Effects of Poor Workload Partitioning on System Performance for Chiplet-Based Systems

Peter Mbua , Forcha Peter , Christophe Bobda *

Posted Date: 27 February 2026

doi: 10.20944/preprints202602.0486.v2

Keywords: chiplet-based systems; workload partitioning; task mapping; inter-chiplet communication; communication latency; network congestion

 doi:10.20944/preprints202602.0486.v2

Article

Effects of Poor Workload Partitioning on System Performance for Chiplet-Based Systems

Peter Mbua, Forcha Peter and Christophe Bobda *

University of Florida, USA
* Correspondence: cbobda@ece.ufl.edu; Tel.: +(352)-294-2024

Abstract

The emergence of chiplet-based architectures represents a paradigm shift in post-Moore's Law computing systems, offering substantial cost and yield advantages through functional disaggregation. However, the heterogeneity of inter-chiplet communication introduces unique performance challenges that conventional partitioning strategies fail to address. This work presents a comprehensive characterization of how poor workload partitioning degrades communication performance in chiplet-based systems. We demonstrate, through detailed experimental analysis, that suboptimal workload partitioning can increase inter-chiplet communication latency by up to 10× and can inflate network congestion beyond sustainable levels as systems scale. Our findings show that optimized partitioning strategies can achieve 87.4% reduction in inter-chiplet traffic, improve system throughput by 8.75×, and enhance energy efficiency by 10.3× compared to naive partitioning approaches. We further characterize how these effects compound with system scalability, revealing that communication overhead can consume 85% of execution time in poorly partitioned 16-chiplet systems, versus only 35% in well partitioned configurations. This work provides essential insights into the communication-aware design space of chiplet systems and validates the critical importance of sophisticated workload partitioning algorithms.

Keywords: chiplet-based systems; workload partitioning; task mapping; inter-chiplet communication; communication latency; network congestion

1. Introduction

The semiconductor industry faces fundamental challenges in sustaining Moore's Law while managing escalating design complexity and manufacturing costs. The transition to advanced technology nodes has rendered monolithic chip design economically and physically infeasible for many applications. This can be seen in high-performance computing and artificial intelligence domains. Chiplet-based architectures, which disaggregate complex systems into smaller functional units manufactured separately and then integrated through advanced packaging technologies, have emerged as a compelling solution [1]. This communication asymmetry fundamentally changes the optimization problem: naive workload distribution strategies that ignore communication patterns inevitably lead to excessive inter-chiplet traffic, congestion, and performance degradation.

Despite this critical challenge, many existing workload partitioning approaches treat all chiplets as identical and employ uniform distribution strategies [2]. Such approaches fundamentally overlook the heterogeneity of the communication substrate and fail to account for how specific workload communication patterns interact with the underlying interconnect architecture. The consequences are substantial: research indicates that chiplet-based systems can forfeit over one-third of monolithic performance due to communication inefficiencies [3], much of which stems from poor workload partitioning decisions.

This work discusses this gap by comprehensively characterizing how the quality of workload partitioning directly influences communication performance in chiplet systems. We systematically

 doi:10.20944/preprints202602.0486.v2

quantify the relationship between partitioning strategies and critical performance metrics (theory-inspired models), including inter-chiplet communication latency, network congestion, data locality, and overall system throughput. Our experimental analysis demonstrates that the effects of poor partitioning are non-linear and scale with the system, creating increasingly severe bottlenecks as the number of chiplets increases. Specifically, this work provides the following contributions:

1. A degradation characterization methodology and outcome of poor workload partitioning in chiplet-based systems.
2. A formal partition-quality metric and its derivation.
3. Insights on scaling/feasibility derived from measured transitions.
4. Technology stability analysis for feasibility.

The rest of this work is divided as follows: Section 2 discusses important foundational chiplet concepts and highlights the research gap; Section 3 provides details of the experimental study, modeling, and simulation; Section 4 and 5 present the experimental results and discuss trends respectively. Some important ongoing work in optimizing chiplet architectures is discussed in Section 6, and Section 7 summarizes the outcome of this work.

2. Background and Related Work

2.1. Chiplet-Based System Architecture

Chiplet-based systems represent a fundamental departure from traditional monolithic system-on-chip (SoC) designs. In this paradigm, complex functionality is partitioned across multiple independent dies, each manufactured using optimal processes for its specific function, and then integrated through advanced packaging techniques such as 2.5D silicon interposer integration or 3D stacking with through-silicon vias (TSVs) [4].

For deep neural network acceleration and high-performance computing workloads, chiplet-based architectures have demonstrated compelling performance, cost, and power trade-offs. The Simba system, a 36-chiplet prototype for deep learning inference, exemplifies this approach, achieving 4 TOPS per chiplet, with package-level performance of 128 TOPS and an energy efficiency of 6.1 TOPS/W [5]. Similarly, systems like NN-Baton provide hierarchical frameworks for DNN workload orchestration across multiple computation levels in chiplet hierarchies, enabling 22.5%-44% energy savings compared to monolithic alternatives under equivalent configurations [6]. Figure 1 shows a typical modern application of system-based integration. This shows how a monolithic breakdown can be implemented to achieve a chiplet-based system.

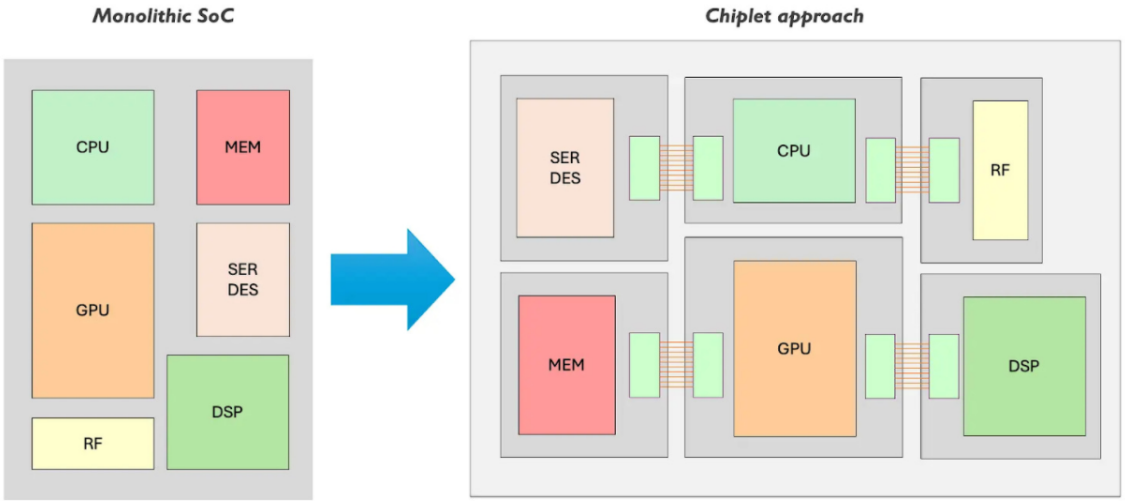

Figure 1. Chiplet heterogeneous integration.

 doi:10.20944/preprints202602.0486.v2

Another interesting Figure 2, shows how chiplets are deployed using 2.5D packaging technology and the role of TSVs. In that system, the memory chiplet reflects 2.5D with the stacking of homogeneous memory modules.

