

Article

Feasibility-Aware Design-Space Exploration of Transparent Coarse-Grained Reconfigurable Architectures

Thiago R. B. S. Soares ^{1,*} and Ivan S. Silva ^{2,†}¹ Direção de Ensino, Instituto Federal do Piauí (IFPI), Teresina 64018-000, PI, Brazil² Departamento de Ciência da Computação, Universidade Federal do Piauí (UFPI), Teresina 64049-550, PI, Brazil; ivan@ufpi.edu.br

* Correspondence: thiago.soares@ifpi.edu.br

† These authors contributed equally to this work.

Abstract

Coarse-Grained Reconfigurable Architectures (CGRAs) execute compute-intensive kernels on a reconfigurable processing mesh. Transparent CGRAs extend this model by generating configurations at runtime and storing them in a dedicated cache, removing compiler dependence and enabling adaptive behavior. Although prior work has explored mapping strategies and mesh scaling, the feasibility of the configuration cache remains unaddressed, as it is commonly treated as a generic storage block. This paper presents a feasibility study of configuration cache organizations and a design-space exploration of Transparent CGRAs, introducing a parameterized cache geometry model that relates cache parameters to the processing mesh and configuration structure. The model enables realistic estimates of area, latency, and energy at the digital system level and is applied to three Transparent CGRAs from the literature and five additional designs covering a wide range of spatial and temporal organizations. The results show that mesh scaling must be balanced with cache feasibility: wide I/O paths and large configurations lead to impractical caches, whereas well-proportioned meshes achieve competitive performance with modest overheads. Under the proposed exploration, selected expanded meshes outperform a two-issue out-of-order processor by up to 1.4× while increasing area by only 14.8% and energy by 2%. These findings demonstrate that Transparent CGRAs are viable, but their scalability depends on a realistic configuration cache design. The proposed parameterized cache model provides a structured and reproducible basis for analyzing transparency overheads and guiding future CGRA designs.

Keywords: coarse-grained reconfigurable architecture; transparent CGRAs; configuration cache; feasibility analysis; design-space exploration



Academic Editors: Yanpeng Shi, Yafei Ning and Jie Su

Received: 18 December 2025

Revised: 4 January 2026

Accepted: 6 January 2026

Published: 10 January 2026

Copyright: © 2026 by the authors.

Licensee MDPI, Basel, Switzerland.

This article is an open access article distributed under the terms and conditions of the [Creative Commons Attribution \(CC BY\)](https://creativecommons.org/licenses/by/4.0/) license.

1. Introduction

Rapid technological and social development continues to demand fast and energy-efficient computing systems. Historically, these demands were addressed through transistor scaling and frequency improvements; however, the slowdown of Moore's law and Dennard's scaling has driven the search for architectural alternatives [1–3].

Among these alternatives, Coarse-Grained Reconfigurable Architectures (CGRAs) have emerged as a promising option, delivering performance gains at low energy cost across multiple application domains [4–7]. A CGRA operates as an accelerator coupled to a host processor, exploiting instruction-level parallelism through a reconfigurable mesh

of processing elements (PEs). However, conventional CGRAs still face programmability challenges due to compilation overheads, motivating the development of execution models that reduce compiler dependence and enable runtime adaptability [7,8].

Within this context, Transparent CGRAs were proposed to automate configuration generation at runtime [9–13]. They integrate a configuration-generation block, typically a binary translator [11], that maps instructions onto the processing mesh and stores them in a local configuration cache for reuse. As a result, Transparent CGRAs eliminate dependence on external compilation tools and enable adaptive behavior based on workload characteristics [14].

Several Transparent CGRAs have explored mesh scaling and mapping strategies through design-space exploration (DSE) [9,11–13,15]. However, the physical feasibility and architectural implications of the configuration cache—a fundamental component for transparency—remain largely unexplored. Recent CGRA research continues to investigate aggressive DSE and architectural trade-offs in emerging domains, including system-on-chip accelerators for sparse machine learning [16], processing-in-memory optimized CGRAs [17], and radiation-tolerant designs for space applications [18]. While these works demonstrate that CGRAs remain an active research topic, they rely on static, compiler-driven, or host-managed configuration mechanisms and do not address the feasibility of configuration storage and reuse in transparent execution models. The following subsection reviews prior Transparent CGRA proposals, with particular attention paid to how configuration caches have been characterized.

Related Work

One of the pioneering Transparent CGRAs is the Configurable Compute Accelerator (CCA) [9,10], which aims to improve performance through application-specific instruction-set extensions (ISEs) derived from data-flow graphs (DFGs). The authors integrated the CCA into the execute stage of the host processor, while a control generator unit generates the configuration bits. The CCA employs a tree-based mesh topology and presents design limitations, including the lack of speculative execution, the inability to issue memory operations, and a restricted I/O context. To support configuration execution, the branch target buffer (BTB) was expanded to store configuration operands, whereas the configuration cache stores the control bits. The BTB is four-way associative with 512 entries, and each configuration contains 172 control bits.

The Dynamic Instruction Merging (DIM) [11] architecture employs a four-stage binary translation pipeline to generate its configurations. It uses a greedy mapping algorithm that speculatively merges up to three sequential basic blocks into a single configuration to improve performance. DIM features a high-capacity, 2D processing mesh, and the authors conducted a DSE by varying both the mesh size and the configuration cache capacity to evaluate their impact on system performance. The configuration cache is four-way associative, with each configuration occupying a single cache line and requiring approximately 400 bytes of data.

The TransRec [19] is derived from DIM and uses the same binary translator to generate its configurations. Unlike DIM, TransRec adopts a smaller and more constrained processing mesh without dedicated multiplication units, prioritizing low area overhead and reduced energy consumption over peak processing capability. Each configuration occupies 256 bytes of data, and the configuration cache comprises 128 lines, totaling 32 KB, which is comparable in size to a typical instruction cache.

A THin rEcoNfigurable Array (ATHENA [12]) also features a regular 2D topology and generates configurations using a greedy mapping algorithm embedded in its five-stage binary translator. The authors designed this architecture to be tightly integrated with

superscalar processors while incorporating several optimization techniques to improve performance, including basic-block speculation, configuration prefetching, and the write-back of intermediate results. The authors explored different mesh sizes to evaluate their impact on system performance, area, and energy consumption, considering a four-way associative configuration cache with 128 entries and a Least Frequently Used (LFU) replacement policy.

The Microarchitecture Extensions for Spatial Architecture Generation (MESA) [13] is a transparent CGRA framework that extends the conventional 2D mesh topology with a half-ring network interconnecting multiple PE slices. MESA integrates runtime loop detection with a local hardware mapping pipeline and was evaluated in various grid sizes with selective floating-point support. However, unlike prior Transparent CGRAs, the work does not provide architectural details about its configuration cache, including capacity, associativity, or configuration size, as the cache is treated only as a lightweight buffer for reusing previously generated configurations.

