The Architectural Sustainability Indicator

Jaime Roelandts, *Graduate Student Member, IEEE*, Ajeya Naithani, *Member, IEEE*, and Lieven Eeckhout, *Fellow, IEEE*

***Abstract*—Computing devices are responsible for a significant fraction of the world's total carbon footprint. Designing sustainable systems is a challenging endeavor because of the huge design space, the complex objective function, and the inherent data uncertainty. To make matters worse, a design that seems sustainable at first, might turn out to not be when taking rebound effects into account. In this paper, we propose the Architectural Sustainability Indicator (ASI), a novel metric to assess the sustainability of a given design and determine whether it is strongly, weakly, or unsustainable. ASI provides insight and hints for turning unsustainable and weakly sustainable design points into strongly sustainable ones that are robust against potential rebound effects. A case study illustrates how ASI steers Scalar Vector Runahead, a weakly sustainable hardware prefetching technique, into a strongly sustainable one while offering a 3.2× performance boost.**

***Index Terms*—Computer architecture, modeling, sustainable development.**

I. Introduction

SUSTAINABILITY is a major challenge for our society and generation due to its impact on global warming and the continuous quest for raw materials. Information and Communication Technology (ICT) is responsible for 2.1% to 3.9% of global warming — on par with the aviation industry — and this contribution is expected to continue to increase substantially in the near future [1]. Without a significant and continuous effort from our community to reduce computing devices' carbon footprint, the sustainability gap between ICT's actual contribution to global warming versus the Paris agreement will continue to grow [2].

Even though energy- and power-efficient computing has been a 'hot' topic for several decades now [3], only recently have computer system engineers realized that improving the environmental footprint of a computing device involves addressing both its embodied and operational footprint [4], [5]. The former refers to the upfront environmental footprint due to manufacturing, while the latter refers to the environmental footprint due to device use during its lifetime. While the embodied footprint dominates for mobile devices, the operational footprint tends to dominate for connected always-on devices [5]. Carbon models have been proposed to assess a device's carbon footprint at design time, including the bottom-up data-driven GreenChip [4] and ACT [6] models, as well as the top-down first-order FOCAL model [7].

Received 15 May 2025; accepted 27 May 2025. Date of publication 5 June 2025; date of current version 17 July 2025. This work was supported by the Research Foundation Flanders (FWO) under Grant G018722N. *(Corresponding author: Jaime Roelandts.)*

Jaime Roelandts and Lieven Eeckhout are with Ghent University, 9052 Gent, Belgium (e-mail: jaime.roelandts@ugent.be).

Ajeya Naithani is with the Eindhoven University of Technology, 5600 MB Eindhoven, The Netherlands.

Digital Object Identifier 10.1109/LCA.2025.3576891

Designing sustainable computer architectures is a major challenge because of the huge and complex design space, i.e., there are many design parameters which all impact performance, power, energy, chip area, and thus ultimately a design's environmental footprint. To make things worse, rebound effects may turn a presumably sustainable design into an unsustainable one as efficient designs tend to be used more intensively and thus perform more work, which in turn leads to an increased environmental footprint. How to guide computer architects towards sustainable design points is an open challenge.

This paper presents the *Architectural Sustainability Indicator (ASI)* which enables computer architects to assess whether a design is strongly, weakly or unsustainable compared to a reference design. Weakly sustainable means that a design is subject to a rebound effect, i.e., while a more efficient design may reduce the footprint under a fixed-work scenario, it may lead to an increased footprint under a fixed-time scenario when more work is done. Strongly sustainable means that a design is (more) robust to rebound effects, i.e., a design reduces the footprint under both the fixed-work and fixed-time scenarios. While the FOCAL model [7] introduced the notion of strongly, weakly and unsustainable design options, it did not introduce the ASI metric. ASI is a powerful metric by capturing a design's sustainability profile in a single number. Visualizing a design's ASI versus performance trade-off relative to a reference design provides insight and a hint for how to make a design more sustainable, i.e., whether one needs to focus on reducing chip area and/or power consumption versus improving performance.

