SHARE: Sustainable Heterogeneous Architectures can Reduce Emissions by Sharing Memory

Jack Toubes

a senior thesis submitted in partial fulfillment
of the requirements for the degree of
Bachelor of Science in Engineering
Department of Electrical and Computer Engineering
Princeton University

Adviser: Margaret Martonosi

July 4, 2025

SHARE: Sustainable Heterogeneous Architectures can Reduce Emissions by Sharing Memory

Jack Toubes

Though hardware accelerators decrease the operational carbon emissions of contemporary systems-on-chip (SoCs), their significant area, and thus embodied carbon cost, makes their impact on the overall lifetime sustainability of a chip less clear. As opposed to instantiating separate hardware for each accelerated application kernel, time sharing hardware resources for multiple kernels can decrease the total chip area required for acceleration, at the cost of increased energy consumption. This paper examines a spectrum of methods for this hardware sharing, in order to find the architectures which optimally trade off operational and embodied carbon emissions.

Our analysis quantifies the carbon impact of resource sharing for *both* compute and memory hardware. We show that the decreased area and power consumption which comes with highly specialized compute hardware outweighs the benefits of sharing compute hardware with a reconfigurable fabric. Importantly, however, the power and area overhead required to share memory hardware is low, and so sharing memory comes with a substantial carbon-efficiency improvement. Hence, the best option is fixed-function accelerators which keep compute separate but maximally share on-chip memory hardware. Our analysis also shows sustainable sizing options for tailoring these shared memory pools to the memory needs of an application's accelerator collection.

Acknowledgements

There aren't words elegant enough to adequately describe my gratitude to the support system which has coalesced around me in the spaces I call home—both here in Princeton and in Los Angeles. It is not lost on me that I have far more champions than adversaries. Here are some of them.

My research advisers, Professors Margaret Martonosi and Sharad Malik. Your guidance and support have been a treasure. Neither this project, nor my path in research, could have been achieved without the two of you behind me. I feel privileged to have had to opportunity to sit across the table from each of you, once a week, for an extended period of time, and perform my best impression of someone who understands the world of computing.

Professor Stefanos Kaxiras. Your guidance early on shaped the direction of this project. Your sharp yet roaming questions made this project better. They also exemplified the beauty of the phase of research which was so artfully introduced to me by Professor Martonosi as "wandering in the desert".

Leon Schuermann. I hope you feel your gamble on me a year and a half ago paid off. It's been a pleasure and a privilege working with you, even when it meant staying up until three in the morning to submit a paper we didn't think we could possibly finish when we woke up the previous day. Thank you for the opportunities you've provided me, and even more for your friendship.

Aaron Meng and Ella Rubinshtein. Thank you for showing me that there can be joy in the lab, even when the rest of the world outside is dreary. And thank you for the Grimace sweater.

Phillip Jeong. Your picture is still up on my wall. It reminds me that I have an obligation to do the things I love and to make the world a better place. It also reminds me to use my Burt's Bees Cucumber Mint.

Michael Kavounas and Max Nemoy. You're my bedrock. The comfort and ease of falling back into conversation with you after months apart is a testament to our enduring friendships.

Ian Henriques, Ernest McCarter, Kevin Phan, Alicia Barbieri, Lana Gilsic, Jimmy Tran, and all of PURC's current leadership team. The space we cultivated here for ourselves and for future engineers is truly something special. Your friendships have been a blessing.

Gabriel Marin and Masha Musthafa. You're my gotos. I don't know what I will do without the two of you around to make fun of me for listening to music. Whether you know it or not, your friendships got me through my toughest moments at Princeton.

Aux "Nous" et tous les "personnes". You all made this place feel like a home for me. Thank you for making me talk about the things in my life which I am normally hesitant to share. Late nights in the loft and north of campus will be my favorite memories from my time in Princeton.

My extended family of grandparents, aunts, uncles, cousins, and Rogers. Thank you for being the architecture of my life. Sitting down at the table like we are the family in Avalon, laughing as the words whiz past me, sharp as tacks, I feel so lucky.

Max. I love you. I miss you when I'm away. Your brilliance shines through even though you don't realize it.

Mom and Abba. You showed me how to love and how to care and all that it means to do so. To love and to care for what I do so that my work and my actions take on meaning to me. To love and to care for the world so that my work and my actions take on meaning for everyone. To love and to care for others so that we all feel that we are important and worthwhile. To love and to care for myself so that on the days when it is hard, I remember that there are days where it won't be so hard.

There have been and there will continue to be hard days, but every day I know it will be enough to move through the world with the love and the care which the two of you exude. It is enough because when life is rooted in this love and care, it is easy to work hard, it is easy to make the right decisions, and it is easy to feel fulfilled in the relationships with the people around me. I love you both so much.

There are so many more people who have meaningfully impacted my life and my work, but the music playing me off the stage started three minutes ago, and so I must end this exercise in self-indulgence. Thank you all!

Contents

List of Tables

List of Figures

Chapter 1

Introduction

Information and Computing Technology (ICT) is responsible for between 2.1% and 3.9% of all greenhouse gas emissions globally [1]. Current trends threaten to increase these percentages significantly. These include the growth in AI [2] and datacenter construction [3], as well as the expected uptick with the roll-out of autonomous vehicles [4]. Considering these circumstances, even optimistic perspectives on ICT's carbon footprint believe that a call to action is necessary [5].

Processors and systems-on-chip (SoCs) are a significant contributor to ICT's emissions. For example, Apple's A17 Pro SoC contributes between 14% and 21% of the iPhone 15 Pro's total emissions [6, 7].[1] In Microsoft's datacenters, despite massive amounts of storage and memory, processors are responsible for around 14% of carbon emissions [8].

The carbon footprint of an SoC can be credited to two main sources. The first is a processor's *operational emissions*, the result of its power/energy consumption. The second is the processor's *embodied emissions*, which account for the carbon footprint of fabricating, and to a lesser extent transporting and disposing of the processor. Optimizing for operational emissions has historically been the focus of power-efficient and sustainable computing [9], a reality easily seen by scrolling through the titles of papers published in past conferences focused on sustainable computing [10].

Over the years, hardware designers have noticed with surprise that operational emissions may no longer be the dominant factor. Instead, a processor's embodied emissions must now be considered on an equal footing with its operational emissions [12]. Returning to the example from before, embodied emissions are responsible for between 48% and 73% of the Apple A17 Pro SoC's total emissions [6, 7]. Furthermore, embodied emissions are growing in importance, since new process technologies will continue to decrease operational emissions but, as shown in Figure 1.1, the increased complexity

[1]Range based on the SoC consuming between 25% and 75% of the iPhone's energy.

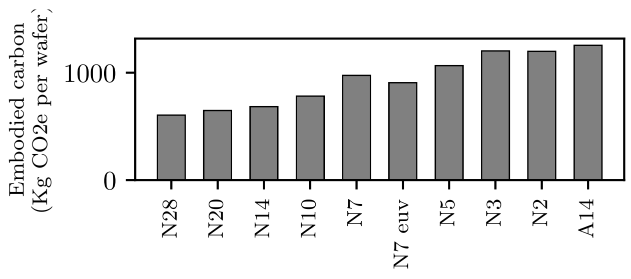


Figure 1.1: Embodied emissions per unit area of logic have steadily increased with new technology nodes. [11]

of these newer nodes leads to higher embodied emissions per wafer [11].

Although a chip's embodied emissions are highly dependent on the technology node in which it is fabricated, and can be amortized over a longer device lifespan [13], holding technology node and lifetime constant leads to approximately constant embodied emissions per wafer [14, 15]. This means that, as architects, we can optimize for embodied emissions by decreasing a processor's area, such that we get more chips out of a wafer.

Nowadays, hardware acceleration is often the biggest contributor to an SoC's area—as long ago as 2018, fixed-function, domain-specific accelerators (DSAs) already took up over two-thirds of the total silicon area on state-of-the-art smartphone SoCs [16]. As such, accelerators increase embodied emissions while decreasing the operational emissions through more energy efficient processing. This is especially true for fixed-function DSAs as each DSA requires its own distinct, low-utilization "dark" silicon area [17, 18]. Though *seas of DSAs* are the norm in many of today's mobile SoCs [19, 20], they are not the only option. Energy-constrained systems have been using FPGAs alongside their processors for a long time. Additionally, recent research into coarse-grained reconfigurable arrays (CGRAs) has demonstrated that they can offer significantly faster reconfigurability and lower power and area overheads than FPGAs for a given level of performance [21, 22, 23].

At first glance, the DSA may seem like the most carbon-efficient option for acceleration. For a given level of performance, because a DSA is highly *specialized*, it will generally take up less die area and consume less power than a reconfigurable fabric like a CGRA or FPGA, or than a Single Instruction Multiple Thread device like a GPU, for a given level of performance. However, as demonstrated in [24], a CGRA can in some cases decrease the carbon footprint of an SoC by replacing multiple DSAs with one reconfigurable fabric which is *shared* for multiple application kernels, such that the total area of the chip decreases enough to offset the increase in power consumption. In cases where embodied emissions dominate, a CGRA may only need to be shared by 3 application

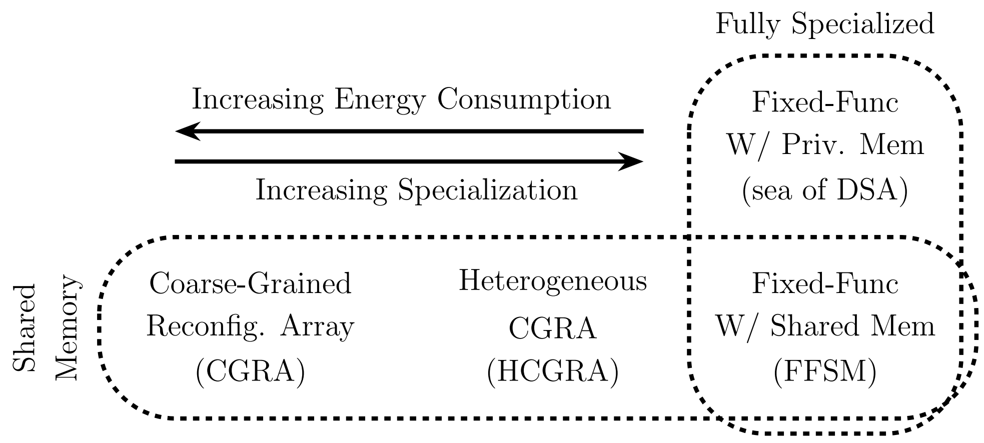


Figure 1.2: The architectures in our exploration. From left to right, they increase in compute specialization and decrease in energy consumption. The architectures on the bottom share memory between multiple kernels.

kernels [24].