Most studies have treated the configuration cache in a simplified manner, typically reporting only its capacity or loosely correlating it with the instruction cache while neglecting parameters such as word size, block size, and I/O bandwidth. Even recent CGRA architectures that perform detailed DSE and physical evaluation focus on architectural optimization, domain-specific acceleration, or deployment constraints [16–18], without analyzing the internal organization, physical feasibility, or architectural impact of configuration caches in Transparent CGRAs.

To address this gap, this work conducts a feasibility study of configuration cache organizations and a comprehensive DSE of Transparent CGRAs. We introduce a parameterized cache geometry model not as an end in itself, but as a systematic tool for assessing viability, enabling accurate simulation of cache structures under realistic constraints. The model establishes a clear relationship between the mesh dimensions, the configuration size, and the internal organization of the cache for storage and access, enabling accurate estimations of area and energy consumption and clarifying how mesh scaling impacts cache overheads.

Using this model, this work evaluates the DIM [11], TransRec [19], and ATHENA [12] CGRAs, along with five additional mesh designs that span a broad spectrum of spatial parallelism and temporal depth. We coupled each CGRA to a two-issue out-of-order processor to form a complete heterogeneous system. This simulation setup allowed us to quantify how variations in the processing mesh affect configuration cache behavior and overall system efficiency, revealing how cache organization and mesh size jointly determine scalability and the best performance–cost trade-offs.

Under the proposed cache geometry, the expanded mesh designs achieved up to 1.4× higher performance than the baseline processor, while maintaining an area overhead of 14.8% and limiting the energy increase to approximately 2%. These results indicate that Transparent CGRAs are practical, with performance gains that significantly outweigh the modest area and energy costs introduced by scaling the configuration cache under realistic cache parameters.

The main contributions of this work are as follows:

- A parameterizable cache geometry model that relates the CGRA’s processing mesh, the configuration structure, and realistic cache parameters, enabling accurate feasibility assessments and providing a detailed characterization of how mesh dimensions influence cache scalability.
- A replicable and systematic simulation methodology that quantifies transparency overheads through detailed evaluation of performance, area, and energy across multiple CGRA organizations.

- A comprehensive DSE of baseline and newly derived meshes, exposing scalability trends and identifying balanced and practical architectural configurations for Transparent CGRAs.

Within this context, we develop a parameterized configuration cache model that relates cache parameters to the CGRA's processing capacity and standardizes the description of configuration cache design, enabling feasibility assessments, structured design-space exploration, and fair comparisons across different Transparent CGRAs. In addition, it is fully compatible with the CACTI tool, allowing for the accurate simulation and estimation of cache area, latency, and energy consumption.

The remainder of this paper is organized as follows. Section 2 describes the proposed cache geometry and its relationship to the CGRA processing mesh. Section 3 describes the ATHENA model used for this work. Section 4 presents the experimental setup used in our simulations. Section 5 shows and discusses the obtained results. Finally, Section 6 summarizes our findings and outlines directions for future work.

2. Parameterized Configuration Cache Model

The proposed cache geometry provides a systematic foundation for evaluating feasibility and scalability in Transparent CGRAs. To achieve this, it is essential to understand how cache parameters affect key metrics such as area, energy consumption, and access latency. To this end, we conduct a sensitivity analysis exploring cache capacity, block size, and I/O bus width to identify which factors most strongly influence cache viability. All simulations are performed using the CACTI tool configured for a 45 nm technology node.

In the first experiment, we fixed the block size and bus width at 64 bytes and varied the cache capacity from 32 to 512 KB using an 8-way set-associative organization. The simulations show that increasing capacity leads to a substantial increase in static power, from 42 to 614 mW (up to 13×), along with a threefold increase in area, as expected from the larger total storage. In the second experiment, we varied the block size from 64 to 512 bytes while keeping the cache at 64 KB with a 64-byte bus width. In this scenario, the simulations show negligible variation: access latency remains constant, area is virtually unchanged, and static power decreases slightly. These results indicate that cache capacity is the dominant factor influencing feasibility, whereas block size has minimal impact.

The analysis shows that the I/O bus width is the most critical parameter for assessing cache feasibility. For this experiment, we simulated a configuration cache designed for DIM (400-byte configurations) [11]. To evaluate the design implications, we modeled a 4-way set-associative cache capable of storing 128 configurations, with each memory block corresponding to a single 512-byte configuration (64 KB total capacity). Figure 1a–c illustrates the impact of varying the I/O bus width on access latency, dynamic energy per read, and total area, respectively.

The graph shows that attempting to read an entire configuration in a single access (BUS = 512 bytes) would result in a slow, large, and power-hungry cache. For comparison, the graph includes data from an 8-way, 32 KB instruction cache with a 64-byte I/O bus. For example, if the CGRA attempted to read half of a configuration per access (BUS = 256 bytes), the configuration cache would still be approximately eight times larger than the instruction cache and would exhibit nearly twice its static power. These simulations indicate that CGRAs with larger configurations must fetch them in smaller portions across multiple clock cycles; otherwise, the configuration cache would become impractical. We refer to these portions as configuration words (CWs).

To assess feasibility and to keep cache access practical under realistic I/O widths, we adopt configuration words that correspond to the amount of work performed by the CGRA in a single clock cycle. Thus, the CGRA can execute one word while fetching the next,

forming a fetch–execute pipeline. Moreover, CGRAs with regular processing meshes, such as those analyzed in this work, can further exploit this approach by employing smaller physical meshes that execute operations sequentially through CWs. This strategy allows the CGRA to effectively scale its processing capacity over time without modifying the physical array by scheduling its operations across CWs. Therefore, we conceptually divide the CGRA’s processing mesh into physical and logical parts (as shown in Figure 2). The physical level represents the mesh’s actual hardware resources, whereas the logical level defines the temporal scheduling of CWs.

