACM/IEEE 41st International Symposium on Computer Architecture (ISCA)*. ISSN: 1063-6897. 2014-06, pp. 361–372. DOI: [10.1109/ISCA.2014.6853210](https://doi.org/10.1109/ISCA.2014.6853210) (cit. on p. 12).
- [21] Micron. *176-Layer NAND*. en. 2022. URL: <https://www.micron.com/products/nand-flash/176-layer-nand> (visited on 2022-07-13) (cit. on p. 13).
- [22] *Lecture 21: Storage*. URL: <https://www.cs.utexas.edu/users/mckinley/352/lectures/21.pdf> (visited on 2022-07-13) (cit. on p. 14).

- [23] *IBM Makes Tape Storage Better Than Ever*. IEEE Spectrum. 2020-12-17. URL: <https://spectrum.ieee.org/tape-is-back-and-better-than-ever> (visited on 2022-07-13) (cit. on p. 14).
- [24] M. M. Sabry Aly et al. “Energy-Efficient Abundant-Data Computing: The N3XT 1,000x”. In: *Computer* 48.12 (2015-12), pp. 24–33. ISSN: 0018-9162. DOI: [10.1109/MC.2015.376](https://doi.org/10.1109/MC.2015.376) (cit. on pp. 17, 27).
- [25] Shimeng Yu and Pai-Yu Chen. “Emerging Memory Technologies: Recent Trends and Prospects”. In: *IEEE Solid-State Circuits Magazine* 8.2 (2016), pp. 43–56. ISSN: 1943-0590. DOI: [10.1109/MSSC.2016.2546199](https://doi.org/10.1109/MSSC.2016.2546199) (cit. on pp. 18–21).
- [26] SHARP KABUSHIKI KAISHA. *EUIPO - eSearch*. 2003. URL: <https://euipo.europa.eu/eSearch/#details/trademarks/003062791> (visited on 2022-09-06) (cit. on p. 17).
- [27] Alexander Hankin et al. “Evaluation of Non-Volatile Memory Based Last Level Cache Given Modern Use Case Behavior”. In: *2019 IEEE International Symposium on Workload Characterization (IISWC)*. 2019-11, pp. 143–154. DOI: [10.1109/IISWC47752.2019.9042051](https://doi.org/10.1109/IISWC47752.2019.9042051) (cit. on p. 18).
- [28] H.-S. Philip Wong et al. “Metal–Oxide RRAM”. In: *Proceedings of the IEEE* 100.6 (2012-06), pp. 1951–1970. ISSN: 1558-2256. DOI: [10.1109/JPROC.2012.2190369](https://doi.org/10.1109/JPROC.2012.2190369) (cit. on pp. 18, 21).
- [29] Yangyin Chen. “ReRAM: History, Status, and Future”. In: *IEEE Transactions on Electron Devices* 67.4 (2020-04), pp. 1420–1433. ISSN: 0018-9383, 1557-9646. DOI: [10.1109/TED.2019.2961505](https://doi.org/10.1109/TED.2019.2961505). URL: <https://ieeexplore.ieee.org/document/8961211/> (visited on 2022-06-04) (cit. on p. 18).
- [30] M. Ezzadeen et al. “Low-Overhead Implementation of Binarized Neural Networks Employing Robust 2T2R Resistive RAM Bridges”. In: *ESSCIRC 2021 - IEEE 47th European Solid State Circuits Conference (ESSCIRC)*. 2021-09, pp. 83–86. DOI: [10.1109/ESSCIRC53450.2021.9567742](https://doi.org/10.1109/ESSCIRC53450.2021.9567742) (cit. on p. 19).
- [31] Young-Bae Kim et al. “Bi-layered RRAM with unlimited endurance and extremely uniform switching”. In: *2011 Symposium on VLSI Technology - Digest of Technical Papers*. 2011-06, pp. 52–53 (cit. on p. 19).
- [32] Y. S. Chen et al. “Highly scalable hafnium oxide memory with improvements of resistive distribution and read disturb immunity”. In: *2009 IEEE International Electron Devices Meeting (IEDM)*. 2009-12, pp. 1–4. DOI: [10.1109/IEDM.2009.5424411](https://doi.org/10.1109/IEDM.2009.5424411) (cit. on p. 19).
- [33] H.-S. Philip Wong et al. “Phase Change Memory”. In: *Proceedings of the IEEE* 98.12 (2010-12), pp. 2201–2227. ISSN: 1558-2256. DOI: [10.1109/JPROC.2010.2070050](https://doi.org/10.1109/JPROC.2010.2070050) (cit. on pp. 19–21).
- [34] Moinuddin K. Qureshi et al. “Enhancing lifetime and security of PCM-based Main Memory with Start-Gap Wear Leveling”. In: *2009 42nd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*. 2009-12, pp. 14–23. DOI: [10.1145/1669112.1669117](https://doi.org/10.1145/1669112.1669117) (cit. on p. 20).

- [35] Andre Seznec. “A Phase Change Memory as a Secure Main Memory”. In: *IEEE Computer Architecture Letters* 9.1 (2010-01), pp. 5–8. ISSN: 2473-2575. DOI: [10.1109/L-CA.2010.2](https://doi.org/10.1109/L-CA.2010.2) (cit. on p. 20).
- [36] T. Nirschl et al. “Write Strategies for 2 and 4-bit Multi-Level Phase-Change Memory”. In: *2007 IEEE International Electron Devices Meeting*. 2007-12, pp. 461–464. DOI: [10.1109/IEDM.2007.4418973](https://doi.org/10.1109/IEDM.2007.4418973) (cit. on p. 20).
- [37] Benjamin C. Lee et al. “Phase-Change Technology and the Future of Main Memory”. In: *IEEE Micro* 30.1 (2010-01), pp. 143–143. ISSN: 1937-4143. DOI: [10.1109/MM.2010.24](https://doi.org/10.1109/MM.2010.24) (cit. on p. 20).
- [38] Jaehyun Park, Donghwa Shin, and Hyung Gyu Lee. “Design space exploration of row buffer architecture for phase change memory with LPDDR2-NVM interface”. In: *2015 IFIP/IEEE International Conference on Very Large Scale Integration (VLSI-SoC)*. 2015-10, pp. 104–109. DOI: [10.1109/VLSI-SoC.2015.7314400](https://doi.org/10.1109/VLSI-SoC.2015.7314400) (cit. on p. 20).
- [39] Benjamin C. Lee et al. “Architecting phase change memory as a scalable dram alternative”. In: *Proceedings of the 36th annual international symposium on Computer architecture*. ISCA '09. New York, NY, USA: Association for Computing Machinery, 2009-06, pp. 2–13. ISBN: 978-1-60558-526-0. DOI: [10.1145/1555754.1555758](https://doi.org/10.1145/1555754.1555758). URL: <https://doi.org/10.1145/1555754.1555758> (visited on 2022-09-05) (cit. on p. 20).
- [40] Yiming Huai et al. “High Density 3D Cross-Point STT-MRAM”. In: *2018 IEEE International Memory Workshop (IMW)*. 2018-05, pp. 1–4. DOI: [10.1109/IMW.2018.8388833](https://doi.org/10.1109/IMW.2018.8388833) (cit. on p. 21).
- [41] An Chen. “A review of emerging non-volatile memory (NVM) technologies and applications”. en. In: *Solid-State Electronics*. Extended papers selected from ESSDERC 2015 125 (2016-11), pp. 25–38. ISSN: 0038-1101. DOI: [10.1016/j.sse.2016.07.006](https://doi.org/10.1016/j.sse.2016.07.006). URL: <http://www.sciencedirect.com/science/article/pii/S0038110116300867> (visited on 2020-09-07) (cit. on p. 21).
- [42] Sanjay Prajapati and Brajesh Kumar Kaushik. “Area and Energy Efficient Series Multilevel Cell STT-MRAMs for Optimized Read–Write Operations”. In: *IEEE Transactions on Magnetics* 55.1 (2019-01). Conference Name: IEEE Transactions on Magnetics, pp. 1–10. ISSN: 1941-0069. DOI: [10.1109/TMAG.2018.2875885](https://doi.org/10.1109/TMAG.2018.2875885) (cit. on p. 21).
- [43] Sparsh Mittal, Jeffrey S. Vetter, and Dong Li. “A Survey Of Architectural Approaches for Managing Embedded DRAM and Non-Volatile On-Chip Caches”. In: *IEEE Transactions on Parallel and Distributed Systems* 26.6 (2015-06), pp. 1524–1537. ISSN: 2161-9883. DOI: [10.1109/TPDS.2014.2324563](https://doi.org/10.1109/TPDS.2014.2324563) (cit. on p. 21).
- [44] Jalil Boukhobza et al. “Emerging NVM: A Survey on Architectural Integration and Research Challenges”. In: *ACM Transactions on Design Automation of Electronic Systems* 23 (2018-01). DOI: [10.1145/3131848](https://doi.org/10.1145/3131848) (cit. on p. 21).