We illustrate the utility of the ASI metric by assessing the sustainability of Scalar Vector Runahead (SVR) [8], a hardware runahead data prefetching technique that speculatively executes the future instruction stream in a vector-scalar manner. The default SVR design with a 16-wide vector length is weakly sustainable though: while yielding an average 3.3× performance speedup, it also leads to an ASI that is 38% less compared to the reference design. The ASI versus performance trade-off provides a hint for how to turn SVR — by itself a weakly sustainable microarchitecture technique — into a strongly sustainable one. Indeed, ASI hints towards reducing chip area and/or power consumption, which can be achieved by reducing the size of the last-level cache (LLC). In particular, we identify a strongly sustainable SVR design point (16-wide vector length with a 128 KiB LLC) with similar ASI as the reference design (with 512 KiB LLC) *even when subject to rebound effects* while still achieving an average 3.2× performance speedup. In other words, ASI provides a pathway towards architecture optimizations that improve performance and sustainability in the context of potential rebound effects.

II. Background

A. Strongly, Weakly and Unsustainable Designs

The FOCAL model [7] computes the *Normalized Carbon Footprint (NCF)* to quantify the carbon footprint of a system *X* relative to a reference system *Y*. NCF is a weighted sum of a system's normalized embodied and operational footprints. The weight α ($0 < \alpha < 1$) determines the relative importance of embodied versus operational footprint for the system, i.e., mobile devices tend to be dominated by the embodied footprint ($\alpha \simeq 0.8$) while always-on devices are mostly dominated by the operational footprint ($\alpha \simeq 0.2$) [5]. To cope with the inherent data uncertainty regarding sustainability, FOCAL uses proxies for the embodied and operational footprint when comparing two systems in the same technology node. The embodied footprint is represented by *chip area (A)*, while the operational footprint is represented by either *energy (E)* in a *fixed-work* scenario and *power (P)* in a *fixed-time* scenario. The fixed-work scenario assumes that the amount of work done is constant when comparing systems, while the fixed-time scenario assumes that the amount of time spent is constant, implying that a more efficient system performs more work. This is to foreshadow the infamous rebound effect, also known as Jevons' paradox [9].

Equations (1) and (2) report NCF for the fixed-work and fixed-time scenarios, respectively, with $A_n = \frac{A_X}{A_Y}$ normalized chip area, and P_n and E_n normalized power and energy, respectively.

$$NCF_{fw} = \alpha \times A_n + (1-\alpha) \times E_n \tag{1}$$

$$NCF_{ft} = \alpha \times A_n + (1-\alpha) \times P_n \tag{2}$$

As described in the FOCAL work [7], a system *X* is called *strongly sustainable* relative to system *Y* when it reduces the carbon footprint under both the fixed-work and fixed-time scenarios, thus $NCF_{fw} < 1$ and $NCF_{ft} < 1$. A system is classified as *weakly sustainable* if it reduces the carbon footprint under only one of the two scenarios, i.e., $NCF_{fw} < 1 < NCF_{ft}$ or $NCF_{fw} > 1 > NCF_{ft}$. A system is *unsustainable* if the carbon footprint increases under both the fixed-work and fixed-time scenarios, i.e., $NCF_{fw} > 1$ and $NCF_{ft} > 1$.

B. Scalar Vector Runahead

We employ Scalar Vector Runahead (SVR) [8] to illustrate the utility of the ASI metric proposed in this work. SVR is a low-overhead hardware data prefetching technique to improve the performance of workloads with chains of dependent memory accesses, as typically seen in graph analytics workloads among others. SVR achieves substantial performance speedups through speculative execution of prefetch requests on an energy-efficient in-order core. The key idea underpinning SVR stems from the Vector Runahead (VR) proposal [10], in which the processor core speculatively vectorizes future instructions during runahead mode in a hardware-only fashion to prefetch data when the processor core stalls on a long-latency load. Decoupled Vector Runahead (DVR) [11] solves some of the fundamental limitations of VR and further boosts performance by decoupling vectorization from the main thread into a separate microthread context, by limiting the number of inaccurate prefetches through loop bound detection and prediction, by prefetching across nested loops, and by efficiently handling branch divergence.