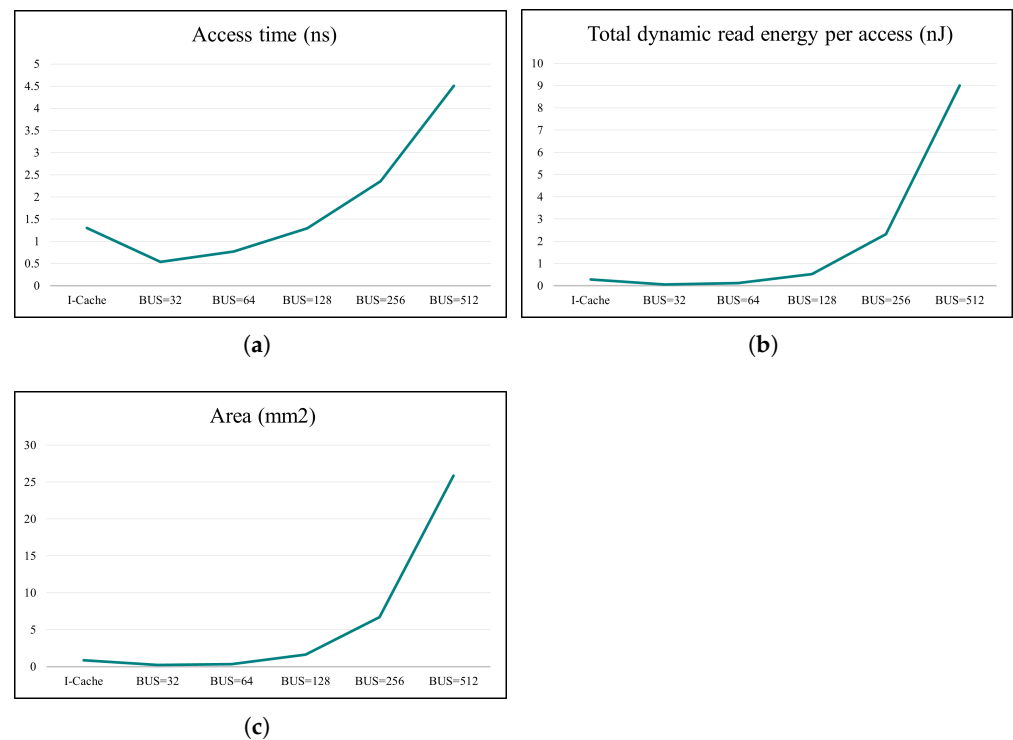


Figure 1. Impact of I/O bus width on configuration cache: (a) latency; (b) dynamic energy per read; (c) total area.

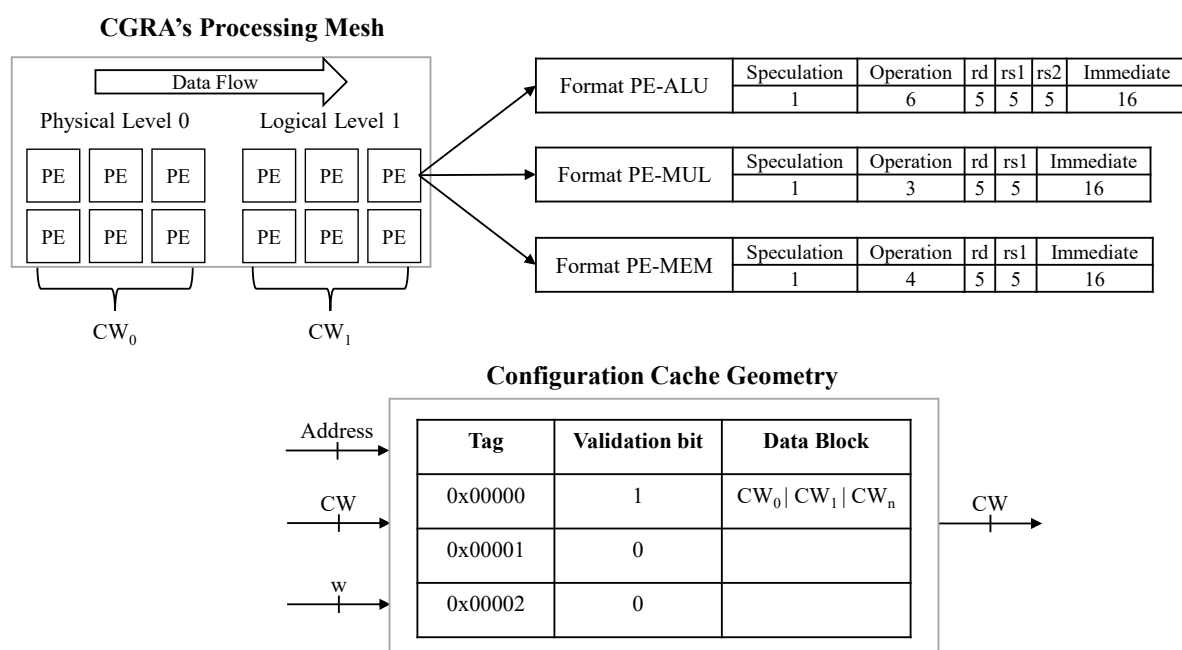


Figure 2. Configuration cache geometry and its relation to the CGRA’s processing mesh.

In our experimental setup, we consider the RISC-V ISA and processing meshes composed of three types of Processing Elements (PEs) for integer operations: PE-MEM, responsible for memory instructions; PE-MUL, for multiplications; and PE-ALU, which executes the remaining arithmetic, logic, and branch operations supported by the CGRA. Each PE type has a specific configuration bit format and length, as defined in Figure 2. By determining how each PE contributes to the total configuration size, variations in the CGRA's processing capacity directly influence the configuration size and, consequently, the cache parameters.

To ensure the feasibility of the cache simulation, we used power-of-two values for both the word and block sizes. This choice is not architectural but methodological, ensuring CACTI compatibility and supporting a uniform exploration of feasible cache organizations. Caches with parameters that deviate from this base-two alignment tend to produce irregular layouts and often cannot be simulated in CACTI. However, adopting a power-of-two alignment may introduce intra-word and intra-block inefficiencies due to unused bits. To mitigate this issue, designers can adjust the CGRA's processing capacity by varying the number of PEs and logical levels. It is also possible to refine the configuration bit length and the instruction encoding of each PE through an application-guided analysis, reducing the number of supported operations and thereby improving cache utilization.

A configuration word size is determined by the number and type of PEs in the physical level (level 0) of the mesh. For example, TransRec features four ALUs and one memory PE per level, across 12 levels, resulting in 183 bits per configuration word. In this case, the CW is aligned to 256 bits (32 bytes), leading to approximately 28% intra-word overhead. Similarly, each configuration occupies 384 bytes (12 levels, each requiring 32 bytes), which rounds up to 512 bytes when aligned to the next power of two, producing an intra-block overhead of 25%. This result shows that it is possible to increase the CGRA's processing capacity by up to 4 logical levels without modifying either the processing mesh or the cache, since the configuration size would remain 512 bytes.

ATHENA was selected as the primary CGRA model for the experiments and analyses. ATHENA is a small-scale CGRA with a straightforward and reproducible greedy mapping algorithm. It is designed to accelerate superscalar processors and already incorporates performance-oriented optimizations, including temporal scheduling of operations through configuration words.

To obtain a broader and more insightful evaluation, we also included the DIM and TransRec architectures, as these three meshes represent distinct and complementary design points. DIM embodies a spatially oriented architecture with high parallelism within each level, enabling many operations to execute simultaneously. In contrast, ATHENA adopts an opposite strategy, relying on deep temporal scaling to schedule operations across multiple logical levels while keeping the physical mesh relatively small. TransRec represents a third design point, featuring few PEs per level, shallow temporal depth, and no multiplication units, resulting in a more constrained processing capability.

