Carbon-Efficient Design Optimization for Computing Systems

Mariam Elgamal
mariamelgamal@g.harvard.edu
Harvard University
USA

Doug Carmean
carmean@meta.com
Meta
USA

Elnaz Ansari
elnazans@meta.com
Meta
USA

Okay Zed
okay@meta.com
Meta
USA

Ramesh Peri
rvperi@meta.com
Meta
USA

Srilatha Manne
bmanne@meta.com
Meta
USA

Udit Gupta
uditg@meta.com
Meta
USA

Gu-Yeon Wei
guyeon@seas.harvard.edu
Harvard University
USA

David Brooks
dbrooks@eecs.harvard.edu
Harvard University / Meta
USA

Gage Hills
ghills@seas.harvard.edu
Harvard University
USA

Carole-Jean Wu
carolejeanwu@meta.com
Meta
USA

ABSTRACT

The world's push toward an environmentally sustainable society is highly dependent on the semiconductor industry, due to carbon footprints of global-scale sources such as computing systems for virtual and extended reality applications (VR and XR). Despite previous carbon *modeling* efforts for such computing systems, there lacks a wide range of design tools to *optimize* total life cycle carbon footprint (during manufacturing and also during day-to-day operation), while meeting application-level constraints (power, performance, area). To address this need, we have developed a carbon-aware design framework that optimizes *carbon efficiency* of computing systems—quantified by metrics such as total Carbon Delay Product (tCDP: the product of total carbon and total application execution time)—while also identifying key design parameters for improving carbon efficiency. As a case study, we demonstrate the effectiveness of our framework to improve tCDP of hardware accelerators for artificial intelligence (AI) and XR applications. We show: (1) optimizing for *carbon efficiency* (tCDP) instead of *energy efficiency* (Energy-Delay Product or EDP) improves carbon efficiency by up to 6.9×—i.e., optimizing for EDP is insufficient; (2) for multi-core CPUs inside production VR headsets, optimizing number of cores (from 8 to 4) improves tCDP by 1.25× (over their entire lifetime); (3) leveraging an advanced three-dimensional integration (3D) technique (3D stacking of separately-fabricated logic and memory chips) can improve tCDP by 6.9× vs. conventional systems (no 3D stacking).

HotCarbon '23, July 9, 2023, Boston, MA, USA

ACM ISBN 979-8-4007-0242-6/23/07…$15.00
https://doi.org/10.1145/3604930.3605712

CCS CONCEPTS

• **Hardware → Impact on the environment**; **Integrated circuits**; *3D integrated circuits*; • **Computer systems organization → Architectures**.

KEYWORDS

sustainable computing systems, carbon-efficient optimization

ACM Reference Format:
Mariam Elgamal, Doug Carmean, Elnaz Ansari, Okay Zed, Ramesh Peri, Srilatha Manne, Udit Gupta, Gu-Yeon Wei, David Brooks, Gage Hills, and Carole-Jean Wu. 2023. Carbon-Efficient Design Optimization for Computing Systems. In *2nd Workshop on Sustainable Computer Systems (HotCarbon '23), July 9, 2023, Boston, MA, USA.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3604930.3605712

1 INTRODUCTION

The carbon footprint of the Information and Communication Technology (ICT) sector already accounts for 2.1-3.9% of global carbon emissions (quantified in equivalent grams of carbon dioxide: CO_{2e}) [1, 9], and is expected to grow due to our ever-increasing demand for computing. Collaborative efforts across the computing stack are essential for the ICT sector to improve its environmental sustainability (for example, to meet a target set by the International Telecommunication Union: 45% reduction in emissions by 2030 [28]).

In particular, *designers* of computing systems have opportunities to significantly improve computing's carbon footprint, with access to extensive design choices across process technologies, logic/memory devices, computer architectures, advanced 3D integration and packaging techniques, and application use cases over a system's entire lifetime. However, minimizing carbon footprint in such a massive design space—while also meeting power, performance, and area constraints—is especially challenging.

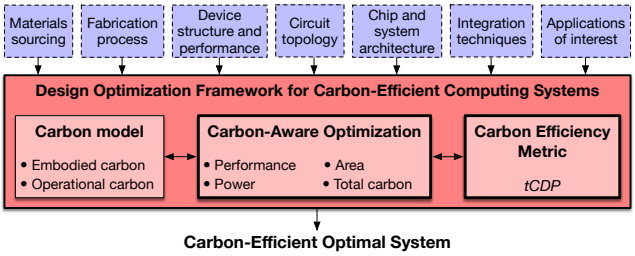

Figure 1: Summary of our framework to optimize carbon efficiency over multiple layers of the computing stack.

Key challenges are:

- **Lack of cross-stack carbon-aware design tools** – designers must account for *total life cycle carbon* (C_{total}) of computing systems, including both *operational carbon*—CO_{2e} due to energy consumption during use—and *embodied carbon*—CO_{2e} from manufacturing and production. Embodied CO_{2e}($C_{embodied}$) now accounts for 50% of cloud computing's carbon footprint and over 70% of consumer electronics carbon footprint, and thus must be considered [12]. Useful tools for quantifying $C_{embodied}$ are becoming accessible (ACT [11], GreenChip [15]), but must be integrated with existing design methodologies to enable carbon-aware optimization, i.e., trading off C_{total} vs. conventional performance metrics.
- **Imprecise quantification of carbon footprint** – due to both (a) *transparency*: designers may not have full access to detailed carbon emission numbers from manufacturing; (b) *varying energy sources*: carbon consumption of a computing task varies depending on the energy source (e.g., renewable vs. non-renewable), which also changes vs. time.
- **Nascent design guidelines for carbon-aware optimization** – design communities are still learning which parameters are most important for improving carbon efficiency, including the impact of recent trends in technology, e.g., 3D integration.

To address these outstanding challenges, we present:

(1) **A framework for carbon-efficient design across multiple layers of the computing stack.** Our framework (Figure 1) enables designers to optimize *total life cycle carbon* of computing systems—the sum of embodied carbon and operational carbon—while meeting application constraints (power, performance, area).

