

Beyond the Roofline: Cache-aware Power and Energy-Efficiency Modeling for Multi-cores

Aleksandar Ilic, *Member, IEEE*, Frederico Pratas, *Member, IEEE*,
and Leonel Sousa, *Senior Member, IEEE*

Abstract—To foster the energy-efficiency in current and future multi-core processors, the benefits and trade-offs of a large set of optimization solutions must be evaluated. For this purpose, it is often crucial to consider how key micro-architecture aspects, such as accessing different memory levels and functional units, affect the attainable power and energy consumption. To ease this process, we propose a set of insightful cache-aware models to characterize the upper-bounds for power, energy and energy-efficiency of modern multi-cores in three different domains of the processor chip: cores, uncore and package. The practical importance of the proposed models is illustrated when optimizing matrix multiplication and deriving a set of power envelopes and energy-efficiency ranges of the micro-architecture for different operating frequencies. The proposed models are experimentally validated on a computing platform with a quad-core Intel 3770K processor by using hardware counters, on-chip power monitoring facilities and assembly micro-benchmarks.

Index Terms—Multicore architectures, Modeling and simulation, Performance evaluation and measurement.

1 INTRODUCTION

ON the road to pursuing highly energy-efficient execution, current and future trends in computer architecture move towards complex and heterogeneous designs by incorporating specialized functional units and cores [1]. In this process, it is often crucial to consider the characteristics/demands of different applications and the capability of relevant micro-architecture components to satisfy these demands. In such a broad design space, evaluating benefits and trade-offs for a set of different optimization goals and approaches is a hard and time consuming task.

Although cycle-accurate simulators can precisely model the functionality of architectures and applications, these environments are rather complex, hard to use and difficult to develop [2]. In contrast, to perform a rapid analysis of different design solutions, insightful modeling is of great practical importance for computer architects and application designers, especially in early design stages and for fast prototyping. This type of modeling is usually focused on describing the vital functional aspects of the architecture via intuitive graphical representation, such as our recently proposed Cache-Aware Roofline Model (CARM) [3].

CARM is a novel insightful *Roofline Modeling* concept that allows a description of the *performance* upper bounds for multi-core architectures with complex memory hierarchy [3]. Later works, such as [4]–[7], also corroborate the advantages of incorporating multiple memory levels into a single performance model. However, to this date, there are no insightful power/energy consumption and/or energy-efficiency models based on the CARM principles. The work proposed in this paper aims at closing this gap.

The key contribution of this paper is a set of insightful cache-aware models to characterize the upper-bounds for power, energy and energy-efficiency of modern multi-core architectures. The proposed *Power CARMs* explicitly consider how the accesses to different memory levels and utilization of specific functional units impact the power consumption in different domains of the processor chip – cores, off-core components (uncore) and the overall chip. This paper also proposes the *Total Power CARM* that defines the complete power envelope of a multi-core processor at a single frequency level. By coupling two fundamental CARMs (i.e., performance [3] and the proposed *Power CARMs*) different models can be derived to express the architecture efficiency limits. Due to their practical importance, we focus herein on two pivotal models, namely the *Energy* and *Energy-efficiency CARMs*, using which the maximum energy-efficiency of the micro-architecture is formalized.

The proposed models are experimentally validated on a computing platform with a quad-core Intel 3770K processor by using hardware counters, on-chip power monitoring facilities and assembly micro-benchmarks, while their usefulness is illustrated when optimizing matrix multiplication. The proposed models are also used to analyze the Dynamic Voltage and Frequency Scaling (DVFS) benefits from the application and micro-architecture perspectives, where different power envelopes and energy-efficiency ranges are derived for a range of supported operational frequencies.

2 BACKGROUND: ROOFLINE MODELING

The *Roofline Modeling* represents an insightful approach for describing the attainable performance upper bounds of the micro-architecture (e.g., F_a in $flops/s$). It is based on the observation that the execution time T can be limited either by the time to perform computations (T_ϕ) or memory operations (T_β), i.e., it assumes a perfect overlap of these operations in time ($T = \max\{T_\phi, T_\beta\}$). Hence, in the

- Aleksandar Ilic and Leonel Sousa are with INESC-ID, IST, Universidade de Lisboa, Lisbon, Portugal
E-mail: {Aleksandar.Ilic, Leonel.Sousa}@inesc-id.pt
- Frederico Pratas is with Imagination Technologies Limited.
E-mail: Frederico.Pratas@imgtec.com

Manuscript received August 11, 2015; revised March 20, 2016; May 20, 2016.

	Original Roofline Model (ORM)	Cache-aware Roofline Model (CARM)
INTENSITY (x-axis)	Operational (e.g., <i>flops/DRAMbytes</i>)	Arithmetic (<i>flops/bytes</i>)
MODEL TYPE	Multi-plot (one per memory level)	Single-plot (one for all memory levels)
MEMORY SCOPE	Between two memory levels	Complete memory hierarchy
MODEL CONSTRUCTION	Memory bound region	
	ORM	Data-sheets
	CARM	Micro-benchmarking
	Compute bound region	
EXPERIMENTAL VALIDATION	ORM	Possible
	CARM	Possible (achievable for all models)
EXISTING MODELS	ORM	Williams, et.al. [8]
	CARM	Choi, et.al. [10]

Fig. 1: CARM and ORM: Differences and modeling domains.

micro-architecture, the execution can be limited either by the processor computation capabilities (e.g., peak Floating Point (FP) performance, F_p in *flops/s*) or by the memory subsystem capabilities (i.e., the memory bandwidth, B).

To date, there are two main approaches for performance Roofline modeling, the Original Roofline Model (ORM) [8] and our recently proposed CARM [3]. Although both approaches model F_a , the fundamental differences lie in the way how memory traffic is considered and how intensity is defined, i.e., the x-axis in the plots. The ORM considers data traffic between two subsequent memory levels, thus the x-axis refers to *Operational Intensity* (OI), e.g., *flops/DRAMbytes*, where *DRAMbytes* refers to amount of data traffic between the Last Level Cache (LLC) and DRAM [8]. In contrast, CARM considers *Arithmetic Intensity* (AI) on the x-axis, i.e., *flops/bytes*, where *bytes* reflect data traffic at the memory ports of the processor pipeline (i.e., as seen by the cores) [3]. This fundamental difference has direct repercussions in how the two models are constructed, experimentally validated, and used for application characterization and optimization, as summarized in Fig. 1.

Model construction: The memory hierarchy of modern multi-cores consists of a set of private and shared caches memory levels, as depicted in Fig. 2 for an Intel 3770K processor. Hence, for this micro-architecture, it is required to construct 4 different ORM instances (one for each memory level). Each instance is built by considering the peak theoretical bandwidth (B_x) of the selected memory level and the OI that reflects the data transfers at that level (OI_x), where $x \in \{DRAM(D) \rightarrow L3, L3 \rightarrow L2, \dots\}$. As a result, the attainable performance in ORM is expressed as $F_a(OI_x) = \min \{B_x \cdot OI_x, F_p\}$ [8]. All ORM instances can be constructed from processor specifications (data-sheets). For the Intel 3770K¹, Fig. 3-D presents the most commonly used ORM DRAM instance [8].