Techniques such as VR/DVR can only be implemented within large, typically out-of-order, processor cores with large vector execution units and register files. VR/DVR is hence not suitable for energy-constrained edge devices which typically lack vector units and registers. How to improve the performance of challenging memory-intensive workloads on edge devices which feature small, typically in-order, cores is what SVR contributes. First, it does not require any vector execution units by speculatively executing future (vectorized) instructions as scalars. Second, unlike DVR which requires a decoupled microthread context for vectorization, Scalar Vector Runahead (SVR) executes in a lockstep fashion alongside the main thread. Third, SVR does not require a discovery pass for loop bound detection and prediction, but recycles a (small) physical register file among speculatively executing scalar instructions. These features make SVR a lean hardware data prefetching technique well suited for low-power embedded processors.

SVR is a good candidate to illustrate the utility of the ASI metric. In fact, we find that SVR, as originally proposed, is a weakly sustainable technique, i.e., SVR incurs some hardware chip area overhead adding to the embodied footprint, while reducing energy consumption (due to improved performance) which in turn reduces the operational footprint under a fixed-work scenario. However, because power consumption increases, SVR is susceptible to a rebound effect up to the point that it is weakly sustainable, i.e., SVR leads to an increased carbon footprint under a fixed-time scenario. The ASI metric proposed in this work clearly demonstrates this and provides a hint to turn SVR from a weakly into a strongly sustainable solution.

III. ASI

The Architectural Sustainability Indicator (ASI) enables computer architects to assess whether a design is strongly, weakly or unsustainable using a singe metric, and more importantly provide insight and hints to improve the sustainability of a design. Recall from Section II-A that a design is strongly, weakly or unsustainable depending on whether the value of NCF_{fw} and NCF_{ft} is larger or smaller than one as denoted below in Equation 3. Using the symbols α, A_n and P_n as defined in Section II-A, and by defining T_n as normalized execution time or $T_n = T_X/T_Y$, Equations 4 through 6 rewrite the conditions, as follows:

$$\begin{cases} NCF_{fw} \lesseqgtr 1 \\ NCF_{ft} \lesseqgtr 1 \end{cases} \tag{3}$$

$$\iff \begin{cases} \alpha A_n + (1-\alpha) P_n T_n \lesseqgtr 1 \\ \alpha A_n + (1-\alpha) P_n \lesseqgtr 1 \end{cases} \tag{4}$$

$$\iff \begin{cases} (1-\alpha) P_n T_n \lesseqgtr 1 - \alpha A_n \\ (1-\alpha) P_n \lesseqgtr 1 - \alpha A_n \end{cases} \tag{5}$$

$$\iff \begin{cases} T_n \lesseqgtr \frac{1-\alpha A_n}{(1-\alpha) P_n} \\ 1 \lesseqgtr \frac{1-\alpha A_n}{(1-\alpha) P_n} \end{cases} \tag{6}$$

$$\iff \begin{cases} T_n \lesseqgtr ASI \\ 1 \lesseqgtr ASI \end{cases} \tag{7}$$

TABLE I
CLASSIFYING SUSTAINABILITY OF A SYSTEM USING ASI

Strongly sustainable	$ASI > \max(1, T_n)$
Unsustainable	$ASI < \min(1, T_n)$
Weakly sustainable	$\min(1, T_n) < ASI < \max(1, T_n)$

By defining ASI as

$$ASI = \frac{1 - \alpha A_n}{(1 - \alpha) P_n}, \tag{8}$$

we reach the final form of the conditions in (7). ASI is a *higher-is-better* metric, and measures the architectural sustainability of a system relative to another reference system. The numerator computes the normalized total carbon footprint of the reference system *Y* (equal to one) minus the weighted normalized embodied footprint of system *X*; this is divided by the denominator which quantifies the weighted normalized operational footprint of system *X*.