(2) **Techniques to optimize tCDP despite imprecise quantification of C_{total}.** Using regularization techniques [5], we formulate tCDP = $C_{embodied} \cdot D + \beta \cdot C_{operational} \cdot D$, where β is the regularization parameter. Optimizing tCDP for all non-negative values of β (0 to ∞) enables designers to make intelligent design decisions despite uncertainty in quantifying C_{total} (Section 2.1).

(3) **Developing design guidelines for carbon-efficient computing.** As a case study, we optimize tCDP for XR applications, though our framework can also be applied to other use cases. We show: (a) optimizing for *energy efficiency* (Energy-Delay Product or EDP) is insufficient for carbon-aware optimization, since it does not consider $C_{embodied}$. Instead, optimizing for tCDP improves *carbon efficiency* by up to 6.9×. (b) Total lifetime and hardware overprovisioning (adding extra computing resources, e.g., extra cores) are key parameters for carbon-efficient design; for a multi-core CPU inside production VR headsets, optimizing number of cores (from 8 to 4) improves tCDP by 1.25×. (c) 3D stacking of separately-fabricated logic and memory chips (as in [29]) improves tCDP by 1.1× to 6.9×.

2 CARBON-AWARE OPTIMIZATION FRAMEWORK

2.1 Optimizing Carbon Efficiency

In this section, we formulate the design optimization of carbon-efficient systems. We consider total life cycle carbon, as shown in equation 2.1. x is the vector of parameters defining the computing system (see Table 1), including area, process technology node, energy source, and fabrication facility specifications. Parameters a_i, q_j, and p_l are area, quality of service (QoS), and power optimization constraints, respectively.

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & (C_{\text{operational}}(x) + C_{\text{embodied}}(x)) \cdot D(x) \\ \text{subject to} \quad & \text{Area}_i(x) \le a_i, \qquad i = 1, \ldots, I \\ & \text{QoS}_j(x) \ge q_j, \qquad j = 1, \ldots, J \\ & \text{Power}_l(x) \le p_l, \qquad l = 1, \ldots, L \end{aligned} \tag{2.1}$$

Equation 2.2 shows an example of the optimization for in-production VR headsets. x is the vector of parameters specifying the VR headset, including number of CPU cores ranging from four to eight. Area constraints (for the System-on-Chip (SoC) and CPU) are based on dimensions approximated from die images [26], the QoS constraint is target frame rate to ensure quality of user-experience [14], and power constraint is 8.3 W based on the SoC's Thermal Design Power (TDP) [7]. The corresponding results are discussed in Section 3.2.

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & (C_{\text{operational}}(x) + C_{\text{embodied}}(x)) \cdot D(x) \\ \text{subject to} \quad & \text{Area}_{\text{SoC}}(x) \le 2.25 \text{ cm}^2, \\ & \text{Area}_{\text{CPU}}(x) \le 0.45 \text{ cm}^2, \\ & \text{QoS}(x) \ge 60 \text{ FPS}, \\ & \text{Power}_{\text{SoC}}(x) \le 8.3 \text{ W} \end{aligned} \tag{2.2}$$

We define a task T as a set of kernels K required to run a target application. Each task can comprise a single kernel, or multiple kernels running in parallel, depending on the number of kernel calls per task ($N_{T,K}$). For example, an XR gaming *task* can include eye-tracking, motion-tracking, and UnityEngine gaming *kernels*. A zero value of $N_{T,K}$ indicates that a kernel K is not part of task T. Equations 2.3-2.7 allow designers to optimize for multiple kernels and tasks while meeting design constraints. Note that, $\mathbf{1}$ is a vector where every element is equal to 1 (e.g., $\mathbf{1}^\intercal \mathbf{E}$ is the sum of all elements in vector $\mathbf{E}$). Refer to Table 1 for a description of each parameter.

$$\mathbf{D} = \begin{bmatrix} D_{T_1} \\ \vdots \\ D_{T_t} \end{bmatrix} = \begin{bmatrix} N_{T_1,K_1} & .. & N_{T_1,K_k} \\ \vdots & \vdots & \vdots \\ N_{T_t,k_1} & .. & N_{T_t,K_k} \end{bmatrix} \begin{bmatrix} D_{k_1} \\ \vdots \\ D_{K_k} \end{bmatrix} \tag{2.3}$$

$$C_{\text{embodied}} = \frac{\mathbf{1}^\intercal \mathbf{D}}{LT - D_{\text{idle}}} \cdot \begin{bmatrix} C_{\text{embodied},\, x_1} \\ \vdots \\ C_{\text{embodied},\, x_m} \end{bmatrix}^\intercal \begin{bmatrix} 0 \\ \vdots \\ 1 \end{bmatrix} \tag{2.4}$$

$$\mathbf{E} = \begin{bmatrix} E_{T_1} \\ \vdots \\ E_{T_t} \end{bmatrix} = \begin{bmatrix} N_{T_1,K_1} & .. & N_{T_1,K_k} \\ \vdots & \vdots & \vdots \\ N_{T_t,K_1} & .. & N_{T_t,K_k} \end{bmatrix} \begin{bmatrix} P_{\text{dyn},K_1} D_{K_1} \\ \vdots \\ P_{\text{dyn},K_k} D_{K_k} \end{bmatrix} + P_{\text{leak}} \begin{bmatrix} D_{T_1} \\ \vdots \\ D_{T_t} \end{bmatrix} \tag{2.5}$$

$$C_{\text{embodied},\, x_m} = (CI_{\text{fab}} \cdot EPA + MPA + GPA) \cdot \frac{A}{Y} \tag{2.6}$$

$$C_{\text{operational}} = CI_{\text{use}} \cdot \mathbf{1}^\intercal \mathbf{E} \tag{2.7}$$

Table 1: Parameters in our carbon-aware optimization framework.