These contrasting characteristics allow us to observe how parallelism, temporal depth, and functional-unit availability influence performance, area, energy, and the physical feasibility of configuration cache organizations. Therefore, evaluating these CGRAs provides a comprehensive understanding of how mesh topology affects the system's overall efficiency and transparency behavior.

3. The ATHENA CGRA

Figure 3 illustrates the internal organization of ATHENA. The system comprises a Binary Translator (1), a Configuration Cache (2), a Controller (3), and the Processing Mesh (4). The base processor integrates the CGRA through four primary interfaces: (a) reading com-

mitted instructions, (b) reading and writing the Program Counter (PC), (c) accessing the Register File (RF) for both reads and writes, and (d) controlling the processor's execution flow via a Flush and Stall signal.

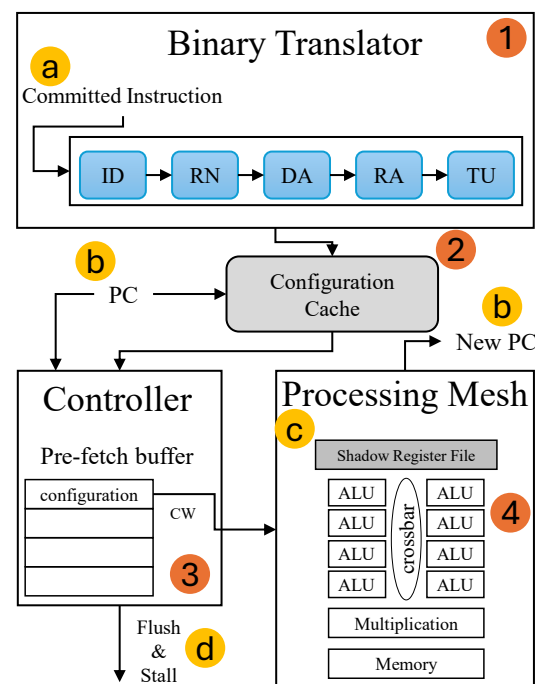


Figure 3. Visual organization of ATHENA.

The Binary Translator (1) generates configurations using a mapping algorithm embedded in its five-stage pipeline. The Instruction Decode (ID) stage decodes the instruction recently committed by the processor, and the Register Rename (RN) stage performs register renaming. Next, the Dependency Analysis (DA) stage identifies the last row in the mesh that produced the instruction's operands and informs the Resource Allocation (RA) stage of the row where it should begin searching for an available PE. The RA stage then maps the instruction to the first available PE capable of executing it in the identified row or in subsequent rows. Finally, the Table Update (TU) stage consolidates modifications to the tables from the previous stages.

The mapping process begins at the start of a basic block and can be interrupted under three conditions: (i) when the basic-block speculation limit is reached; (ii) when an invalid instruction is encountered; and (iii) when no suitable hardware resources are available. At the end of the mapping process, the resulting configuration is indexed by the memory address of the first mapped instruction and stored in the Configuration Cache (2).

The Controller (3) accesses the configuration cache using the PC of the instruction recently fetched by the processor. In the case of a hit, the configuration is read and placed in the prefetch buffer. This optimization anticipates loading configurations associated with in-flight instructions into the processor, thereby reducing CGRA reconfiguration latency. The controller triggers the execution of a configuration when the processor commits its identifying instruction. At that moment, the Flush and Stall signal (d) is sent to the processor, forcing it to discard all active instructions and enter a stall state while the CGRA executes the configuration. During configuration execution, memory operations may introduce additional stalls if cache misses occur. Once execution completes, the CGRA updates the processor's register file and PC.

The Processing Mesh (4) of ATHENA includes a live copy of the processor's register file (Shadow Register File), a 4×2 ALU mesh, a multiplier, and a memory access PE.

The ALUs are simple units which allows the execution of two sequential operations in a single clock cycle. The multiplier has a latency of three cycles, while load and store instructions incur latencies of two and one cycles, respectively, in a cache hit. The PEs are connected via a crossbar, enabling results to be forwarded among PEs according to the mapping algorithm.

A more detailed description of ATHENA's microarchitecture, mapping pipeline, and optimization mechanisms can be found in the original publications [12,20].

4. Materials and Methods

We used gem5 v23.1 [21], CACTI v7.0 [22], and McPAT v1.3 [23] to analyze the performance, area, and energy of the evaluated CGRAs, targeting a 45 nm technology node. The host processor is a modified version of the gem5 O3 model, configured as a two-issue design. All system parts, including the host processor and the CGRA components, were configured to operate at the same clock frequency of 1 GHz. We selected a subset of 15 applications from MiBench [24] to evaluate the systems—bitcount, blowfish_d, blowfish_e, crc32, dijkstra, fft, fft_i, jpeg_c, jpeg_d, qsort, sha, stringsearch, susan_c, susan_e, susan_s. All applications were compiled using the RISC-V GNU Compiler Toolchain with the -O3 optimization flag. Energy values were computed offline using CACTI and McPAT, accounting for both dynamic and static components, and aggregated per benchmark according to each benchmark's execution time.

Table 1 lists the parameters of the simulated systems. We simulated the CGRA's binary translator and processing mesh as two distinct processors in McPAT, and the configuration cache as an independent component in CACTI. The processor that models the binary translator is a five-stage pipeline that fetches, decodes, renames, and dispatches instructions. The processor that models the processing mesh, in turn, is configured according to the CGRA's processing capacity and operates as a two-stage pipeline that fetches and executes CWs. By simulating each CGRA subcomponent independently, we can precisely identify the area and energy costs associated with each part of the architecture, deepening our understanding of the transparency overhead in Transparent CGRAs.

Table 1. System description for McPat and CACTI simulations.

Component	Parameters
Host Processor	16-byte fetch buffer, 8-entry memory queue, 16-entry IQ, 48-entry ROB, 2 integer ALUs, 128 physical integer and FP registers each. 8-way 32 KB L1 I-Cache and D-Cache, and 4-way 512 KB L2
Binary Translator	5-stage pipeline with decode, rename, and issue units
Processing Mesh	2-stage pipeline with custom execution stage
Configuration Cache	128 or 256 entry 8-way with LRU replacement policy

Table 2 presents the processing capacity of the evaluated CGRAs. To investigate how spatial parallelism and temporal depth affect CGRA metrics, we derived five models from ATHENA by varying the number and organization of ALUs as well as the total number of logical levels. To maintain a consistent design space, we prioritized regular two-dimensional meshes and avoided irregular organizations with 5 or 7 ALUs per level. ATHENA-8 uses a 4×2 ALU mesh, matching the original design but with varying processing depths, and serves as our baseline. In ATHENA-4, we reduce spatial parallelism to a 2×2 organization, while ATHENA-3 further decreases the available ALUs to a 1×3 layout.

Conversely, ATHENA-6 and ATHENA-9 increase intra-level parallelism through 2×3 and 3×3 organizations, respectively.