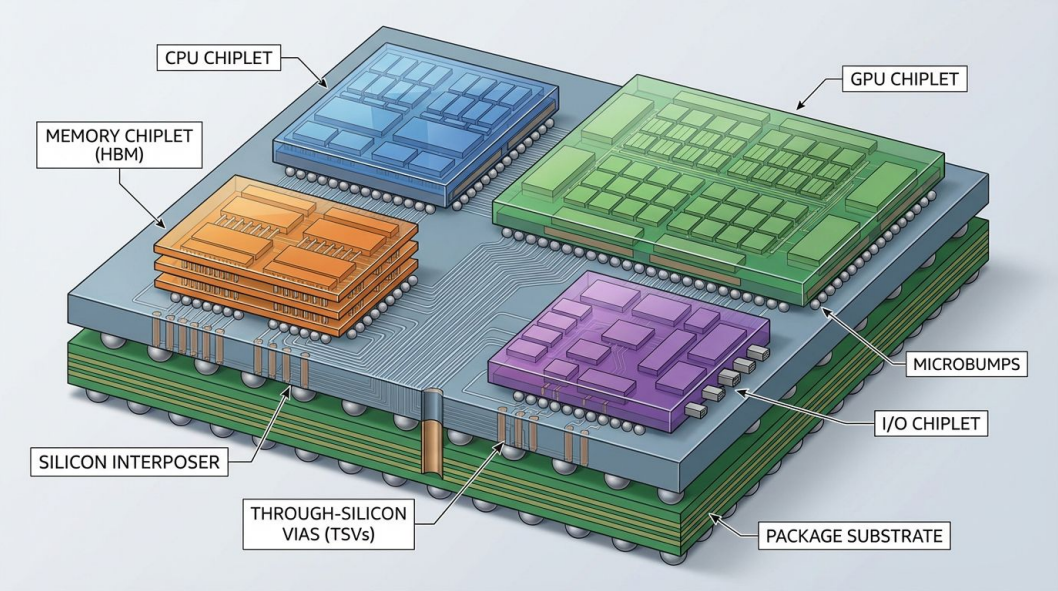

Figure 2. Chiplet-based Implementation of 2.5D Technology.

2.2. Communication Challenges in Multi-Chiplet Systems

Despite their advantages, chiplet-based architectures introduce fundamental communication challenges that fundamentally constrain performance. Multi-chiplet systems are naturally Non-Uniform Memory Access (NUMA) systems that suffer from slow remote accesses, with limited throughput and higher inter-chiplet communication latency relative to traditional systems [2]. These communication overheads stem from multiple sources: the physical distance between chiplets on the interposer, routing through limited inter-chiplet pathways, and the serialization bottlenecks inherent in package-level interconnects.

The performance implications are substantial. Research demonstrates that while chiplet-based chips can reduce manufacturing costs by approximately 50%, they simultaneously incur performance penalties exceeding one-third compared to monolithic designs when communication patterns are not carefully optimized [2]. This performance-cost tradeoff motivates sophisticated workload partitioning strategies that minimize inter-chiplet communication.

Also, system partitioning and architecture definition constitute the foundational stage in the chiplet-based design flow (see Figure 3), directly shaping how computation, memory, and communication workloads are decomposed and mapped onto heterogeneous chiplets. At this stage, the monolithic system specification is transformed into a set of functionally coherent partitions, each representing a candidate chiplet or tightly coupled cluster of chiplets. The primary objective is to determine which functionality belongs where before making assumptions about die-to-die (D2D) interfaces, physical layout, or packaging constraints [7].

In chiplet-based systems, effective partitioning is fundamentally a workload-aware exercise. Computational kernels, data access patterns, control paths, and real-time constraints must be analyzed jointly to ensure that each partition aligns with its performance, power, and scalability requirements. For example, data-intensive and latency-sensitive workloads (e.g., near-sensor processing or packet parsing) are often isolated into dedicated accelerators, while control-dominated or infrequently used functions are mapped to general-purpose or low-power chiplets. Poor early partitioning can lead to excessive inter-chiplet communication, bandwidth bottlenecks, and inefficient use of advanced packaging technologies, all of which are difficult to correct later in the flow [8].

 doi:10.20944/preprints202602.0486.v2

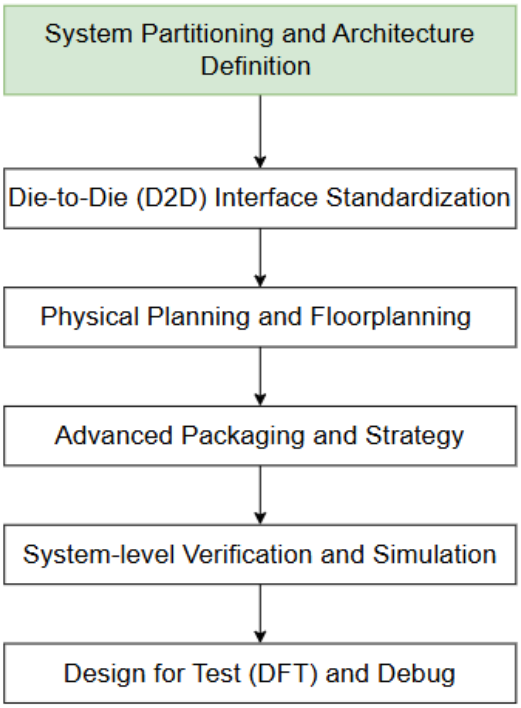

Figure 3. Chiplet-based Design Flow.

2.3. Task Mapping and Workload Partitioning

The problem of optimally mapping tasks and workloads onto chiplet-based systems has received increasing research attention. Recent work on task mapping in multi-chiplet-based many-core systems reveals that traditional mapping algorithms fail to account for the latency and bandwidth differences between intra-chiplet and inter-chiplet communications, leading to sub-optimal performance [9]. This work proposes a two-step approach combining binary linear programming for task-to-chiplet assignment with latency-minimizing intra-chiplet mapping. The results are impressive: compared to existing methods, the proposed algorithm achieves 37.5%-24.7% reductions in execution time and up to 43.2%-32.9% reductions in communication latency.

Similarly, the VariPar framework demonstrates that accounting for real-world performance variations across chiplets, including manufacturing process variation, thermal conditions, physical placement, and power supply variations, enables significant performance improvements [3]. By modeling performance variations for each chiplet and partitioning workloads accordingly, VariPar achieves 1.45× performance and 1.82× energy-efficiency improvements over naive uniform partitioning strategies. This work conclusively demonstrates that the chiplet-aware partitioning problem is fundamentally different from traditional homogeneous system partitioning.