Building on this insight, we sought to answer the following questions: how much specialization do we have to give up in order to share hardware, is it more sustainable to do so, and how much does this differ between compute logic and on-chip memory? In other words, what are the overheads we must endure to share compute logic and on-chip memory for multiple accelerated kernels, and are these overheads worth enduring from a carbon emissions perspective?

In this work, we demonstrate that CGRA is far from the best we can do. We explored the design space of architectures which are more specialized than CGRA, but include more hardware sharing than traditional sea of DSA architectures, to find the carbon-optimal design points. Specifically, in addition to the CGRA and sea of DSA architectures, we examined a *heterogeneous CGRA* (HCGRA) architecture and an architecture with fully specialized fixed-function compute which shares a pool of memory (FFSM).

An HCGRA is built by removing low utilization hardware, like underutilized ALU operators or unneeded configuration storage from some of the processing elements in a CGRA [25, 26, 27].

Our FFSM architecture uses a low-overhead hardware block to interface highly specialized compute hardware for many distinct kernels to a shared pool of SRAM. These architectures are summarized in Figure 1.2.

We compiled, generated, and synthesized hardware descriptions of these different architectures to acquire realistic power and area estimates. These power and area estimates were used to perform a first order carbon emissions analysis of each architecture with a variety of different memory sizes and relative weights of operational and embodied emissions.

Broadly, we find that past work focused on sharing compute hardware to decrease an SoC's carbon footprint has misattributed the benefits of sharing memory to using reconfigurable compute hardware. Specifically, our major findings are as follows:

- The overhead for sharing compute hardware is high—an 8x8 CGRA needs ¿100X the area of an average DSA—and the potential savings are low since memory dominates an accelerator's area. However, the potential savings for sharing memory are high, and the overhead for sharing memory is small—as low as 3X the area of an average DSA for our FFSM architecture.

- In line with this, the FFSM architecture is the most sustainable for hardware acceleration in 90% of the scenarios we explored. When sharing hardware for 8 application kernels, FFSM was up to 17.1X, 8.9X, and 3.6X more carbon efficient than CGRA, HCGRA, and a sea of DSAs. Only a sea of DSAs ever had a lower carbon footprint, and only when operational emissions were highly dominant.

- Systems with changing or unclear applications benefit from using a reconfigurable fabric. For these architectures, we show that HCGRAs are significantly less carbon emitting than CGRAs. By reducing power and area overheads by 49.1% and 36.7% respectively, an HCGRA's carbon footprint is over 38% lower than a traditional CGRA in all operationally dominant operating conditions tested. HCGRA was more sustainable than CGRA in all considered scenarios.

- Although memory sharing has high leverage, only a handful of DSAs can connect to a single shared memory pool, so kernels must be clustered into groups to share memory. We show that grouping together kernels with similar memory capacity requirements has an outsize effect on carbon efficiency. Doubling the ratio between the average and the maximum memory capacity required for the kernels sharing a pool of memory results in a 2X decrease in carbon footprint in a range of our experiments.

Chapter 2

Background

2.1 Analyzing a Processor's Carbon Footprint

In order to make more sustainable processors, it is important to have architecture-level models that can accurately estimate a processor's carbon footprint. Such models have been the focus of multiple recent publications [13, 9, 28], each of which uses the summation of operational and embodied emissions as the starting point for estimating the carbon footprint (CF) of any hardware component.

$$CF_{total} = CF_{embodied} + CF_{operational} \tag{2.1}$$

For processors, operational emissions are modeled by the power consumption (P) multiplied by the carbon emitted per Watt of power (C_p). Embodied emissions are modeled by chip area (A) multiplied by the carbon emitted per unit area during manufacturing (C_a).

$$CF_{processor} = C_a A + C_p P \tag{2.2}$$

Different models then take a different approach to filling in the values for C_a and C_p. For example, *ACT* uses real power-grid carbon intensity data for C_p, and a summation of emissions from sourcing materials and from using gases, chemicals, and electricity during manufacturing for C_a [13]. In this work, we use a different approach, based on the *FOCAL* carbon modeling framework [28]. Specifically, we model C_a and C_p as hyperparameters, which we can sweep over different values to see which architectures are the most sustainable in a wide range of operating conditions. By

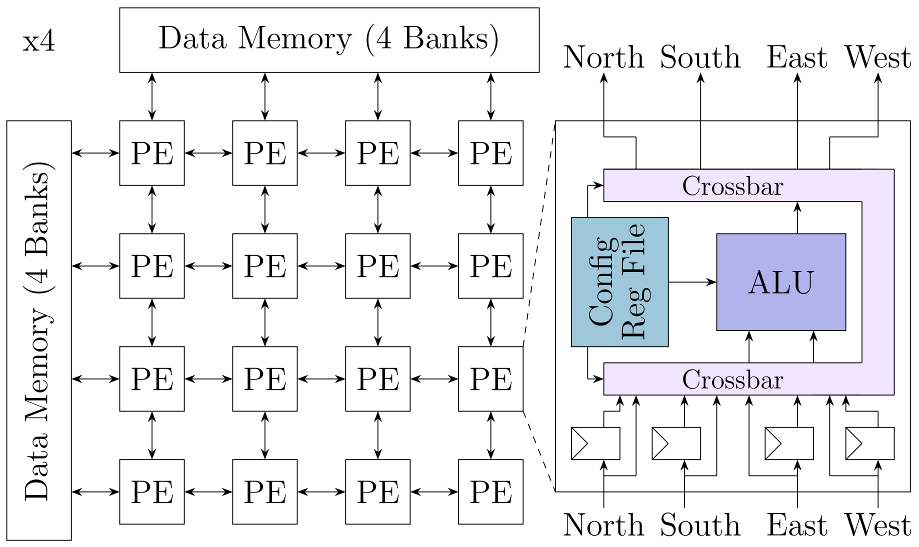


Figure 2.1: The CGRA, based on HyCube [29], analyzed in this work. We modeled an 8x8 CGRA using 4, 4x4 CGRAs.

modeling C_a and C_p in this way, we can more easily and quickly explore a wide design space. Our *FOCAL* based carbon modeling is described in detail in Section 3, and we map each architecture onto our carbon models in Section 4.2.

2.2 Compute Specialization: Accelerator Architectures from CGRA to DSA

2.2.1 Coarse grained reconfigurable array (CGRA)

A Coarse-grained reconfigurable array, or CGRA, is a dataflow architecture consisting of a reconfigurable grid of processing elements, as shown in Figure 2.1. Each PE can be configured each cycle to take different inputs from neighboring PEs, perform different computations on these inputs, and then send the result back out to neighboring PEs. Unlike an FPGA, which computes on data at the bit granularity, data moves through the CGRA in entire words. The result of this difference is that CGRAs can be reconfigured significantly faster than FPGAs, making them well-suited for being time-shared for multiple kernels. Additionally, CGRAs have significantly lower power and area overheads compared to FPGAs. CGRA are the least specialized architecture we consider.

2.2.2 Heterogeneous CGRA

If a traditional CGRA's processing elements can all perform the same operations, but some of those operations are rarely or never used by the algorithms the CGRA is performing, then that implies that there is hardware which has been instantiated unnecessarily. Similarly, and more significantly, as highlighted in [25], the configuration register files inside of those processing elements, may not need to be able to represent every possible configuration, if certain configurations are highly unlikely, or if the configuration is changing infrequently. *Heterogeneous CGRA* is a more *specialized* architecture, which capitalizes on these and similar insights in order to reduce the overheads of a CGRA. For example, an HCGRA may have operators cut out of its ALUs, and configuration register files of reduced size compared to a CGRA. Heterogeneous CGRAs have been shown to reduce area and power conumption by factors of 30% to 50% compared to corresponding traditional CGRAs [25, 26, 27]. Furthermore, these improvements can be made with little to no performance cost, especially when the exact algorithms the HCGRA will be performing are known in advance. HCGRA architectures represent a middle ground between CGRA and fixed-function DSAs in terms of specialization.

2.2.3 Sea of Domain-Specific Accelerators (DSAs)

Fixed-function domain-specific accelerators consist of highly-specialized, purpose-built hardware designed to perform a particular, repeated computational task with high performance and power efficiency. To do this, DSAs will often perform many independent computations in parallel, with custom designed hardware blocks, maximally optimized for a specific task. Modern day SoCs often contain many tens of these DSAs [20, 17]. Because it can be fine tuned to the algorithm it is performing, a DSA will have a smaller area and power consumption than a reconfigurable fabric configured to execute the same algorithm, however to perform many distinct operations, you will need to instantiate many distinct DSAs.