Table 2. Level description of the CGRAs.

CGRA	ALUs	Memory	Multiplication	Level Count
ATHENA	4×2	1	1	80
DIM	8×3	1	1	8
TransRec	2×2	1	0	12
ATHENA-8	4×2	1	1	16-32-64
ATHENA-4	2×2	1	1	16-32-64
ATHENA-3	1×3	1	1	16-32-64
ATHENA-6	2×3	1	1	16-32-64
ATHENA-9	3×3	1	1	16-32-64

The ATHENA-3 and ATHENA-4 models generate 32-byte configuration words with 32% and 17% intra-word waste, respectively, while ATHENA-6, ATHENA-8, and ATHENA-9 produce 64-byte configuration words, resulting in 44%, 28%, and 21% intra-word waste. We simulated each model with 16, 32, and 64 levels, as these configurations eliminate intra-block waste in the configuration cache. Configuration words of 32 and 64 bytes ensure that the resulting cache remains feasible, with access latency on the order of 1 ns. From the cache perspective, differences in processing capacity among models will primarily reflect variations in size and energy consumption. All models can generate configurations containing up to ten basic blocks and support a subset of 54 integer RISC-V instructions, including multiplication, load/store, branch/jump, and basic ALU operations.

5. Results

The host processor serves as the baseline for assessing both the performance improvements and the architectural overhead introduced by the CGRAs. It occupies 21.44 mm^2 , of which 7.26 mm^2 corresponds to the core and 14.18 mm^2 to the L2 cache. The processor achieves an aggregate instructions per cycle (IPC) of 1.47 and consumes 0.88 J when considering all evaluated benchmarks, with 97.25% of this energy attributed to the core and 2.75% to the L2. These values provide the reference point against which the CGRAs' area, energy, and performance variations are measured. In the following sections, we first analyze the baseline CGRAs to understand how each design interacts with the proposed geometry and the resulting system metrics. We then evaluate the new mesh models to assess how scaling processing capacity influences transparency efficiency and overall system behavior. All results presented in the following sections were obtained through simulations conducted specifically for this work, using the same benchmark set and identical experimental conditions for all evaluated systems.

5.1. Baseline CGRAs Results

Table 3 summarizes the geometric mean speedup, area overhead, and energy variation of the complete system formed by coupling the host processor with each baseline CGRA. Area variation is computed as the relative difference between the baseline processor area and the total system area, obtained by summing the processor and all CGRA components' areas. Energy variation represents the relative difference between the aggregated energy consumed by all benchmarks on the baseline processor and on the complete system.

Table 3. Performance, area, and energy variation for ATHENA, DIM, and TransRec using a 128-entry and 256-entry configuration caches.

Cache Size	CGRA Model	Speedup	Area (%)	Energy (%)
128-entry	ATHENA	1.36	24.79	20.588
	DIM	1.145	24.4	37.122
	TransRec	1.167	3.89	−5.81
256-entry	ATHENA	1.393	46.55	43.926
	DIM	1.155	26.18	40.425
	TransRec	1.177	5.45	−4.647

Under the 128-entry configuration cache, all models improved the host processor's performance, with ATHENA achieving the highest speedup of 1.36×. ATHENA's high performance results from its large processing capacity, enabled by scheduling up to 80 CWs per configuration. However, simulation metadata revealed that 67% of mappings were terminated upon reaching the basic block speculation limit, while only 0.23% were terminated due to resource unavailability. This data indicates an imbalance between the mesh's processing capacity and the speculation limit, which could be addressed by either increasing the speculation limit or decreasing the mesh's level count.

ATHENA increases the host processor's area by approximately 25%, with the binary translator, processing mesh, and configuration cache accounting for 0.88%, 2.26%, and 16.73% of the total area, respectively. Considering the complete system (host processor and CGRA components), total energy consumption increases by 20.56%, primarily due to the processing mesh and configuration cache. Although application acceleration reduces the host processor's energy consumption by 28%, this reduction is outweighed by the additional energy consumed by the CGRA, resulting in a net increase in system-level energy. In the overall system energy breakdown, the binary translator, processing mesh, and configuration cache account for 3.36%, 10.39%, and 26.37% of the total energy consumption, respectively.

DIM achieved the lowest performance among the baseline architectures, with an area increase comparable to ATHENA (24.4%) and the highest energy consumption (37.1%). Despite its 24 ALU per-level processing mesh, its limited depth constrains overall performance and accounts for 4.73% of the system's area and 22.5% of its energy consumption. DIM's mesh also requires a proportionally larger configuration word, resulting in a large cache that accounts for 14.01% of the total area and 11.7% of the energy consumption. These results indicate that large physical meshes require proportionally larger, power-hungry configuration caches, yet still fail to deliver significant performance gains due to their shallow processing depth.

TransRec achieved a slightly higher speedup than DIM, but with a significantly lower area overhead and reduced energy consumption. Since TransRec does not include multiplication units in its mesh, this result suggests that the evaluated benchmarks contain few multiplication instructions. Simulation metadata revealed that 17% of mappings were terminated due to speculation limit, 63% due to resource unavailability, and 20% due to invalid instructions, a trend opposite that observed in ATHENA and indicating an imbalance in the opposite direction between speculation limits and processing capacity. Its compact design—featuring only four ALUs and a correspondingly small configuration cache—consumes less energy and, when combined with the performance gains achieved through acceleration, reduces the system's energy consumption. As a reference point, adding a single multiplier to the processing mesh increases its area and energy consumption by 39.5% and 44.6%, respectively, reinforcing the energy efficiency of simpler mesh organizations.

The results show that doubling the configuration cache capacity resulted in marginal performance improvements across all models—roughly 2–3%. ATHENA was the most negatively affected by this change, revealing a clear feasibility limit: beyond a specific cache size, area, and static power dominate, making further scaling impractical under current geometry constraints. Its speedup increased only from 1.36× to 1.39×, yet its area overhead nearly doubled (from 24.8% to 46.6%), and its energy cost rose from 20.6% to 43.9%. DIM exhibited modest increases in area (from 24.4% to 26.2%) and energy (from 37.1% to 40.4%), mainly due to CACTI generating a compact, efficient cache layout for the 256-entry configuration. In contrast, TransRec maintained a low area overhead with reduced energy consumption; however, its total area still increased by 40% (from 3.9% to 5.45%), and its energy savings decreased by 20% (from −5.8% to −4.6%). Overall, the results demonstrate that all evaluated architectures operate efficiently with low-capacity configuration caches, as larger caches introduce substantial area and energy overheads while providing minimal performance gains.

5.2. Design Space Exploration Results

Table 4 presents the performance results, while Table 5 reports the variation in area and energy consumption of the proposed models with 16, 32, and 64 processing levels.

Table 4. Speedup for newly derived meshes using a 128-entry and 256-entry configuration caches.