Whether a design is strongly, weakly or unsustainable can now be derived from analyzing the ASI value:

Strongly sustainable: A system is strongly sustainable if its normalized footprint under both the fixed-work and fixed-time scenarios is smaller than the reference system. In other words, $NCF_{ft} < 1$ and $NCF_{fw} < 1$. This implies that $1 < ASI$ and $T_n < ASI$, or $ASI > \max(1, T_n)$.

Unsustainable: A system is unsustainable if $NCF_{ft} > 1$ and $NCF_{fw} > 1$. This implies that $1 > ASI$ and $T_n > ASI$, or $ASI < \min(1, T_n)$.

Sustainable under fixed-work (S-FW): This is a system that improves sustainability under a fixed-work scenario but not under a fixed-time scenario. This can happen when a performance increase results in a decrease in energy, but at a cost of an increase in power consumption. The inequalities become $NCF_{ft} > 1$ and $NCF_{fw} < 1$. This implies that $1 > ASI$ and $T_n < ASI$, or $T_n < ASI < 1$.

Sustainable under fixed-time (S-FT): This is a system that improves sustainability under a fixed-time scenario but not under a fixed-work scenario. This can happen when the reduction in power consumption is less than the increase in speedup, which results in higher energy consumption. The inequalities become $NCF_{ft} < 1$ and $NCF_{fw} > 1$, which implies that $1 < ASI$ and $T_n > ASI$, or $T_n > ASI > 1$.

Note that the latter two categories denote *weakly sustainable* systems because they improve sustainability in either a fixed-work scenario or a fixed-time scenario, but not both. Combining both categories of weakly sustainable systems with respective conditions, $T_n < ASI < 1$ and $T_n > ASI > 1$, we obtain $\min(1, T_n) < ASI < \max(1, T_n)$. Interestingly, these boundary conditions are identical to the ones for the strongly and unsustainable designs. This leads to the overall summary as presented in Table I.

Fig. 1 illustrates the relationship between ASI (vertical axis) versus speedup ($S = 1/T_n$ along the horizontal axis). It depicts the different sustainability regions of a new system *X* versus a reference system *Y*, which is represented by the reference point $(1, 1)$. The blue line denotes the $\max(1, T_n)$ boundary condition while the green line denotes the $\min(1, T_n)$ boundary condition.

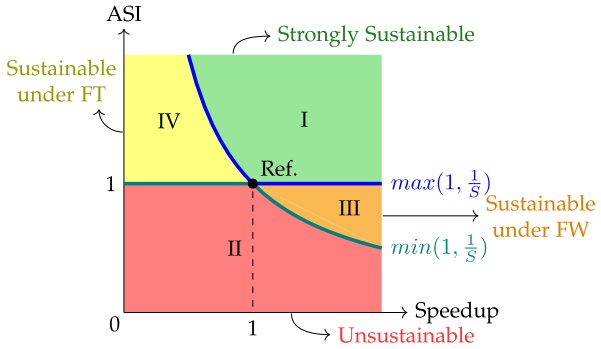

Fig. 1. This graph depicts the relationship between ASI and speedup S defined as $S = \frac{1}{T_n}$. The four regions denote whether a system is strongly, weakly, or unsustainable with respect to the reference system.

We identify four regions: (I) the top right region (green) is the strongly sustainable region, (II) the bottom left region (red) is the unsustainable region, (III) the right-most region (orange) represents weakly sustainable designs under the fixed-work scenario (S-FW), and (IV) the top-left region (yellow) denotes weakly sustainable designs under the fixed-time scenario (S-FT).

The power of this representation is that it visualizes what a computer architect should focus on to improve the sustainability of a system under design and make it strongly sustainable. If a system is situated in region I (green), the system is strongly sustainable. If the system is located in region III (orange, S-FW), we need to improve ASI which implies that we need to reduce chip area A_n or power consumption P_n, or both. If the system is located in region IV (S-FT), we need to focus on improving performance, more so than on improving ASI. Finally, if the system is located in region II (red), we need to improve both performance and ASI to become sustainable. An interesting observation of region III, and partially II, is that regardless how much performance is improved, it will never become strongly sustainable, without any improvement in ASI. This possibly goes against the intuition that better performance implies a more sustainable system. It is further worth noting that ASI is a relative metric, specific to a reference design and a set of workloads; the α value on the other hand only affects ASI, but does not affect the boundary conditions.