2.3 Memory Sharing

2.3.1 Fixed-function compute with shared memory (FFSM)

SRAM often dominates the area of fixed-function DSAs [30, 31]. Additionally, though the logic required for a DSA is tightly intertwined with the algorithm being performed, most SRAMs, such as those used for caches and scratchpad memories, are highly structurally similar. For these reasons, we consider an architecture which interfaces the compute hardware of many fixed-function DSAs with

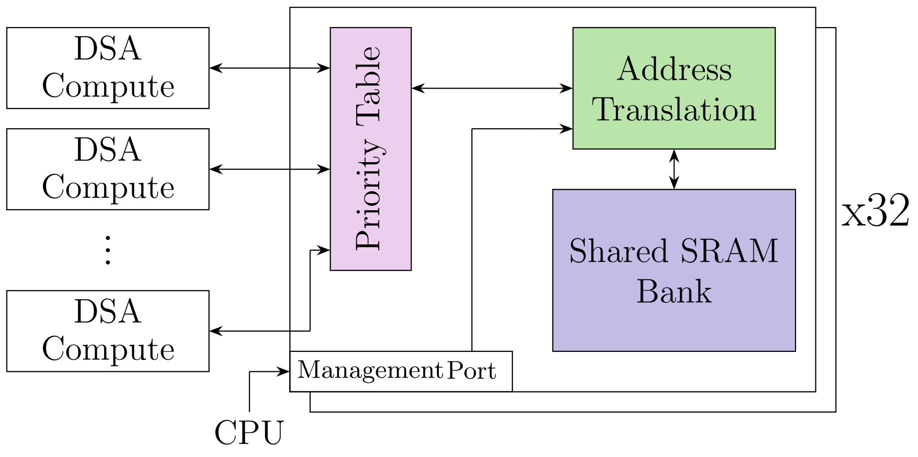


Figure 2.2: The hardware, inspired by an Accelerator Store [30], used by our FFSM architecture for sharing memory between the compute hardware for multiple, fixed-function DSAs

a shared pool of SRAM. By sharing SRAM, we reduce the total amount of memory needed on the chip, and thus the chip's area, in exchange for small power overheads. Inspired by the Accelerator Store [30], we implemented low-overhead hardware for sharing a pool of memory between multiple DSAs, as shown in Figure 2.2. Like the Accelerator Store, our hardware uses a priority table for selecting which accelerator is allowed to perform a memory access, and an address translation layer which can be configured to use an SRAM bank as a FIFO buffer or for random access. Unlike the Accelerator Store, each SRAM bank is shared independently, and each accelerator port connects to only a single SRAM bank, as opposed to a routing system based on multiple shared channels. Throughout this paper, we refer to the architecture using this hardware as the FFSM (fixed-function shared memory) architecture.

2.3.2 CGRA and HCGRA Memory Sharing

Because CGRA and HCGRA use the same hardware to perform different computations, memory is inherently shared for each different application kernel.

2.4 Accelerator Memory Hierarchies

We consider only the simplified case of accelerator memory hierarchies with a single level on-chip, which can either be shared (for CGRA, HCGRA, and FFSM), or unshared (sea of DSAs). However, different systems implement accelerators with a range of different memory structures, many of which are discussed in [31]. Our model is consistent with accelerators which access DRAM directly using DMA and are not cache coherent with the rest of the system. On the other hand, some accelerators' memory hierarchies contain many levels on chip, some of which are shared and some of which are unshared. We believe our simplification is not overly reductive for a first-order carbon analysis, and makes it easier to see the effects of sharing memory (and of making more of the memory hierarchy shared) on carbon emissions. For more complex accelerator memory hierarchies, our results motivate maximizing the amount of shared memory by, for example, replacing private L2 caches with shared L2 caches for accelerators.

Chapter 3

Carbon Analysis

Our carbon emissions model is based on *FOCAL* [28]. As explained in Section 2.1, the following equation can be used to compute a processor's total carbon emissions, where P is power consumption, C_p is carbon emitted per Watt of power, A is chip area, and C_a is carbon emitted per unit area during manufacturing.

$$CF_{processor} = C_a A + C_p P \tag{2.2}$$

The *FOCAL* model swaps out equation 2.2 for equation 3.1, which is very similar. To get to this equation, *FOCAL* first replaces C_a and C_p with a single parameter, α_{E2O}, which represents the relative importance of embodied emissions compared to operational emissions. We refer to α_{E2O} as the *operating conditions.* This allows for sweeping over many different operating conditions easily, by only changing one variable. Then, *FOCAL* normalizes the area and power consumption for an architecture under test to a reference architecture, and multiplies the normalized area by α_{E2O} and the normalized power (or energy in the constant work case) by $(1 - \alpha_{E20})$, to yield a normalized carbon footprint (CF_{norm}). If this CF_{norm} is less than 1, then *FOCAL* concludes that the test architecture is more sustainable than the reference, and vice versa.

$$CF_{norm} = \alpha_{E20} \frac{A_{test}}{A_{ref}} + (1 - \alpha_{E20}) \frac{P_{test}}{P_{ref}} \tag{3.1}$$

Unfortunately, *FOCAL* is easily subject to misunderstanding because α_{E2O} specifies the operating conditions of *only* the reference architecture. This distinction is critical, as the relative importance of embodied and operational emissions can differ significantly for two architectures because they have different area and power consumptions. One result of this is that *FOCAL* does not yield mirrored results when swapping the test and reference architectures in equation 3.1, as changing the reference architecture changes the meaning of α_{E2O}. Another is that a *FOCAL* analysis may appear to be considering a case which is dominated by embodied emissions ($\alpha_{E2O} > 0.5$), when only the reference is dominated by embodied emissions, but the test architecture is dominated by operational emissions. To highlight that α_{E2O} is tied to the reference architecture, we rearrange equation 3.1 into equation 3.2.

$$CF_{test} = \frac{\alpha_{E2O}}{A_{Ref}} A_{test} + \frac{(1 - \alpha_{E2O})}{P_{Ref}} P_{test} \tag{3.2}$$

To increase clarity, in Section 3.1 we show a derivation of equation 3.2 from equation 2.2 to be clear what the operating conditions are, and where they are derived from.

First, however, we emphasize that our reference architecture throughout this paper will be the CGRA architecture. As such, moving forward, α_{E2O} will be referred to as α_{cgra}. The CGRA is the best choice as a reference because, unlike the FFSM and sea of DSAs architectures, the CGRA's operating conditions are not a function of the number of application kernels which share it, and we want this number of kernels to be something that our model can solve for. This way, we can find the critical number of kernels which must share an architecture in order for it to be more sustainable than a sea of DSAs. Additionally, whereas we expect the carbon footprint for the sea of DSAs architecture to be dominated by embodied emissions, since each kernel requires it's own dedicated, low-utilization area, it is less clear what operating conditions are likely for the CGRA, making it more important to examine multiple CGRA operating conditions.

3.1 *FOCAL* Derivation

The key parameter of our *FOCAL* carbon analysis is α_{cgra}, the proportion of our CGRA's carbon footprint which is attributed to embodied emissions. For example, if embodied emissions are 80% of the CGRA's total carbon emissions, then $\alpha_{cgra} = 0.8$. We can set up the following equation using α_{cgra}. As a reminder C_A is the carbon emitted per unit area during manufacturing, and CF_{total} is total carbon footprint.

$$\alpha_{cgra} CF_{processor} = CF_{emb} = C_a A_{cgra} \tag{3.3}$$

Substituting equation 2.2 into equation 3.3, yields the following.

$$\alpha_{cgra}(C_a A_{cgra} + C_p P_{cgra}) = C_a A_{cgra} \tag{3.4}$$

$$=> \alpha_{cgra} C_p P_{cgra} = C_a A_{cgra} - \alpha_{cgra} C_a A_{cgra} \tag{3.5}$$

$$=> \alpha_{cgra} C_p P_{cgra} = (1 - \alpha_{cgra}) C_a A_{cgra} \tag{3.6}$$

From here there are many solutions for C_a and C_p, all of which are mathematically sound and can be used. However, the following solutions are used by *FOCAL*.

$$C_a = \frac{\alpha_{cgra}}{A_{cgra}} \qquad C_p = \frac{(1 - \alpha_{cgra})}{P_{cgra}} \tag{3.7}$$

Plugging these equations for C_a and C_p back into equation 2.2 yields equation 3.2

$$CF_{test} = \frac{\alpha_{cgra}}{A_{cgra}} A_{test} + \frac{(1 - \alpha_{cgra})}{P_{cgra}} P_{test} \tag{3.2}$$

In summary, α is the proportion of the reference architecture's (CGRA in this paper) emissions which are attributed to embodied emissions, and is used to derive and sweep over values for C_a and C_p which, when plugged into equation 2.2, yield equation 3.2.

Chapter 4

Methodology

4.1 Power and Area Estimation

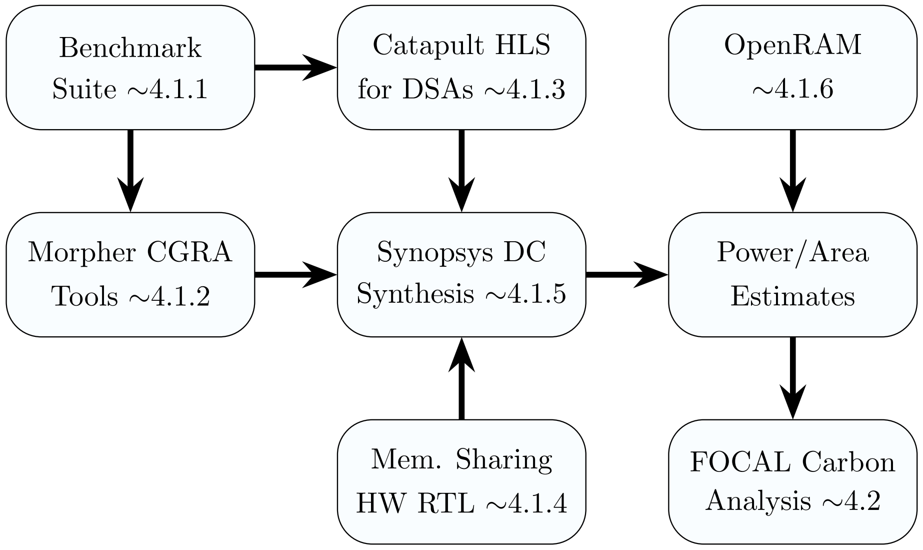
Figure 4.1 shows our toolflow for analyzing the carbon footprint of each architecture under study.