In contrast to ORM, the attainable performance in CARM is modeled as $F_{a,y}(AI) = \min \{B_y \cdot AI, F_p\}$, where $y \in \{D \rightarrow C, L3 \rightarrow C, \dots, L1 \rightarrow C\}$ [3]. Since AI refers to the memory traffic as seen by the cores (C), the CARM includes all memory levels in a *single plot*, as depicted in Fig. 3-C for the Intel 3770K. In CARM, B_y reflects the realistically attainable bandwidth from a certain memory level to the core, which is lower than the corresponding peak bandwidth (B_x), since it includes the time to traverse all higher memory levels (see Fig. 2). Hence, CARM observes the data traffic, FP opera-

EXPERIMENTAL PLATFORM Intel 3770K Ivy Bridge (4 Cores @ 3.5GHz)				AVX Double Precision (DP) Floating Point (FP) Arithmetic	
Peak Memory Bandwidth (GB/s): AVX Load+Store				Peak AVX Performance F_p	
Private: 1 Core		Shared: 4 Cores		4 Cores	
L1	L2	L3	DRAM	CARM	ORM
32kB, bus: 384b	256kB, bus: 256b	8MB, bus: 256b	2ch, 2x933MHz	28 Gflops/s (1 AVX MUL-ADD/cik)	112 Gflops/s (4 cores x 28)
$B_{L1 \rightarrow C}$	$B_{L2 \rightarrow C}$	$B_{L3 \rightarrow C}$	$B_{D \rightarrow C}$	Hardware Counters	
168	40.9	121.8	13.6	RAPL: PP0 and PP1	
memory bandwidth to the core [7]				TIME	
$B_{L1 \rightarrow L2}$	$B_{L2 \rightarrow L1}$	$B_{L3 \rightarrow L2}$	$B_{D \rightarrow L3}$	CPU_CLK_UNHALTED.CORE	
168	112	448	29.9	SIMD_FP_256.PACKED_DOUBLE	
ORM L1	ORM L2	ORM L3	ORM DRAM	MEM_UOPS_RETIRED.ALL_LOADS	
peak bandwidth between two memory levels				MEM_UOPS_RETIRED.ALL_STORES	
DISABLED Enhanced Intel SpeedStep Technology (TurboBoost) / Hyperthreading / Turbo Mode / BIOS SETTINGS Hardware Prefetcher / Adjacent Cache Line Prefetch / EPU Power Saving				OFFCORE_RESPONSE.DATA_IN_SOCKET.LLC_MISS.LOCAL_DRAM_0	
				x64*	

Fig. 2: Experimental platform, setup and hardware counters.

tions and time (clock cycles) from a consistent architecture point of view. Although F_p can be derived from data-sheets, the B_y values are usually experimentally determined [3].

Model interpretation and experimental validation: The ORM and CARM are similarly interpreted. In Fig. 3-C/D, the slanted lines mark the memory-bound regions, while the F_p limits the compute-bound region (horizontal line). The ridge point (where the slanted and horizontal lines intersect) is the minimum intensity to reach maximum performance.

The experimental validation of both ORM and CARM is performed with (micro-)benchmarks capable of fully exercising the functional units and memory subsystem. The CARM is experimentally validated for a range of different Intel micro-architectures with accuracy higher than 90% for all modeled regions [3], [9]. The impossibility of validating ORM with real benchmarks is discussed in [4], while adopting the micro-benchmark methodology from [3] and [10].

Application characterization and optimization: Roofline models are typically used to simplify detection of the main application bottlenecks. To depict the differences between CARM and ORM, Fig. 3 presents a hand-tuned dense matrix multiplication example based on a layered AVX Z-Morton ordering ($C=A \cdot B$) [11]², where 5 different optimizations are applied (see Fig. 3-B). The numbered markers represent experimental points obtained for each code version (median value from 8192 runs) using the hardware counters and experimental setup in Fig. 2.

In ORM, the cache-oblivious baseline code 1 is characterized as strictly memory-bound ($OI \approx 1$). Its performance can be potentially improved to fully exploit the theoretical DRAM bandwidth ($B_{D \rightarrow LLC}$). In CARM, code 1 has $AI \approx 0.5$, which belongs to the memory-bound regions of all memory levels except L1, where it is compute-bound. According to the plotted performance, CARM shows code 1 as DRAM-bound, while fully exploiting the achievable DRAM bandwidth ($B_{D \rightarrow C}$). The DRAM-bound nature of code 1 is confirmed with the Intel top-down method [9], [12].

Since both models hint DRAM accesses as potential bottlenecks, the B matrix is transposed in code 2 to improve the L3 data re-use. In ORM, code 2 is strictly memory-bound, while almost reaching the theoretical $B_{D \rightarrow LLC}$. This observation may have two interpretations: a) no further optimizations can be applied to improve performance; or b) switch to a different ORM instance to get more insights is required. In contrast, CARM suggests that L3 accesses should

1. To fully exploit the architecture capabilities, Double Precision (DP) Advanced Vector Extensions (AVX) instructions are considered.

2. Algorithm implementation details and all analytical derivations are presented in the online Supplemental Material.

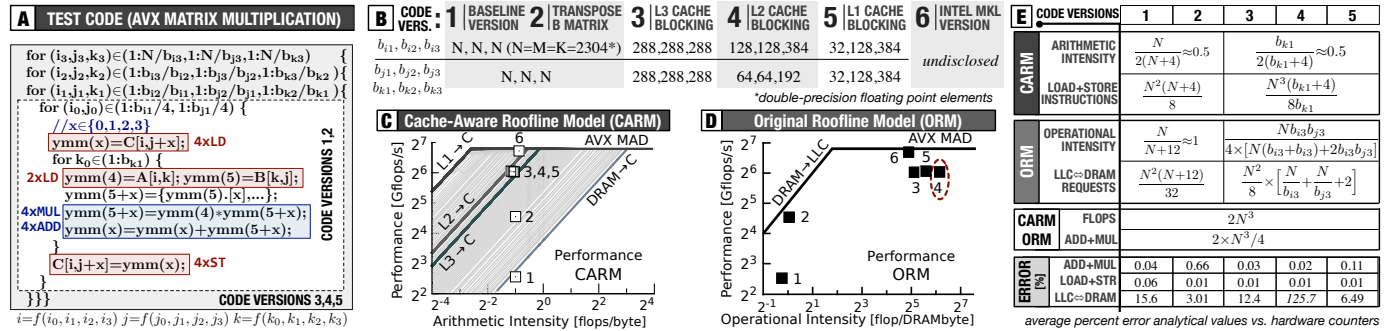


Fig. 3: Matrix multiplication optimization: Code versions, experimental and analytical characterization in CARM and ORM.

be improved to reach higher performance. By following the CARM guidelines, cache blocking for L3, L2 and L1 levels is applied in codes 3, 4 and 5, respectively. Since no reordering of FP, LD or ST instructions is performed, the AI is kept at ≈ 0.5 (see Fig. 3-A and 3-C). In CARM, these codes allowed surpassing the $L3 \rightarrow C$ and reaching the $L2 \rightarrow C$ slopes, while further improving performance by $\approx 4\times$. Finally, the Intel MKL version (code 6) reaches the initially predicted CARM L1 compute-bound region, without changing AI. However, when codes 3, 4, 5 and 6 are plotted in ORM, a significant shift in OI can be observed (from ≈ 1 to $\approx 2^6$), where the codes are characterized as strictly compute-bound.