- [45] Gianluca O. Puglia et al. “Non-Volatile Memory File Systems: A Survey”. In: *IEEE Access* 7 (2019), pp. 25836–25871. ISSN: 2169-3536. DOI: [10.1109/ACCESS.2019.2899463](https://doi.org/10.1109/ACCESS.2019.2899463) (cit. on p. 21).
- [46] Martin Hilbert and Priscila López. “The World’s Technological Capacity to Store, Communicate, and Compute Information”. en. In: *Science* 332.6025 (2011-04), pp. 60–65. ISSN: 0036-8075, 1095-9203. DOI: [10.1126/science.1200970](https://doi.org/10.1126/science.1200970). URL: <https://www.science.org/doi/10.1126/science.1200970> (visited on 2022-07-13) (cit. on p. 22).
- [47] *Data Durability, and Back-up at scale: A tale of "the Tape"*. en. URL: <https://community.ibm.com/community/user/storage/blogs/shawn-brume1/2020/07/14/data-durability-and-back-up-at-scale-a-tale-of-the> (visited on 2022-07-13) (cit. on pp. 22, 23).
- [48] Jingcheng Wang et al. “A 28-nm Compute SRAM With Bit-Serial Logic/Arithmetic Operations for Programmable In-Memory Vector Computing”. In: *IEEE Journal of Solid-State Circuits* (2019-11), pp. 1–11. ISSN: 1558-173X. DOI: [10.1109/JSSC.2019.2939682](https://doi.org/10.1109/JSSC.2019.2939682) (cit. on p. 24).
- [49] W. H. Kautz. “Cellular Logic-in-Memory Arrays”. In: *IEEE Transactions on Computers* C-18.8 (1969-08), pp. 719–727. ISSN: 0018-9340. DOI: [10.1109/T-C.1969.222754](https://doi.org/10.1109/T-C.1969.222754) (cit. on p. 25).
- [50] H. S. Stone. “A Logic-in-Memory Computer”. In: *IEEE Transactions on Computers* C-19.1 (1970-01), pp. 73–78. ISSN: 0018-9340. DOI: [10.1109/TC.1970.5008902](https://doi.org/10.1109/TC.1970.5008902). URL: <https://ieeexplore.ieee.org/abstract/document/5008902> (cit. on p. 25).
- [51] D. G. Elliott, W. M. Snelgrove, and M. Stumm. “Computational Ram: A Memory-simd Hybrid And Its Application To Dsp”. In: *1992 Proceedings of the IEEE Custom Integrated Circuits Conference*. 1992 Proceedings of the IEEE Custom Integrated Circuits Conference. 1992-05, pp. 30.6.1–30.6.4. DOI: [10.1109/CICC.1992.591879](https://doi.org/10.1109/CICC.1992.591879). URL: <http://www.eecg.toronto.edu/~dunc/cram/> (cit. on p. 25).
- [52] Wm. A. Wulf and Sally A. McKee. “Hitting the memory wall: implications of the obvious”. In: *ACM SIGARCH Computer Architecture News* 23.1 (1995-03), pp. 20–24. URL: <https://dl.acm.org/citation.cfm?id=216588> (cit. on p. 25).
- [53] M. Gokhale, B. Holmes, and K. Iobst. “Processing in memory: the Terasys massively parallel PIM array”. In: *Computer* 28.4 (1995-04), pp. 23–31. ISSN: 0018-9162. DOI: [10.1109/2.375174](https://doi.org/10.1109/2.375174). URL: <https://ieeexplore.ieee.org/abstract/document/375174> (cit. on p. 25).
- [54] D. Patterson et al. “A case for intelligent RAM”. In: *IEEE Micro* 17.2 (1997-03), pp. 34–44. ISSN: 0272-1732. DOI: [10.1109/40.592312](https://doi.org/10.1109/40.592312) (cit. on p. 25).

- [55] Yiran Chen. “Reshaping Future Computing Systems With Emerging Nonvolatile Memory Technologies”. In: *IEEE Micro* 39.1 (2019-01), pp. 54–57. ISSN: 1937-4143. DOI: [10.1109/MM.2018.2885588](https://doi.org/10.1109/MM.2018.2885588) (cit. on p. 25).
- [56] Roman Gauchi. “Exploration of Reconfigurable Tiles of Computing-in-Memory Architecture for Data-intensive Applications”. fr. PhD thesis. Université Grenoble Alpes: Université Grenoble Alpes, 2021.
- [57] K. C. Akyel et al. “DRC2: Dynamically Reconfigurable Computing Circuit based on memory architecture”. In: *2016 IEEE International Conference on Rebooting Computing (ICRC)*. 2016-10, pp. 1–8. DOI: [10.1109/ICRC.2016.7738698](https://doi.org/10.1109/ICRC.2016.7738698). URL: <http://ieeexplore.ieee.org/document/7738698/>.
- [58] S. Aga et al. “Compute Caches”. In: *2017 IEEE International Symposium on High Performance Computer Architecture (HPCA)*. 2017-02, pp. 481–492. DOI: [10.1109/HPCA.2017.21](https://doi.org/10.1109/HPCA.2017.21). URL: <http://ieeexplore.ieee.org/document/7920849/>.
- [59] Jianmin Zeng et al. “DM-IMCA: A dual-mode in-memory computing architecture for general purpose processing”. In: *IEICE Electronics Express* 17.4 (2020), pp. 20200005–20200005. DOI: [10.1587/elex.17.20200005](https://doi.org/10.1587/elex.17.20200005).
- [60] R. Khaddam-Aljameh et al. “An SRAM-Based Multibit In-Memory Matrix-Vector Multiplier With a Precision That Scales Linearly in Area, Time, and Power”. In: *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* (2020), pp. 1–14. ISSN: 1557-9999. DOI: [10.1109/TVLSI.2020.3037871](https://doi.org/10.1109/TVLSI.2020.3037871).
- [61] H. Chen et al. “Configurable 8T SRAM for Enabling in-Memory Computing”. In: *2019 2nd International Conference on Communication Engineering and Technology (ICCET)*. 2019-04, pp. 139–142. DOI: [10.1109/ICCET.2019.8726871](https://doi.org/10.1109/ICCET.2019.8726871).
- [62] Zhiting Lin et al. “In-Memory Computing With Double Word Lines and Three Read Ports for Four Operands”. In: *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 28.5 (2020-05), pp. 1316–1320. ISSN: 1557-9999. DOI: [10.1109/TVLSI.2020.2976099](https://doi.org/10.1109/TVLSI.2020.2976099).
- [63] A. Agrawal et al. “X-SRAM: Enabling In-Memory Boolean Computations in CMOS Static Random Access Memories”. In: *IEEE Transactions on Circuits and Systems I: Regular Papers* 65.12 (2018-12), pp. 4219–4232. ISSN: 1549-8328. DOI: [10.1109/TCSI.2018.2848999](https://doi.org/10.1109/TCSI.2018.2848999).
- [64] C. Eckert et al. “Neural Cache: Bit-Serial In-Cache Acceleration of Deep Neural Networks”. In: *2018 ACM/IEEE 45th Annual International Symposium on Computer Architecture (ISCA)*. 2018-06, pp. 383–396. DOI: [10.1109/ISCA.2018.00040](https://doi.org/10.1109/ISCA.2018.00040). URL: <https://ieeexplore.ieee.org/document/8416842>.
- [65] Y. Zhang et al. “Recryptor: A reconfigurable in-memory cryptographic Cortex-M0 processor for IoT”. In: *2017 Symposium on VLSI Circuits*. 2017-06, pp. C264–C265. DOI: [10.23919/VLSIC.2017.8008501](https://doi.org/10.23919/VLSIC.2017.8008501). URL: <http://ieeexplore.ieee.org/document/8008501/>.