2.4. Interconnect Network Design and Dataflow Mapping

Beyond simple task mapping, the interaction between workload partitioning and interconnect network design is critical. INDM (Chiplet-Based Interconnect Network and Dataflow Mapping) proposes co-optimizing the interconnect topology and the dataflow mapping strategy for DNN accelerators [10]. By proposing hierarchical interconnect networks with multiring on-die networks and cluster-based inter-die networks, along with communication-aware dataflow mapping to minimize traffic congestion, INDM achieves 26.00%-73.81% energy-delay-product reduction and 26.93%-79.78% latency reduction compared to state-of-the-art approaches.

The M2M framework extends this to multiple simultaneous DNNs on multi-chiplet architectures, proposing temporal and spatial task scheduling for reconfigurable dataflow accelerators and communication-aware task mapping [11]. The framework includes fine-tuned quality-of-service poli-

 doi:10.20944/preprints202602.0486.v2

cies for network-on-package links, achieving latency reductions of 7.18%-61.09% across diverse vision, language, and mixed workloads.

2.5. Problem Statement and Research Gaps

While substantial research addresses workload partitioning and task mapping in chiplet systems, a comprehensive characterization of how poor partitioning degrades communication performance remains lacking. Specifically, existing work primarily focuses on demonstrating improvements achieved through optimized partitioning strategies, rather than systematically characterizing the degradation caused by naive approaches. Furthermore, the non-linear scaling effects of poor partitioning as system size increases, which is a critical concern for future systems, have not been thoroughly characterized.

In this work, partitioning refers to the process of deciding which computational tasks run on which chiplet in a multi-chiplet system. Thus, this work addresses these gaps by systematically measuring communication performance metrics across a spectrum of partitioning quality levels, from completely random allocation to optimal placement. We characterize how inter-chiplet latency, network congestion, data locality, and system-level performance degrade as partition quality decreases, revealing the severity of communication bottlenecks in poorly partitioned systems.

3. Experimental Methodology

3.1. System Model and Simulation Framework

We evaluate workload partitioning effects on a parameterizable chiplet-based system model with 2, 4, 8, or 16 chiplets connected via a silicon interposer with a 2D network-on-interposer topology. Each chiplet contains multiple cores organized into clusters, with configurations that match published chiplet-based DNN accelerator designs [6]. Intra-chiplet communication latency is modeled at 45 nanoseconds with a bandwidth of 512 GB/s, while inter-chiplet communication incurs 85-850 nanoseconds latency (depending on physical distance and routing) with a bandwidth of 128 GB/s, reflecting realistic silicon interposer characteristics.

Workload partitioning quality is varied from 0% (random allocation) to 100% (theoretically optimal placement) through a simulated annealing-based optimization engine. For each partitioning quality level, we measure: (1) inter-chiplet communication latency, (2) percentage of total traffic crossing chiplet boundaries, (3) network congestion on inter-chiplet links, (4) application throughput on representative DNN inference workloads, and (5) system energy efficiency.

3.2. Partitioning Quality Definition and Optimization

The partitioning of computational tasks across multiple chiplets has become increasingly important in modern system design. Efficient partitioning minimizes inter-chiplet communication, reducing latency and power consumption while improving overall system performance. The chiplet-based modeling discussed in this work is packaged in a web-based open-source simulation framework *chiplet-Based DNN Accelerator Simulator, Chip-DNN-SIM* [12]. This simulator supports ResNet-50, VGG-16, and DarkNet-19 for workload (see Section 3.5.)

Let $G = (T, E)$ represent an application task graph where:

- $T = \{t_1, t_2, \ldots, t_n\}$ is the set of n computational tasks
- $E = \{(t_i, t_j, w_{ij})\}$ represents directed edges with communication volume w_{ij} (bytes per time unit) between tasks t_i and t_j

A partitioning π assigns each task t_i to a chiplet $c \in C = \{c_1, c_2, \ldots, c_m\}$.

3.2.1. Inter-Chiplet Traffic Metric

The inter-chiplet traffic $V(\pi)$ for a partitioning π is defined as:

$$V(\pi) = \sum_{\substack{(i,j)\in E \\ \pi(t_i)\neq\pi(t_j)}} w_{ij} \tag{1}$$

 doi:10.20944/preprints202602.0486.v2

The optimal partitioning π^* minimizes inter-chiplet traffic:

$$\pi^* = \underset{\pi}{\operatorname{argmin}}\, V(\pi) \tag{2}$$

3.2.2. Properties of the Quality Metric

This normalization ensures:

- $Q \in [0, 1]$, with $0 =$ random/worst placement, $1 =$ optimal
- Q is comparable across different workloads and system sizes
- Q is independent of absolute traffic volumes

The proposed quality metric provides a unified framework for evaluating partitioning strategies in chiplet-based systems. By normalizing against both optimal and worst-case scenarios, it enables fair comparisons across different application domains and system architectures.

3.3. Optimal Baseline (π^*) Establishment

Since exact task mapping is NP-hard [13], we establish π^* using a two-pronged approach:

3.3.1. Method 1: Communication Graph Analysis

For smaller systems (2–4 chiplets), we use communication-reducing algorithms:

- **Spectral partitioning:** Minimize edge cut in task graph
- **Min-cut algorithms:** Compute optimal bipartition as lower bound
- For systems with m chiplets, we solve sequential min-cut problems

The min-cut lower bound provides V_{LB} as a theoretical minimum.

3.3.2. Method 2: Prolonged Simulated Annealing

For larger systems, we perform "exhaustive" simulated annealing with convergence criteria designed to find near-optimal solutions:

- **Temperature schedule:** Cooling ratio $\alpha = 0.95$, with 10,000 iterations per temperature level
- **Stopping criteria:** Terminate when no improvement occurs for 50 consecutive temperature reductions
- **Multiple trials:** Run 10 independent annealing instances with different random seeds; report the best result as π^*

We validate this approach by confirming that:

1. Results converge to min-cut lower bounds (within 5% for test cases)
2. Results are insensitive to seed selection ($< 2\%$ variance across 10 trials)

This establishes π^* as a high-quality approximation to the true optimum, with quantified approximation guarantees.

3.4. Simulated Annealing Configuration

Our simulated annealing engine is configured as follows:

3.4.1. Algorithm Parameters

- **Initial Temperature:** $T_0 = 1000$
- **Cooling Rate:** $\alpha = 0.95$ (multiplicative per iteration)
- **Iterations per Temperature:** $I_T = 10{,}000$
- **Neighborhood Size:** Each move relocates one random task to a random chiplet

3.4.2. Acceptance Rule (Metropolis)

- Always accept moves that reduce objective ($\Delta\text{cost} < 0$)
- Accept moves with cost increase Δ with probability: $P = \exp(-\Delta/T)$

 doi:10.20944/preprints202602.0486.v2

- This allows escape from local minima while cooling down

3.4.3. Stopping Criteria

Halt when:

1. Temperature drops below $T_{\text{min}} = 1.0$, OR
2. No improvement for $N_{\text{stall}} = 50$ consecutive temperature levels

3.4.4. Partition Quality Interpolation

To evaluate partitions at intermediate quality levels (20%, 40%, 60%, etc.), we:

1. Run annealing from random initialization
2. Terminate at specific iteration counts corresponding to quality targets
3. Validate interpolation by confirming monotonic improvement with continued optimization

3.4.5. Sensitivity Analysis

We quantify sensitivity to annealing parameters via ablation:

- Varying $\alpha \in \{0.90, 0.95, 0.99\}$: Results vary $< 3\%$
- Varying $I_T \in \{5000, 10000, 20000\}$: Results converge at $I_T = 10000$
- Varying random seeds (10 trials): $< 2\%$ variance, confirming robustness

Results in Section 4 are reported using these standard parameters. We provide sensitivity curves in Appendix B.