Application behavior and intensity prediction: To understand the application behavior in the Roofline models the intensity must be analytically tractable [13]. The AI in CARM can be derived by simply counting the number of FP and memory (LD+ST) instructions. For complex algorithms, AI can be obtained with automatic code analysis tools, e.g., Intel Analyzer [14]. As shown in Fig. 3-E, the analytical AI for different code versions match the obtained empirical counter values with an average percent error of 0.09%.

By considering the traffic at a single memory level (e.g., DRAM), the OI in ORM corresponds to Kung's I/O complexity [15]. From this aspect, several works analyzed the ORM applicability for linear algebra kernels [13], [16]. These studies agree that predicting the OI (thus, understanding the application behavior in the ORM) is not a trivial task, since both the algorithm details and micro-architecture features (e.g., cache sizes, replacement policy) need to be considered. Generally, one can only derive the OI bounds [13], [16]. For the code versions in Fig. 3-A, the analytical ORM analysis is performed by estimating the number of LLC misses. The derived OI for codes 1 and 2 roughly matches the experimental values. As also referred in [13], the derived OI for codes 3, 4 and 5 depends on the block size, which explains their shift in OI towards the compute-bound region. However, the obtained average percent error is relatively high (16.4%), mainly due to the unpredictable behavior of code 4 and the difficulty in predicting all LLC replacements [13], [16].

Use-cases and model extensions: Several works rely in the ORM for application characterization and optimization [8], [17], in visualization tools [4], or for analyzing its practical applicability [13], [16]. The ORM was also applied to aid hardware/software co-design [2], [18] and for different device architectures [19]. Several studies extend the ORM usability by including additional micro-architecture

features (e.g., latency, throughput) [20], dynamic characterization [21] or performance prediction [22].

Although recent, the CARM [3] was used for optimization and characterization of real-world applications [9], [23], and for exploring architecture designs [24]. Several counter-level tools were proposed to facilitate the CARM-based analysis [25], [26]. For the CARM, the effects of prefetching, different instruction types/mixes and minimum modeling were analyzed in [7], [9]. The need for cache-awareness is also exemplified in recent works of the ORM authors [17], while adopting the CARM-based principles in [4].

Power, Energy and Efficiency: the ORM principles are applied to energy modeling in [10], [19] for processors with a two-level memory hierarchy, by assuming that energy of FP and memory operations can not be overlapped. Power consumption and energy-efficiency ORMs are derived from the energy ORM. The scope is the complete CPU package, and external power meters were used for validation.

In this paper, we propose power, energy and efficiency modeling of multi-cores with complex memory hierarchy based on the CARM principles. The proposed models inherit all differences between the CARM and ORM, thus they offer fundamentally different architecture modeling when compared to [10], [19]. The scope of this work is modeling the power consumption of different micro-architecture domains (cores, uncore and package) by specifically considering the impact of accessing different levels of the memory hierarchy. The energy and energy-efficiency CARMs are derived from the proposed power and performance CARMs.

3 POWER CONSUMPTION MODELING

To model the power consumption upper-bounds of a micro-architecture based on the CARM principles (**Power CARM**), a relation must be established between the AI and the power consumption when FP operations (*flops*) and memory operations (*mops*) are performed simultaneously. The Power CARM methodology consists of three steps: *i*) experimental analysis of the real micro-architecture (to assess the fundamental principles behind the power variation when flops and accesses to different memory levels are performed separately); *ii*) analytical derivation (based on the micro-architecture analysis); and *iii*) experimental validation.

Being a micro-architecture model, the Power CARM considers three different (internal) domains of the processor chip: *i*) **cores domain** (P_c) - the power consumed by the components involved in instruction execution (e.g.,

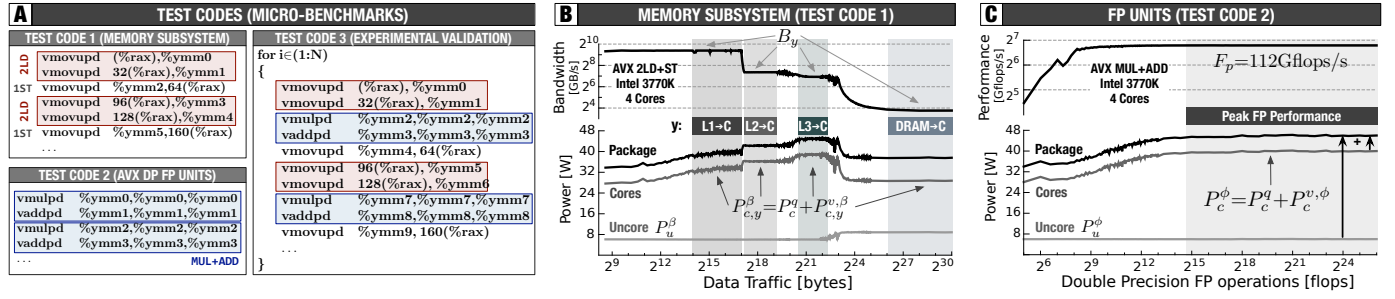


Fig. 4: Platform experimental evaluation: Micro-benchmarks and power variation with data traffic and FP operations.

pipeline, functional units and caches); *ii*) **uncore domain** (P_u) - power consumed by the other on-chip components, e.g., memory controller and interconnects; and *iii*) **package domain** (P_p) - the overall chip power consumption.

Micro-architecture experimental analysis: To evaluate the micro-architecture upper-bounds, i.e., to fully exercise the FP units and memory subsystem, two different assembly micro-benchmarks were designed (the test codes 1 and 2 in Fig. 4-A). This evaluation must be performed with architecture-specific micro-benchmarks, since real applications are not tailored for deep micro-architecture testing [3], [4]. Because the modeling scope of the Power CARM covers *on-chip components*, the power consumption can only be assessed with *internal* monitoring facilities. Due to space limitations, we present the Intel 3770K (Ivy Bridge) evaluation conducted with the counters and setup in Fig. 2, Intel RAPL [27] and precise monitoring tools [9], [25], [26]³. However, the derived conclusions and models are also valid and experimentally verified for other architectures [9].