- [66] Z. Jiang et al. “XNOR-SRAM: In-Memory Computing SRAM Macro for Binary/Ternary Deep Neural Networks”. In: *2018 IEEE Symposium on VLSI Technology*. 2018-06, pp. 173–174. DOI: [10.1109/VLSIT.2018.8510687](https://doi.org/10.1109/VLSIT.2018.8510687).
- [67] J. Wang et al. “A Compute SRAM with Bit-Serial Integer/Floating-Point Operations for Programmable In-Memory Vector Acceleration”. In: *2019 IEEE International Solid-State Circuits Conference - (ISSCC)*. San Francisco, USA, 2019-02, pp. 224–226. DOI: [10.1109/ISSCC.2019.8662419](https://doi.org/10.1109/ISSCC.2019.8662419).
- [68] D. Jeon et al. “A 23-mW Face Recognition Processor with Mostly-Read 5T Memory in 40-nm CMOS”. In: *IEEE Journal of Solid-State Circuits* 52.6 (2017-06), pp. 1628–1642. ISSN: 0018-9200. DOI: [10.1109/JSSC.2017.2661838](https://doi.org/10.1109/JSSC.2017.2661838). URL: <http://ieeexplore.ieee.org/document/7859309/>.
- [69] L. Fick et al. “Analog in-memory subthreshold deep neural network accelerator”. In: *2017 IEEE Custom Integrated Circuits Conference (CICC)*. 2017-04, pp. 1–4. DOI: [10.1109/CICC.2017.7993629](https://doi.org/10.1109/CICC.2017.7993629).
- [70] H. E. Sumbul et al. “A 2.9–33.0 TOPS/W Reconfigurable 1-D/2-D Compute-Near-Memory Inference Accelerator in 10-nm FinFET CMOS”. In: *IEEE Solid-State Circuits Letters* 3 (2020), pp. 118–121. ISSN: 2573-9603. DOI: [10.1109/LSSC.2020.3007185](https://doi.org/10.1109/LSSC.2020.3007185).
- [71] A. Biswas and A. P. Chandrakasan. “Conv-RAM: An energy-efficient SRAM with embedded convolution computation for low-power CNN-based machine learning applications”. In: *2018 IEEE International Solid-State Circuits Conference - (ISSCC)*. 2018-02, pp. 488–490. DOI: [10.1109/ISSCC.2018.8310397](https://doi.org/10.1109/ISSCC.2018.8310397).
- [72] J. Saikia et al. “K-Nearest Neighbor Hardware Accelerator Using In-Memory Computing SRAM”. In: *2019 IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED)*. 2019-07, pp. 1–6. DOI: [10.1109/ISLPED.2019.8824822](https://doi.org/10.1109/ISLPED.2019.8824822).
- [73] William Simon et al. “A Fast, Reliable and Wide-Voltage-Range In-Memory Computing Architecture”. In: ACM, 2019-02, p. 83. ISBN: 978-1-4503-6725-7. DOI: [10.1145/3316781.3317741](https://doi.org/10.1145/3316781.3317741). URL: <http://dl.acm.org/citation.cfm?id=3316781.3317741> (visited on 2019-07-08).
- [74] S. Srinivasa et al. “ROBIN: Monolithic-3D SRAM for Enhanced Robustness with In-Memory Computation Support”. In: *IEEE Transactions on Circuits and Systems I: Regular Papers* 66.7 (2019-07), pp. 2533–2545. ISSN: 1549-8328. DOI: [10.1109/TCSI.2019.2897497](https://doi.org/10.1109/TCSI.2019.2897497).
- [75] Onur Mutlu et al. “Processing data where it makes sense: Enabling in-memory computation”. In: *Microprocessors and Microsystems* 67 (2019-06), pp. 28–41. ISSN: 0141-9331. DOI: [10.1016/j.micpro.2019.01.009](https://doi.org/10.1016/j.micpro.2019.01.009). URL: <http://www.sciencedirect.com/science/article/pii/S0141933118302291> (visited on 2019-07-09).

- [76] S. Jeloka et al. "A 28 nm Configurable Memory (TCAM/BCAM/SRAM) Using Push-Rule 6T Bit Cell Enabling Logic-in-Memory". In: *IEEE Journal of Solid-State Circuits* 51.4 (2016-04), pp. 1009–1021. ISSN: 0018-9200. DOI: [10.1109/JSSC.2016.2515510](https://doi.org/10.1109/JSSC.2016.2515510). URL: <http://ieeexplore.ieee.org/document/7400984/>.
- [77] W. Khwa et al. "A 65nm 4Kb algorithm-dependent computing-in-memory SRAM unit-macro with 2.3ns and 55.8TOPS/W fully parallel product-sum operation for binary DNN edge processors". In: *2018 IEEE International Solid - State Circuits Conference - (ISSCC)*. 2018-02, pp. 496–498. DOI: [10.1109/ISSCC.2018.8310401](https://doi.org/10.1109/ISSCC.2018.8310401).
- [78] Jinseok Kim et al. "Area-Efficient and Variation-Tolerant In-Memory BNN Computing using 6T SRAM Array". en. In: *VLSI Circuits* (2019), p. 2.
- [79] J. Zhang, Z. Wang, and N. Verma. "In-Memory Computation of a Machine-Learning Classifier in a Standard 6T SRAM Array". In: *IEEE Journal of Solid-State Circuits* 52.4 (2017-04), pp. 915–924. ISSN: 0018-9200. DOI: [10.1109/JSSC.2016.2642198](https://doi.org/10.1109/JSSC.2016.2642198).
- [80] K. Ando et al. "BRein memory: A 13-layer 4.2 K neuron/0.8 M synapse binary/ternary reconfigurable in-memory deep neural network accelerator in 65 nm CMOS". In: *2017 Symposium on VLSI Circuits*. 2017-06, pp. C24–C25. DOI: [10.23919/VLSIC.2017.8008533](https://doi.org/10.23919/VLSIC.2017.8008533).
- [81] M. Kang, S. K. Gonugondla, and N. R. Shanbhag. "A 19.4 nJ/decision 364K decisions/s in-memory random forest classifier in 6T SRAM array". In: *ESSCIRC 2017 - 43rd IEEE European Solid State Circuits Conference*. 2017-09, pp. 263–266. DOI: [10.1109/ESSCIRC.2017.8094576](https://doi.org/10.1109/ESSCIRC.2017.8094576).
- [82] Jian-Wei Su et al. "15.2 A 28nm 64Kb Inference-Training Two-Way Transpose Multibit 6T SRAM Compute-in-Memory Macro for AI Edge Chips". In: *2020 IEEE International Solid- State Circuits Conference - (ISSCC)*. 2020-02, pp. 240–242. DOI: [10.1109/ISSCC19947.2020.9062949](https://doi.org/10.1109/ISSCC19947.2020.9062949).
- [83] Vivek Seshadri et al. "RowClone: Fast and energy-efficient in-DRAM bulk data copy and initialization". In: *2013 46th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*. 2013-12, pp. 185–197.
- [84] Dongping Zhang et al. "TOP-PIM: Throughput-oriented Programmable Processing in Memory". In: *Proceedings of the 23rd International Symposium on High-performance Parallel and Distributed Computing*. HPDC '14. New York, NY, USA: ACM, 2014, pp. 85–98. ISBN: 978-1-4503-2749-7. DOI: [10.1145/2600212.2600213](https://doi.org/10.1145/2600212.2600213). URL: <http://doi.acm.org/10.1145/2600212.2600213> (visited on 2019-02-08).