4.1.1 Assembling a Benchmark Suite

To start, we assembled a comprehensive suite of commonly accelerated benchmark kernels on which to base our instantiations of each architecture. Because a DSAs structure is tightly intertwined with the function it performs, a representative sample of benchmarks must incorporate accelerators from a wide variety of domains. On the other hand, for an HCGRA to be able to be maximally specialized, it must be used for application kernels with similar computational needs. For these reasons, we curated a suite of 16 kernels, clustered into groups of four kernels from each of four application domains: security, machine learning, image processing, and digital signal processing. Table 4.1 lists our kernels by domain. These kernels are derived from a wide variety of open source repositories [32, 33, 34, 35, 36, 37, 38, 39] and use helper functions from others [40, 41].

4.1.2 (H)CGRA Generation and Mapping

We mapped each benchmark onto CGRA and HCGRA. The CGRA we used for this analysis is depicted in Figure 2.1. Specifically, we used an 8x8 CGRA composed of four, 4x4 CGRAs, each based on the *HyCUBE* architecture [29]. Each PE is connected to its north, south, east, and west neighbors, and PEs on two of the edges are connected to banks of data memory. Each 4x4 CGRA is connected to 8 banks of SRAM, for a total of 32 banks. Each memory bank is an equal size, and



Benchmark Suite ∼4.1.1
Catapult HLS for DSAs ∼4.1.3
OpenRAM ∼4.1.6
Morpher CGRA Tools ∼4.1.2
Synopsys DC Synthesis ∼4.1.5
Power/Area Estimates
Mem. Sharing HW RTL ∼4.1.4
FOCAL Carbon Analysis ∼4.2

Figure 4.1: The flow of our power and area estimation tools into our carbon analysis.

Security	AI/ML	Image Processing	DSP
AES Encrypt	2 Matrix Mult	DCT	FFT
ChaCha20 Encrypt	2d-Convolution	Histogram EQ	DFT
Blake2s Hash	Max Pooling	RGB to YUV	FIR
SHA-256 Hash	Batch Norm	Auto Focus	IIR

Table 4.1: The set of benchmark kernels analyzed. Four kernels were chosen from each of four application domains.

this size is a parameter we vary in our analysis. This architecture was chosen to be consistent with prior work [24].

Using *Morpher* [21], we generated a dataflow graph (DFG) for each benchmark kernel. We passed these DFGs, as well as CGRA configuration files describing our base homogeneous CGRA architecture, into an extension of *Morpher* called *Revamp* [25] to generate an HCGRA based on the distribution of operations in the kernels. For example, a set of kernels with very few shift operations will lead to an HCGRA with very few shifters in its ALUs. Using *REVAMP*, each HCGRA can also be instantiated with a reduced size configuration register file as discussed in Section 2.2.2, providing an even larger area and power consumption decrease. We generated a separate HCGRA for each of the four domains of kernels in Table 4.1, and used the average of their areas and power consumptions in our carbon analysis. Next, using *Morpher*, we mapped the DFG for each application kernel onto the CGRA and then the generated HCGRA. If the HCGRA performed worse than the CGRA, we added resources back into the HCGRA until it met the performance of the CGRA.

This step (adding resources back into the HCGRA to meet performance) is necessary. To effectively compare the carbon footprint of these architectures, they must each achieve approximately the same level of performance. This requirement equalizes the runtimes of each architecture for a given workload, enabling direct comparisons of their dynamic power consumption, and also removes the need to consider the constant-work and constant-time cases separately. Additionally, it approximately constrains the hardware resources each architecture must dedicate to performing a given kernel, enabling a more fair leakage power and area comparison.

4.1.3 Generating DSAs with HLS

We used Siemens' high-level synthesis (HLS) tool *Catapult* to generate RTL descriptions of domain specific accelerators for the algorithms in our benchmark suite. *Catapult* and *Morpher* take almost identical C-language inputs. To create DSAs with equal performance to the CGRAs, we modulated factors of loop unrolling, pipeline initiation interval, clock frequency, and number of duplicate instances of the DSA for an algorithm. Area was used as the optimization target in *Catapult*.

4.1.4 Instantiating Low Overhead Memory Sharing Hardware

As discussed in Section 2.3.1, we implemented and tested an RTL implementation of low-overhead memory sharing hardware, as shown in Figure 2.2. For our carbon analysis, we implement this hardware with 32 banks of SRAM (to be consistent with the (H)CGRA), and a variable number of accelerator ports.

4.1.5 Synthesis With Synopsys Design Compiler

We used Synopsys *Design Compiler* to synthesize each of the architectures. After synthesis, *Design Compiler* was queried for power and area reports for each architecture. We used the provided *Design Compiler* synthesis scripts from *Catapult* for our DSAs, and from *Morpher* for our CGRA and HCGRAs. We used a modified version of the CGRA synthesis scripts to synthesize our memory sharing hardware for our FFSM (fixed-function compute with shared memory) architecture. We used the Nangate/Silvaco open cell library based on the 45nm FreePDK predictive silicon technology library for all synthesis [42, 43]. Without switching characteristics available for representative executions of each kernel for all architectures, our estimates rely on *Design Compiler*'s default switching activity profile.

4.1.6 SRAM Compilation with *OpenRAM*

To characterize SRAMs, we used OpenRAM, an open-source memory compiler [44]. Each memory bank was compiled with a single read/write port, and with a word size of 32 bits. We compiled memories using the same 45nm technology library as used for synthesizing the compute hardware. OpenRAM's outputs include leakage power, and area estimates, as well as dynamic power estimates per read/write. For our analysis, we assume that each kernel accesses its lowest level memory twice per cycle on average, which is consistent with the schedules produced by *Catapult*.

4.2 Carbon Modeling for Each Architecture

We now show how the power and area of each of the architectures we examine can be mapped onto equation 3.2. For CGRA and HCGRA, this mapping is fairly simple. In the equation, we set A_{test} equal to the area of the CGRA, or the average area of the HCGRAs for all four sets of kernels. Similarly, we set P_{test} to the average power consumption per cycle for the CGRA and the average over all of the HCGRAs. Because CGRA is our reference architecture, its carbon footprint will always evaluate to 1.

$$CF_{cgra} = \frac{\alpha_{cgra}}{A_{cgra}} A_{cgra} + \frac{(1-\alpha_{cgra})}{P_{cgra}} P_{cgra} = 1 \tag{4.1}$$

$$CF_{hcgra} = \frac{\alpha_{cgra}}{A_{cgra}} A_{hcgra} + \frac{(1-\alpha_{cgra})}{P_{cgra}} P_{hcgra} \tag{4.2}$$

The sea of DSAs architecture can be mapped onto equation 3.2 similarly. First, we take the average area and power consumption of all of the DSAs synthesized, including their memory requirements, and use this as A_{test} and P_{test}. We also multiply A_{test} by a new parameter, N_{dsa}, which represents the number of DSAs in our sea of DSAs, to get the total area for the sea of DSAs. We do not multiply P_{dsa} by N_{dsa}, since we assume in our comparison that only one DSA is active at a time (to match the CGRA).

$$CF_{dsa} = \frac{\alpha_{cgra}}{A_{cgra}} N_{dsa} A_{dsa} + \frac{(1-\alpha_{cgra})}{P_{cgra}} P_{dsa} \tag{4.3}$$

To map the FFSM architecture onto equation 3.2, we consider both the total area and power consumption of the memory sharing hardware itself (A_{sm} and P_{sm}), including the memory it contains, and the area and power consumption of the compute hardware for an average DSA, since this architecture keeps all of the compute hardware from the sea of DSAs architecture. The average DSA area is multiplied by the number of kernels, like in equation 4.3.

$$CF_{ffsm} = \frac{\alpha_{cgra}}{A_{cgra}}(N_{dsa}A_{ff.comp} + A_{sm}) + \frac{(1-\alpha_{cgra})}{P_{cgra}}(P_{ff.comp} + P_{sm}) \tag{4.4}$$

From these equations, we can find a critical threshold for N_{dsa}, where a given architecture switches from having a higher carbon footprint than another architecture to a lower one. We can also quantify, for specific values of N_{dsa}, how much more or less sustainable one architecture is compared to another.

Chapter 5

Results

5.1 Compute Hardware Area and Power Consumption

Table 5.1 summarizes the area and power estimates for each architecture as synthesized by *Design Compiler*. Compared to the CGRA, the average heterogeneous CGRA requires 36.7% less chip area, and 49.1% less power. These results are consistent with existing research comparing HCGRA to CGRA [27, 26, 25]. Notably, 3 of the 4 HCGRAs synthesized had highly similar area and power consumptions to each other. The HCGRA generated for executing security kernels, on the other hand, required a significantly higher area and power consumption than the other HCGRAs because it could not meet the CGRA's performance with *REVAMP*'s changes to the configuration register files of each PE, so these register files were left unchanged.

The average DSA is over 100 times smaller and less power consuming than the CGRA architecture. Though this result may be surprising at first, it makes sense when considering how much more hardware a CGRA requires than a DSA for a single kernel. Specifically, our 8x8 CGRA requires 64 configuration register files, 64 ALUs, each with many different operators, 64 crossbar routers, and a lot of interconnect. On the other hand, to match the CGRA's performance, even the largest synthesized DSA only required a few operators and registers, some simple state logic, and significantly less interconnect. Furthermore, because each PE in the CGRA can only execute one operation per cycle, pipeline initiation intervals become larger, indicating that the CGRA is less efficient with its resources, and because it is limited to a single bitwidth, operations which require no delay and no hardware in the DSA, like shifting by a constant, require an entire cycle and PE in a CGRA. All of this is to say that the large difference in area and power between the CGRA and DSA is not

	Av. Power (μW)	Av. Area (μm^2)
CGRA	$142,999$	$1,716,528$
HCGRA	$72,727$	$1,086,105$
Average DSA	$1,372$	$16,619$
Mem Sharing overhead	$3,372 + 222N_{dsa}$	$48,027 + 10603N_{dsa}$

Table 5.1: The average area and power consumption estimates synthesized by Synopsys *Design Compiler* for the compute hardware for each architecture in our sustainability analysis.

unreasonable and is to be expected.