Since the Intel 3770K pipeline contains 3 memory ports (2LD+1ST), the test code 1 is tailored to traverse the memory hierarchy by increasing the amount of 2LD+1ST AVX operations in different code runs. Each experimental point in Fig. 4-B represents the median of 8192 runs of a single code instance (fixed 2LD+1ST amount, warm-caches with counter training) [9]. The B_y maximums, for $y \in \{L1 \rightarrow C, \dots, D \rightarrow C\}$, were obtained by accessing successive memory locations (for all 4 cores in parallel). For example, in Fig. 4-B the experimental $B_{L1 \rightarrow C}$ matches the theoretical L1 bandwidth.

As presented in Fig. 4-B, B_y decreases when accessing different memory levels (from L1 to DRAM) [3]. However, while the power consumption in the *cores domain* ($P_{c,y}^{\beta}$) increases when accessing the caches (from L1 to L3), it decreases for DRAM accesses. The power consumption increase in the caches is caused by the combined activity of all lower cache levels, which prevails the decrease in data-fetch rate (B_y). When servicing DRAM requests, the very low $B_{D \rightarrow C}$ causes a power consumption drop, due to reduced activity in the caches (stalled while waiting for the data). In the *uncore domain* (P_u^{β}), the power remains constant for cache-only traffic, while it increases for DRAM accesses (due to a more intensive utilization of the on-chip memory

controller and interconnects)⁴. In the *package domain*, the power consumption ($P_{p,y}^{\beta}$) is the sum of $P_{c,y}^{\beta}$ and P_u^{β} .

In the Intel 3770K pipeline, two ports provide simultaneous access to AVX arithmetic units, which are exercised by varying the amount of MUL+ADD FP operations (test code 2 in Fig. 4-A). As shown in Fig. 4-C (shaded region), the theoretical quad-core peak FP performance ($F_p = 112 \text{ Gflops/s}$) was experimentally reached. The lower performance on the left corresponds to an insufficient number of flops to fill the execution pipeline. Thus, the power consumption in the *cores* (P_c^{ϕ}) and *package domains* (P_p^{ϕ}) initially increases, and it saturates to a constant value, when F_p is reached. In the *uncore domain* (P_u^{ϕ}), the power is constant (only arithmetic units are used) and equal to the uncore power of caches.

Power CARM (cores domain): As shown in Fig. 4, the B_y regions correspond to stable $P_{c,y}^{\beta}$ power regions, while the F_p region matches the P_c^{ϕ} . To model the performance upper-bounds (F_a) in CARM [3], it is sufficient to relate AI , B_y and F_p (i.e., $F_{a,y} = \min\{B_y \cdot AI, F_p\}$) by assuming a perfect overlap of flops and mops. However, this strategy can not be directly applied for deriving the Power CARM.

For a single memory level, the micro-architecture performance “sweet spot” lies at the ridge point, i.e., $AI_{ry} = F_p / B_y$. However, in the Power CARM, the highest power consumption is expected to occur at the ridge point. Since the mops and flops are overlapped and take the same amount of time ($T = T_{\beta} = T_{\phi}$), both operation types must have the maximum contribution to the overall power (all components actively used during T). However, this contribution of mops and flops does not directly correspond to $P_{c,y}^{\beta}$ and P_c^{ϕ} in Fig. 4 (the power when mops and flops are separately performed).

When mops and flops are executed simultaneously, they must share a portion of the pipeline (e.g., instruction cache, pipeline stages, scheduler). Hence, the $P_{c,y}^{\beta}$ and P_c^{ϕ} in Fig. 4 include the static power of the chip and the dynamic power of the shared components. This power contribution is referred herein as the constant power P_c^q . As such, $P_{c,y}^{\beta}$ and P_c^{ϕ} in Fig. 4 include P_c^q and the variable power (P_c^v) of logical blocks that are only active for a specific operation type, i.e., $P_c^{\phi} = P_c^q + P_c^{v,\phi}$ and $P_{c,y}^{\beta} = P_c^q + P_{c,y}^{v,\beta}$, where $P_c^{v,\phi}$ and $P_{c,y}^{v,\beta}$ refer to the variable power consumption of flops and mops (at the y memory level), respectively. As a result, at the ridge point of the Power CARM, i.e., $P_{c,y}(AI_{ry})$, the

3. Extensive analysis, experimental evaluation, testing methodology and validation details for 4 different Intel processors can be found in [9], including different types/mixes of FP and memory operations, prefetching, and a set of additional efficiency models.

4. As stated in [27], the DRAM power corresponds to the consumption of the on-chip components servicing the DRAM requests (e.g., memory controller), while the ring, L3 cache and snooping agent reside in the same clock and voltage domain as the cores, i.e., the *cores domain*.

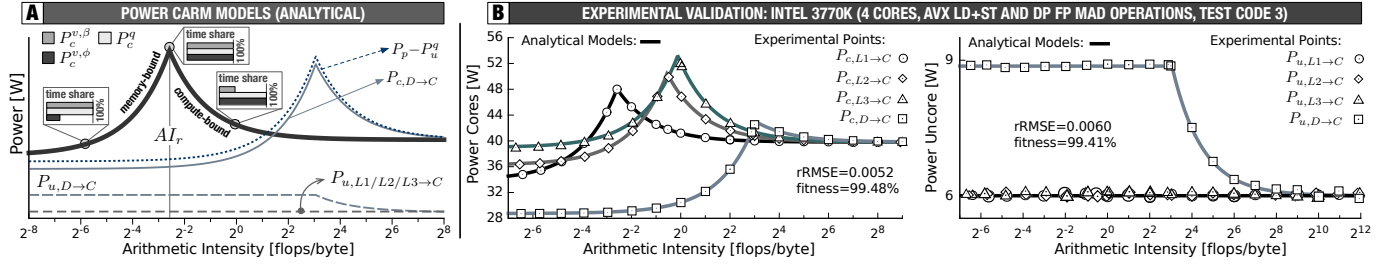


Fig. 5: Power CARM: Analytical models and experimental validation (cores and uncore domains).

(maximum) power consumption is equal to $P_c^q + P_c^{v,\beta} + P_c^{v,\phi}$.

As shown in Fig. 5-A, when the mops and flops are fully overlapped in time (delivering the best performance), their P_c^q , $P_c^{v,\beta}$ and $P_c^{v,\phi}$ power contributions are fully superposed (delivering the most undesirable power consumption). In the compute-bound region, the execution is dominated by flops ($T=T_\phi > T_\beta$), thus P_c^q and $P_c^{v,\phi}$ are consumed during the whole time T , while the contribution of $P_c^{v,\beta}$ depends on the share of T_β in T (i.e., the time period when mops-specific components are active). Conversely, in the memory-bound region, the execution is dominated by mops ($T=T_\beta > T_\phi$), thus P_c^q and $P_c^{v,\beta}$ are consumed during T , while the $P_c^{v,\phi}$ contribution depends on the share of T_ϕ in T .