Parameter	Description	Units	Example Range (Section 3.2)
K	Kernels or workloads software (e.g., a program written in Python or C++)	from software profiling	IOT Tracking, UnityEngine, etc.
N	Number of kernel calls per task	1 or more	multiple calls per kernel
T	Task, which could be a set of workloads or an application defined by number of kernel calls	from software profiling	M-1
D	Task execution time	seconds	40s
x	Hardware target system	from hardware design	VR headset CPU
E	Operational energy consumption	J per task	332J per task
P_{leak}	Hardware leakage power	W	-
P_{dyn}	Hardware dynamic power	W	-
P_{total}	Total power = P_{leak} + P_{dyn}	W	8.3W
CI_{use}	Use-phase carbon intensity	g CO_{2e} per kWh	380g CO_{2e}/kWh
CI_{fab}	Carbon intensity of the energy source used by fabrication facility for chip manufacturing	g CO_{2e} per kWh	820g CO_{2e}/kWh
EPA	Energy Per die Area consumed by the fabrication facility during manufacturing	kWh per cm^2	2.15 kWh/cm^2
MPA	Materials Per die Area is carbon footprint of procured materials used in fabrication	g CO_{2e} per mm^2	500g CO_{2e}/cm^2
GPA	Gases Per die Area are direct gases emitted at fabrication facility during manufacturing	g CO_{2e} per cm^2	300g CO_{2e}/cm^2
Y	Yield (e.g., estimated using yield models such as Murphy Yield [18])	0-1	0.98
A	Die area of components in system x	cm^2	2.25cm^2
$C_{operational}$	Operational carbon	g CO_{2e}	3.154g CO_{2e} per hour of use
$C_{embodied,\ x_m}$	Embodied carbon per component m (e.g. component m can be a CPU core), in system x	g CO_{2e} per component	895.89g CO_{2e} per gold core
LT	Overall hardware lifetime	1-10 years	5 years
D_{idle}	Idle time when the system is not in use throughout the system's lifetime	hours	22 hours per day for 5 years
$C_{embodied}$	Embodied carbon of a system dependent on operational lifetime = LT - D_{idle}	g CO_{2e} per system	5375.33g CO_{2e}

Task delay. Task delay (D_T) is the product of $N_{T,K}$ and kernel execution time (D_K) (equation 2.3). We compute total task delay by taking the sum of all elements in the task delay vector **D**. Designers can also compute **D** using alternative performance metrics, e.g., SPEC scores [6], or frames-per-second.

Total life cycle carbon. $C_{total} = C_{operational} + C_{embodied}$.

Embodied carbon. Embodied carbon footprint of an integrated circuit (IC) (equation 2.6) depends on the fabrication facility and on the process technology node [11, 19] (among other dependencies). To compute $C_{embodied}$, we multiply the vector $C_{embodied,\ x}$ (equation 2.4) with a vector whose elements are either 1 or 0; "1" indicates that a component is included as part of a computing system (and "0" otherwise), e.g., there is a 1/0 element for each CPU core, GPU, DRAM, on-chip SRAM, and off-chip SRAM that can potentially be included in system x. This formulation enables hardware provisioning as a design parameter (Section 3.2), where designers can compare different system configurations of their hardware to optimize tCDP. Finally, we amortize embodied carbon ($C_{embodied}$) over *execution time*, which is not necessarily the same as the system's total lifetime in years (equation 2.4). This ensures that embodied carbon is correctly accounted for when the system is not in use, and when a task is computed many times over the system's lifetime.

Operational carbon. Energy consumption depends on N_{Tk}, dynamic and leakage power, and delay (equation 2.5); each of these parameters is extracted using electronic design automation (EDA) tools. $C_{operational}$ (equation 2.7) is the total energy (sum of the components in the task energy vector **E**) multiplied by the "use-phase carbon intensity" (CI_{use}). CI_{use} is dependent on the energy source, e.g., considering renewable vs. non-renewable sources, and the electric power grid.

2.2 Optimizing tCDP Despite Uncertain C_{total}

Optimizing tCDP can be especially challenging when there is uncertainty in quantifying total carbon footprint. Practical challenges include: *transparency* (designers not having full access to carbon footprint data from manufacturing) and *varying energy sources* (with varying carbon footprint) over a system's lifetime. For example, CI_{use} (Table 1) may change dramatically from year-to-year (as renewable energy sources become more prevalent), or depending on the time of day (e.g., depending on the availability of renewable energy sources such as solar). Thus, it is essential to develop techniques for designers to improve carbon efficiency—*even without precise quantification of total carbon footprint.*

Here, we demonstrate that even when CI_{use} is unknown or changing over time, a designer can still make informed design decisions about optimizing carbon efficiency. Specifically, given a set of hardware targets: $X = \{x_1, x_2, ..., x_n\}$, we show how to identify hardware targets that are not tCDP-optimal for *any* value of CI_{use} over time. Such hardware targets can thus be *eliminated* from the set of candidate hardware targets, *even when CI_{use} is unknown.*

To justify this claim, let $CI_{use}(t)$ be the value of CI_{use} vs. time over a hardware target's operational lifetime. Similarly, let $E(t)$ be the operational energy consumption vs. time. We assume that $E(t)$ is fixed and known for a given hardware target (e.g., it can be accurately predicted using mature EDA tools); a short discussion of the benefits and limitations of this assumption is at the end of this section. Using these definitions, equation 2.7 becomes:

$$C_{operational} = \int_0^{LT} CI_{use}(t)E(t)dt. \tag{2.8}$$

Additionally, the objective function in equation 2.1 becomes:

$$C_{embodied} \cdot D + \left(\int_0^{LT} CI_{use}(t)E(t)dt\right) D. \tag{2.9}$$

However, if $CI_{use}(t)$ is unknown, then a designer can't evaluate (or optimize) this objective function. Instead, we leverage mathematical regularization techniques to eliminate designs that can't be tCDP-optimal, even when we can't find the tCDP-optimal design. Regularization is a common technique used in ill-posed optimization problems when the relative importance (weight) of two objective functions is unknown [5]. It allows us to recast the objective function from equation 2.1 as:

$$C_{embodied} \cdot D + \beta \cdot E \cdot D, \tag{2.10}$$

where E is the known total operational energy consumption over the entire lifetime, and β is the regularization parameter. We argue that there exists *some value of* β in the range $[0, \infty)$, such that

the objective functions are *exactly equal* (equations 2.1 and 2.10). Call this value β'. Both $CI_{use}(t)$ and E are always non-negative (i.e., computation always *consumes* energy, which always *increases* carbon emissions), and thus, β' must be non-negative as well.