The overhead of the memory sharing hardware for the FFSM architecture is significant compared to the size of a DSA, but is significantly less than 5% of the overhead of CGRA. The specific overhead depends on the number of DSAs the hardware was configured to interface with, as each new interface port requires more interconnect and logic to be instantiated in the sharing hardware. Specifically, each new accelerator adds $10603\mu m^2$ and $222\mu W$ to the overhead.

One way to quantify the sustainability of a sharing architecture is to specify how many kernels must share hardware for the architecture to be more carbon-efficient than a sea of DSAs where one DSA is instantiated for each kernel. Specifically, the number of kernels which must share silicon area, for an area decrease (embodied emissions) that can offset the increased power consumption (operational emissions) of a sharing architecture compared to a DSA. We refer to this quantity throughout this paper as the *kernel sharing threshold*.

When just considering compute (ignoring cache/scratchpad memories), neither CGRA nor HCGRA provides a sustainability benefit with a reasonable kernel sharing threshold. **Even in cases highly dominated by embodied emissions ($\alpha_{cgra} = 0.95$), the CGRA would need to be shared by 109 application kernels, and the HCGRA by 68 application kernels.** These numbers balloon to 513 and 272 in operationally dominated conditions.

It is unrealistic to expect a CGRA to be shared by 109 kernels, or an HCGRA to be shared by 68, without a significant cost to the performance of the chip, as it is highly unlikely that this many application kernels can share a single CGRA without timing conflicts. This is to say that specialized compute hardware is plainly more carbon-efficient than reconfigurable, shared compute hardware (assuming a predetermined set of kernels). To see how a CGRA, HCGRA, or the FFSM architecture can decrease emissions, we need to look at the area and power consumption of *memory*.

	Leak Power (μW)	Dynamic Power per Access (μW)	Av. Area (μm^2)
128B	1.4	36.5	$9,030$
256B	2.4	63.1	$9,243$
512B	4.7	116.1	$14,858$
1kB	8.9	210.1	$19,460$
2kB	17.7	410.7	$34,469$
4kB	34.6	788.8	$64,790$
8kB	69.1	1575.4	$118,617$
16kB	136.7	3134.1	$240,817$
32kB	273.1	6352.8	$447,698$

Table 5.2: The area and power consumption values acquired from OpenRAM for different sized banks of SRAM.

5.2 SRAM Area and Power Consumption

Table 5.2 details the power and area estimates for SRAM banks of different sizes, compiled using *OpenRAM*. Generally, the area and power consumption of a bank of SRAM are linearly proportional to it's size. For smaller banks of SRAM, however, the overhead of the hardware the bank requires outside of it's array of bit storage is more significant, especially as a proportion of the bank's area.

Comparing the memory banks to the compute hardware, it is clear that SRAM contributes a significant proportion of both the area and power consumption of any hardware acceleration. A single, 1kB bank of SRAM, requires more chip area than all of the compute for an average sized DSA (with CGRA levels of performance).

Because the smallest bank size OpenRAM could compile was 128B, and our CGRA and HCGRA architectures use 32 banks of SRAM, we set our smallest considered memory size to 4kB (32 banks of 128B). As such, even the smallest considered memory size dominates the area of an average DSA, and consumes a slim majority of it's power. For small memory sizes, the ratio between our DSA power and area estimates and our memory power and area estimates is consistent with existing work [30, 45]. We consider average kernel memory capacity requirements up to 256kB, to be consistent with prior work [24].

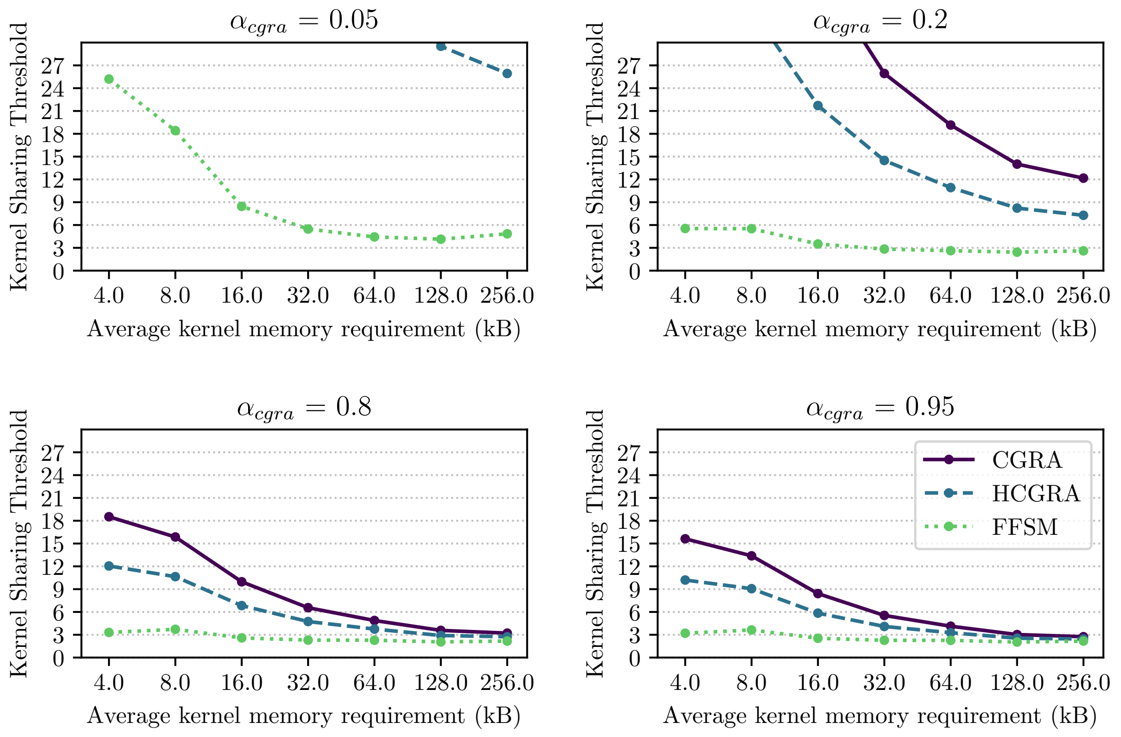


Figure 5.1: The number of kernels which must share hardware for a given architecture to have a lower carbon footprint than a sea of DSAs at four different operating conditions and with different average kernel memory capacity requirements. Each architecture is allocated twice the memory required for an average kernel. Lower is better.

5.3 Critical Kernel Sharing Thresholds

Figure 5.1 plots the number of application kernels which must share a CGRA, HCGRA, or FFSM architecture in order to emit less carbon than a sea of DSAs for those kernels. For example, if 9 kernels can share hardware without a performance penalty, then all architectures can provide a sustainability win at $\alpha_{cgra} = 0.8$ if the kernels require 256kB of memory on average, but only FFSM can if the kernels require 4kB on average (see bottom left panel in Figure 5.1).

We measured this kernel sharing threshold at different CGRA operating conditions (as discussed in Section 3), spanning from a highly operationally dominated case ($\alpha_{cgra} = 0.05$) to a highly embodied dominated case ($\alpha_{cgra} = 0.95$), and for seven different average kernel memory capacity requirements.

In the embodied dominant cases, at large memory sizes, all of the memory sharing architectures approach an asymptotic minimum value of 2 kernels. 2 is the asymptotic minimum because each memory sharing architecture is allocated a pool of memory twice the size of the average kernel memory capacity requirement (the value along the x-axis). This ratio of average kernel memory requirement to shared memory pool size is explored further in Section 5.5. Since each kernel is generally active only a small proportion of time, 2 kernels can easily share hardware.

At smaller memory sizes and in operationally dominated cases, the critical kernel sharing threshold for CGRA quickly rises. As such, it is clear that CGRA can only provide a sustainability benefit over a sea of DSAs in ideal conditions, with embodied dominant carbon emissions, high memory requirements, and/or large numbers of kernels which can share hardware without a performance penalty. Outside of these conditions, the power and area overhead required for a CGRA's reconfigurability is too high.

The added specialization of Heterogeneous CGRA leads to a substantial improvement over CGRA. For $\alpha_{cgra} = 0.8$ and $\alpha_{cgra} = 0.95$ with a 4kB average DSA memory size, HCGRA reduces the critical number of kernels which must share hardware by 35% and 32% respectively. In the operationally dominated cases, the HCGRA reduced the kernel sharing threshold by more than 40% at *every* measured memory size. There is no case where the CGRA emits less carbon than the HCGRA.

However, the fixed-function compute with shared memory (FFSM) architecture is the clear winner of the three. The threshold of shared kernels for FFSM increases only a small amount over it's asymptotic minimum for all but the most operationally dominated case. There is no measured case where this architecture is less sustainable than either the HCGRA or the CGRA. For $\alpha_{cgra} = 0.8$ with a small average kernel memory size like 4kB, the FFSM architecture's kernel sharing threshold is 3.6X and 5.6X smaller than than the critical threshold for HCGRA and CGRA respectively. In the operationally dominated cases, the threshold is always more than 2.8X and 4.7X smaller than HCGRA and CGRA. In the highly operationally dominant case the CGRA's kernel sharing threshold does not even make it onto the graph, and the HCGRA's threshold hardly does. However, FFSM still provides a reasonable kernel sharing threshold at all but the two smallest considered memory sizes.