Cache Size	CGRA Model	16 Levels	32 Levels	64 Levels
128-entry	ATHENA-8	1.27	1.35	1.36
	ATHENA-4	1.21	1.31	1.31
	ATHENA-3	1.15	1.25	1.26
	ATHENA-6	1.25	1.34	1.35
	ATHENA-9	1.28	1.37	1.38
256-entry	ATHENA-8	1.28	1.37	1.39
	ATHENA-4	1.22	1.33	1.34
	ATHENA-3	1.17	1.26	1.28
	ATHENA-6	1.27	1.36	1.38
	ATHENA-9	1.31	1.40	1.41

Table 5. Area overhead and energy variation for newly derived meshes using a 128-entry and 256-entry configuration caches.

Cache Size	CGRA Model	Area (%)			Energy (%)		
		16 Levels	32 Levels	64 Levels	16 Levels	32 Levels	64 Levels
128-entry	ATHENA-8	9.23	11.02	14.61	−0.04	−1.44	6.95
	ATHENA-4	4.47	6.02	7.91	−4.91	−8.72	−3.85
	ATHENA-3	4.28	5.83	7.72	−2.48	−7.14	−2.08
	ATHENA-6	8.84	10.64	14.23	−1.20	−2.62	5.74
	ATHENA-9	9.42	11.21	14.81	0.42	−1.46	6.96
256-entry	ATHENA-8	11.02	14.62	24.80	1.68	2.22	16.64
	ATHENA-4	6.02	7.91	13.35	−4.63	−8.06	−0.56
	ATHENA-3	5.83	7.72	13.15	−2.12	−5.61	1.83
	ATHENA-6	10.64	14.23	24.42	0.20	1.15	15.52
	ATHENA-9	11.22	14.81	24.99	1.33	1.99	16.36

5.2.1. Performance Analysis

The speedup results range from 1.15× to 1.41× and reveal a clear relationship between mesh width and depth with the performance. The ATHENA-8 model with the 128-entry cache configuration achieves speedups comparable to baseline ATHENA (1.36×), reaching 1.35× and 1.36× at 32 and 64 levels, respectively. This result indicates that reducing the number of levels does not significantly impact performance once temporal depth exceeds 32 levels.

The ATHENA-4 model, whose per-level topology is similar to that of TransRec, consistently outperforms it across all scenarios. For example, with a 128-entry configuration cache, TransRec reaches a speedup of 1.17×, whereas ATHENA-4 achieves 1.31× with the 128-entry cache at 32 levels. Moreover, ATHENA-4 delivers performance comparable to ATHENA-8 while using only half of its processing capacity.

ATHENA-6, which is four times smaller than DIM, outperforms it in every scenario. While DIM with a 128-entry configuration cache reaches a 1.145× speedup, ATHENA-6 achieves 1.34× at 32 levels. The ATHENA-9 delivers the highest performance among all models. With a 256-entry cache, ATHENA-9 reaches a 1.41× speedup at 64 levels, surpassing all other variants and even outperforming the baseline ATHENA.

As observed with the baseline CGRAs, increasing the cache capacity from 128 to 256 configurations in the new mesh models also yields small performance gains at the cost of higher area overhead and degrading energy efficiency. These results reinforce the conclusion that Transparent CGRAs achieve good performance results with low-capacity configuration caches.

The data in Table 4 also show that increasing the number of levels from 16 to 32 yields the most significant performance improvement across all models, whereas extending the mesh from 32 to 64 levels provides only marginal gains, indicating a practical saturation point in temporal scalability. Analyzing the simulation metadata of ATHENA-9 with the 128-entry configuration cache reveals that the percentage of mappings terminated due to resource unavailability dropped from 41% at 16 levels to 7% at 32 levels and 2% at 64 levels. This trend indicates an increasing imbalance between the mesh's processing capacity and the fixed speculation limit as the number of levels grows.

Figure 4a–c shows the average number of instructions, levels, and density for ATHENA-9 with the 128-entry configuration cache, respectively. As mesh depth increases, the binary translator produces larger configurations spanning more levels, but the density quickly stabilizes at about three instructions per level. This plateau indicates that deeper meshes expand the temporal window without producing proportionally denser configurations, as instruction-level parallelism limits how much work can be packed per level. As a result, configuration growth often reflects the combination of smaller sub-configurations rather than true densification, which explains why performance gains do not scale proportionally. This trend also reflects in the number of executed configurations, which decreases from 6.67 million (16 levels) to 5.52 million (32 levels) and 5.19 million (64 levels) as configuration size grows.

5.2.2. Area and Energy Analysis

The area overhead ranges from 4.28% to 24.99%, while energy consumption varies from −8.72% to 16.64%. The component-level breakdown shows that the configuration cache accounts for most of the area and energy overhead. The binary translator contributes only about 1% of the area and 3.19% to 4.63% of the total energy consumption across all scenarios, whereas the processing mesh accounts for 1.65% to 2.75% of the area and 8.42% to 13.7% of the total energy consumption. In contrast, the configuration cache ranges from 1.26% to 16.78% of the total area and from 2.99% to 26.6% of the final energy consumption.

These results demonstrate that the primary area and energy cost of transparency arises from storing and accessing configurations, rather than from additional compute or control logic.

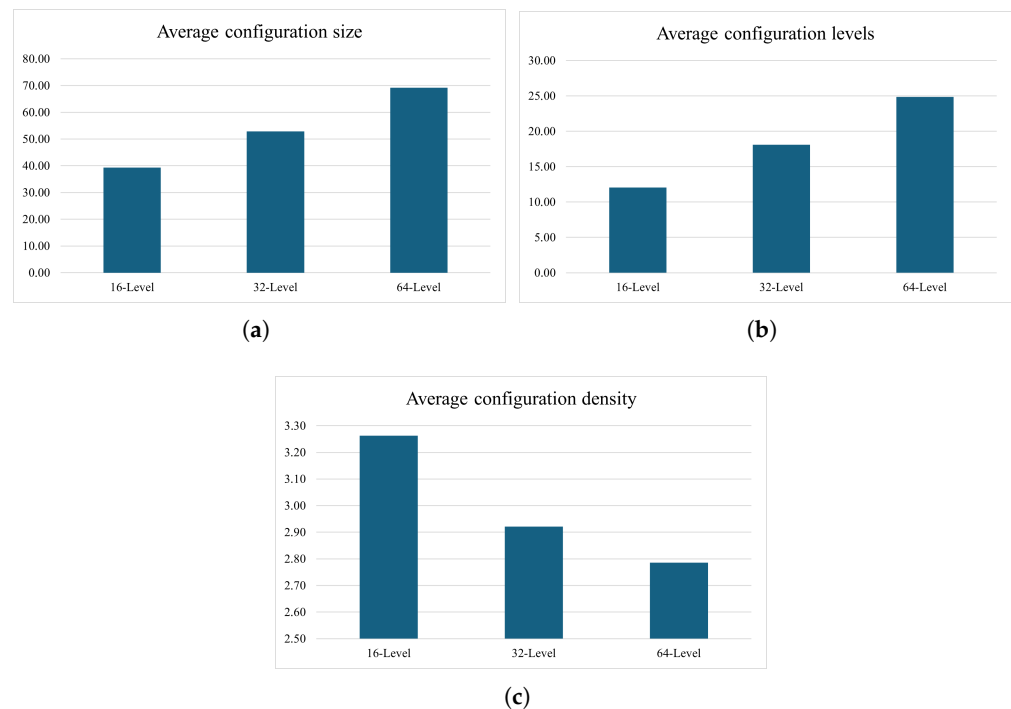


Figure 4. Metadata of ATHENA-9 configurations: (a) average instruction per configuration; (b) average levels per configuration; (c) average density per configuration.