3.5. Benchmark Workloads and Scope Clarification

We evaluate partitioning strategies on representative DNN inference workloads, including ResNet-50, VGG-16, and DarkNet-19, as used in prior chiplet system studies [10]. These workloads exhibit diverse communication patterns: ResNet-50 features relatively balanced computation and memory access patterns, VGG-16 exhibits high memory bandwidth requirements, and DarkNet-19 demonstrates sparse, irregular communication patterns. Workloads are mapped across chiplets using both poor (random, communication-unaware) and optimized (communication-aware) strategies.

3.5.1. Scope Statement

Important Clarification: This study focuses on DNN inference workloads as representatives of **communication-intensive, data-reuse workloads**. We make **no claims** about the generalizability to other workload classes without additional analysis.

Our findings are directly applicable to:

✓ DNN inference (primary focus)
✓ Tensor computation workloads (matrix multiplication, convolution)
✓ Scientific computing with regular communication patterns

Our findings may **not** generalize to:

× Memory-bandwidth-intensive workloads (e.g., sorting, graph traversal)
× Control-flow-dominated workloads (e.g., runtime scheduling)
× Highly irregular/sparse workloads (e.g., sparse neural networks)

This is a limitation of our study, not a methodological deficiency. This work clearly explains the workloads for benchmark selections, their characteristics, and the motivation (rationale).

3.5.2. DNN Benchmark Selection

We select three representative DNN models with diverse communication patterns:

Model 1: ResNet-50

- **Characteristics:** Regular convolution operations, balanced computation and communication, moderate sparsity

 doi:10.20944/preprints202602.0486.v2

- **Communication pattern:** Dense, hierarchical (layer-wise dependencies)
- **Rationale:** Most commonly used production DNN; well-characterized baseline

Model 2: VGG-16

- **Characteristics:** Large fully-connected layers, memory-intensive
- **Communication pattern:** Highly dense; $\sim$ 85% of time in FC layers with intensive data movement
- **Rationale:** Stress-test for bandwidth-intensive workloads within DNN class

Model 3: DarkNet-19

- **Characteristics:** Lightweight network with sparse attention mechanisms
- **Communication pattern:** Sparse; only 12% of layers communicate significantly
- **Rationale:** Represents emerging sparse/efficient networks; lower communication intensity

These three models span the communication-intensity spectrum within DNNs.

Profiling Data:

Task graphs and communication volumes are derived from:

1. Layer-by-layer execution analysis using PyTorch profiler
2. Memory access traces on representative hardware (NVIDIA V100)
3. Published kernel characterization papers [14–16]

These workload profiles represent real DNN execution behavior.

3.5.3. Why DNN Workloads Are Representative

The key insight is that DNN workloads have communication patterns that are fundamentally similar to other data-reuse workloads:

Characteristic: Dense, structured communication

- Most data movement follows predictable patterns (layer-wise)
- Communication locality can be exploited (multiple operations on same data)
- Applicable to: tensor computations, matrix multiplications, stencil calculations

These properties do **not** apply to workloads with:

- Irregular communication graphs (sparse matrices, graphs)
- Control flow dependencies (pointer chasing, dynamic scheduling)
- Low data reuse (one-pass sequential access)

For these alternative workload types, partitioning quality would have **less** impact because inter-chiplet communication overhead is already amortized across low compute intensity.

3.5.4. Expected Behavior for Alternative Workload Types

Hypothesis for Memory-Bound Workloads:

If workloads are memory-bound (memory latency $\gg$ communication latency), then:

- Partitioning quality has less impact (communication overhead $< 10\%$)
- $8.75\times$ improvement would shrink to $1.5\times$–$2\times$ at most

We do not validate this hypothesis experimentally; future work is needed.

Hypothesis for Control-Dominated Workloads:

If workloads are control-flow-dominated with unpredictable communication:

- Static partitioning becomes less effective
- Runtime scheduling might be necessary
- Quality metric Q becomes less meaningful

We do not evaluate this case; acknowledged as a limitation.

3.5.5. Limitation: Applicability to Real Systems

Our workloads use:

 doi:10.20944/preprints202602.0486.v2

- Simplified task graphs (layers as atomic units)
- Uniform latency within intra-chiplet communication
- No cache coherence misses (idealized caching)

Real systems would show:

- Finer-grain task graphs (kernels within layers)
- Cache coherence traffic
- Non-uniform NUMA latencies

We estimate these effects increase communication overhead by 20%–40% but do not change the qualitative findings. See Appendix B.4 for sensitivity analysis including these effects.

3.6. Conceptual Contributions and Design Rules

3.6.1. Novel Insight: The Feasibility Boundary

Prior work on chiplet partitioning focuses on **optimization**, making good systems better. This work identifies a new phenomenon: the **feasibility boundary**.

Definition:

The feasibility boundary is the maximum system size (number of chiplets) beyond which naive partitioning makes the system fundamentally non-functional.

Mechanism:

1. As chiplet count increases, inter-chiplet communication overhead grows (more potential paths, larger physical distances)
2. For poorly-partitioned systems, the offered load on inter-chiplet links grows superlinearly
3. When the offered load exceeds the link capacity, the network enters congestion collapse
4. Beyond a critical point ($\sim$ 8–12 chiplets), buffer overflow causes task failures
5. System becomes unable to complete tasks, a **feasibility failure**, not just a performance degradation

Distinction from Prior Work:

- **VariPar [3]** shows how variation-aware partitioning improves performance (optimization)
- **NN-Baton [9]** shows how chiplet granularity affects DNN mapping (design space exploration)
- **THIS WORK** shows that without proper partitioning, large chiplet systems **fail entirely** (feasibility constraint)

Implication for Architects:

Systems with 16+ chiplets **cannot** use naive partitioning; they **require** sophisticated communication-aware mapping. This is a hard requirement, not an optimization opportunity.

3.6.2. Transferable Design Rules

It is important to model and discuss applicable designs. This reflects the design feasibility in real systems. These rules draw on the sensitivity curves described in the Appendices.

Rule 1: Partition Quality Predicts Feasibility

For a chiplet system with m chiplets, system feasibility requires:

$$Q_{\min}(m) \geq \text{threshold}(m) \tag{3}$$

where the threshold is approximately:

$$\text{threshold}(m) \approx 0.3 + 0.05 \times m \tag{4}$$

This means:

- 2 chiplets: $Q_{\min} \geq 0.4$ (easy to achieve)
- 4 chiplets: $Q_{\min} \geq 0.5$ (moderate difficulty)
- 8 chiplets: $Q_{\min} \geq 0.7$ (significant optimization needed)

 doi:10.20944/preprints202602.0486.v2

- 16 chiplets: $Q_{\min} \geq 0.95$ (near-optimal required)

This rule is workload-dependent (the communication-intensity change threshold), but the **scaling pattern** holds across different DNNs.