5.4 DSAs with Higher Performance than a CGRA

Because we consider DSAs with performance equivalent to that of a CGRA, we are necessarily considering relatively small DSAs. To examine how this impacts our results, we used *Catapult* to instantiate and synthesize versions of each DSA with 4 times the performance of the CGRA. To do this, we used combinations of inner loop unrolling, shrinking pipeline iteration intervals, increasing clock frequency, and instantiating more instances of our DSAs. The average area and

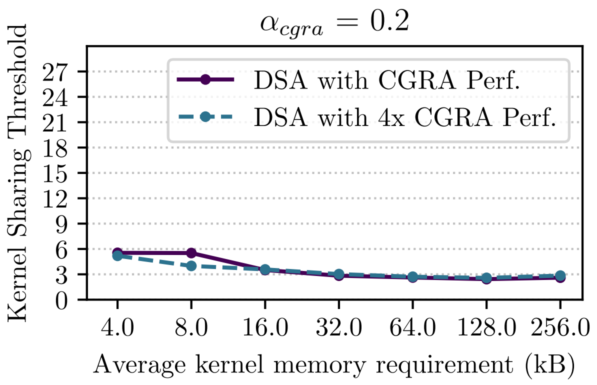


Figure 5.2: The number of kernels which must share hardware for an FFSM architecture to be more carbon-efficient than a sea of DSAs if the compute has CGRA levels of performance, or 4X CGRA levels of performance. Each architecture is allocated twice the memory required for an average DSA. Lower is better.

power consumption respectively for these higher performance DSAs were 1.83X and 3.23X those of the DSAs with CGRA levels of performance. Figure 5.2 illustrates the small effect that this change has on the critical threshold for the FFSM architecture. Though the area and power consumptions of the compute hardware rise relative to the memory hardware, which makes FFSM less effective, they also rise relative to the overhead of the hardware instantiated for sharing memory. These two effects mostly cancel out. These results generalize to all considered values of α_{cgra}.

5.5 Maximum to Average Memory Requirement Ratio

A contemporary SoC may contain many tens of DSAs [20], and it is infeasible from a floor planning and a time sharing perspective for this many kernels to share a single pool of lowest level memory. As such, it is important to consider how to best cluster these kernels.

The decrease in area achievable by a memory sharing architecture depends significantly on the ratio between the average memory required by a kernel and the amount of memory which must be allocated in the shared memory pool. Figure 5.3 visualizes this importance. The architectures on the left can both be replaced by the same sharing architecture on the right with 64kB of SRAM, however doing so for the architecture in the top left saves significantly more area than doing so for the architecture in the bottom left. This motivates clustering kernels by their memory requirements in order to maximally decrease total chip area.

To examine this effect quantitatively, we evaluated the kernel sharing threshold for the FFSM architecture with different ratios between size of the memory pool allocated to the memory sharing

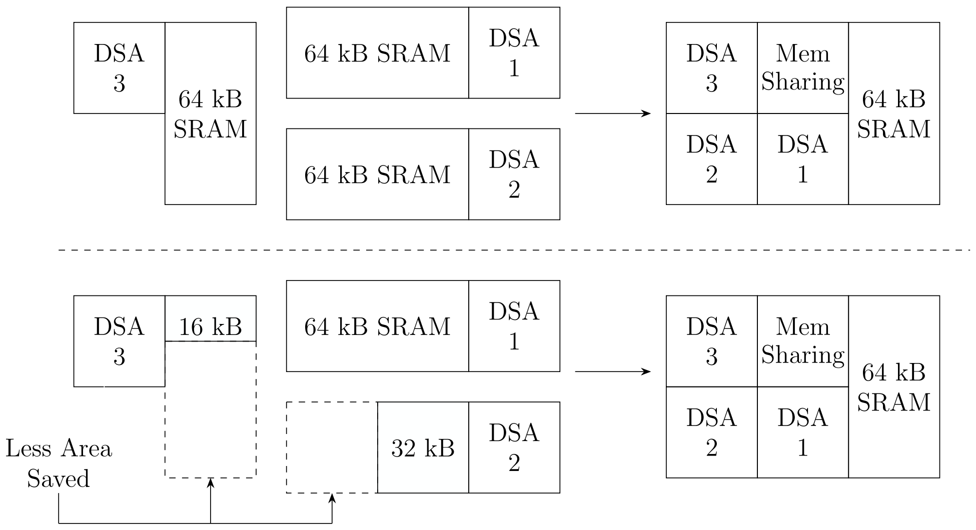


Figure 5.3: The ratio between maximum and average DSA memory requirements affect the ability of a memory sharing architecture to reduce area.

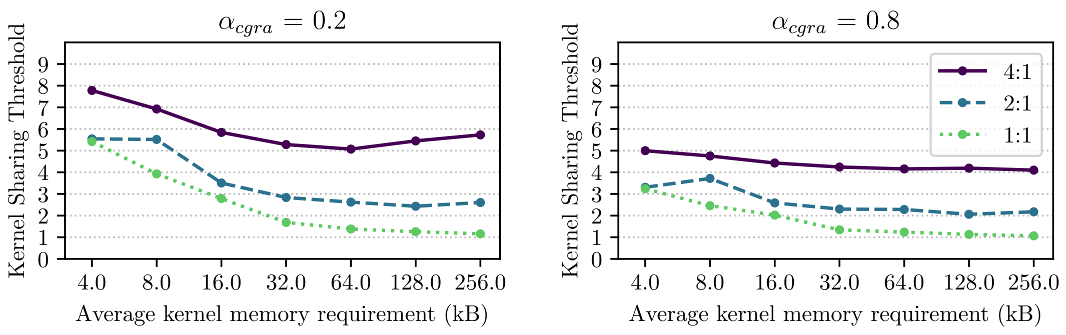


Figure 5.4: The number of kernels which must share an FFSM architecture for it to be more carbon-efficient than a sea of DSAs for different ratios between the memory allocated for the shared pool and the average memory required by a DSA in the sea of DSAs. Lower is better.

architecture, and the average memory size required by each kernel being shared. For example, for a ratio of 4:1, and an average kernel memory requirement of 4kB, 16kB of SRAM will be allocated in the memory sharing architecture's pool of memory. The results of this evaluation are plotted in Figure 5.4 for two operating conditions. Unless otherwise specified, all other data presented throughout this work uses a 2:1 ratio between the size of the shared memory pool and the average DSA memory requirement.

Changing between different ratios shifts the critical threshold along the y axis. As expected, decreasing the ratio to 1:1 decreases the number of kernels which must share, and increasing the ratio to 4:1 increases the threshold. Notably, this ratio is the asymptotic minimum number of kernels which must share to reduce emissions—this means that switching from a 2:1 ratio to a 1:1

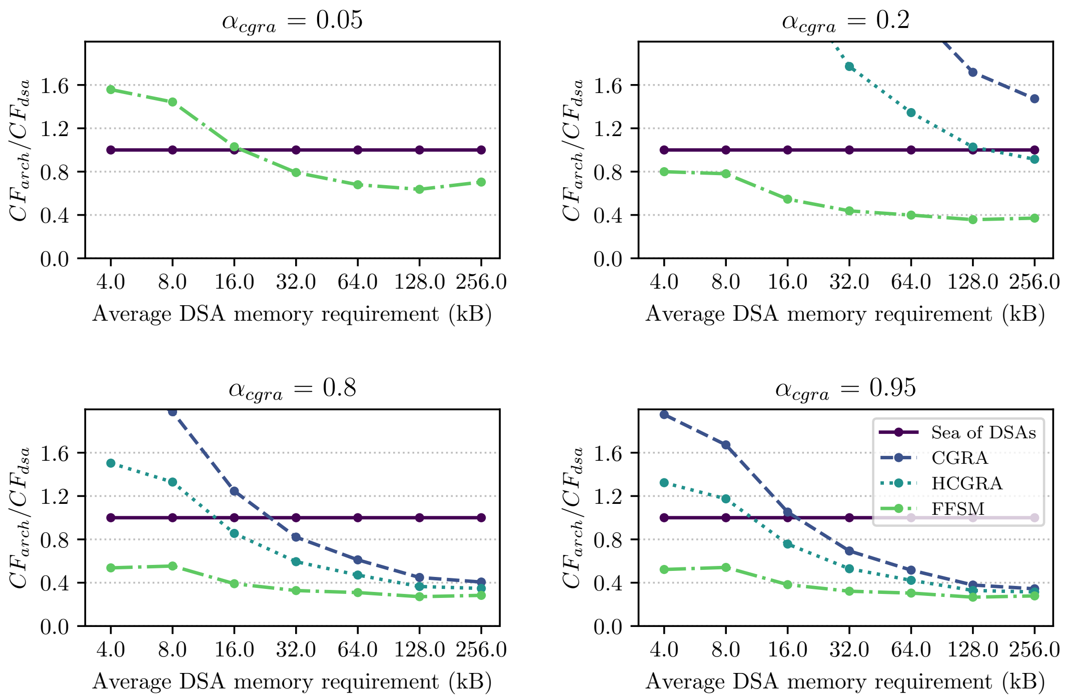


Figure 5.5: The carbon footprint of each architecture normalized to a sea of DSAs architecture, for a system with 8 accelerated kernels, at four different operating conditions and with different average DSA memory requirements. Each architecture is allocated twice the memory required for an average DSA. Lower is better.

ratio cuts the minimum number of kernels in half. Though these effects are only shown for the FFSM architecture, this ratio has the same impact for CGRA and HCGRA.

As mentioned, this observation is significant when considering how to implement these memory sharing architectures to minimize their carbon emissions. Specifically, if instantiating multiple memory sharing architectures on a single chip, then it is significantly more sustainable for kernels with *similar memory capacity requirements* to be grouped together for hardware sharing.

5.6 Carbon Footprint for a Fixed Number of Application Kernels

We now directly compare the carbon footprints of each of the four architectures assuming that the memory sharing architectures are able to be shared by 8 application kernels without a performance penalty. This approximately represents a system where the average accelerator is active less than 12.5% of the time, and which can handle the light restrictions on concurrent operation which our CGRA, HCGRA, and FFSM architectures impose (i.e. concurrency is possible but with significantly diminished performance for CGRA and HCGRA and variably diminished performance with FFSM, as discussed in Section 6).

Figure 5.5 plots the carbon footprint of each of the four architectures when shared by 8 kernels, normalized to the carbon footprint of a sea of 8 DSAs. In this scenario, at embodied dominant operating conditions and with large memory sizes, any memory sharing architecture achieves similar emissions decreases over the sea of DSAs. However, for smaller memory sizes and operationally dominant operating conditions, this is no longer the case—CGRA and HCGRA are less carbon efficient than a sea of DSAs in these cases.

Notably, an HCGRA's carbon footprint is over 38% lower than a traditional CGRA in *all* operationally dominant operating conditions and over 34% lower in embodied dominant cases with small memory sizes.