Next, we can optimize the objective in equation 2.10 for all values of β in $[0, \infty)$, which must include β'. For each value of β, the optimization can result in a different hardware target being tCDP-optimal. We call this set of hardware targets X^*: the set of designs that are tCDP-optimal for some value of β. Note that, if we plot $E \cdot D$ vs. $C_{embodied} \cdot D$ for all designs in X, then the designs in X^* define the *Pareto-optimal curve* for $E \cdot D$ vs. $C_{embodied} \cdot D$. In other words, for all hardware targets in X^*, there is no other hardware target in X that has better $E \cdot D$ *and* better $C_{embodied} \cdot D$ *simultaneously* (see Figure 7 for an example). *Importantly, X^* contains the tCDP-optimal design* (i.e., the tCDP-optimal design is on the Pareto-optimal curve of $E \cdot D$ vs. $C_{embodied} \cdot D$). Thus, even if $CI_{use}(t)$ is unknown, designers can eliminate all hardware targets in X that are not in X^*.

As an example, Figure 7 (in Section 3) shows $E \cdot D$ vs. $C_{embodied} \cdot D$ for a set of seven different hardware targets (details in Section 3.3); only two of these seven hardware targets are on the Pareto-optimal curve (i.e., in X^*). Thus, even without knowing $CI_{use}(t)$, five of the seven hardware targets are ensured not to be tCDP-optimal. Since precise quantification of total carbon (including both operational carbon and embodied carbon) may not always be fully accessible to designers, practical design techniques (such as this) are essential to guide designers toward carbon-efficient design decisions. While we relied on various assumptions to formulate this result (e.g., our limitation that $E(t)$ is fixed and known), we encourage designers to continue developing optimization techniques that are robust to uncertainty in quantifying carbon emissions. For example, we can also leverage regularization when parameters for *embodied carbon* are unknown at design time, such as the carbon intensity of fabrication facilities (CI_{fab}), and identifying additional scenarios will aid in robust optimization of carbon efficiency.

3 OPTIMIZATION RESULTS AND ANALYSIS

3.1 The Need for Carbon-Aware Optimization

Here, we demonstrate that targeting *energy efficiency* is insufficient for carbon-aware optimization. We characterize five tasks comprising a variety of AI and XR kernels. The kernels include ResNets [13], GoogleNet [25], MobileNet-V2 [22], eye tracking [2], depth estimation [16, 24], emotion detection [27], hand tracking [17], image denoising [20, 30] and super-resolution [3], using an accelerator simulator based on a scaled-up version of Simbal et al's work [23]. We characterize five tasks: All kernels, XR (10 kernels), AI (10 kernels), XR (5 kernels), AI (5 kernels). To characterize each kernel, a neural network model from PyTorch is provided as input into the accelerator simulator, which then outputs latency and energy consumption for that kernel running on a user-specified accelerator architecture. We then use equations 2.3-2.7 to compute tCDP for a range of accelerator architectures (varying the number of Multiply-Accumulate units (MACs) and size of activation memory). Figure 2 shows that by optimizing tCDP instead of EDP (a metric of energy efficiency [10]), *carbon efficiency* improves by 2.6× to 6.9×. EDP does not consider embodied carbon, and thus is insufficient for carbon-aware optimization.

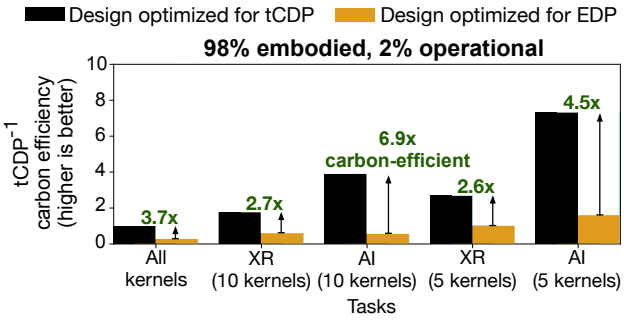

Figure 2: Optimizing for tCDP instead of EDP improves carbon efficiency up to 6.9×.

Table 2: XR accelerators latency, energy, and embodied CO_{2e}.

Accelerator	Relative latency	Relative energy	Embodied carbon
A-1	1	1	23.5 gCO_{2e}
A-2	0.7	1	67.6 gCO_{2e}
A-3	0.69	1.16	29.4 gCO_{2e}

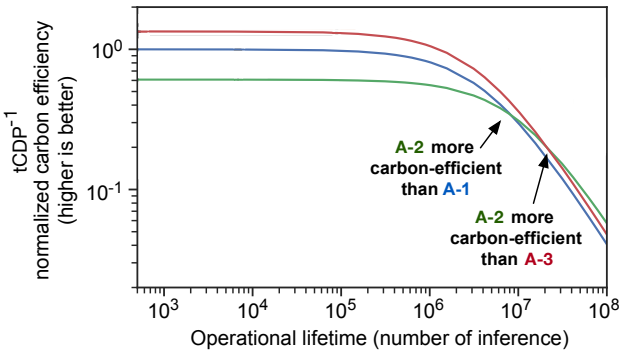

Figure 3: Carbon efficiency (y-axis) vs. operational lifetime (x-axis) to execute the "All kernels" task (Figure 2) for accelerators A-1, A-2, A-3 (Table 2). As lifetime increases, systems with better energy efficiency (lower operational carbon) become more carbon-efficient despite higher embodied carbon.