- [85] Vivek Seshadri et al. “Ambit: In-memory Accelerator for Bulk Bitwise Operations Using Commodity DRAM Technology”. In: *Proceedings of the 50th Annual IEEE/ACM International Symposium on Microarchitecture*. MICRO-50 '17. New York, NY, USA: ACM, 2017, pp. 273–287. ISBN: 978-1-4503-4952-9. DOI: [10.1145/3123939.3124544](https://doi.org/10.1145/3123939.3124544). URL: <http://doi.acm.org/10.1145/3123939.3124544> (visited on 2019-02-08).
- [86] Amirali Boroumand et al. “Google Workloads for Consumer Devices: Mitigating Data Movement Bottlenecks”. en. In: *Proceedings of the Twenty-Third International Conference on Architectural Support for Programming Languages and Operating Systems - ASPLOS '18*. Williamsburg, VA, USA: ACM Press, 2018, pp. 316–331. ISBN: 978-1-4503-4911-6. DOI: [10.1145/3173162.3173177](https://doi.org/10.1145/3173162.3173177). URL: <http://dl.acm.org/citation.cfm?doid=3173162.3173177> (visited on 2019-02-08).
- [87] Vasileios Zois et al. “Massively Parallel Skyline Computation for Processing-in-memory Architectures”. In: *Proceedings of the 27th International Conference on Parallel Architectures and Compilation Techniques*. PACT '18. New York, NY, USA: ACM, 2018, 1:1–1:12. ISBN: 978-1-4503-5986-3. DOI: [10.1145/3243176.3243187](https://doi.org/10.1145/3243176.3243187). URL: <http://doi.acm.org/10.1145/3243176.3243187> (visited on 2019-02-08).
- [88] Fabrice Devaux and Jean-François Roy. “Memory circuit with integrated processor”. en. US10324870B2. 2019-06. URL: <https://patents.google.com/patent/US10324870B2/en> (visited on 2019-10-24).
- [89] Mingxuan He et al. “Newton: A DRAM-maker’s Accelerator-in-Memory (AiM) Architecture for Machine Learning”. In: *2020 53rd Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*. 2020-10, pp. 372–385. DOI: [10.1109/MICRO50266.2020.00040](https://doi.org/10.1109/MICRO50266.2020.00040).
- [90] Shaahin Angizi and Deliang Fan. “GraphiDe: A Graph Processing Accelerator leveraging In-DRAM-Computing”. In: *Proceedings of the 2019 on Great Lakes Symposium on VLSI*. GLSVLSI '19. New York, NY, USA: Association for Computing Machinery, 2019-05, pp. 45–50. ISBN: 978-1-4503-6252-8. DOI: [10.1145/3299874.3317984](https://doi.org/10.1145/3299874.3317984). URL: <https://doi.org/10.1145/3299874.3317984> (visited on 2022-07-30).
- [91] Shuangchen Li et al. “SCOPE: A Stochastic Computing Engine for DRAM-Based In-Situ Accelerator”. In: *2018 51st Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*. 2018-10, pp. 696–709. DOI: [10.1109/MICRO.2018.00062](https://doi.org/10.1109/MICRO.2018.00062).
- [92] *UPMEM*. 2017. URL: <http://www.upmem.com/> (visited on 2017-10-23).
- [93] Dominique Lavenier et al. *BLAST on UPMEM*. Research Report RR-8878. INRIA Rennes - Bretagne Atlantique, 2016-03, p. 20. URL: <https://hal.archives-ouvertes.fr/hal-01294345> (visited on 2019-02-08).

- [94] Fei Gao, Georgios Tziantzioulis, and David Wentzlaff. “ComputeDRAM: In-Memory Compute Using Off-the-Shelf DRAMs”. In: *Proceedings of the 52nd Annual IEEE/ACM International Symposium on Microarchitecture*. MICRO ’52. New York, NY, USA: Association for Computing Machinery, 2019-10, pp. 100–113. ISBN: 978-1-4503-6938-1. DOI: [10.1145/3352460.3358260](https://doi.org/10.1145/3352460.3358260). URL: <https://doi.org/10.1145/3352460.3358260> (visited on 2022-07-30).
- [95] Shanshan Xie et al. “16.2 eDRAM-CIM: Compute-In-Memory Design with Reconfigurable Embedded-Dynamic-Memory Array Realizing Adaptive Data Converters and Charge-Domain Computing”. In: *2021 IEEE International Solid-State Circuits Conference (ISSCC)*. Vol. 64. 2021-02, pp. 248–250. DOI: [10.1109/ISSCC42613.2021.9365932](https://doi.org/10.1109/ISSCC42613.2021.9365932).
- [96] Samsung. *PIM| Technology*. en. 2021. URL: <https://semiconductor.samsung.com/content/semiconductor/global/insights/technology/pim.html> (visited on 2022-08-06).
- [97] Samsung. *Samsung Brings In-Memory Processing Power to Wider Range of Applications*. en. 2021. URL: <https://news.samsung.com/global/samsung-brings-in-memory-processing-power-to-wider-range-of-applications> (visited on 2022-08-06).
- [98] Thomas Vogelsang. “Understanding the Energy Consumption of Dynamic Random Access Memories”. In: *2010 43rd Annual IEEE/ACM International Symposium on Microarchitecture*. 2010-12, pp. 363–374. DOI: [10.1109/MICRO.2010.42](https://doi.org/10.1109/MICRO.2010.42).
- [99] Peng Li, Kevin Gomez, and David J. Lilja. “Exploiting free silicon for energy-efficient computing directly in NAND flash-based solid-state storage systems”. In: *2013 IEEE High Performance Extreme Computing Conference (HPEC)*. 2013-09, pp. 1–6. DOI: [10.1109/HPEC.2013.6670317](https://doi.org/10.1109/HPEC.2013.6670317).
- [100] Panni Wang et al. “Three-Dimensional nand Flash for Vector–Matrix Multiplication”. In: *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 27.4 (2019-04), pp. 988–991. ISSN: 1557-9999. DOI: [10.1109/TVLSI.2018.2882194](https://doi.org/10.1109/TVLSI.2018.2882194).
- [101] Hang-Ting Lue et al. “Optimal Design Methods to Transform 3D NAND Flash into a High-Density, High-Bandwidth and Low-Power Nonvolatile Computing in Memory (nvCIM) Accelerator for Deep-Learning Neural Networks (DNN)”. In: *2019 IEEE International Electron Devices Meeting (IEDM)*. 2019-12, pp. 38.1.1–38.1.4. DOI: [10.1109/IEDM19573.2019.8993652](https://doi.org/10.1109/IEDM19573.2019.8993652).
- [102] Wonbo Shim and Shimeng Yu. “Technological Design of 3D NAND-Based Compute-in-Memory Architecture for GB-Scale Deep Neural Network”. In: *IEEE Electron Device Letters* 42.2 (2021-02), pp. 160–163. ISSN: 1558-0563. DOI: [10.1109/LED.2020.3048101](https://doi.org/10.1109/LED.2020.3048101).