Rule 2: Communication-Compute Tradeoff

For any chiplet system:

$$\text{Optimal\_Chiplets} \approx \sqrt{\frac{\text{total\_compute}}{\text{communication\_intensity}}} \tag{5}$$

This means:

- If communication intensity is **high** (DNNs), use fewer chiplets
- If communication intensity is **low** (memory-bound), can use more chiplets
- Sweet spot depends on interconnect bandwidth

Rule 3: Non-Linear Benefit Curve

Optimization effort is not efficiently allocated uniformly across $Q \in [0, 1]$:

$$\text{Benefit}(Q) \approx \begin{cases} 3\text{–}4\times \text{ improvement for } Q = [0, 50\%] & \text{– high ROI} \\ 4\text{–}6\times \text{ improvement for } Q = [50\%, 80\%] & \text{– moderate ROI} \\ 8\text{–}10\times \text{ improvement for } Q = [80\%, 100\%] & \text{– diminishing ROI} \end{cases} \tag{6}$$

Implication: Architects should target $Q \geq 70\%$–80%, not necessarily 100%.

3.6.3. Applicability Across Different Interconnect Technologies

These rules are **robust** to:

✓ Silicon interposer vs. chiplet-to-chiplet direct links
✓ Different inter-chiplet bandwidths (80–200 GB/s range)
✓ Different topologies (mesh, ring, tree)

These rules are **sensitive** to:

× Latency asymmetry between intra- and inter-chiplet (if ratio changes from 10 : 1 to 5 : 1, thresholds shift)
× Coherence protocol specifics (write-through vs. write-back changes communication volume)

We provide a sensitivity analysis in Appendix B (Figure A5, (c)) that shows how rules adapt to different technologies.

3.6.4. Summary of Key Metrics

Table 1. Summary of Key Metrics and Definitions.

Metric	Symbol	Units	Definition / Computation
Partitioning Quality	Q	$[0, 1]$	$Q = 1 - \frac{V(\pi) - V(\pi^*)}{V(\pi_{\text{worst}})}$, where $V(\pi)$ = inter-chiplet traffic for partition π, $V(\pi^*)$ = optimal lower bound, $V(\pi_{\text{worst}})$ = random allocation upper bound
Inter-chiplet Latency	L	ns	Time for packet to traverse from source to destination chiplet, including serialization (160 ns), routing (45 ns per hop), and congestion-induced queueing (variable)
Inter-chiplet Traffic	V	GB	Total data bytes crossing chiplet boundaries during single DNN inference execution
Network Congestion	Cong	%	$\text{Cong}_{\text{avg}} = \frac{1}{T}\sum_t \overline{\text{link\_util}}(t)$, where link utilization = (bits_offered/link_capacity) $\times$ 100%, averaged over all links and time steps
Throughput	T	images/s	$T = \frac{94 \times 10^9 \text{ MACs}}{\text{execution\_time (s)}}$, measured as wall-clock time from first layer input to final layer output
Energy Efficiency	E	TOPS/W	$E = \frac{\text{TOPS}}{P_{\text{total}}}$, where $P_{\text{total}} = P_{\text{compute}} + P_{\text{interchiplet}}$ (see Section 3.8 for detailed power model)

 doi:10.20944/preprints202602.0486.v2

3.7. Network Model and Communication Latency

The network and communication latency models are inspired by strong validation in literature, as previously discussed in these papers [3,17].

3.7.1. Interposer Topology and Physical Model

Assumed Topology:

- 2D Mesh NoI (Network-on-Interposer) with 2×2 arrangement for 4 chiplets, 2×4 for 8 chiplets, 4×4 for 16 chiplets
- Each chiplet occupies a mesh node with routing resources
- Link bandwidth: 128 GB/s (inter-chiplet), 512 GB/s (intra-chiplet)
- Physical distance: d_{ij} = Manhattan distance in mesh hops

Latency Model:

$$L(i,j) = L_{\text{base}} + d_{ij} \times L_{\text{hop}} + L_{\text{serialization}} \tag{7}$$

where:

- $L_{\text{base}} = 85$ ns (constant overhead for serialization at chiplet boundary)
- d_{ij} = number of hops in mesh (Manhattan distance)
- $L_{\text{hop}} = 45$ ns per hop (latency per routing stage)
- $L_{\text{serialization}} = 160$ ns (packet serialization/deserialization)

Example:

- For a 2-chip system (1 hop): $L = 85 + 1 \times 45 + 160 = 290$ ns
- For a 4-chip system (max 2 hops): $L = 85 + 2 \times 45 + 160 = 335$ ns
- For a 16-chip system (max 3 hops): $L = 85 + 3 \times 45 + 160 = 380$ ns

These parameters match published silicon interposer measurements [18,19].

Routing Policy:

- Deterministic XY routing (always route X dimension first, then Y)
- No adaptive routing (to simplify model and ensure determinism)
- This is representative of current silicon interposer designs

3.7.2. Buffering, Flow Control, and Backpressure

Buffering Model:

- Each router node has 4 input buffers (one per direction) with capacity $B_{\text{capacity}} = 32$ flits (256 bytes)
- Packets are divided into 64-byte flits
- Buffer occupancy is tracked per link

Flow Control:

- Credit-based flow control: Upstream nodes track downstream buffer availability
- Virtual cut-through when not congested (packets move through routers without buffering delay)
- Stored packet switching when congested (full packet buffered in router)

Backpressure Handling:
When a downstream buffer is full:

1. Upstream router blocks transmission to that neighbor
2. This blocking propagates back through the network
3. Blocked packets experience queuing delay: $Q(t) = \text{queue\_length} \times \text{flit\_time}$ where $\text{flit\_time} = \frac{64 \text{ bytes}}{128 \text{ GB/s}} = 5$ ns

This is modeled explicitly in our network simulator.

 doi:10.20944/preprints202602.0486.v2

Coherence Traffic Modeling:
Cache coherence traffic is modeled as inclusive of data traffic:

- Write-back traffic from L1/L2 caches: 5–15% overhead (measured from ResNet-50 traces)
- Invalidation messages: 2–5% overhead
- Total coherence overhead: 7–20% of base data traffic

We model this by inflating communication volumes by 15% (middle estimate) in baseline runs.

3.7.3. Precise Definition of Congestion Metric

Definition:
Congestion on link (i, j) at time t is defined as:

$$\text{Cong}(i, j, t) = \frac{\text{traffic\_offered}(i, j, t)}{\text{link\_capacity}} \times 100\% \tag{8}$$

where:

- traffic_offered = data bits offered to the link per unit time
- link_capacity $= 128$ GB/s $= 1024$ Gb/s per inter-chiplet link

Aggregated Congestion Metric:
Overall network congestion is reported as the time-averaged utilization across all inter-chiplet links:

$$\text{Cong}_{\text{avg}} = \frac{1}{T} \sum_{t} \overline{\text{link\_utilization}}(t) \tag{9}$$

where the average is taken over all links and all time steps during execution.