3.2 Guidelines for Carbon-Efficient Design

We present two results to guide carbon-efficient computing.

The first result highlights the importance of *operational lifetime*. We quantify tCDP of three accelerators (A-1, A-2, and A-3) for production XR systems. The accelerators have varying numbers of MAC units and on-chip SRAM capacity (for activation memory), which affect their relative performance (latency), relative energy (related to operational carbon), and embodied carbon (related to die area), as shown in Table 2. Figure 3 demonstrates that for *short lifetimes* (quantified in units of total inferences), A-3 is the most carbon-efficient accelerator (best tCDP) since it has lower embodied carbon vs. A-2, despite its higher operational carbon. As operational lifetime increases, A-2 becomes the most carbon-efficient accelerator, since *operational carbon* becomes a higher fraction of *total carbon*. Thus, for systems with long lifetimes, designers should consider techniques that decrease operational carbon at the cost of increased embodied carbon (e.g., allocating extra die area to incorporate energy-efficient accelerators into their designs).

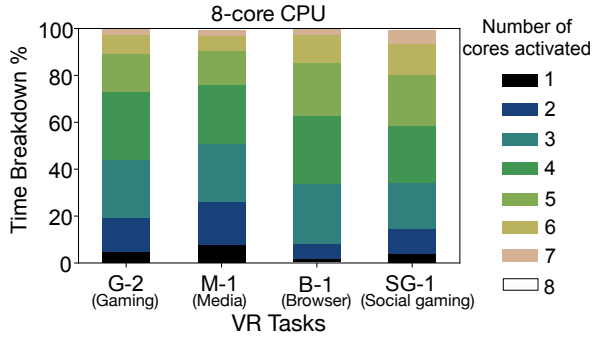

Figure 4: Time breakdown of number of cores activated per application task for in-production VR headsets. Average TLP is 3.9, indicating there are over 3 unused cores on average, highlighting opportunities to improve tCDP by hardware provisioning (optimizing number of CPU cores per task).

Table 3: VR SoC parameters before and after carbon-efficient optimization for M-1 application, with number of cores and area having high impact on carbon efficiency.

Parameter	Before Optimization	After Optimization	Improvement
P_{total}	8.3W	8.3W	-
E	332J	332J	-
A	2.25cm^2	1.35cm^2	1.67×
$C_{operational}$ CPU cores	3.15gCO_{2e} per hour 4 gold + 4 silver cores	3.15gCO_{2e} per hour 2 gold + 2 silver cores	- reduced 4 cores
$C_{embodied}$	5375.33gCO_{2e}	2687.67gCO_{2e}	2×
C_{total}	12273gCO_{2e}	9696gCO_{2e}	1.27×
D	1.0 normalized FPS	0.98 normalized FPS	0.98×
EDP	1	1.02	0.98×
tCDP	1	0.8	1.25×

The second result identifies hardware provisioning as a key design parameter for carbon-efficient computing. We analyze the top 100 tasks running on 8-core CPUs inside deployed Meta Quest 2 devices. We group the tasks into four categories: general gaming (G), social gaming (SG), browser and virtual desktop (B), and media (M). The *top 10 tasks account for over 85% of overall computation time*, and we quantify the benefits of optimal hardware provisioning for four of those top-10 tasks here (labeled G-2, M-1, B-1, and SG-1).

We quantify the degree of thread-level parallelism (TLP) to indicate the level of hardware over-provisioning on the 8-core CPU (results in Figure 4). TLP is computed as the number of cores activated concurrently divided by the total task execution time [4, 8]. For the four VR tasks, TLP ranges from 3.52 to 4.15, indicating there are over three unused cores on average. Thus, we can remove unused cores to improve total carbon (lowering die area) without significant performance degradation, improving tCDP.

Figure 5 shows that the reducing number of cores from 8 to 5 improves tCDP by 1.25× for the M-1 task (media category), with parameter values before and after optimization quantified in Table 3. Note that, B-1 (browser) and SG-1 (social gaming) tasks suffer degraded tCDP for the 4-core configuration, due to relatively higher TLP and frame-rate requirements. However, even for the "All Tasks" category, reducing cores from 8 to 5 improves tCDP by 1.08×. Thus, designers should carefully optimize hardware provisioning (based on application requirements) to develop carbon-efficient systems.

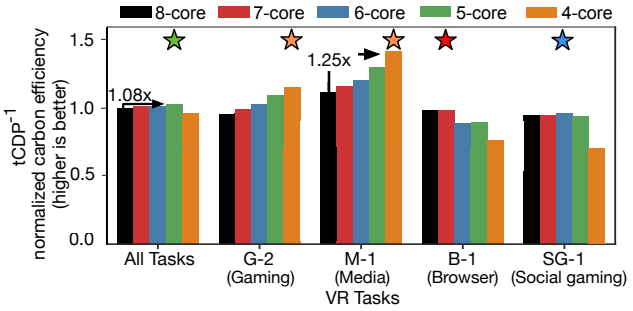

Figure 5: Carbon efficiency of VR Tasks on Meta Quest 2 vs. CPU core count (stars indicate optimal configuration). Optimal hardware provisioning improves tCDP by up to 1.25×.