For a memory level y , the Power CARM (cores domain) is defined as $P_{c,y}(AI) = P_c^q + P_c^{v,\beta} \cdot T_\beta / T + P_c^{v,\phi} \cdot T_\phi / T$, with $T = \max\{T_\beta, T_\phi\}$. By substitution, the analytical Power CARM (in the cores domain) is expressed as⁵:

$$P_{c,y}(AI) = P_c^q + P_c^{v,\beta} \min\left\{1, \frac{F_p}{B_y AI}\right\} + P_c^{v,\phi} \min\left\{1, \frac{B_y AI}{F_p}\right\} \quad (1)$$

As presented in Fig. 5, the proposed Power CARM has a hill shape, where the hill top corresponds to the ridge point. When moving away from the hill top, into the compute- or memory-bound regions, the power consumption asymptotically decreases towards the power consumption of the dominating operation type, i.e., P_c^{β} when $AI \rightarrow 0$ or P_c^{ϕ} when $AI \rightarrow +\infty$. When the complete memory hierarchy is considered, the proposed Power CARM (cores domain) is defined with several hill-shaped lines in the same plot (one for each memory level). These lines correspond to the different $P_{c,y}^{\beta}$ regions in Fig. 4-B with the hill-tops located at $AI_{ry} = F_p / B_y$, where $y \in \{L1 \rightarrow C, \dots, D \rightarrow C\}$.

Power CARM (uncore domain): As presented in Fig. 4, the uncore power only varies for DRAM requests ($P_{u,D \rightarrow C}$), while it is constant for FP operations and cache accesses (i.e., $P_u^{\phi} = P_{u,y}^{\beta} = P_u^q$, $y \neq D \rightarrow C$). Hence, the Power CARM for the uncore domain $P_{u,y}(AI)$ is defined as:

$$P_{u,y}(AI) = P_u^q + P_u^v \cdot \frac{T_D}{T(AI)} = P_u^q + P_u^v \min\left\{1, \frac{F_p}{B_{D \rightarrow C} AI}\right\} \quad (2)$$

where P_u^v is the variable power contribution of the uncore components and T_D is the time spent when servicing DRAM requests. For the DRAM accesses, the uncore CARM has the shape depicted at the bottom of Fig. 5-A, it is constant when data is accessed from the caches ($T_D = P_u^v = 0$).

5. Detailed analytical derivations for all proposed models are provided in the online Appendix.

Power CARM (package domain): The power consumption of the overall chip (package domain), $P_{p,y}(AI)$, corresponds to a superposition of $P_{c,y}(AI)$ and $P_{u,y}(AI)$, i.e., $P_{c,y}(AI) + P_{u,y}(AI)$. For the cache levels, power hills have the same shape as in the cores domain, but shifted up by P_u^q . For the DRAM accesses, in addition to the P_u^q , the power consumption in the memory-bound region also includes the P_u^v contribution (see $P_p - P_u^q$ vs. $P_{c,D \rightarrow C}$ in Fig. 5-A).

Experimental validation: Figure 5-B presents the analytical Power CARM models (solid lines) for cores and uncore domains for the quad-core Intel 3700K (Ivy Bridge)⁶. The proposed models are experimentally validated with the AVX assembly test code 3 in Fig. 4 using the hardware counters and setup in Fig. 2, Intel RAPL [27] and precise monitoring tools [9]. To experimentally reach the maximum power consumption, for different levels of the memory hierarchy and power domains, our testing methodology is based on performing thousands of assembly tests with different AI . This is achieved by controlling the mix of 2LD+1ST and MUL+ADD AVX instructions in test code 3, upon which the constant and variable power contributions are determined by applying (1). Each experimentally obtained point represents the median value across 8192 test repetitions for a fixed AI . To show the accuracy, the relative root-mean squared error (rRMSE) and curve fitness ($100/(1+rRMSE)$) are reported in Fig. 5-B. For all domains and memory levels, the experimentally obtained points match the analytical Power CARMs with an rRMSE of about 0.01 and fitness above 99%.

Total Power CARM: The Power CARM considers distinct power consumption values for different memory levels ($P_{c,y}^{\beta}$), which also results in a fixed number of clearly defined power hills (one per memory level). However, as presented in Fig. 4, the power consumption transitions between memory levels and for different number of flops are gradual and they may affect the analysis and conclusions obtained from the models. For these reasons, the **Total Power CARM** is proposed herein, which considers a full range of possible power consumption states and values by including transitional memory regions and different number of flops. As presented in Fig. 6, for the Intel 3770K, the transitional states provoke a spatial disposition of the hill shapes, thus creating additional overlapping regions. By considering all the hill tops and Total Power CARM regions, the **Total Power Roofline** is constructed, which defines the complete CARM power envelope for the micro-architecture (the topmost dark thick line in Fig. 6). The *Total Power Roofline* provides

6. Extensive experimental evaluation of the other architectures is provided in our technical report [9].

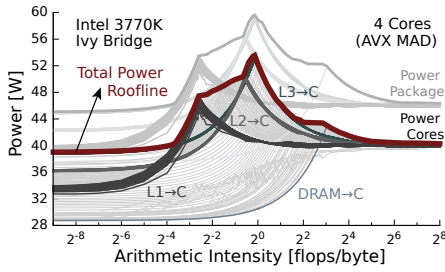


Fig. 6: Total Power CARM.

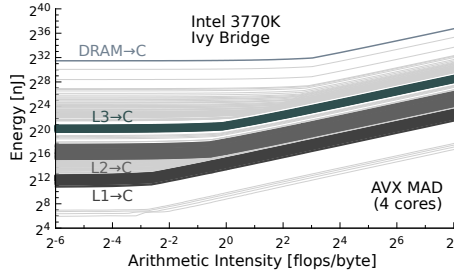


Fig. 7: Total Energy CARM.

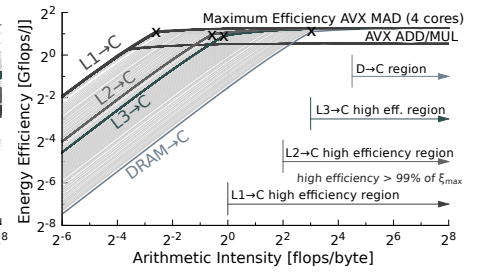


Fig. 8: Total Energy-efficiency CARM.

the insights on the power consumption upper-bounds when traversing the memory hierarchy, e.g., in Fig. 6, for the Intel 3770K, the power consumption reaches the highest values when flops are overlapped with the L3 accesses, while it is significantly lower for the other memory levels.