Wider meshes such as ATHENA-8, ATHENA-6, and ATHENA-9 require larger configuration words and, consequently, larger configuration caches, pushing the cache contribution toward the upper bounds of 16–17% of total area and 25–26% of total energy consumption in the 64-level/256-entry scenarios. Meshes, such as ATHENA-3 and ATHENA-4, on the other hand, keep configuration word smaller, so their configuration caches remain near the lower end of these ranges, which explains why they maintain much lower global area and energy overheads even as temporal depth increases. Altogether, these results reinforce the conclusion that the configuration cache should be the primary focus for controlling area and energy costs in transparent CGRAs.

Increasing the number of levels from 16 to 32 effectively doubles the configuration size and, consequently, the overall cache capacity. Even though this expansion increases the cache's power, the performance improvement achieved by the new models was sufficient to offset this cost, resulting in a net reduction in total energy consumption. The same trend does not hold when increasing from 32 to 64 levels: the marginal performance gain was insufficient to offset the additional cache power, resulting in degraded energy efficiency across all models.

5.2.3. Trade-Off Analysis

Across all evaluated mesh models and scenarios, ATHENA-4 with a 128-entry configuration cache at 32 levels provides the best tradeoff between performance, area, and energy efficiency. It achieves a $1.31\times$ speedup and 1.84 IPC, which is competitive with the higher-capacity ATHENA-8 and ATHENA-9 models, while maintaining a moderate area overhead of only 6.02% and reducing energy consumption by 8.72%. In contrast, larger meshes such as ATHENA-9 offer slightly higher speedups but at significantly higher area and energy costs. Conversely, smaller meshes such as ATHENA-3 provide lower area overhead but fail to deliver comparable performance. Overall, the results indicate that

ATHENA-4 achieves the most balanced efficiency point, delivering substantial performance gains while reducing energy consumption and incurring only a small area overhead.

5.2.4. Architectural Findings

A synthesis of the feasibility analysis and design-space exploration reveals three primary factors that bound the scalability of Transparent CGRAs.

- **Temporal scalability limit.** While increasing temporal depth from 16 to 32 levels consistently improves performance, extending to 64 levels yields marginal gains. Simulation metadata shows that deeper meshes generate configurations that grow in height but not in density, stabilizing at roughly three instructions per level for the evaluated benchmarks. This plateau indicates that performance is ultimately limited by the ILP available in the workload, and that increasing depth beyond 32 levels results in significant area and energy overheads without meaningful performance gains. Although higher inherent instruction-level parallelism could shift this temporal scalability limit, the evaluated benchmarks are representative of typical embedded and general-purpose applications, for which ILP saturation occurs at relatively shallow temporal depths.
- **Dominance of configuration cache overheads.** Across all evaluated designs, the configuration cache accounts for the majority of transparency-related costs. Cache scaling is driven by configuration size and I/O bus width, both of which expand with wider meshes. These effects make the cache the primary architectural bottleneck and a first-order determinant of feasible design points.
- **Sensitivity to functional-unit selection.** Functional-unit choices significantly impact feasibility. Removing the multiplier in TransRec led to dramatic reductions in area and energy with minimal performance loss, indicating that complex units are unjustified under sparse utilization. Designers must therefore tailor PE functionality to the target workload domain to avoid unnecessary costs.

Collectively, these findings identify a region of the design space—moderate spatial width and temporal depth—where performance, area, and energy efficiency align most favorably. Balanced spatial-temporal scaling emerges as essential for preserving the viability of Transparent CGRAs and should guide future research aimed at reducing configuration size and cache bandwidth requirements.

6. Conclusions

This work conducted a feasibility study and a systematic design-space exploration of Transparent CGRAs. Using a parameterized cache geometry model, this work evaluates how realistic cache parameters—such as word and block size and I/O width—affect the physical feasibility of Transparent CGRAs.

The results identify three fundamental constraints that govern the scalability of Transparent CGRAs. First, increasing temporal depth enlarges configurations without uncovering additional parallelism, yielding negligible performance gains while imposing substantial area and energy overheads, as reported in prior works on Transparent CGRAs [11,12]. Second, configuration cache scaling dominates physical feasibility; growth in configuration size and I/O width leads to cache structures whose latency, area, and power exceed practical limits. Third, functional-unit selection must be guided by workload characteristics, as high-cost units, such as multipliers, impose disproportionate area and energy overheads when sparsely used.

Despite these constraints, balanced meshes demonstrate that Transparent CGRAs are viable accelerators. Designs with moderate spatial width and temporal depth achieve performance comparable to that of larger meshes while maintaining manageable config-

uration size and modest cache overhead. Among all evaluated scenarios, ATHENA-4 at 32 processing levels with a 128-entry configuration cache offers the most balance between performance, area, and energy efficiency.

Overall, the parameterized geometry model and the DSE methodology introduced in this work provide a structured, reproducible foundation for evaluating new CGRA meshes under realistic physical constraints. By identifying architectural limits stemming from the available ILP and the configuration cache behavior, this study outlines clear directions for future research aimed at improving transparency efficiency and expanding the practical design space of Transparent CGRAs, including the extension of the proposed cache-geometry model and evaluation methodology to additional Transparent CGRA organizations, alternative mapping strategies, and emerging workloads.

Author Contributions: Conceptualization, T.R.B.S.S. and I.S.S.; Investigation, T.R.B.S.S.; Methodology, T.R.B.S.S. and I.S.S.; Writing—original draft preparation, T.R.B.S.S.; Writing—review and editing, T.R.B.S.S. and I.S.S.; Supervision, I.S.S. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The original contributions presented in this study are included in the article. Further inquiries can be directed to the corresponding author.