For reference, equations 3.11-3.13 illustrate how we optimized hardware provisioning using the framework in Section 2. "MT" is a Motion Tracking kernel, "A" is an Audio kernel, and the "4-c" subscript indicates that these equations are used to compute tCDP for the 4-core configuration. We used similar equations to compute tCDP for 5-core, 6-core, 7-core, and 8-core configurations, which correspond to the results in Figure 5.

$$D_{M\text{-}1,\,4\text{-}c} = \begin{bmatrix} N_{M\text{-}1,MT} & .. & N_{M\text{-}1,A} \end{bmatrix} \begin{bmatrix} D_{MT,\,4\text{-}c} \\ : \\ D_{A,\,4\text{-}c} \end{bmatrix} \tag{3.11}$$

$$E_{M\text{-}1,\,4\text{-}c} = \begin{bmatrix} N_{M\text{-}1,MT} & .. & N_{M\text{-}1,A} \end{bmatrix} \begin{bmatrix} P_{dyn,MT} \cdot D_{MT,\,4\text{-}c} \\ : \\ P_{dyn,A} \cdot D_{A,\,4\text{-}c} \end{bmatrix} + P_{leak,\,4\text{-}c} \begin{bmatrix} D_{M\text{-}1,\,4\text{-}c} \end{bmatrix} \tag{3.12}$$

$$C_{embodied,\,overall} = \begin{bmatrix} C_{embodied,\,silver\ core\ 1} \\ C_{embodied,\,silver\ core\ 2} \\ C_{embodied,\,silver\ core\ 3} \\ C_{embodied,\,silver\ core\ 4} \\ C_{embodied,\,gold\ core\ 1} \\ C_{embodied,\,gold\ core\ 2} \\ C_{embodied,\,gold\ core\ 3} \\ C_{embodied,\,prime\ gold\ core} \end{bmatrix}^{\top} \begin{bmatrix} 1 \\ 1 \\ 0 \\ 0 \\ 1 \\ 0 \\ 0 \\ 1 \end{bmatrix} \tag{3.13}$$

3.3 Example tCDP Benefits of 3D Integration

State-of-the-art technologies are now using 3D integration techniques to increase connectivity between processor and memory, which can address *system-level bottlenecks* such as the "memory wall" (excessive time and energy overheads for transferring data back-and-forth between processors and memory [21]). However, the *carbon emissions* of 3D integration techniques are still in the process of becoming well-understood. Here, we evaluate the carbon efficiency of *3D stacking*: connecting separately-fabricated dice in 3D (using hybrid bumps to connect through silicon vias (TSVs) of vertically-adjacent dice, as described in [29]). Figure 6a shows the baseline A-3 accelerator memory architecture compared to a variety of 3D-stacked configurations. The 3D-stacked configurations include various combinations of computing resources (1,000 MAC units or 2,000 MAC units), and activation memory (in MegaBytes: 2 MB, 4 MB, 8 MB, or 16 MB). The activation memory *per memory die* is 2 MB for configurations with 1,000 (1K) MACs, and 4 MB for configurations with 2,000 (2K) MACs.

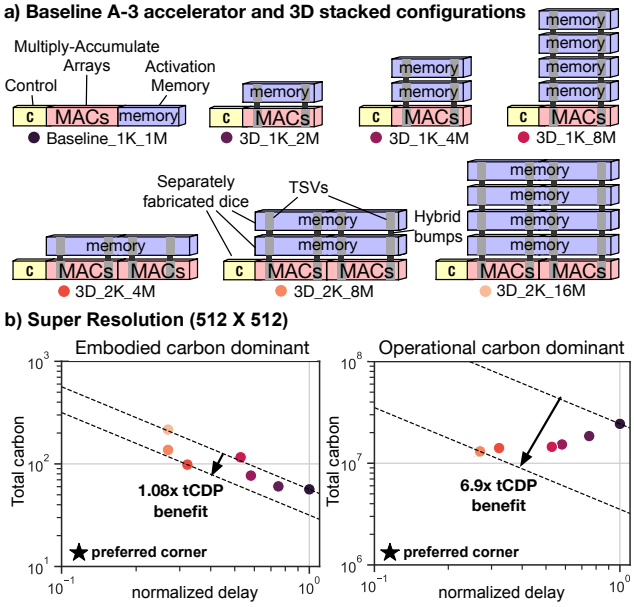

Figure 6: (a) Baseline A-3 accelerator (derived from [23]) and six 3D stacked configurations [29]. 1K/2K is number of MACs (1K = 1,000), 2M/4M/8M/16M is SRAM capacity (in MegaBytes). (b) For SR (512×512) kernel, 3D stacking improves tCDP by 1.08× to 6.9× vs. baseline (depending on operational lifetime).

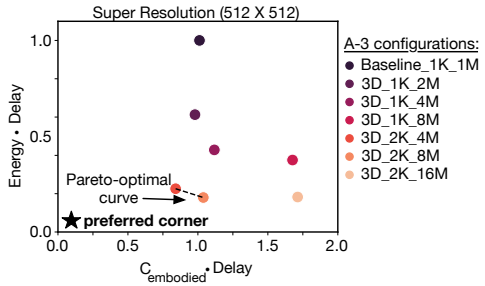

Figure 7: E · D vs. $C_{embodied}$ · D for the various A-3 accelerator configurations running SR (512×512) (same data as in Figure 6b). The Pareto-optimal curve indicates the only configurations that can be tCDP-optimal for any possible value of $CI_{use}(t)$, as described in Section 2.2.

To demonstrate the tCDP benefits of 3D stacking, we analyze the A-3 accelerator running a super-resolution kernel (SR), as an example. This SR kernel is used on XR systems to improve low-resolution image quality [3], and we use the term "inference" to refer to the SR kernel running on a single image. We quantify tCDP in two cases: for an "embodied carbon dominant" case, and an "operational carbon dominant" case. Both cases run the same task (SR 512x512), but operate for different lifetimes (quantified by the number of "inferences", as in Figure 2). The embodied carbon dominant has shorter lifetime (fewer inferences) such that the embodied carbon accounts for 80% of total carbon (average across all seven A-3 configurations), and operational carbon accounts for the remaining 20%. In contrast, the operational carbon dominant has longer lifetime, such that embodied carbon accounts for 8% of total carbon, and operational carbon accounts the remaining 92% (on average).