4 ENERGY AND ENERGY-EFFICIENCY MODELING

Based on the fundamental micro-architecture models, i.e., the proposed Performance and Power CARMs, a variety of models can be derived for different execution metrics. We focus herein on the Energy and Energy-efficiency CARMs.

Energy CARM: By relying on the Power CARM, the Energy CARM (cores domain), for a memory level y , is defined as $E_{c,y}(AI) = P_{c,y}(AI) \cdot T(AI)$, such that:

$$E_{c,y}(AI) = \phi \cdot \left[\frac{P_c^q}{\min\{B_y \cdot AI, F_p\}} + \frac{P_{c,y}^{v,\beta}}{B_y \cdot AI} + \frac{P_c^{v,\phi}}{F_p} \right], \quad (3)$$

where ϕ is the number of flops, while the addition terms represent the energy consumption relative to the constant power, and variable power of mops and flops, respectively.

Figure 7 presents the **Total Energy CARM** for the Intel 3770K. The energy consumption is approximately constant in the memory-bound region, while it increases with AI in the compute-bound region as the number of flops and its associated computation time increase, following the time domain trend in [9]. This trend is evidenced for all memory levels, with increasing energy from L1 to DRAM. In contrast, the energy ORM from [10] assumes: *i*) fixed energy per flop and mop; and *ii*) the superposition of energy consumption for flops and mops. However, according to the experimental evaluation in Fig. 4, the energy per flop and per mop can vary with the number of flops and accessed memory level, thus they can not be regarded as constant (fixed) values, while the contributions of mops and flops are only partially superposed in the *power* domain (see Section 3).

Energy-efficiency CARM: To characterize the micro-architecture efficiency limits in flops per unit of energy (J), the Energy-efficiency CARM (cores domain) is defined as $\xi_{c,y}(AI) = F_{p,y}(AI) / P_{c,y}(AI) = \phi / E_{c,y}(AI)$, and from (3):

$$\xi_{c,y}(AI) = \frac{B_y \cdot AI \cdot F_p}{P_c^q \cdot \max\{F_p, B_y \cdot AI\} + P_{c,y}^{v,\beta} \cdot F_p + P_c^{v,\phi} \cdot B_y \cdot AI}. \quad (4)$$

The model can be similarly derived for the other power domains, i.e., uncore, $\xi_{u,y}(AI)$, and package, $\xi_{p,y}(AI)$, by considering $P_{u,y}(AI)$ and $P_{p,y}(AI)$, respectively.

For the Intel 3770K, the **Total Energy-efficiency CARM** for the package domain is shown in Fig. 8, which considers the complete memory hierarchy and all transitional states. As it can be seen, accessing different memory levels and performing different FP operations (AVX MAD or ADD/MUL) results in different energy-efficiency curves. In the memory-bound region, the lowest efficiency is achieved when accessing the DRAM, while the highest efficiency occurs for the L1. As AI increases, all energy-efficiency curves converge towards the maximum efficiency of the micro-architecture.

Theoretically, the maximum energy-efficiency of the micro-architecture (ξ_{max}) can only be achieved for $AI \rightarrow \infty$, i.e., when the variable power of mops reaches 0, while maintaining the peak performance F_p . Hence, it can be analytically derived from (4) as $\xi_{max} = F_p / P_c^\phi$. In practice, one can only asymptotically approach to the maximum efficiency. For these reasons, when interpreting the proposed Energy-efficiency CARM, we claim that there are several energy-efficiency regions, where only a certain percentage of the maximum efficiency can be achieved and that these regions are a property of the micro-architecture.

In Fig. 8, for the Intel 3770K, the regions with energy-efficiency above 99% are marked for different memory levels (regions delimited with vertical lines). As it can be seen, the entry points to the high efficiency regions (AI) differ for different memory levels and increase when accessing memory levels from L1 to DRAM. Typically, the high efficiency entry points do not correspond to the performance ridge points (marked with an "X" in Fig. 8). In fact, since the ridge points correspond to the maximum power consumption, they do not guarantee achieving the maximum energy-efficiency.

5 USE CASES: APPLICATIONS AND DVFS

The proposed Power, Energy and Energy-efficiency CARMs were used for the micro-architecture analysis in Sections 3 and 4, while this section focuses on their practical usability.

Application optimization and characterization: To show the benefits of different modeling strategies, the code versions for matrix multiplication optimization from Section 2 are analyzed herein in the proposed CARMs and in the state-of-the-art power and energy-efficiency ORMs [10]. Figure 9 presents experimental results obtained for the different code versions (numbered markers, median value across 8192 runs) using the hardware counters and setup in Fig. 2, the Intel RAPL [27] and the monitoring tools from [9].

In all domains (power, energy and energy-efficiency), the proposed CARMs and the models from [10] inherit all fundamental differences between the performance CARM [3]

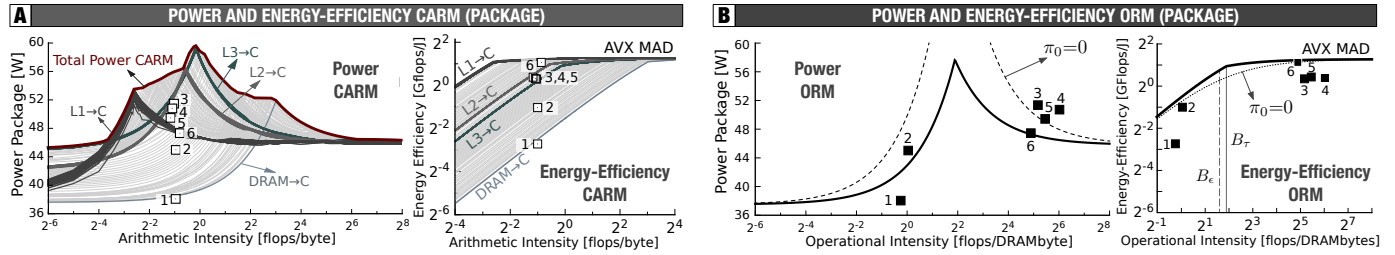


Fig. 9: Matrix multiplication code versions in Power and Energy-efficiency CARMs and ORMs (Intel 3770K, 4 Cores).

and ORM [8] stated in Section 2. Additionally, the proposed CARMs consider three different domains of the processor chip (cores, uncore, package) by relying on the on-chip monitoring facilities. The models from [10] focus on the package domain and peripheral components (e.g., cooler, off-chip interconnects) by using external power meters. Hence, the proposed CARMs and the models from [10] offer different perspectives when analyzing the micro-architecture upper-bounds, and they are also differently constructed, interpreted and used for application characterization.