- [103] Won Ho Choi et al. “An In-Flash Binary Neural Network Accelerator with SLC NAND Flash Array”. In: *2020 IEEE International Symposium on Circuits and Systems (ISCAS)*. 2020-10, pp. 1–5. DOI: [10.1109/ISCAS45731.2020.9180920](https://doi.org/10.1109/ISCAS45731.2020.9180920).
- [104] Wen Zhou et al. “Temporal Correlation Detection Based on 3D NAND Flash In-Memory Computing”. In: *IEEE Electron Device Letters* 43.6 (2022-06), pp. 874–877. ISSN: 1558-0563. DOI: [10.1109/LED.2022.3170593](https://doi.org/10.1109/LED.2022.3170593).
- [105] Minsu Kim et al. “An Embedded nand Flash-Based Compute-In-Memory Array Demonstrated in a Standard Logic Process”. In: *IEEE Journal of Solid-State Circuits* 57.2 (2022-02), pp. 625–638. ISSN: 1558-173X. DOI: [10.1109/JSSC.2021.3098671](https://doi.org/10.1109/JSSC.2021.3098671).
- [106] Samsung. *Smart SSD | SSD Card*. en. 2020. URL: <https://semiconductor.samsung.com/content/semiconductor/global/ssd/smart-ssd.html> (visited on 2022-08-09).
- [107] S. Kvatinsky et al. “Memristor-Based Material Implication (IMPLY) Logic: Design Principles and Methodologies”. In: *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 22.10 (2014-10), pp. 2054–2066. ISSN: 1063-8210. DOI: [10.1109/TVLSI.2013.2282132](https://doi.org/10.1109/TVLSI.2013.2282132).
- [108] S. Kvatinsky et al. “MAGIC—Memristor-Aided Logic”. In: *IEEE Transactions on Circuits and Systems II: Express Briefs* 61.11 (2014-11), pp. 895–899. ISSN: 1549-7747. DOI: [10.1109/TCSII.2014.2357292](https://doi.org/10.1109/TCSII.2014.2357292). URL: <https://ieeexplore.ieee.org/abstract/document/6895258>.
- [109] N. Talati et al. “Practical challenges in delivering the promises of real processing-in-memory machines”. In: *2018 Design, Automation Test in Europe Conference Exhibition (DATE)*. 2018-03, pp. 1628–1633. DOI: [10.23919/DATE.2018.8342275](https://doi.org/10.23919/DATE.2018.8342275). URL: <https://ieeexplore.ieee.org/document/8342275>.
- [110] A. Haj-Ali et al. “Not in Name Alone: A Memristive Memory Processing Unit for Real In-Memory Processing”. In: *IEEE Micro* 38.5 (2018-09), pp. 13–21. ISSN: 0272-1732. DOI: [10.1109/MM.2018.053631137](https://doi.org/10.1109/MM.2018.053631137). URL: <https://ieeexplore.ieee.org/abstract/document/8474943>.
- [111] A. Haj-Ali et al. “Efficient Algorithms for In-Memory Fixed Point Multiplication Using MAGIC”. In: *2018 IEEE International Symposium on Circuits and Systems (ISCAS)*. 2018-05, pp. 1–5. DOI: [10.1109/ISCAS.2018.8351561](https://doi.org/10.1109/ISCAS.2018.8351561). URL: <https://ieeexplore.ieee.org/document/8351561>.
- [112] R. Ben Hur et al. “Simple magic: Synthesis and in-memory Mapping of logic execution for memristor-aided logic”. In: *2017 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*. 2017-11, pp. 225–232. DOI: [10.1109/ICCAD.2017.8203782](https://doi.org/10.1109/ICCAD.2017.8203782). URL: <https://ieeexplore.ieee.org/document/8203782>.
- [113] P. Gaillardon et al. “The Programmable Logic-in-Memory (PLiM) computer”. In: *2016 Design, Automation Test in Europe Conference Exhibition (DATE)*. 2016-03, pp. 427–432. URL: <https://ieeexplore.ieee.org/document/7459349>.

- [114] M. Abu Lebdeh et al. “An Efficient Heterogeneous Memristive xnor for In-Memory Computing”. In: *IEEE Transactions on Circuits and Systems I: Regular Papers* 64.9 (2017-09), pp. 2427–2437. ISSN: 1549-8328. DOI: [10.1109/TCSI.2017.2706299](https://doi.org/10.1109/TCSI.2017.2706299).
- [115] João Vieira et al. “A Product Engine for Energy-Efficient Execution of Binary Neural Networks Using Resistive Memories”. In: *2019 IFIP/IEEE 27th International Conference on Very Large Scale Integration (VLSI-SoC)*. 2019-10, pp. 160–165. DOI: [10.1109/VLSI-SoC.2019.8920343](https://doi.org/10.1109/VLSI-SoC.2019.8920343).
- [116] Tianqi Tang et al. “Binary convolutional neural network on RRAM”. In: *2017 22nd Asia and South Pacific Design Automation Conference (ASP-DAC)*. 2017-01, pp. 782–787. DOI: [10.1109/ASPDAC.2017.7858419](https://doi.org/10.1109/ASPDAC.2017.7858419).
- [117] Mahdi Nazm Bojnordi and Engin Ipek. “Memristive Boltzmann machine: A hardware accelerator for combinatorial optimization and deep learning”. In: *2016 IEEE International Symposium on High Performance Computer Architecture (HPCA)*. 2016-03, pp. 1–13. DOI: [10.1109/HPCA.2016.7446049](https://doi.org/10.1109/HPCA.2016.7446049).
- [118] Jaesung Park et al. “TiOx-Based RRAM Synapse With 64-Levels of Conductance and Symmetric Conductance Change by Adopting a Hybrid Pulse Scheme for Neuromorphic Computing”. In: *IEEE Electron Device Letters* 37.12 (2016-12), pp. 1559–1562. ISSN: 1558-0563. DOI: [10.1109/LED.2016.2622716](https://doi.org/10.1109/LED.2016.2622716).
- [119] F. Su et al. “A 462GOPS/J RRAM-based nonvolatile intelligent processor for energy harvesting IoE system featuring nonvolatile logics and processing-in-memory”. In: *2017 Symposium on VLSI Circuits*. 2017-06, pp. C260–C261. DOI: [10.23919/VLSIC.2017.8008585](https://doi.org/10.23919/VLSIC.2017.8008585).
- [120] W. Chen et al. “A 16Mb dual-mode ReRAM macro with sub-14ns computing-in-memory and memory functions enabled by self-write termination scheme”. In: *2017 IEEE International Electron Devices Meeting (IEDM)*. 2017-12, pp. 28.2.1–28.2.4. DOI: [10.1109/IEDM.2017.8268468](https://doi.org/10.1109/IEDM.2017.8268468).
- [121] W. Chen et al. “A 65nm 1Mb nonvolatile computing-in-memory ReRAM macro with sub-16ns multiply-and-accumulate for binary DNN AI edge processors”. In: *2018 IEEE International Solid - State Circuits Conference - (ISSCC)*. 2018-02, pp. 494–496. DOI: [10.1109/ISSCC.2018.8310400](https://doi.org/10.1109/ISSCC.2018.8310400).
- [122] Qi Liu et al. “33.2 A Fully Integrated Analog ReRAM Based 78.4TOPS/W Compute-In-Memory Chip with Fully Parallel MAC Computing”. In: *2020 IEEE International Solid- State Circuits Conference - (ISSCC)*. 2020-02, pp. 500–502. DOI: [10.1109/ISSCC19947.2020.9062953](https://doi.org/10.1109/ISSCC19947.2020.9062953).
- [123] Zhuo-Rui Wang et al. “Efficient Implementation of Boolean and Full-Adder Functions With 1T1R RRAMs for Beyond Von Neumann In-Memory Computing”. In: *IEEE Transactions on Electron Devices* 65.10 (2018-10), pp. 4659–4666. ISSN: 1557-9646. DOI: [10.1109/TED.2018.2866048](https://doi.org/10.1109/TED.2018.2866048).
- [124] A Sebastian et al. “Computational memory-based inference and training of deep neural networks”. en. In: *VLSI Technology* (2019), p. 2.