Figure 6b shows: for the embodied carbon dominant case (left side), configuration "3D_2K_4M" improves tCDP by 1.08× vs. the baseline, and for the operational carbon dominant case (right side), configuration "3D_2K_8M" improves tCDP by 6.9× vs. the baseline. In both cases, 3D stacking improves tCDP. When operational carbon exceeds embodied carbon, the energy efficiency benefits of 3D stacking (more computing resources, higher processor to memory bandwidth) provide significantly larger tCDP benefit. The tCDP benefit is relatively smaller for the embodied carbon case, which is dominated by the physical manufacturing of the individual compute and memory components; our analysis currently accounts for the area overhead of including TSVs on each die, and can be extended to incorporate future models that account for the embodied carbon cost of physically connecting multiple dice. Developing carbon footprint models for various bonding and packaging techniques will be required for quantifying tCDP of computing systems.

Finally, Figure 7 illustrates the relationship between E · D vs. $C_{embodied}$ · D for the same data in Figure 6b, where E corresponds to "Energy per inference" (which is the same for both the embodied carbon dominant and operational carbon dominant cases). As described in Section 2.2, any A-3 configuration that is not on the Pareto-optimal curve cannot be tCDP-optimal for any values of $CI_{use}(t)$. Thus, even when $CI_{use}(t)$ is unknown or changing over time, five of the seven configurations can be eliminated: Baseline_1K_1M, 3D_1K_2M, 3D_1K_4M, 3D_1K_8M, and 3D_2K_16M. Instead, either of 3D_2K_4M and 3D_2K_8M will be tCDP-optimal depending on $CI_{use}(t)$; indeed, 3D_2K_4M is tCDP-optimal for the embodied carbon dominant case in Figure 6b, and 3D_2K_8M is tCDP-optimal for the operational carbon dominant case. Note that, the change in *lifetime* (between the embodied carbon dominant case and the operational carbon dominant case) has the same effect as a change in $CI_{use}(t)$ *for this particular analysis*—both have the effect of changing the scaling factor from E to $C_{operational}$ (as in equation 2.7). Thus, we can leverage the regularization technique in Section 2.2. This example demonstrates the ability to make informed design decisions even when there is uncertainty in quantifying C_{total}.

CONCLUSION

Designers must consider *total life cycle carbon* when optimizing carbon-efficient computing systems. We present a carbon-aware design framework to optimize parameters at multiple layers of the computing stack (across manufacturing, design, and operational use) to improve tCDP of computing systems. We identify key parameters to improve carbon efficiency, including operational lifetime, hardware provisioning, and 3D integration, which can guide designers in developing future carbon-efficient computing systems. We also provide—and justify—regularization techniques for designing carbon-efficient systems even when quantifying total carbon is challenging. We lay the foundation for a carbon-efficient design framework that can be also extended to include additional models (e.g. yield, CI_{use}, manufacturing, and more), as the community continues to adopt carbon-aware optimization.

ACKNOWLEDGMENTS

We thank our collaborators at Meta for valuable discussions: Jordan Tse, Noah VanGorder, Lita Yang, and Edith Beigne.

REFERENCES

[1] Anders S. G. Andrae and Tomas Edler. 2015. On Global Electricity Usage of Communication Technology: Trends to 2030. *Challenges* 6, 1 (2015), 117–157. https://doi.org/10.3390/challe6010117

[2] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. 2017. SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 39, 12 (2017), 2481–2495. https://doi.org/10.1109/TPAMI.2016.2644615

[3] Goutam Bhat, Martin Danelljan, Luc Van Gool, and Radu Timofte. 2021. Deep Burst Super-Resolution. https://doi.org/10.48550/ARXIV.2101.10997

[4] Geoffrey Blake, Ronald G. Dreslinski, Trevor Mudge, and Krisztián Flautner. 2010. Evolution of Thread-Level Parallelism in Desktop Applications. In *Proceedings of the 37th Annual International Symposium on Computer Architecture* (Saint-Malo, France) *(ISCA '10)*. Association for Computing Machinery, New York, NY, USA, 302–313. https://doi.org/10.1145/1815961.1816000

[5] Stephen P Boyd and Lieven Vandenberghe. 2004. *Convex optimization*. Cambridge university press.

[6] Standard Performance Evaluation Corporation. 2023. SPEC CPU2017 Results. http://www.spec.org/cpu2017/results/index.html

[7] Mariam Elgamal, Doug Carmean, Elnaz Ansari, Okay Zed, Ramesh Peri, Srilatha Manne, Udit Gupta, Gu-Yeon Wei, David Brooks, Gage Hills, and Carole-Jean Wu. 2023. Design Space Exploration and Optimization for Carbon-Efficient Extended Reality Systems. arXiv:2305.01831 [cs.AR]

[8] Kristián Flautner, Rich Uhlig, Steve Reinhardt, and Trevor Mudge. 2000. Thread-Level Parallelism and Interactive Performance of Desktop Applications. In *Proceedings of the Ninth International Conference on Architectural Support for Programming Languages and Operating Systems* (Cambridge, Massachusetts, USA) *(ASPLOS IX)*. Association for Computing Machinery, New York, NY, USA, 129–138. https://doi.org/10.1145/378993.379233

[9] Charlotte Freitag, Mike Berners-Lee, Kelly Widdicks, Bran Knowles, Gordon Blair, and Adrian Friday. 2021. The climate impact of ICT: A review of estimates, trends and regulations. https://doi.org/10.48550/ARXIV.2102.02622

[10] R. Gonzalez and M. Horowitz. 1996. Energy dissipation in general purpose microprocessors. *IEEE Journal of Solid-State Circuits* 31, 9 (1996), 1277–1284. https://doi.org/10.1109/4.535411

[11] Udit Gupta, Mariam Elgamal, Gage Hills, Gu-Yeon Wei, Hsien-Hsin S. Lee, David Brooks, and Carole-Jean Wu. 2022. ACT: Designing Sustainable Computer Systems with an Architectural Carbon Modeling Tool. In *Proceedings of the 49th Annual International Symposium on Computer Architecture*. Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/3470496.3527408

[12] U. Gupta, Y. Kim, S. Lee, J. Tse, H. S. Lee, G. Wei, D. Brooks, and C. Wu. 2021. Chasing Carbon: The Elusive Environmental Footprint of Computing. In *2021 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*. IEEE Computer Society, Los Alamitos, CA, USA, 854–867. https://doi.org/10.1109/HPCA51647.2021.00076