By adhering to the CARM principles, each code version has the same *AI* in the Power, Energy-efficiency and Performance CARMs in Fig. 9-A and 3-C. Although codes 1 to 6 provide performance improvements (Fig. 3-C), they affect power consumption and energy-efficiency differently. Since code 1 is close to the maximum DRAM→C power in Fig. 9-A, the Power CARM unveils that any performance (cache utilization) improvement must further increase power consumption. Hence, the power increases for code 2, reaches the maximum for code 3, and then gradually decreases for codes 4 to 6. This observation corroborates the Power CARM postulates from Section 3, where fostering L3 data re-use (in codes 2 and 3) corresponds to the highest power consumption of the micro-architecture, while improving L1 and L2 accesses (in codes 4 to 6) reduces power consumption.

The Energy-efficiency CARM in Fig. 9-A helps visualizing performance vs. power consumption trade-offs. Significant efficiency improvements are achieved by surpassing the memory-bound lines with codes 1 to 5, while code 6 is close to the modeled L1 upper-bound. These observations reflect a high discrepancy between the Intel 3770K performance and power ranges (see Sections 2 and 3), where a significant performance improvement can be achieved with a relatively small power consumption increase. Although code 6 lies at the entry point of the L1 high efficiency region (see Fig. 8), its efficiency can possibly be improved by code re-structuring to deliver a higher *AI*, thus approaching to the ξ_{max} by keeping the performance with lower L1 power.

By following the ORM principles, the same codes in the Power and Energy-efficiency DRAM ORMs [10] and in the performance ORM [8] have the same and highly dispersed *OI* in Fig. 9-B and 3-D. Although code 1 lies in the Power ORM memory-bound region, codes 2 to 5 break the roofline (the maximum modeled DRAM power). As shown in Fig. 9-B, points 3 and 4 are beyond the modeling scope, since they break the absolute maximum Power DRAM ORM⁷ (the constant power $\pi_0=0$ from [10]). As referred in [10], this

behavior may be caused by the increased *energy* for cache accesses, since the Energy ORM is considered in [10] as a fundamental model (upon which the Power and Energy-efficiency ORMs are derived). In [10], the cache Energy ORM instances are built by applying a set of linear regressions on the experimental data (one per memory level, over a set of parameters obtained from the previous regressions). Hence, for the Intel 3770K, 4 linear regressions are required to fit 6 different parameters using 4 different experimental setups [10]. In contrast, for the proposed Power CARMs, only P_c^q needs to be determined, as referred in Section 3.

The code versions show a similar behavior in the Energy-efficiency [10] and performance ORMs [8] in Fig. 9-B and 3-D. By considering the DRAM traffic, the Energy-efficiency ORM provides insights for the least efficient execution domain. In [10], this model is interpreted according to: *i*) the energy-balance point (B_e) - the *OI* where flops and mops consume an equal amount of energy; and *ii*) the balance gap - the ratio between B_e and the ORM ridge point (B_r). As shown in Fig. 9-B, the Intel 3770K balance gap suggests that optimizing for performance implies energy-efficiency [10], and codes 1 and 2 are energy memory-bound, while codes 3 to 6 are compute-bound. However, optimizing for the B_e does not always imply the best energy-efficiency. This is evident in [28], where energy savings are achieved by separating (in time) mops and flops for DRAM-bound applications. According to the proposed CARMs, this energy reduction is a trade-off in the power-time domain, i.e., avoiding the power hill top by slightly increasing the execution time.

Cache-aware DVFS Roofline Modeling: The proposed CARMs can also be used for DVFS analysis, as shown in Fig. 10 for the Intel 3770K (core frequency range from 1.6 to 3.5GHz). In Fig. 10-A, the Performance CARM roofs steadily scale across frequency levels, namely: *i*) for caches, the ridge points have constant *AI*, since F_p and $B_{y \rightarrow C}$ depend on the frequency (see R_y arrows in Fig. 10-A, $y \in \{L1, L2, L3\}$); and *ii*) since the $B_{D \rightarrow C}$ is nearly constant (the DRAM operates at a fixed off-chip frequency), the *AI* of the DRAM ridge points (R_D) increases with the core frequency. Besides these effects, the power also increases with the frequency for all memory levels in the Power CARM, while in the Energy-efficiency CARM each memory level has a distinct efficiency range (mainly for the DRAM accesses, where the highest efficiency is achieved at the lowest frequency). Hence, a DRAM-bound application can become compute-bound by decreasing the frequency (see X arrows), while increasing frequency does not provide performance benefits and may reduce efficiency due to higher power (see Y arrows). From CARM perspective, the R_y lines may intersect with R_D at low frequencies,

7. In the absolute maximum Power DRAM ORM, the thermal dissipation power of the Intel 3770K chip is surpassed at the hill top.

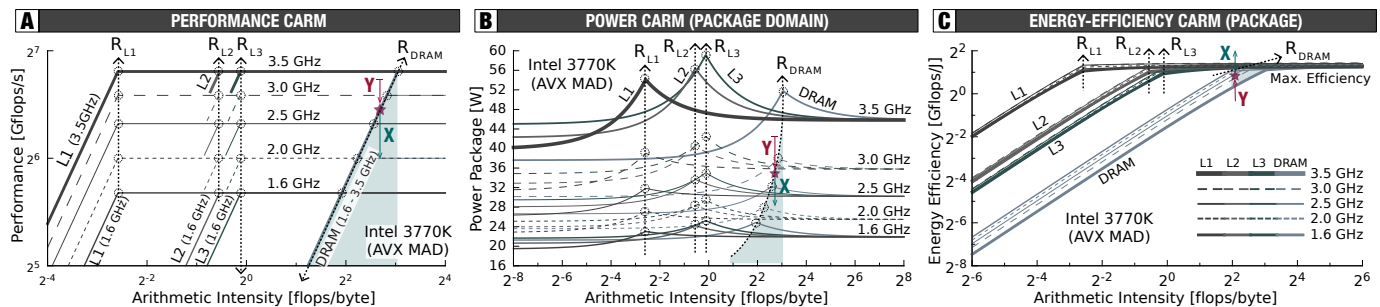


Fig. 10: DVFS CARMs for Performance, Power and Energy-efficiency (Intel 3770K, 4 Cores).

thus suggesting redundancy of the respective cache levels (e.g., at 390MHz for L3 in Fig. 10-A).

6 CONCLUSIONS

In this paper, a set of insightful cache-aware power, energy and energy-efficiency models are proposed, which are based on the CARM principles [3] and consider three domains of the processor chip (cores, uncore and package). The proposed models are constructed and experimentally validated on a platform with a quad-core Intel 3770K processor by using on-chip monitoring facilities and micro-benchmarks. To show their practical importance, the proposed models are used for matrix multiplication optimization and frequency scaling analysis, where different power envelopes and energy-efficiency ranges are derived. Since the proposed models are analytically formulated, the development of visualization tools and integration with existing models (e.g., CACTI [29] and [6]) represent future work directions.