- [125] Jing Li et al. “1 Mb 0.41 μm^2 2T-2R Cell Nonvolatile TCAM With Two-Bit Encoding and Clocked Self-Referenced Sensing”. en. In: *IEEE Journal of Solid-State Circuits* 49.4 (2014-04), pp. 896–907. ISSN: 0018-9200, 1558-173X. DOI: [10.1109/JSSC.2013.2292055](https://doi.org/10.1109/JSSC.2013.2292055). URL: <http://ieeexplore.ieee.org/document/6680770/> (visited on 2020-09-07).
- [126] P. Narayanan et al. “Fully On-Chip MAC at 14 nm Enabled by Accurate Row-Wise Programming of PCM-Based Weights and Parallel Vector-Transport in Duration-Format”. In: *IEEE Transactions on Electron Devices* 68.12 (2021-12), pp. 6629–6636. ISSN: 1557-9646. DOI: [10.1109/TED.2021.3115993](https://doi.org/10.1109/TED.2021.3115993).
- [127] Riduan Khaddam-Aljameh et al. “HERMES-Core—A 1.59-TOPS/ mm^2 PCM on 14-nm CMOS In-Memory Compute Core Using 300-ps/LSB Linearized CCO-Based ADCs”. In: *IEEE Journal of Solid-State Circuits* 57.4 (2022-04), pp. 1027–1038. ISSN: 1558-173X. DOI: [10.1109/JSSC.2022.3140414](https://doi.org/10.1109/JSSC.2022.3140414).
- [128] Wang Kang et al. “In-Memory Processing Paradigm for Bitwise Logic Operations in STT-MRAM”. In: *IEEE Transactions on Magnetics* 53.11 (2017-11), pp. 1–4. ISSN: 1941-0069. DOI: [10.1109/TMAG.2017.2703863](https://doi.org/10.1109/TMAG.2017.2703863).
- [129] Shubham Jain et al. “Computing in Memory With Spin-Transfer Torque Magnetic RAM”. In: *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 26.3 (2018-03), pp. 470–483. ISSN: 1557-9999. DOI: [10.1109/TVLSI.2017.2776954](https://doi.org/10.1109/TVLSI.2017.2776954).
- [130] Masoud Zabihi et al. “In-Memory Processing on the Spintronic CRAM: From Hardware Design to Application Mapping”. In: *IEEE Transactions on Computers* 68.8 (2019-08), pp. 1159–1173. ISSN: 1557-9956. DOI: [10.1109/TC.2018.2858251](https://doi.org/10.1109/TC.2018.2858251).
- [131] Tifenn Hirtzlin et al. “Stochastic Computing for Hardware Implementation of Binarized Neural Networks”. In: *IEEE Access* 7 (2019), pp. 76394–76403. ISSN: 2169-3536. DOI: [10.1109/ACCESS.2019.2921104](https://doi.org/10.1109/ACCESS.2019.2921104).
- [132] Tung-Cheng Chang et al. “13.4 A 22nm 1Mb 1024b-Read and Near-Memory-Computing Dual-Mode STT-MRAM Macro with 42.6GB/s Read Bandwidth for Security-Aware Mobile Devices”. In: *2020 IEEE International Solid-State Circuits Conference - (ISSCC)*. 2020-02, pp. 224–226. DOI: [10.1109/ISSCC19947.2020.9063072](https://doi.org/10.1109/ISSCC19947.2020.9063072).
- [133] Peter Deaville et al. “A Maximally Row-Parallel MRAM In-Memory-Computing Macro Addressing Readout Circuit Sensitivity and Area”. In: *ESSCIRC 2021 - IEEE 47th European Solid State Circuits Conference (ESSCIRC)*. 2021-09, pp. 75–78. DOI: [10.1109/ESSCIRC53450.2021.9567807](https://doi.org/10.1109/ESSCIRC53450.2021.9567807).
- [134] Shuangchen Li et al. “Pinatubo: A Processing-in-memory Architecture for Bulk Bitwise Operations in Emerging Non-volatile Memories”. In: *Proceedings of the 53rd Annual Design Automation Conference*. DAC '16. New York, NY, USA: ACM, 2016, 173:1–173:6. ISBN: 978-1-4503-4236-0. DOI: [10.1145/2897937](https://doi.org/10.1145/2897937).

2898064. URL: <http://doi.acm.org/10.1145/2897937.2898064> (visited on 2019-02-08).
- [135] M. Imani, S. Gupta, and T. Rosing. “GenPIM: Generalized processing in-memory to accelerate data intensive applications”. In: *2018 Design, Automation Test in Europe Conference Exhibition (DATE)*. 2018-03, pp. 1155–1158. DOI: [10.23919/DATE.2018.8342186](https://doi.org/10.23919/DATE.2018.8342186). URL: <https://ieeexplore.ieee.org/document/8342186>.
- [136] Shun Okamoto et al. “Application Driven SCM and NAND Flash Hybrid SSD Design for Data-Centric Computing System”. In: *2015 IEEE International Memory Workshop (IMW)*. 2015-05, pp. 1–4. DOI: [10.1109/IMW.2015.7150277](https://doi.org/10.1109/IMW.2015.7150277).
- [137] Ken Takeuchi. “Data-aware NAND flash memory for intelligent computing with deep neural network”. In: *2017 IEEE International Electron Devices Meeting (IEDM)*. 2017-12, pp. 28.4.1–28.4.4. DOI: [10.1109/IEDM.2017.8268470](https://doi.org/10.1109/IEDM.2017.8268470).
- [138] Linbin Chen et al. “CCE: A Combined SRAM and Non Volatile Cache for Endurance of Next Generation Multilevel Non Volatile Memories in Embedded Systems”. In: *Proceedings of the 14th IEEE/ACM International Symposium on Nanoscale Architectures*. NANOARCH '18. New York, NY, USA: ACM, 2018, pp. 58–64. ISBN: 978-1-4503-5815-6. DOI: [10.1145/3232195.3232196](https://doi.org/10.1145/3232195.3232196). URL: <http://doi.acm.org/10.1145/3232195.3232196> (visited on 2019-01-07).
- [139] Xueyong Zhang, Vivek Mohan, and Arindam Basu. “CRAM: Collocated SRAM and DRAM With In-Memory Computing-Based Denoising and Filling for Neuromorphic Vision Sensors in 65 nm CMOS”. In: *IEEE Transactions on Circuits and Systems II: Express Briefs* 67.5 (2020-05), pp. 816–820. ISSN: 1558-3791. DOI: [10.1109/TCSII.2020.2980125](https://doi.org/10.1109/TCSII.2020.2980125).
- [140] Marco Rios et al. “Running Efficiently CNNs on the Edge Thanks to Hybrid SRAM-RRAM In-Memory Computing”. In: *2021 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. 2021-02, pp. 1881–1886. DOI: [10.23919/DATE51398.2021.9474233](https://doi.org/10.23919/DATE51398.2021.9474233).
- [141] Hwajung Kim, Heon Y. Yeom, and Hanul Sung. “Understanding the Performance Characteristics of Computational Storage Drives: A Case Study with SmartSSD”. en. In: *Electronics* 10.21 (2021-10), p. 2617. ISSN: 2079-9292. DOI: [10.3390/electronics10212617](https://doi.org/10.3390/electronics10212617). URL: <https://www.mdpi.com/2079-9292/10/21/2617> (visited on 2022-08-06).
- [142] Di Gao et al. “Eva-CiM: A System-Level Performance and Energy Evaluation Framework for Computing-in-Memory Architectures”. In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* (2020), pp. 1–1. ISSN: 1937-4151. DOI: [10.1109/TCAD.2020.2966484](https://doi.org/10.1109/TCAD.2020.2966484).