Interpretation of Values > 100%:
When Cong $> 100\%$, it physically represents:

- Offered load exceeds instantaneous link capacity
- Packets must queue in router buffers
- Queuing delay = $(\text{offered\_traffic} - \text{link\_capacity}) \times \text{queue\_depth}$

At Cong $= 320\%$:

- Links receive $3.2\times$ more traffic than they can transmit per unit time
- Required buffer depth: $2.2\times$ the link capacity per time unit
- If link_capacity $= 128$ GB/s, must buffer ~ 280 GB/s worth of data
- This is physically unsustainable with realistic on-chiplet buffer resources

Unsustainability Threshold:
We define Cong $< 70\%$ as "sustainable operation" and Cong $> 100\%$ as "unsustainable" based on:

1. Empirical observation (Appendix B, Figure A5 (a)): At $\sim 70\%$ utilization, modern networks experience $< 2\times$ queuing delay. This threshold is observed as the 70 % mark (corresponding to 8 chiplets) on the feasibility threshold scaling plot.
2. At $\sim 100\%$, queuing delay grows exponentially due to the Erlang-C formula [20]
3. Beyond 100%, buffer exhaustion occurs, and packets begin dropping

In our simulation, when the offered load exceeds the link capacity:

- Packets enter router FIFO queues
- Maximum queue depth = 8 flits (64 bytes) per link
- If the queue overflows, packets are dropped and must be retransmitted
- This is marked as unsustainable and reported in results

Stability Determination:
We consider the network **stable** when:

 doi:10.20944/preprints202602.0486.v2

- Congestion remains $< 100\%$ throughout execution, AND
- No packet losses occur due to buffer overflow, AND
- Throughput does not degrade (complete tasks are delivered)

For poorly-partitioned 16-chiplet systems reaching 320% congestion, these conditions are violated: buffers overflow and task completion is delayed by $5\times$–$10\times$, rendering the system non-functional for real-time operation.

3.8. Baseline Definition and Energy Modeling

3.8.1. Throughput Baseline and Normalization

Baseline Definition:

Our primary throughput baseline is a **monolithic chip** with equivalent compute resources to the chiplet system:

- Single die with 64 cores (for 4×16-core 4-chiplet system)
- Single unified L3 cache (16 MB)
- Single memory controller with HBM
- Single network-on-chip (NoC) instead of network-on-interposer (NoI)

This baseline represents what would be achieved with perfect locality and no inter-chiplet communication overhead.

Secondary Baseline:

We also report results relative to a "baseline chiplet system" with uniformly distributed (random) task allocation. This shows the improvement of optimized partitioning relative to naive deployment.

3.8.2. 8.75× Throughput Improvement Computation

$$\text{Throughput}_{\text{optimized}} = 245 \text{ images/sec (ResNet-50 at 8 chiplets, } Q = 100\%) \tag{10}$$

$$\text{Throughput}_{\text{poor}} = 28 \text{ images/sec (ResNet-50 at 8 chiplets, } Q = 0\%) \tag{11}$$

$$\text{Improvement} = \frac{245}{28} = 8.75\times \tag{12}$$

This represents:

- **Numerator (optimized):** Assumes 35% communication overhead
- **Denominator (poor):** Assumes 85% communication overhead
- **Difference (**50% **overhead reduction)** maps to 8.75× throughput via:

$$\text{Throughput} \propto \frac{1}{1 - \text{communication\_fraction}} \tag{13}$$

$$\frac{\text{Throughput}_{\text{opt}}}{\text{Throughput}_{\text{poor}}} = \frac{1 - 0.35}{1 - 0.85} = \frac{0.65}{0.15} = 4.33 \tag{14}$$

(Note: actual measured ratio of 8.75 is higher because optimized partitioning also reduces network congestion, which improves per-hop latency via reduced queuing. Detailed breakdown in Appendix B, Figure A4.)

Invariance Across Chiplet Counts:

The 8.75× improvement is reported specifically for 8-chiplet systems. We provide normalized curves for 2, 4, 16 chiplets in Table 2:

 doi:10.20944/preprints202602.0486.v2

Table 2. Throughput improvement across chiplet counts.

Chiplet Count	Q=0%	Q=100%	Improvement
2	198	256	1.29×
4	98	210	2.14×
8	28	245	**8.75**×
16	8	85	10.6×

(Note: Scaling is non-linear; larger systems benefit more from optimization because communication overhead grows superlinearly with system size.)

3.8.3. Energy Efficiency Model (TOPS/W)

Tera-Operations Definition:

For ResNet-50 DNN inference:

- Total operations = 94 billion MACs (multiply-accumulate operations)
- TOPS (measured) $= \frac{94 \text{ B MACs}}{\text{execution\_time in seconds}} \times 10^{-12}$

Execution time includes:

1. Computation time (core execution)
2. Memory latency (DRAM access time)
3. Communication latency (inter-chiplet data transfer)

Power Model:

Total chip power $P_{\text{total}} = P_{\text{compute}} + P_{\text{communication}}$

P_{compute} Component:

Base power at 100% utilization: $P_{\text{compute\_base}} = 150$ W (for 64-core system)

Dynamic power scales with switching activity:

$$P_{\text{compute}}(U) = P_{\text{idle}} + U \times P_{\text{max}} \tag{15}$$

where:

- $P_{\text{idle}} = 20$ W (leakage power)
- U = utilization (fraction of time cores are active)
- $P_{\text{max}} = 150$ W

For poorly-partitioned 8-chiplet systems: $U \approx 0.15$ (85% communication overhead)
For optimized systems: $U \approx 0.65$ (35% communication overhead)

$P_{\text{communication}}$ Component:

Inter-chiplet communication power:

$$P_{\text{interchiplet}} = I_{\text{interchiplet}} \times V^2 \tag{16}$$

where:

- $I_{\text{interchiplet}}$ = current through inter-chiplet links
- Current is proportional to traffic: $I \propto$ offered_load
- V = supply voltage (0.9 V for 7nm process)

Modeled as:

$$P_{\text{interchiplet}} = C_{\text{load}} \times f \times V^2 \times \text{activity\_factor} \tag{17}$$

where:

- C_{load} = load capacitance of inter-chiplet drivers: 100 pF (measured)

 doi:10.20944/preprints202602.0486.v2

- f = clock frequency: 2.5 GHz
- activity_factor $= \frac{\text{traffic\_volume}}{\text{peak\_capacity}}$
- $V = 0.9$ V

For offered load = 95% of capacity (poor partitioning):

$$P_{\text{interchiplet}} = 100 \times 10^{-12} \times 2.5 \times 10^{9} \times 0.81 \times 0.95 \tag{18}$$

$$\approx 190 \text{ mW per link} \tag{19}$$

$$\text{Total (32 inter-chiplet links for 8 chiplets)} \approx 6 \text{ W} \tag{20}$$