ACKNOWLEDGMENTS

We thank Philippe Thierry (Intel France) for useful discussions. This work was supported by national funds through Fundação para a Ciência e a Tecnologia (FCT) with reference UID/CEC/50021/2013 and PTDC/EEI-ELC/3152/2012.

REFERENCES

- [1] M. B. Taylor, "Is Dark Silicon Useful? Harnessing the Four Horsemen of the Coming Dark Silicon Apocalypse," in *Proc. Design Automation Conf.* ACM, 2012, pp. 1131–1136.
- [2] J. Guo, J. Meng, Q. Yi, V. Morozov, and K. Kumar, "Analytically modeling application execution for software-hardware co-design," in *Int. Symp. Parallel Distributed Process.* IEEE, 2014, pp. 468–477.
- [3] A. Ilic, F. Pratas, and L. Sousa, "Cache-aware Roofline model: Upgrading the loft," *IEEE Comput. Archit. Lett.*, vol. 13, no. 1, pp. 21–24, Jan. 2014.
- [4] Y. J. Lo et al., "Roofline Model Toolkit: A Practical Tool for Architectural and Program Analysis," in *Int. Workshop on Perf. Modeling, Benchmarking and Simulation.* Springer, 2015, pp. 129–148.
- [5] S. K. Bhat, A. Saya, H. K. Rawat, A. Barbalace, and B. Ravindran, "Harnessing energy efficiency of heterogeneous-ISA platforms," in *Workshop on Power-Aware Comp. and Syst.* ACM, 2015, pp. 6–10.
- [6] H. Stengel et al., "Quantifying Performance Bottlenecks of Stencil Computations Using the Execution-Cache-Memory Model," in *Proc. Int. Conf. on Supercomputing.* ACM, 2015, pp. 207–216.
- [7] A. Ilic, F. Pratas, and L. Sousa, "CARM: Cache-Aware Performance, Power and Energy-Efficiency Roofline Modeling," in *Compiler, Architecture and Tools Conf.* Intel CATC, 2015.
- [8] S. Williams, A. Waterman, and D. Patterson, "Roofline: An Insightful Visual Performance Model for Multicore Architectures," *Commun. of the ACM*, vol. 52, no. 4, pp. 65–76, Apr. 2009.

- [9] A. Ilic, F. Pratas, and L. Sousa, "Insightful Cache-aware Performance, Power and Efficiency Modeling for Multi-core Architectures," INESC-ID, Tech. Rep., Feb. 2016.
- [10] J. W. Choi, D. Bedard, R. Fowler, and R. Vuduc, "A Roofline Model of Energy," in *IEEE Int. Par. Distr. Process. Symp.*, 2013, pp. 661–672.
- [11] E. Elmroth, F. Gustavson, I. Jonsson, and B. Kågström, "Recursive Blocked Algorithms and Hybrid Data Structures for Dense Matrix Library Software," *SIAM Review*, vol. 46, no. 1, pp. 3–45, 2004.
- [12] A. Yasin, "A Top-Down method for performance analysis and counters architecture," in *Proc. Int. Symp. on Performance Anal. of Syst. and Software.* IEEE, 2014, pp. 35–44.
- [13] G. Offenbeck, R. Steinmann, V. Caparros, D. Spampinato, and M. Puschel, "Applying the roofline model," in *Proc. Int. Symp. on Performance Anal. of Syst. and Software.* IEEE, 2014, pp. 76–85.
- [14] Intel, "Intel Architecture Code Analyzer," *Manual*, 2012.
- [15] H. Jia-Wei and H. T. Kung, "I/O Complexity: The Red-blue Pebble Game," in *Symp. on Theor. of Comput.* ACM, 1981, pp. 326–333.
- [16] V. Elango et al., "On using the roofline model with lower bounds on data movement," *ACM Trans. Archit. Code Optim.*, vol. 11, no. 4, pp. 67:1–67:23, Jan. 2015.
- [17] H. M. Aktulga, A. Buluç, S. Williams, and C. Yang, "Optimizing sparse matrix-multiple vectors multiplication for nuclear configuration interaction calculations," in *Proc. Int. Parallel and Distributed Process. Symp.* IEEE, 2014, pp. 1213–1222.
- [18] C. Zhang et al., "Optimizing FPGA-based accelerator design for deep convolutional neural networks," in *Proc. Int. Symp. on Field-Programmable Gate Arrays.* ACM, 2015, pp. 161–170.
- [19] J. Choi, M. Dukhan, X. Liu, and R. Vuduc, "Algorithmic time, energy, and power on candidate HPC compute building blocks," in *Int. Parallel Distributed Process. Symp.* IEEE, 2014, pp. 1–11.
- [20] V. C. Cabezas and M. Puschel, "Extending the roofline model: Bottleneck analysis with microarchitectural constraints," in *Int. Symp. on Workload Characterization.* IEEE, 2014, pp. 222–231.
- [21] Ó. G. Lorenzo, T. F. Pena, J. C. Cabaleiro, J. C. Pichel, and F. F. Rivera, "3DyRM: a dynamic roofline model including memory latency," *J. Supercomputing*, vol. 70, no. 2, pp. 696–708, 2014.
- [22] C. Nugteren and H. Corporaal, "The Boat Hull Model: Enabling Performance Prediction for Parallel Computing Prior to Code Development," in *ACM Conf. Comput. Frontiers*, 2012, pp. 203–212.
- [23] T. Ferreira et al., "Acceleration of stochastic seismic inversion in OpenCL-based heterogeneous platforms," *Comput. & Geosci.*, vol. 78, pp. 26–36, 2015.
- [24] J. Andrade, F. Pratas, G. Falcao, V. Silva, and L. Sousa, "Combining flexibility with low power: Dataflow and wide-pipeline LDPC decoding engines in the Gbit/s era," in *Proc. Int. Conf. on Application-specific Syst., Architectures and Processors.* IEEE, 2014, pp. 264–269.
- [25] D. Antão et al., "Monitoring Performance and Power for Application Characterization with the Cache-Aware Roofline Model," in *Int. Conf. Parallel Process. Appl. Math.* Springer, 2014, pp. 747–760.
- [26] L. Taniça, A. Ilic, P. Tomás, and L. Sousa, "SchedMon: A Performance and Energy Monitoring Tool for Modern Multi-cores," in *Euro-Par: Parallel Process. Workshops.* Springer, 2014, pp. 1–10.
- [27] Intel, "Intel 64 and IA-32 Architectures Software Developer's Manual: Combined Volumes 1–3," *Manual*, 2012.
- [28] K. Koukos, D. Black-Schaffer, V. Spiliopoulos, and S. Kaxiras, "Towards more efficient execution: A decoupled access-execute approach," in *Int. Conf. Supercomputing.* ACM, 2013, pp. 253–262.
- [29] N. Muralimanoohar, R. Balasubramanian, and N. P. Jouppi, "CACTI: A tool to model large caches," *HP Lab.*, pp. 22–31, 2009.