- [143] R. Gauchi et al. “Reconfigurable tiles of computing-in-memory SRAM architecture for scalable vectorization”. In: *Proceedings of the ACM/IEEE International Symposium on Low Power Electronics and Design*. ISLPED '20. New York, NY, USA: Association for Computing Machinery, 2020-08, pp. 121–126. ISBN: 978-1-4503-7053-0. DOI: [10.1145/3370748.3406550](https://doi.org/10.1145/3370748.3406550). URL: <https://doi.org/10.1145/3370748.3406550> (visited on 2020-09-11).
- [144] Junwhan Ahn et al. “PIM-enabled instructions: A low-overhead, locality-aware processing-in-memory architecture”. In: *2015 ACM/IEEE 42nd Annual International Symposium on Computer Architecture (ISCA)*. 2015-06, pp. 336–348. DOI: [10.1145/2749469.2750385](https://doi.org/10.1145/2749469.2750385).
- [145] Maciej Besta et al. “SISA: Set-Centric Instruction Set Architecture for Graph Mining on Processing-in-Memory Systems”. In: *arXiv:2104.07582 [cs]* (2021-04). URL: <http://arxiv.org/abs/2104.07582> (visited on 2021-04-26).
- [146] J. v Lunteren et al. “Coherently Attached Programmable Near-Memory Acceleration Platform and its application to Stencil Processing”. In: *2019 Design, Automation Test in Europe Conference Exhibition (DATE)*. 2019-03, pp. 668–673. DOI: [10.23919/DATE.2019.8715088](https://doi.org/10.23919/DATE.2019.8715088).
- [147] Liang Chang et al. “DASM: Data-Streaming-Based Computing in Nonvolatile Memory Architecture for Embedded System”. In: *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 27.9 (2019-09), pp. 2046–2059. ISSN: 1557-9999. DOI: [10.1109/TVLSI.2019.2912941](https://doi.org/10.1109/TVLSI.2019.2912941).
- [148] N. Verma et al. “In-Memory Computing: Advances and Prospects”. In: *IEEE Solid-State Circuits Magazine* 11.3 (2019), pp. 43–55. ISSN: 1943-0590. DOI: [10.1109/MSSC.2019.2922889](https://doi.org/10.1109/MSSC.2019.2922889).
- [149] Chuan-Jia Jhang et al. “Challenges and Trends of SRAM-Based Computing-In-Memory for AI Edge Devices”. In: *IEEE Transactions on Circuits and Systems I: Regular Papers* 68.5 (2021-05), pp. 1773–1786. ISSN: 1558-0806. DOI: [10.1109/TCSI.2021.3064189](https://doi.org/10.1109/TCSI.2021.3064189).
- [150] Abu Sebastian et al. “Memory devices and applications for in-memory computing”. en. In: *Nature Nanotechnology* 15.7 (2020-07), pp. 529–544. ISSN: 1748-3395. DOI: [10.1038/s41565-020-0655-z](https://doi.org/10.1038/s41565-020-0655-z). URL: <https://www.nature.com/articles/s41565-020-0655-z> (visited on 2022-08-10).
- [151] M. Aamir, Somya Sharma, and Anuj Grover. “ChaCha20-in-Memory for Side-Channel Resistance in IoT Edge-Node Devices”. In: *IEEE Open Journal of Circuits and Systems* 2 (2021), pp. 833–842. ISSN: 2644-1225. DOI: [10.1109/OJCS.2021.3127273](https://doi.org/10.1109/OJCS.2021.3127273).
- [152] I. Giannopoulos et al. “8-bit Precision In-Memory Multiplication with Projected Phase-Change Memory”. In: *2018 IEEE International Electron Devices Meeting (IEDM)*. 2018-12, pp. 27.7.1–27.7.4. DOI: [10.1109/IEDM.2018.8614558](https://doi.org/10.1109/IEDM.2018.8614558).

- [153] T. B. Preußner et al. “Inference of quantized neural networks on heterogeneous all-programmable devices”. In: *2018 Design, Automation Test in Europe Conference Exhibition (DATE)*. 2018-03, pp. 833–838. DOI: [10.23919/DATF.2018.8342121](https://doi.org/10.23919/DATF.2018.8342121). URL: <http://ieeexplore.ieee.org/document/8342121/>.
- [154] M. Kooli et al. “Smart instruction codes for in-memory computing architectures compatible with standard SRAM interfaces”. In: *2018 Design, Automation Test in Europe Conference Exhibition (DATE)*. 2018-03, pp. 1634–1639. DOI: [10.23919/DATF.2018.8342276](https://doi.org/10.23919/DATF.2018.8342276). URL: <http://ieeexplore.ieee.org/document/8342276/>.
- [155] Henry S. Warren. *Hacker's delight*. en. 2nd ed. Upper Saddle River, NJ: Addison-Wesley, 2013. ISBN: 978-0-321-84268-8.
- [156] Andrew Waterman, Krste Asanovic, and CS Division. “RISC-V Unprivileged ISA specification”. en. In: 1 (2019), p. 238.
- [157] Hossein Valavi et al. “A Mixed-Signal Binarized Convolutional-Neural-Network Accelerator Integrating Dense Weight Storage and Multiplication for Reduced Data Movement”. In: *2018 IEEE Symposium on VLSI Circuits*. 2018 IEEE Symposium on VLSI Circuits. 2018-06, pp. 141–142. DOI: [10.1109/VLSIC.2018.8502421](https://doi.org/10.1109/VLSIC.2018.8502421).
- [158] Louis-Noel Pouchet and Tomofumi Yuki. *PolyBench/C – The polyhedral benchmark suite*. 2015. URL: <https://web.cse.ohio-state.edu/~pouchet.2/software/polybench/> (visited on 2022-08-23).
- [159] Joseph Redmon. *Darknet: Open Source Neural Networks in C*. <http://pjreddie.com/darknet/>. 2013. URL: <https://pjreddie.com/darknet/> (visited on 2022-08-23).
- [160] Bruce Fleischer and Sunil Shukla. *Approximate Computing for On-Chip AI Acceleration: IBM Research at VLSI*. en-US. 2018-06. URL: <https://www.ibm.com/blogs/research/2018/06/approximate-computing-ai-acceleration/> (visited on 2022-08-31).
- [161] Wikipedia. *Bitap algorithm*. en. Page Version ID: 1063296095. 2022-01. URL: https://en.wikipedia.org/w/index.php?title=Bitap_algorithm&oldid=1063296095 (visited on 2022-08-31).
- [162] *BLAS (Basic Linear Algebra Subprograms)*. 2021. URL: <https://netlib.org/blas/> (visited on 2022-08-23).
- [163] Jia Deng et al. “ImageNet: A large-scale hierarchical image database”. In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*. 2009-06, pp. 248–255. DOI: [10.1109/CVPR.2009.5206848](https://doi.org/10.1109/CVPR.2009.5206848).
- [164] Wikichip. *Skylake (client) - Microarchitectures - Intel - WikiChip*. en. 2015. URL: [https://en.wikichip.org/wiki/intel/microarchitectures/skylake_\(client\)](https://en.wikichip.org/wiki/intel/microarchitectures/skylake_(client)) (visited on 2022-09-01).

- [165] Linux Kernel Developers. *PerfWiki*. 2022. URL: https://perf.wiki.kernel.org/index.php/Main_Page (visited on 2022-08-24).
- [166] *Performance Application Programming Interface*. 2022. URL: <https://icl.utk.edu/papi/software/> (visited on 2022-08-24).
- [167] Intel® 64 and IA-32 Architectures Software Developer’s Manual, Volume 3B: *System Programming Guide, Part 2*. Developer’s manual Volume 3B, Part 2, p. 582. (Visited on 2021-07-05).
- [168] John McCalpin. *John McCalpin’s blog “Blog Archive” Notes on the mystery of hardware cache performance counters*. URL: <https://sites.utexas.edu/jdm4372/2013/07/14/notes-on-the-mystery-of-hardware-cache-performance-counters/> (visited on 2021-07-05).
- [169] Fabrice Bellard. “QEMU, a Fast and Portable Dynamic Translator”. en. In: (2005), p. 6.
- [170] Nathan Binkert et al. “The gem5 simulator”. en. In: *ACM SIGARCH Computer Architecture News* 39.2 (2011-05), pp. 1–7. ISSN: 0163-5964. DOI: [10.1145/2024716.2024718](https://doi.org/10.1145/2024716.2024718). URL: <https://dl.acm.org/doi/10.1145/2024716.2024718> (visited on 2022-05-02).
- [171] Anastasiia Butko et al. “Accuracy evaluation of GEM5 simulator system”. In: *7th International Workshop on Reconfigurable and Communication-Centric Systems-on-Chip (ReCoSoC)*. 7th International Workshop on Reconfigurable and Communication-Centric Systems-on-Chip (ReCoSoC). 2012-07, pp. 1–7. DOI: [10.1109/ReCoSoC.2012.6322869](https://doi.org/10.1109/ReCoSoC.2012.6322869).
- [172] Daniel Sanchez and Christos Kozyrakis. “ZSim: fast and accurate microarchitectural simulation of thousand-core systems”. en. In: *Proceedings of the 40th Annual International Symposium on Computer Architecture*. Tel-Aviv Israel: ACM, 2013-06, pp. 475–486. ISBN: 978-1-4503-2079-5. DOI: [10.1145/2485922.2485963](https://doi.org/10.1145/2485922.2485963). URL: <https://dl.acm.org/doi/10.1145/2485922.2485963> (visited on 2022-05-02).
- [173] Trevor E. Carlson, Wim Heirman, and Lieven Eeckhout. “Sniper: exploring the level of abstraction for scalable and accurate parallel multi-core simulation”. en. In: *Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis on - SC ’11*. Seattle, Washington: ACM Press, 2011, p. 1. ISBN: 978-1-4503-0771-0. DOI: [10.1145/2063384.2063454](https://doi.org/10.1145/2063384.2063454). URL: <http://dl.acm.org/citation.cfm?doid=2063384.2063454> (visited on 2022-05-02).
- [174] C. Lattner and V. Adve. “LLVM: A compilation framework for lifelong program analysis & transformation”. en. In: *International Symposium on Code Generation and Optimization, 2004. CGO 2004*. San Jose, CA, USA: IEEE, 2004, pp. 75–86. ISBN: 978-0-7695-2102-2. DOI: [10.1109/CGO.2004.1281665](https://doi.org/10.1109/CGO.2004.1281665). URL: <http://ieeexplore.ieee.org/document/1281665/> (visited on 2022-05-02).