The FFSM architecture has the smallest carbon footprint of all four architectures in 90% of all scenarios. This makes this architecture the clear and obvious winner. Depending on the operating conditions, FFSM is between 1.2X and 17.1X more carbon efficient than CGRA, between 1.1X and 8.9X more carbon efficient than HCGRA, and between 0.6X and 3.6X as carbon efficient as fixed-function compute with private memory. The FFSM architecture is less sustainable than a sea of DSAs architecture only for the three smallest memory sizes in the highly operationally dominated case.

Perhaps what is most useful about these results is that there is little ambiguity about which architecture should be chosen for carbon-efficiency, even despite highly varying operating conditions and memory requirements. The recommendation is clear: the most sustainable way to architect hardware acceleration, is to share as much memory as possible, with as little overhead as possible.

Chapter 6

Discussion

These results demonstrate a clear winner when only considering area and power consumption—the first order models for a processor's carbon emissions. However, other factors may influence an architecture's carbon footprint outside of these first order models.

For example, software updates can enable CGRAs and HCGRAs to perform new kernels, which were not mapped when the system was created. This has the potential, in some cases, to prolong the lifetime of a device whose computation requirements change over time. By increasing the device's lifespan, a CGRA or HCGRA could thus be reducing embodied carbon emissions by preventing a new device from being manufactured to replace the old one. For systems where reconfigurability can decrease the carbon emissions of a system compared to fixed-function compute—by extending the system's lifetime, or otherwise—then the results throughout Section 5 demonstrate that heterogeneous CGRA represents a more sustainable choice than traditional, homogeneous CGRA.

Additionally, memory sharing architectures may see decreased data movement costs compared to a sea of DSAs. Whereas a sea of DSAs must transmit data through main memory, a higher level cache, or from peer-to-peer [46, 31], memory sharing architectures will often not have to move data at all for multiple accelerators to operate on the data.

On the other hand, a sea of DSAs has a much easier time with concurrency. In theory, our model accounts for concurrency by specifying our results as a threshold of kernels which must share hardware to reduce carbon emissions. This allows an architect to compare this threshold to the number of DSAs in their system which can share hardware without a lack of concurrency being an issue. In practice, however, fixed-function DSAs with their own memory hierarchy will have an easier time with concurrency, and won't require any management hardware or for an OS to use CPU cycles to manage the accelerators. That being said, low concurrency isn't an unreasonable expectation. In line with the requirements of dark silicon, many systems only exercise a small number of DSAs concurrently [24].

The sharing architectures do not struggle with concurrency equally. Though a reconfigurable fabric's performance is halved when shared for two kernels (each uses 2 4x4 subarrays of the 8x8 array), an FFSM architecture can handle concurrency more gracefully. Specifically, FFSM performance is only diminished when multiple accelerators try to access the same memory bank in the same cycle. This and similar situations can often be designed around (for example, by interleaving shared data into two memory banks, and having two concurrent accelerators both switch back and forth). The FFSM architecture may even see a performance gain from decreased data movement costs as discussed above.

Finally, for the CGRA and HCGRA architectures, the mappings for each kernel must be stored somewhere. In this work, we have assumed that this storage cost is minimal and that reconfiguration time for the CGRA architectures is negligible. In reality, there is likely a tradeoff between these two costs. For example, storing the mappings on chip will endure an area and thus embodied emissions cost, but will be higher performance. On the other hand, storing them off chip will lead to a large performance penalty when reconfiguring, and require more operational emissions to get the configuration to the chip, but will not cost chip area.

Chapter 7

Related Work

In line with the longstanding perception that energy consumption is the primary component of ICT's carbon emissions, hardware acceleration has long been considered a sustainability win. The sustainability of the low utilization dark silicon that fixed-function accelerators introduce into SoCs was first called into question by Brunvand et al. who demonstrated that the embodied emissions associated with dark silicon are not easily balanced by their operational emissions win over a CPU [17], results which were replicated in [28]. Dangi et al. responded to this work by proposing CGRA as a more sustainable alternative to a sea of fixed-function accelerators, since many DSAs can be replaced with a single CGRA, thereby decreasing chip area and thus embodied emissions [24].

Broadly, considering the full life cycle for more sustainable computing is a growing area of research. Though this full life cycle approach was proposed and argued for sporadically in the past [47, 48], more recent high visibility work highlighting the high carbon footprint of computing [1], and profiling where this carbon footprint stems from [12] has spurred on a significant amount of research into the field. For example, there has been a recent explosion of carbon modeling and optimization frameworks, in order to aid computer engineers in making more sustainable decisions [28, 9, 13, 49]. Sustainable datacenters have also been a focus, with work focused on reducing datacenter emissions using carbon demand response and renewable energy storage [50], better software for reducing storage emissions [51], and more carefully choosing the time between replacement of different hardware components in the datacenter [8].

Sustainable computing isn't limited only to carbon emissions. For example, recent works have considered the water footprint of computing [52], and the use of PFAs chemicals in computing [53].

Chapter 8

Conclusion

In this work we explored different architectural choices for improving the sustainability of hardware acceleration for systems on chip. Like previous work [24], we show that CGRAs can provide a sustainability win over a sea of fixed-function domain specific accelerators in cases where the accelerators have high memory requirements and/or the carbon footprint of the system is dominated by embodied emissions.

We built on this result by demonstrating that more specialized, *heterogeneous* CGRAs are more sustainable options than traditional CGRAs in all cases, often by significant margins. As discussed in Section 7, if reconfigurable compute will make a system more sustainable compared to fixed-function compute by, for example, extending the system's lifetime, then the results of this exploration demonstrate that HCGRA represents a more sustainable choice than traditional, homogeneous CGRA.

More importantly, we showed that when reconfigurability is not needed, it constitutes an unnecessary overhead. We showed that sharing memory with low-overhead hardware is always a more carbon-efficient choice than using a reconfigurable fabric like a CGRA, and is more sustainable than a sea of fixed-function DSAs each with their own memory, in 25 out of 28 cases we explored.

Finally, we highlighted that the ratio between the size of the shared pool of memory, and the average size of the memories which the pool is replacing, asymptotically limits the carbon savings achievable by a memory-sharing architecture. This effect suggests that, when putting DSAs into clusters which share memory, DSAs should be matched based on how much memory they require.

Though our insights regarding the design space between CGRA and fixed-function DSA led us to heterogeneous CGRA and to sharing only memory, these architectures do not represent the entirety of the design space. Future work may consider, for example, architectures which share both compute and memory hardware, but without relying on a reconfigurable fabric. In other words, accelerators which can execute many different kernels, but only the kernels they were designed to execute (they cannot be reconfigured arbitrarily). On the one hand, these accelerators should receive the same sustainability benefit as the FFSM architecture, with added potential for saving resources by sharing compute hardware. On the other hand, because memory is the dominant area consumer, it may not actually provide a meaningful carbon-efficiency improvement, and the difficulty in designing this more complex fixed-function architecture may not be worth it.

Bibliography

[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, no. 9, p. 100340, 2021.

[2] C.-J. Wu, R. Raghavendra, U. Gupta, B. Acun, N. Ardalani, K. Maeng, G. Chang, F. Aga, J. Huang, C. Bai, M. Gschwind, A. Gupta, M. Ott, A. Melnikov, S. Candido, D. Brooks, G. Chauhan, B. Lee, H.-H. Lee, B. Akyildiz, M. Balandat, J. Spisak, R. Jain, M. Rabbat, and K. Hazelwood, "Sustainable ai: Environmental implications, challenges and opportunities," in *Proceedings of Machine Learning and Systems* (D. Marculescu, Y. Chi, and C. Wu, eds.), vol. 4, pp. 795–813, 2022.

[3] S. R. Group, "Hyperscale data centers hit the thousand mark; total capacity is doubling every four years," 2024.

[4] S. Sudhakar, V. Sze, and S. Karaman, "Data centers on wheels: Emissions from computing onboard autonomous vehicles," *IEEE Micro*, vol. 43, no. 1, pp. 29–39, 2023.

[5] D. Patterson, "The carbon footprint of machine learning training will plateau, then shrink," Feb. 2022.

[6] TechInsights, "Techinsights launches the bom data carbon emissions modules," 2023.

[7] Apple, "Product environmental report iphone 15 pro and iphone 15 pro max," tech. rep., Apple, 2023.

[8] J. Wang, D. S. Berger, F. Kazhamiaka, C. Irvene, C. Zhang, E. Choukse, K. Frost, R. Fonseca, B. Warrier, C. Bansal, J. Stern, R. Bianchini, and A. Sriraman, "Designing cloud servers for lower carbon," in *2024 ACM/IEEE 51st Annual International Symposium on Computer Architecture (ISCA)*, pp. 452–470, 2024.

[9] D. Kline, N. Parshook, X. Ge, E. Brunvand, R. Melhem, P. K. Chrysanthis, and A. K. Jones, "Greenchip: A tool for evaluating holistic sustainability of modern computing systems," *Sustainable Computing: Informatics and Systems*, vol. 22, pp. 322–332, 2019.

[10] *IGSC '15: Proceedings of the Sixth International Green and Sustainable Computing Conference (IGSC 2015)*, IEEE, 2015.

[11] L. Boakes, M. Garcia Bardon, V. Schellekens, I.-Y. Liu, B. Vanhouche, G. Mirabelli, F. Sebaai, L. Van Winckel, E. Gallagher, C. Rolin, and L. Ragnarsson, "Cradle-to-gate life cycle assessment of cmos logic technologies," in *2023 International Electron Devices Meeting (IEDM)*, pp. 1–4, 2023.

[12] U. Gupta, Y. G. Kim, S. Lee, J. Tse, H.-H. S. Lee, G.-Y. Wei, D. Brooks, and C.-J. Wu, "Chasing carbon: The elusive environmental footprint of computing," *IEEE Micro*, vol. 42, p. 37–47, July 2022.