IV. MAKING SVR STRONGLY SUSTAINABLE

As alluded to in Section II-B, SVR improves performance while incurring a chip area and power overhead, i.e., it is clear that $T_n < 1$ while $A_n > 1$ and $P_n > 1$. This implies that speedup is larger than one, while ASI is smaller than one. In other words, depending on whether the performance improvement exceeds the power overhead, SVR may be weakly sustainable or unsustainable — it cannot be strongly sustainable. This implies that in the ASI versus speedup graph, SVR is located in region III (sustainable under fixed-work, or S-FW) or region II (unsustainable).

To quantitatively evaluate the sustainability of SVR, we consider the experimental setup previously considered in [8] which assumes a reference superscalar in-order core akin to the Arm Cortex A510 with a 512 KiB shared last-level cache (LLC). SVR's vector length is set at design time and parameterized in

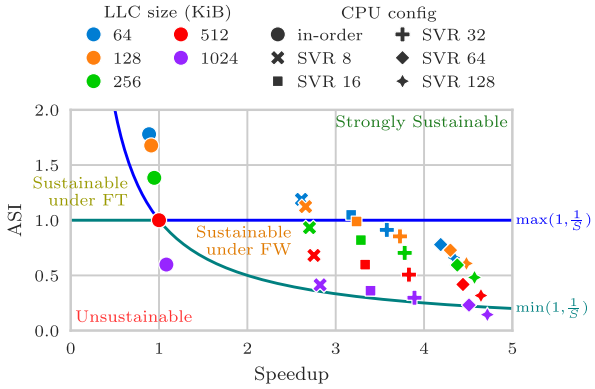

Fig. 2. Exploring ASI versus speedup while varying SVR vector length (8 through 128 in powers of 2) and LLC size (64 to 1024 KiB), assuming $\alpha = 0.8$.

our evaluation. The default vector length is set to 16 which means that, when SVR is engaged, 16 speculative (scalar) instructions are generated and speculatively executed for prefetching purposes. The benchmarks used in the evaluation, as in [8], consist of a broad mix of database and graph analytics applications. Sniper v7.3 [12] (modified to model SVR) was used for timing simulation; McPAT v1.0 [13] assuming a 22 nm technology node was used to estimate chip area and power consumption. In the analysis here we assume $\alpha = 0.8$ to reflect the use case scenario of SVR targeting energy-efficient mobile processors which are mostly dominated by their embodied footprint [5]. For calculating ASI, we first compute average speedup and power consumption across all benchmarks.

Fig. 2 reports ASI as a function of speedup ($S = 1/T_n$) for the SVR design space relative to the reference design without SVR and a 512 KiB LLC, see red circled dot with coordinates (1,1). As noted, assuming a constant LLC size of 512 KiB (red symbols), SVR ends up in region III, meaning it is weakly sustainable and only sustainable under a fixed-work scenario, irrespective of vector length. More specifically, SVR with a default vector length of 16 yields an average 3.3× speedup, however its ASI score is 38% less compared to the reference design. Being located in region III suggests that to make SVR strongly sustainable, we need to focus on improving ASI, and thus reduce chip area and/or power consumption, more so than improving performance. Indeed, reducing vector length reduces power consumption and chip area, however, this also leads to a significant, and proportionally larger, reduction in performance. In fact, as illustrated in Fig. 2, reducing vector length indeed improves ASI but it does not lead to an ASI above one, i.e., SVR remains weakly sustainable. In other words, SVR, by itself, is weakly sustainable.