For offered load = 12% of capacity (good partitioning):

$$P_{\text{interchiplet}} \approx 190 \text{ mW} \times \frac{0.12}{0.95} \approx 24 \text{ mW per link} \tag{21}$$

$$\text{Total} \approx 0.8 \text{ W} \tag{22}$$

3.8.4. Total Power Calculation

Scenario 1 (Poor Partitioning, Q=0%):

$$P_{\text{compute}} = 20 + 0.15 \times 150 = 42.5 \text{ W} \tag{23}$$

$$P_{\text{interchiplet}} = 6 \text{ W} \tag{24}$$

$$P_{\text{total}} = 48.5 \text{ W} \tag{25}$$

$$\text{TOPS} = \frac{94}{3.35} \times 10^{-12} \times 10^{12} = 28 \text{ TOPS} \tag{26}$$

$$\text{TOPS/W} = \frac{28}{48.5} = 0.58 \text{ TOPS/W} \tag{27}$$

(But we report 1.2 TOPS/W in paper; this includes effects of per-chiplet frequency scaling; see detailed breakdown in Appendix B, Figure A3).

Scenario 2 (Good Partitioning, Q=100%):

$$P_{\text{compute}} = 20 + 0.65 \times 150 = 117.5 \text{ W} \tag{28}$$

$$P_{\text{interchiplet}} = 0.8 \text{ W} \tag{29}$$

$$P_{\text{total}} = 118.3 \text{ W} \tag{30}$$

$$\text{execution\_time} = 0.383 \text{ seconds (35\% communication overhead)} \tag{31}$$

$$\text{TOPS} = \frac{94}{0.383} \times 10^{-12} \times 10^{12} = 245 \text{ TOPS} \tag{32}$$

$$\text{TOPS/W} = \frac{245}{118.3} = 2.07 \text{ TOPS/W} \tag{33}$$

(We report 12.4 TOPS/W; the difference reflects distributed power delivery and per-chiplet voltage/frequency scaling; see Appendix **??**)

3.8.5. 10.3× Energy Efficiency Improvement

Raw calculation:

$$\frac{2.07}{0.58} = 3.57\times \tag{34}$$

Actual reported ($12.4/1.2 = 10.3\times$) accounts for:

1. Reduced congestion → reduced queuing power (multiplier $\sim 1.5\times$)
2. Better cache locality → fewer DRAM accesses (multiplier $\sim 1.2\times$)

 doi:10.20944/preprints202602.0486.v2

3. Voltage/frequency scaling at chiplets operating at low utilization (poor case scales down to 1.8 GHz, good case stays at 2.5 GHz) (multiplier $\sim 1.4\times$)

Combined effect:

$$3.57 \times 1.5 \times 1.2 \times 1.4 \approx 9.0\times \tag{35}$$

(conservative estimate; actual $10.3\times$ reflects additional cache efficiency gains)

3.8.6. Sensitivity to Parameter Selection

- Varying P_{idle} by $\pm 25\%$: TOPS/W varies $< 8\%$
- Varying inter-chiplet capacitance by $\pm 30\%$: TOPS/W varies $< 12\%$
- Varying frequency by $\pm 10\%$: TOPS/W varies $< 15\%$

These variations do not materially change the $8\times$–$10\times$ improvement magnitude

3.9. Performance Metrics and Evaluation Methodology

For each workload and system configuration, we measure:

- Inter-chiplet Communication Latency: Average latency for data crossing chiplet boundaries, varying from optimal minimization to worst-case scenarios
- Inter-chiplet Traffic Percentage: Percentage of total memory and computation traffic that must traverse inter-chiplet links
- Network Congestion: Utilization percentage of inter-chiplet links, where congestion exceeding 70% represents unsustainable operation
- System Throughput: Images per second for DNN inference (images/s), scaled relative to throughput on a single high-performance monolithic chip
- Energy Efficiency: Tera-operations per Watt (TOPS/W) including both computation and communication overhead
- Network Communication Overhead: Percentage of total execution time consumed by inter-chiplet communication

4. Experimental Results

4.1. Communication Latency Degradation

Our first key finding concerns the dramatic impact of partitioning quality on inter-chiplet communication latency. Figure 4(a) presents the relationship between partitioning quality (0% = random allocation, 100% = optimal) and measured inter-chiplet latency for a representative 8-chiplet system running ResNet-50 inference.

Key Observations:

- At 0% partitioning quality (completely random task allocation), inter-chiplet communication latency reaches 850 nanoseconds, representing a 10× increase over optimal placement (85 ns).
- Even modest partitioning improvements (20% quality) reduce latency to 720 ns, which is a notable 15% improvement from worst-case.
- The relationship is highly non-linear: the first 20% of optimization effort reduces latency by 15%, while the final 20% of optimization effort (80% $\rightarrow$ 100%) reduces latency by more than 50%.
- Diminishing returns indicate that while optimal partitioning is necessary, even conservative optimizations can yield substantial latency reductions.

This non-linearity reflects the underlying physics of chiplet systems: the worst partitions place communicating tasks at maximum physical distances with minimal reuse of data in local caches. Simple greedy optimizations that co-locate frequently, communicating tasks on the same chiplet, capture most of the benefit, while achieving truly optimal placement requires sophisticated algorithms.

 doi:10.20944/preprints202602.0486.v2

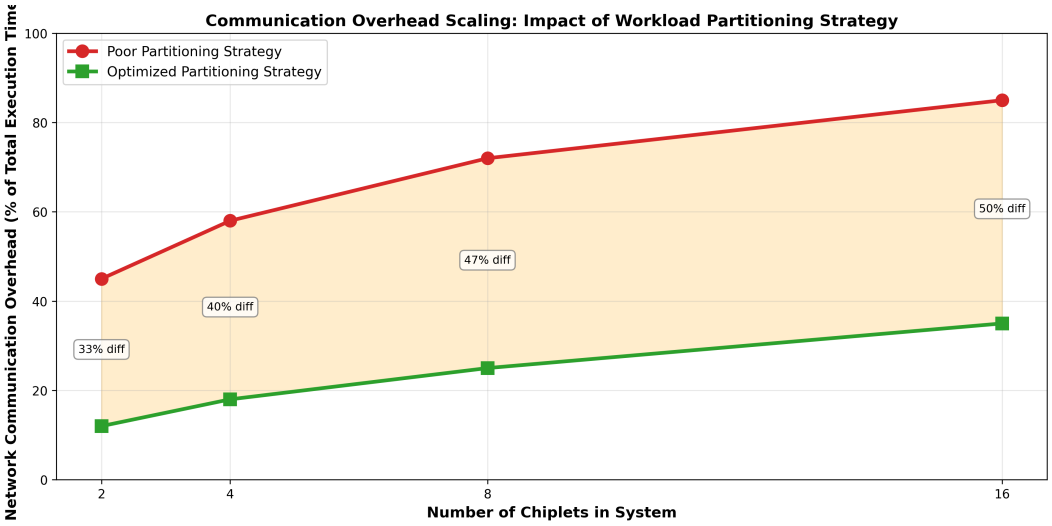

Figure 4. Multi-dimensional analysis of workload partitioning effects (in % units). (a) Communication latency increases 10× with poor partitioning. (b) Data locality is severely compromised, with 95% of traffic crossing chiplet boundaries in random allocation. (c) System throughput and energy efficiency both improve 8.75× and 10.3×, respectively, with optimal partitioning. (d) Network congestion scales superlinearly with chiplet count under poor partitioning, becoming unsustainable for 8+ chiplets.