Acknowledgments: The authors would like to thank the Instituto Federal do Piauí and the Universidade Federal do Piauí for their support.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Moore, G. Cramming more components onto integrated circuits. *Electronics* **1965**, *38*, 114–117. [\[CrossRef\]](#)
2. Dennard, R.; Gaensslen, F.; Yu, H.; Rideout, V.; Bassous, E.; LeBlanc, A. Design of ion-implanted MOSFETs with very small physical dimensions. *IEEE J. Solid-State Circuits* **1974**, *9*, 256–268. [\[CrossRef\]](#)
3. Cavin, R.; Lugli, P.; Zhirnov, V. Science and engineering beyond Moore’s law. *Proc. IEEE* **2012**, *100*, 1720–1749. [\[CrossRef\]](#)
4. Wijtvlit, M.; Waeijen, L.; Corporaal, H. Coarse-grained reconfigurable architectures in the past 25 years: Overview and classification. In Proceedings of the 2016 International Conference on Embedded Computer Systems: Architectures, Modeling and Simulation (SAMOS), Samos, Greece, 17–21 July 2016; pp. 235–244.
5. Lopes, J.; Véstias, M.; Duarte, R.; Neto, H.; Sousa, J. Coarse-grained reconfigurable computing with the Versat architecture. *Electronics* **2021**, *10*, 669. [\[CrossRef\]](#)
6. Qu, T.; Dai, Z.; Liu, Y.; Chen, L. A high flexible shift transformation unit design approach for coarse-grained reconfigurable cryptographic arrays. *Electronics* **2022**, *11*, 3144. [\[CrossRef\]](#)
7. Liu, L.; Zhu, J.; Li, Z.; Lu, Y.; Deng, Y.; Han, J.; Yin, S.; Wei, S. A survey of coarse-grained reconfigurable architecture and design: Taxonomy, challenges, and applications. *ACM Comput. Surv.* **2019**, *52*, 83. [\[CrossRef\]](#)
8. Podobas, A.; Sano, K.; Matsuoka, S. A survey on coarse-grained reconfigurable architectures from a performance perspective. *IEEE Access* **2020**, *8*, 146719–146743. [\[CrossRef\]](#)
9. Clark, N.; Kudlur, M.; Park, H.; Mahlke, S.; Flautner, K. Application-specific processing on a general-purpose core via transparent instruction set customization. In Proceedings of the 37th International Symposium on Microarchitecture, Portland, OR, USA, 4–8 December 2004; pp. 30–40.
10. Clark, N.; Blome, J.; Chu, M.; Mahlke, S.; Biles, S.; Flautner, K. An architecture framework for transparent instruction set customization in embedded processors. In Proceedings of the 32nd International Symposium on Computer Architecture, Madison, WI, USA, 4–8 June 2005; pp. 272–283.
11. Beck, A.; Rutzig, M.; Carro, L. A transparent and adaptive reconfigurable system. *Microprocess. Microsyst.* **2014**, *38*, 509–524. [\[CrossRef\]](#)
12. Silva, F.; Patrocinio, J.; Silva, I.; Jacobi, R. Design space exploration of a reconfigurable accelerator in a heterogeneous multicore. In Proceedings of the 33rd Symposium on Integrated Circuits and Systems Design, Campinas, Brazil, 24–28 August 2020; pp. 1–6.
13. Wang, D.; Lou, J.; Jin, N.; Mascarenhas, E.; Mahapatra, R.; Kinzer, S.; Ghodrati, S.; Yazdanbakhsh, A.; Esmaeilzadeh, H.; Kim, N. MESA: Microarchitecture extensions for spatial architecture generation. In Proceedings of the 50th Annual International Symposium on Computer Architecture, Orlando, FL, USA, 17–21 June 2023.

14. Peccerillo, B.; Mannino, M.; Mondelli, A.; Bartolini, S. A survey on hardware accelerators: Taxonomy, trends, challenges, and perspectives. *J. Syst. Archit.* **2022**, *129*, 102561. [[CrossRef](#)]
15. Liu, F.; Ahn, H.; Beard, S.; Oh, T.; August, D. DynaSpAM: Dynamic spatial architecture mapping using out-of-order instruction schedules. In Proceedings of the 42nd Annual International Symposium on Computer Architecture, Portland, OR, USA, 13–17 June 2015; pp. 541–553.
16. Chen, P.-H.; Cheng, B.W.; Oduoza, M.; Xie, Z.; Lu, R.; Ravipati, S.G.; Koul, K.; Carsello, A.; Mei, Y.; Horowitz, M.; et al. Opal: A 16-nm coarse-grained reconfigurable array SoC for full sparse machine learning applications. *IEEE Solid-State Circuits Lett.* **2025**, *8*, 293–296. [[CrossRef](#)]
17. Salameh, A.A.; Ye, K.M. Design and optimization of coarse-grained reconfigurable array (CGRA) architecture for efficient processing-in-memory (PIM) systems. *J. VLSI Circuits Syst.* **2025**, *7*, 11–18.
18. Zulberti, L.; Monopoli, M.; Nannipieri, P.; Moranti, S.; Thys, G.; Fanucci, L. Efficient coarse-grained reconfigurable array architecture for machine learning applications in space using the DARE65T library platform. *Microprocess. Microsyst.* **2025**, *113*, 105142. [[CrossRef](#)]
19. Brandalero, M.; Shafique, M.; Carro, L.; Beck, A. TransRec: Improving adaptability in single-ISA heterogeneous systems with transparent and reconfigurable acceleration. In Proceedings of the Design, Automation and Test in Europe Conference, Florence, Italy, 25–29 March 2019; pp. 582–585.
20. Silva Junior, F.; Silva, I.; Jacobi, R. Evaluating the performance, energy, and area trade-offs of ATHENA in superscalar processors. In Proceedings of the 34th Symposium on Integrated Circuits and Systems Design (SBCCI), Campinas, Brazil, 23–27 August 2021; pp. 1–6.
21. Binkert, N.; Beckmann, B.; Black, G.; Reinhardt, S.; Saidi, A.; Basu, A.; Hestness, J.; Hower, D.; Krishna, T.; Sardashti, S.; et al. The gem5 simulator. *SIGARCH Comput. Archit. News* **2011**, *39*, 1–7. [[CrossRef](#)]
22. Wilton, S.; Jouppi, N. CACTI: An enhanced cache access and cycle time model. *IEEE J. Solid-State Circuits* **1996**, *31*, 677–688. [[CrossRef](#)]
23. Li, S.; Ahn, J.; Strong, R.; Brockman, J.; Tullsen, D.; Jouppi, N. McPAT: An integrated power, area, and timing modeling framework for multicore and manycore architectures. In Proceedings of the 42nd Annual IEEE/ACM International Symposium on Microarchitecture, New York, NY, USA, 12–16 December 2009; pp. 469–480.
24. Guthaus, M.; Ringenberg, J.; Ernst, D.; Austin, T.; Mudge, T.; Brown, R. MiBench: A free, commercially representative embedded benchmark suite. In Proceedings of the Fourth Annual IEEE International Workshop on Workload Characterization, Austin, TX, USA, 2 December 2001; pp. 3–14.

Disclaimer/Publisher’s Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.