[13] U. Gupta, M. Elgamal, G. Hills, G.-Y. Wei, H.-H. S. Lee, D. Brooks, and C.-J. Wu, "Act: designing sustainable computer systems with an architectural carbon modeling tool," in *Proceedings of the 49th Annual International Symposium on Computer Architecture*, ISCA '22, (New York, NY, USA), p. 784–799, Association for Computing Machinery, 2022.

[14] L. Eeckhout, "A first-order model to assess computer architecture sustainability," *IEEE Computer Architecture Letters*, vol. 21, no. 2, pp. 137–140, 2022.

[15] L. Eeckhout, "Kaya for computer architects: Toward sustainable computer systems," *IEEE Micro*, vol. 43, no. 1, pp. 9–18, 2023.

[16] V. J. Reddi, "Mobile socs: The wild west of domain specific architectures," 2018.

[17] E. Brunvand, D. Kline, and A. K. Jones, "Dark silicon considered harmful: A case for truly green computing," in *2018 Ninth International Green and Sustainable Computing Conference (IGSC)*, pp. 1–8, 2018.

[18] H. Esmaeilzadeh, E. Blem, R. S. Amant, K. Sankaralingam, and D. Burger, "Dark silicon and the end of multicore scaling," in *2011 38th Annual International Symposium on Computer Architecture (ISCA)*, pp. 365–376, 2011.

[19] Qualcomm, "Snapdragon® 8 gen 3 mobile platform." `https://docs.qualcomm.com/bundle/publicresource/87-71408-1_REV_G_Snapdragon_8_gen_3_Mobile_Platform_Product_Brief.pdf`, 2023.

[20] Y. S. Shao, B. Reagan, G.-Y. Wei, and D. Brooks, "Retrospective: Aladdin: a pre-rtl, power-performance accelerator simulator enabling large design space exploration of customized architectures," in *ISCA@50 25-Year Retrospective: 1996-2020* (J. F. Martínez and L. K. John, eds.), ACM SIGARCH and IEEE TCCA, 2023.

[21] D. Wijerathne, Z. Li, M. Karunaratne, T. Mitra, and L.-S. Peh, "Morpher: An open-source integrated compilation and simulation framework for cgra," in *2022 Workshop on Open-Source EDA Technology (WOSET), Article No. 12*, p. 5, 2022.

[22] Y. Qiu, Y. Cao, Y. Dai, W. Yin, and L. Wang, "Tram: An open-source template-based reconfigurable architecture modeling framework," in *2022 32nd International Conference on Field-Programmable Logic and Applications (FPL)*, pp. 61–69, 2022.

[23] J. Anderson, R. Beidas, V. Chacko, H. Hsiao, X. Ling, O. Ragheb, X. Wang, and T. Yu, "Cgra-me: An open-source framework for cgra architecture and cad research : (invited paper)," in *2021 IEEE 32nd International Conference on Application-specific Systems, Architectures and Processors (ASAP)*, pp. 156–162, 2021.

[24] P. Dangi, T. K. Bandara, S. Sheikhpour, T. Mitra, and L. Eeckhout, "Sustainable hardware specialization," in *Proceedings of the 43rd IEEE/ACM International Conference on Computer-Aided Design*, ICCAD '24, (New York, NY, USA), Association for Computing Machinery, 2024.

[25] T. K. Bandara, D. Wijerathne, T. Mitra, and L.-S. Peh, "Revamp: a systematic framework for heterogeneous cgra realization," in *Proceedings of the 27th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*, ASPLOS '22, (New York, NY, USA), p. 918–932, Association for Computing Machinery, 2022.

[26] J. Li, Y. Qiu, G. Zhu, Q. Zhu, W. Yin, and L. Wang, "Thram: A template-based heterogeneous cgra modeling framework supporting fast dse," in *2023 IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1–5, 2023.

[27] Y. Dai, J. Li, Q. Zhu, Y. Qiu, Y. Hu, W. Yin, and L. Wang, "Heta: A heterogeneous temporal cgra modeling and design space exploration via bayesian optimization," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 32, no. 3, pp. 505–518, 2024.

[28] L. Eeckhout, "Focal: A first-order carbon model to assess processor sustainability," in *Proceedings of the 29th ACM International Conference on Architectural Support for Programming*

Languages and Operating Systems, Volume 2, ASPLOS '24, (New York, NY, USA), p. 401–415, Association for Computing Machinery, 2024.

[29] M. Karunaratne, A. K. Mohite, T. Mitra, and L.-S. Peh, "Hycube: A cgra with reconfigurable single-cycle multi-hop interconnect," in *2017 54th ACM/EDAC/IEEE Design Automation Conference (DAC)*, pp. 1–6, 2017.

[30] M. J. Lyons, M. Hempstead, G.-Y. Wei, and D. Brooks, "The accelerator store: A shared memory framework for accelerator-based systems," *ACM Trans. Archit. Code Optim.*, vol. 8, Jan. 2012.

[31] D. Giri, P. Mantovani, and L. P. Carloni, "Accelerators and coherence: An soc perspective," *IEEE Micro*, vol. 38, no. 6, pp. 36–45, 2018.

[32] S. Grauer-Gray, L. Xu, R. Searles, S. Ayalasomayajula, and J. Cavazos, "Auto-tuning a high-level language targeted to gpu codes," in *2012 Innovative Parallel Computing (InPar)*, pp. 1–10, 2012.

[33] S. Bakhurin, R. C. Schmidt, and S. Royz, "Libdspl2," 2018.

[34] J. M. de la Rosa, "Iir_filters," 2018.

[35] J. Wilczek, "fir-simd," 2022.

[36] M. Kohn, "simd_examples," 2018.

[37] L. Project, "Libtomcrypt," 2010.

[38] Kokke, "tiny-aes-c," 2012.

[39] T. Roberts, M. Slaney, D. P. Bouras, and E. Condes, "fix_fft," 2017.

[40] P. Ammon, "Libdivide," 2010. Accessed 2025.

[41] S. Barrett, "stb," 2014.

[42] J. E. Stine, I. Castellanos, M. Wood, J. Henson, F. Love, W. R. Davis, P. D. Franzon, M. Bucher, S. Basavarajaiah, J. Oh, and R. Jenkal, "Freepdk: An open-source variation-aware design kit," in *2007 IEEE International Conference on Microelectronic Systems Education (MSE'07)*, pp. 173–174, 2007.

[43] S. I. I. (SI2), "Open cell and free pdk libraries."

[44] M. R. Guthaus, J. E. Stine, S. Ataei, B. Chen, B. Wu, and M. Sarwar, "Openram: An open-source memory compiler," in *2016 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)*, pp. 1–6, 2016.

[45] K. Prabhu, R. M. Radway, J. Yu, K. Bartolone, M. Giordano, F. Peddinghaus, Y. Urman, W.-S. Khwa, Y.-D. Chih, M.-F. Chang, S. Mitra, and P. Raina, "Minotaur: A posit-based 0.42–0.50-tops/w edge transformer inference and training accelerator," *IEEE Journal of Solid-State Circuits*, pp. 1–13, 2025.

[46] K.-L. Chiu, D. Giri, L. Piccolboni, and L. P. Carloni, "An analysis of accelerator data-transfer modes in noc-based soc architectures," in *2023 IEEE High Performance Extreme Computing Conference (HPEC)*, pp. 1–7, 2023.

[47] S. B. Boyd, A. Horvath, and D. Dornfeld, "Life-cycle energy demand and global warming potential of computational logic," *Environmental Science & Technology*, vol. 43, no. 19, pp. 7303–7309, 2009. PMID: 19848138.

[48] A. K. Jones, L. Liao, W. O. Collinge, H. Xu, L. A. Schaefer, A. E. Landis, and M. M. Bilec, "Green computing: A life cycle perspective," in *2013 International Green Computing Conference Proceedings*, pp. 1–6, 2013.

[49] M. Elgamal, D. Carmean, E. Ansari, O. Zed, R. Peri, S. Manne, U. Gupta, G.-Y. Wei, D. Brooks, G. Hills, and C.-J. Wu, "Cordoba: Carbon-efficient design optimization framework for computing systems," in *2025 IEEE International Symposium on High-Performance Computer Architecture (HPCA)*, 2025.

[50] B. Acun, B. Lee, F. Kazhamiaka, K. Maeng, U. Gupta, M. Chakkaravarthy, D. Brooks, and C.-J. Wu, "Carbon explorer: A holistic framework for designing carbon aware datacenters," in *Proceedings of the 28th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2*, ASPLOS 2023, (New York, NY, USA), p. 118–132, Association for Computing Machinery, 2023.

[51] S. McAllister, Y. S. Wang, B. Berg, D. S. Berger, G. Amvrosiadis, N. Beckmann, and G. R. Ganger, "FairyWREN: A sustainable cache for emerging Write-Read-Erase flash interfaces," in *18th USENIX Symposium on Operating Systems Design and Implementation (OSDI 24)*, (Santa Clara, CA), pp. 745–764, USENIX Association, July 2024.

[52] Y. Jiang, R. B. Roy, R. Kanakagiri, and D. Tiwari, "Waterwise: Co-optimizing carbon- and water-footprint toward environmentally sustainable cloud computing," in *Proceedings of the 30th ACM SIGPLAN Annual Symposium on Principles and Practice of Parallel Programming*, PPoPP '25, (New York, NY, USA), p. 297–311, Association for Computing Machinery, 2025.

[53] M. Elgamal, A. Mahmoud, G.-Y. Wei, D. Brooks, and G. Hills, "Pfasware: Quantifying the environmental impact of per- and polyfluoroalkyl substances (pfas) in computing systems," in *2025 Design, Automation and Test in Europe (DATE)*, 2025.

Appendix A

Engineering and Industrial Standards

The independent project described in this thesis incorporated the following engineering and industrial standards:

- System Verilog was used for modeling hardware for memory sharing
- The C programming language was used for modeling each benchmark kernel
- SI units were used throughout the thesis
- Siemens' Catapult High Level Synthesis was used for instantiating fixed function DSAs
- Synopsys Design Compiler was used for synthesis
- Excel and Google Sheets were used for analyzing data
- Python and Matplotlib were used for plotting data