The question hence is how to turn SVR from a weakly sustainable into a strongly sustainable technique? Or in the words, how to benefit from SVR's performance boost while not increasing the system's overall environmental footprint? ASI provides a hint. Indeed, being located in region III, this suggests that we need to specifically focus on reducing chip area and/or power consumption while limiting the impact on performance. One option to achieve this is to explore SVR under different cache sizes. Indeed, reducing LLC size has a substantial impact on chip area and power consumption, while incurring a relatively small performance degradation. The result is a massive improvement in ASI while incurring a small degradation in speedup. This is also illustrated in Fig. 2, which demonstrates that this strategy indeed brings SVR from the weakly sustainable region to the strongly sustainable region. We identify four architecture configurations that are strongly sustainable, i.e., the four pairs of LLC size and vector length (128 KiB, 16 scalars), (64 KiB, 16 scalars), (128 KiB, 8 scalars) and (64 KiB, 8 scalars). Among these four configurations, SVR with 16-length vectors and a 128 KiB LLC is the design point that yields the highest performance boost (3.2× average speedup) for the same ASI score as the reference design. In other words, this design point has the same environmental footprint as the reference point *even when subject to rebound effects* while achieving a significant performance boost.

V. Conclusion

The Architectural Sustainability Indicator is a novel metric to assess whether a design is strongly, weakly or unsustainable relative to a reference point. The ASI versus performance trade-off provides insight and hints for turning unsustainable and weakly sustainable design points into strongly sustainable ones. The case study illustrates how SVR, by itself a weakly sustainable technique, can be made strongly sustainable with similar ASI as the reference design while boosting performance by on average 3.2×.

References

[1] C. Freitag, M. Berners-Lee, K. Widdicks, B. Knowles, G. S. Blair, and A. Friday, "The real climate and transformative impact of ICT: A critique of estimates, trends, and regulations," *Patterns*, vol. 2, 2021, Art. no. 100340.
[2] L. Eeckhout, "The sustainability gap for computing: Quo Vadis?," *Commun. ACM*, vol. 68, pp. 68–79, Feb. 2025.
[3] M. Själander, M. Martonosi, and S. Kaxiras, *Power-Efficient Computer Architectures: Recent Advances*. Berlin, Germany: Springer Nature, 2022.
[4] D. Kline et al., "GreenChip: A tool for evaluating holistic sustainability of modern computing systems," *Sustain. Comput.: Inform. Syst.*, vol. 22, pp. 322–332, 2019.
[5] U. Gupta et al., "Chasing carbon: The elusive environmental footprint of computing," in *Proc. Int. Symp. High Perform. Comput. Architect.*, 2021, pp. 854–867.
[6] U. Gupta et al., "ACT: Designing sustainable computer systems with an architectural carbon modeling tool," in *Proc. ACM/IEEE Annu. Int. Symp. Comput. Architecture*, 2022, pp. 784–799.
[7] L. Eeckhout, "FOCAL: A first-order carbon model to assess processor sustainability," in *Proc. Int. Conf. Architect. Support Program. Lang. Operating Syst.*, 2024, pp. 401–415.
[8] J. Roelandts, A. Naithani, S. Ainsworth, T. M. Jones, and L. Eeckhout, "Scalar vector runahead," in *Proc. IEEE Int. Symp. Microarchitect.*, 2024, pp. 1367–1381.
[9] B. Alcott, "Jevons' paradox," *Ecological Econ.*, vol. 54, pp. 9–21, 2005.
[10] A. Naithani, S. Ainsworth, T. M. Jones, and L. Eeckhout, "Vector runahead," in *Proc. ACM/IEEE Annu. Int. Symp. Comput. Architect.*, 2021, pp. 195–208.
[11] A. Naithani, J. Roelandts, S. Ainsworth, T. M. Jones, and L. Eeckhout, "Decoupled vector runahead," in *Proc. IEEE Int. Symp. Microarchitect.*, 2023, pp. 17–31.
[12] T. E. Carlson, W. Heirman, S. Eyerman, I. Hur, and L. Eeckhout, "An evaluation of high-level mechanistic core models," *ACM Trans. Architect. Code Optim.*, vol. 11, pp. 1–25, 2014.
[13] S. Li, J. H. Ahn, R. D. Strong, J. B. Brockman, D. M. Tullsen, and N. P. Jouppi, "McPAT: An integrated power, area, and timing modeling framework for multicore and manycore architectures," in *Proc. IEEE Int. Symp. Microarchitect.*, 2009, pp. 469–480.