- [175] M. Kooli et al. “Software Platform Dedicated for In-Memory Computing Circuit Evaluation”. In: *2017 International Symposium on Rapid System Prototyping (RSP)*. 2017-10, pp. 43–49. URL: <https://ieeexplore.ieee.org/document/8547806>.
- [176] Chi-Keung Luk et al. “Pin: building customized program analysis tools with dynamic instrumentation”. In: *ACM SIGPLAN Notices* 40.6 (2005-06), pp. 190–200. ISSN: 0362-1340. DOI: [10.1145/1064978.1065034](https://doi.org/10.1145/1064978.1065034). URL: <https://doi.org/10.1145/1064978.1065034> (visited on 2020-03-10).
- [177] Intel. *Pin – A Dynamic Binary Instrumentation Tool*. URL: <https://www.intel.com/content/www/us/en/developer/articles/tool/pin-a-dynamic-binary-instrumentation-tool.html> (visited on 2022-04-25).
- [178] Xiangyu Dong et al. “NVSim: A Circuit-Level Performance, Energy, and Area Model for Emerging Nonvolatile Memory”. In: *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* 31.7 (2012-07), pp. 994–1007. ISSN: 1937-4151. DOI: [10.1109/TCAD.2012.2185930](https://doi.org/10.1109/TCAD.2012.2185930).
- [179] S.J.E. Wilton and N.P. Jouppi. “CACTI: an enhanced cache access and cycle time model”. In: *IEEE Journal of Solid-State Circuits* 31.5 (1996-05), pp. 677–688. ISSN: 1558-173X. DOI: [10.1109/4.509850](https://doi.org/10.1109/4.509850).
- [180] ITRS. *ITRS Models and Papers*. 2015. URL: <http://www.itrs2.net/itrs-models-and-papers.html> (visited on 2022-09-03).
- [181] Youngdon Choi et al. “A 20nm 1.8V 8Gb PRAM with 40MB/s program bandwidth”. In: *2012 IEEE International Solid-State Circuits Conference*. 2012-02, pp. 46–48. DOI: [10.1109/ISSCC.2012.6176872](https://doi.org/10.1109/ISSCC.2012.6176872).
- [182] R. Gauchi et al. “Memory Sizing of a Scalable SRAM In-Memory Computing Tile Based Architecture”. In: *2019 IFIP/IEEE 27th International Conference on Very Large Scale Integration (VLSI-SoC)*. 2019-10, pp. 166–171. DOI: [10.1109/VLSI-SoC.2019.8920373](https://doi.org/10.1109/VLSI-SoC.2019.8920373).
- [183] Jinsong Ji, Chao Wang, and Xuehai Zhou. “System-Level Early Power Estimation for Memory Subsystem in Embedded Systems”. In: 2008-11-08, pp. 370–375. DOI: [10.1109/SEC.2008.48](https://doi.org/10.1109/SEC.2008.48).
- [184] Yoongu Kim, Weikun Yang, and Onur Mutlu. “Ramulator: A Fast and Extensible DRAM Simulator”. In: *IEEE Computer Architecture Letters* 15.1 (2016-01), pp. 45–49. ISSN: 1556-6064. DOI: [10.1109/LCA.2015.2414456](https://doi.org/10.1109/LCA.2015.2414456).
- [185] *Ramulator: A DRAM Simulator*. 2022-03-01. URL: <https://github.com/CMU-SAFARI/ramulator> (visited on 2022-03-04).
- [186] *VAMPIRE*. 2022-02-11. URL: <https://github.com/CMU-SAFARI/VAMPIRE> (visited on 2022-03-04).

- [187] Saugata Ghose et al. “What Your DRAM Power Models Are Not Telling You: Lessons from a Detailed Experimental Study”. en. In: *Proceedings of the ACM on Measurement and Analysis of Computing Systems* 2.3 (2018-12), pp. 1–41. ISSN: 2476-1249, 2476-1249. DOI: [10.1145/3224419](https://doi.org/10.1145/3224419). URL: <https://dl.acm.org/doi/10.1145/3224419> (visited on 2020-09-08).
- [188] Karthik Chandrasekar et al. *DRAMPower*. URL: <http://www.drampower.info> (visited on 2022-03-04).
- [189] *DRAM Power Model (DRAMPower)*. 2022-02-11. URL: <https://github.com/tukl-msd/DRAMPower> (visited on 2022-03-04).
- [190] Matt Poremba and Yuan Xie. “NVMain: An Architectural-Level Main Memory Simulator for Emerging Non-volatile Memories”. In: *2012 IEEE Computer Society Annual Symposium on VLSI*. 2012 IEEE Computer Society Annual Symposium on VLSI. 2012-08, pp. 392–397. DOI: [10.1109/ISVLSI.2012.82](https://doi.org/10.1109/ISVLSI.2012.82).
- [191] Matthew Poremba, Tao Zhang, and Yuan Xie. “NVMain 2.0: A User-Friendly Memory Simulator to Model (Non-)Volatile Memory Systems”. In: *IEEE Computer Architecture Letters* 14.2 (2015-07), pp. 140–143. ISSN: 1556-6064. DOI: [10.1109/LCA.2015.2402435](https://doi.org/10.1109/LCA.2015.2402435).
- [192] *umd-memsys/DRAMSim2*. 2022-03-04. URL: <https://github.com/umd-memsys/DRAMSim2> (visited on 2022-03-04).
- [193] Matthias Jung, Christian Weis, and Norbert Wehn. “DRAMSys: A Flexible DRAM Subsystem Design Space Exploration Framework”. In: *IPSI Transactions on System LSI Design Methodology* 8 (2015), pp. 63–74. DOI: [10.2197/ipsjtsldm.8.63](https://doi.org/10.2197/ipsjtsldm.8.63).
- [194] *DRAMSys - Fraunhofer IESE*. Fraunhofer Institute for Experimental Software Engineering IESE. URL: https://www.iese.fraunhofer.de/en/innovation_trends/autonomous-systems/memtonomy/DRAMSys.html (visited on 2022-03-12).
- [195] *tukl-msd/DRAMSys*. 2022-03-08. URL: <https://github.com/tukl-msd/DRAMSys> (visited on 2022-03-12).
- [196] *SEAL-UCSB/NVmain*. 2022-01-31. URL: <https://github.com/SEAL-UCSB/NVmain> (visited on 2022-03-04).
- [197] Valentin Egloff et al. “Storage Class Memory with Computing Row Buffer: A Design Space Exploration”. In: *2021 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. 2021-02, pp. 1–6. DOI: [10.23919/DATE51398.2021.9473992](https://doi.org/10.23919/DATE51398.2021.9473992).
- [198] P Rosenfeld, E Cooper-Balis, and B Jacob. “DRAMSim2: A Cycle Accurate Memory System Simulator”. In: *IEEE Computer Architecture Letters* 10.1 (2011-01), pp. 16–19. ISSN: 1556-6056. DOI: [10.1109/L-CA.2011.4](https://doi.org/10.1109/L-CA.2011.4). URL: <http://ieeexplore.ieee.org/document/5732229/> (visited on 2022-03-04).

- [199] Lukas Steiner et al. “DRAMSys4.0: A Fast and Cycle-Accurate SystemC/TLM-Based DRAM Simulator”. In: *Embedded Computer Systems: Architectures, Modeling, and Simulation*. Ed. by Alex Orailoglu, Matthias Jung, and Marc Reichenbach. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2020, pp. 110–126. ISBN: 978-3-030-60939-9. DOI: [10.1007/978-3-030-60939-9_8](https://doi.org/10.1007/978-3-030-60939-9_8).

Appendices

Contents