[13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Deep Residual Learning for Image Recognition. *CoRR* abs/1512.03385 (2015). arXiv:1512.03385 http://arxiv.org/abs/1512.03385

[14] Muhammad Huzaifa, Rishi Desai, Samuel Grayson, Xutao Jiang, Ying Jing, Jae Lee, Fang Lu, Yihan Pang, Joseph Ravichandran, Finn Sinclair, Boyuan Tian, Hengzhi Yuan, Jeffrey Zhang, and Sarita V. Adve. 2021. Exploring Extended Reality with ILLIXR: A new Playground for Architecture Research. arXiv:2004.04643 [cs.DC]

[15] Donald Kline, Nikolas Parshook, Xiaoyu Ge, Erik Brunvand, Rami Melhem, Panos K. Chrysanthis, and Alex K. Jones. 2019. GreenChip: A tool for evaluating holistic sustainability of modern computing systems. *Sustainable Computing: Informatics and Systems* 22 (2019), 322–332. https://doi.org/10.1016/j.suscom.2017.10.001

[16] Zhaoshuo Li, Wei Ye, Dilin Wang, Francis X. Creighton, Russell H. Taylor, Ganesh Venkatesh, and Mathias Unberath. 2021. Temporally Consistent Online Depth Estimation in Dynamic Scenes. https://doi.org/10.48550/ARXIV.2111.09337

[17] Franziska Mueller, Dushyant Mehta, Oleksandr Sotnychenko, Srinath Sridhar, Dan Casas, and Christian Theobalt. 2017. Real-Time Hand Tracking under Occlusion from an Egocentric RGB-D Sensor. In *2017 IEEE International Conference on Computer Vision (ICCV)*. IEEE. https://doi.org/10.1109/iccv.2017.131

[18] B.T. Murphy. 1964. Cost-size optima of monolithic integrated circuits. *Proc. IEEE* 52, 12 (1964), 1537–1545. https://doi.org/10.1109/PROC.1964.3442

[19] L-Å Ragnarsson, M. Garcia Bardon, P. Wuytens, G. Mirabelli, D. Jang, G. Willems, A. Mallik, A. Spessot, J. Ryckaert, and B. Parvais. 2022. Environmental Impact of CMOS Logic Technologies. In *2022 6th IEEE Electron Devices Technology & Manufacturing Conference (EDTM)*. 82–84. https://doi.org/10.1109/EDTM53872.2022.9798208

[20] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. 2015. U-Net: Convolutional Networks for Biomedical Image Segmentation. https://doi.org/10.48550/ARXIV.1505.04597

[21] Mohamed M. Sabry Aly, Tony F. Wu, Andrew Bartolo, Yash H. Malviya, William Hwang, Gage Hills, Igor Markov, Mary Wootters, Max M. Shulaker, H.-S. Philip Wong, and Subhasish Mitra. 2019. The N3XT Approach to Energy-Efficient Abundant-Data Computing. *Proc. IEEE* 107, 1 (2019), 19–48. https://doi.org/10.1109/JPROC.2018.2882603

[22] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. 2018. MobileNetV2: Inverted Residuals and Linear Bottlenecks. In *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 4510–4520. https://doi.org/10.1109/CVPR.2018.00474

[23] H. Ekin Sumbul, Tony F. Wu, Yuecheng Li, Syed Shakib Sarwar, William Koven, Eli Murphy-Trotzky, Xingxing Cai, Elnaz Ansari, Daniel H. Morris, Huichu Liu, Doyun Kim, Edith Beigne, Reality Labs, and Meta. 2022. System-Level Design and Integration of a Prototype AR/VR Hardware Featuring a Custom Low-Power DNN Accelerator Chip in 7nm Technology for Codec Avatars. In *2022 IEEE Custom Integrated Circuits Conference (CICC)*. 01–08. https://doi.org/10.1109/CICC53496.2022.9772810

[24] Ke Sun, Bin Xiao, Dong Liu, and Jingdong Wang. 2019. Deep High-Resolution Representation Learning for Human Pose Estimation. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 5686–5696. https://doi.org/10.1109/CVPR.2019.00584

[25] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. 2015. Going deeper with convolutions. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 1–9. https://doi.org/10.1109/CVPR.2015.7298594

[26] TechInsights. 2018. Samsung Galaxy S9 Teardown. https://www.techinsights.com/blog/samsung-galaxy-s9-teardown

[27] Antoine Toisoul, Jean Kossaifi, Adrian Bulat, Georgios Tzimiropoulos, and Maja Pantic. 2021. Estimation of continuous valence and arousal levels from faces in naturalistic conditions. In *Nature Machine Intelligence*. https://doi.org/10.1038/s42256-020-00280-0

[28] International Telecommunication Union. 2020. Greenhouse gas emissions trajectories for the information and communication technology sector compatible with the UNFCCC Paris Agreement. https://www.itu.int/rec/T-REC-L.1470

[29] Lita Yang, Robert M. Radway, Yu-Hsin Chen, Tony F. Wu, Huichu Liu, Elnaz Ansari, Vikas Chandra, Subhasish Mitra, and Edith Beigné. 2022. Three-Dimensional Stacked Neural Network Accelerator Architectures for AR/VR Applications. *IEEE Micro* 42, 6 (2022), 116–124. https://doi.org/10.1109/MM.2022.3202254

[30] Lucas D. Young, Fitsum A. Reda, Rakesh Ranjan, Jon Morton, Jun Hu, Yazhu Ling, Xiaoyu Xiang, David Liu, and Vikas Chandra. 2022. Feature-Align Network with Knowledge Distillation for Efficient Denoising. In *2022 IEEE/CVF Winter Conference on Applications of Computer Vision Workshops (WACVW)*. 709–718. https://doi.org/10.1109/WACVW54805.2022.00078