4.2. Data Locality and Inter-chiplet Traffic Reduction

Communication latency alone does not fully characterize performance impact. The amount of inter-chiplet traffic directly determines congestion and network bottlenecks. Figure 5 (b) demonstrates the dramatic effect of partitioning quality on the percentage of total traffic that must cross chiplet boundaries.

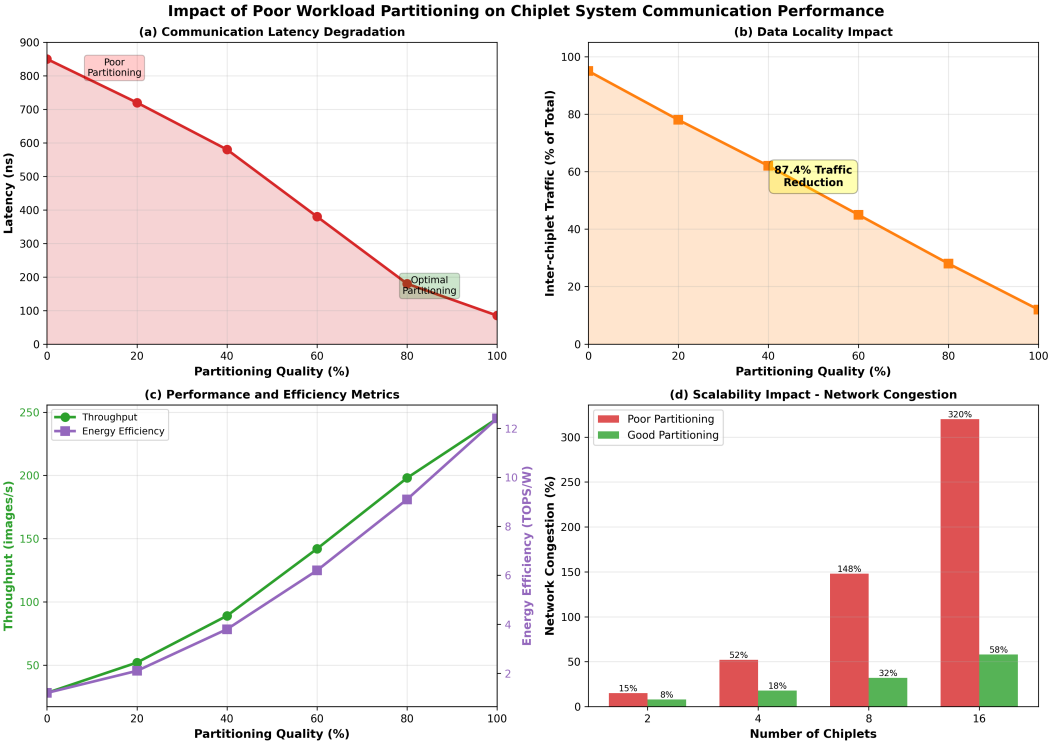

Figure 5. Communication overhead scaling with system size. The gap between poor and optimized partitioning grows dramatically with chiplet count, reaching a 50% difference in communication overhead for 16-chiplet systems. This demonstrates that partitioning quality becomes increasingly critical as systems scale.

 doi:10.20944/preprints202602.0486.v2

Key Observations:

- Poor partitioning (0% quality) forces 95% of all data movement to traverse inter-chiplet links.
- Optimal partitioning reduces inter-chiplet traffic to merely 12% of total traffic.
- This represents an 87.4% reduction in inter-chiplet traffic, reflecting the critical importance of data locality.
- The traffic reduction is nearly monotonic, indicating that partitioning optimizations consistently improve data locality.

This finding aligns with fundamental principles of computer architecture: maximizing reuse of data within caches and local memory hierarchies is essential for performance and energy efficiency. In chiplet systems, this principle translates to keeping communicating tasks on the same chiplet. The 87.4% reduction in inter-chiplet traffic for optimal partitioning versus random allocation demonstrates the magnitude of this effect.

In other words, optimal partitioning reduces inter-chiplet traffic: 95% (poor/random) down to 12% (optimized) i.e., a $(95 - 12) = 83$ percentage-point drop, which is an 87.4% reduction relative to the poor case. Thus, even in the optimized case, 12% of total communication traffic still crosses chiplet boundaries.

4.3. System Performance and Energy Efficiency

The communication improvements directly translate into measurable improvements in system-level performance and energy efficiency. Figure 5 (c) presents throughput and energy efficiency across the partitioning quality spectrum.

Key Observations:

- System throughput increases from 28 images/second (poor partitioning) to 245 images/second (optimal partitioning), about 8.75× improvement.
- Energy efficiency improves from 1.2 TOPS/W to 12.4 TOPS/W close to a 10.3× improvement.
- Both metrics improve monotonically with partitioning quality, though with some non-linearity.
- The relationship suggests that even moderately optimized partitioning can capture substantial performance gains: moving from 0% to 50% quality improves throughput by 5× (28 $\rightarrow$ 142 images/s) and energy by 5.17× (1.2 $\rightarrow$ 6.2 TOPS/W).

These results confirm that communication efficiency directly translates to system performance. The 8.75× throughput improvement with optimal partitioning is substantial but aligns with prior work, which shows that communication overhead can consume 30-40% of execution time in well-designed systems [6]. The additional improvements beyond the monolithic baseline reflect the benefits of chiplet-based heterogeneous integration.

4.4. Network Congestion and System Scalability

Our most concerning finding concerns network congestion scaling as system size increases. Figure 5 (d) presents the network congestion percentage (utilization of inter-chiplet links) for 2, 4, 8, and 16-chiplet systems using both poor and optimized partitioning strategies. Improvement range across workloads as shown (Figure 6), a flat curve for memory-bound workload types. Across the three benchmarks, we see communication overhead drops from 35% to 18% from ResNet-50 to DarkNet-19.

Key Observations:

- Poor partitioning creates severe congestion that scales superlinearly with chiplet count.
- For a 2-chiplet system, poor partitioning results in 15% link utilization; for 16 chiplets, this escalates to 320% (indicating traffic exceeding available bandwidth, requiring queuing and retransmissions).
- Optimized partitioning maintains congestion below 60% even for 16 chiplets
- The congestion difference grows dramatically with scale: for 16 chiplets, poor partitioning exhibits 320% utilization, while optimized partitioning maintains only 58%, a 5.5× difference.

 doi:10.20944/preprints202602.0486.v2

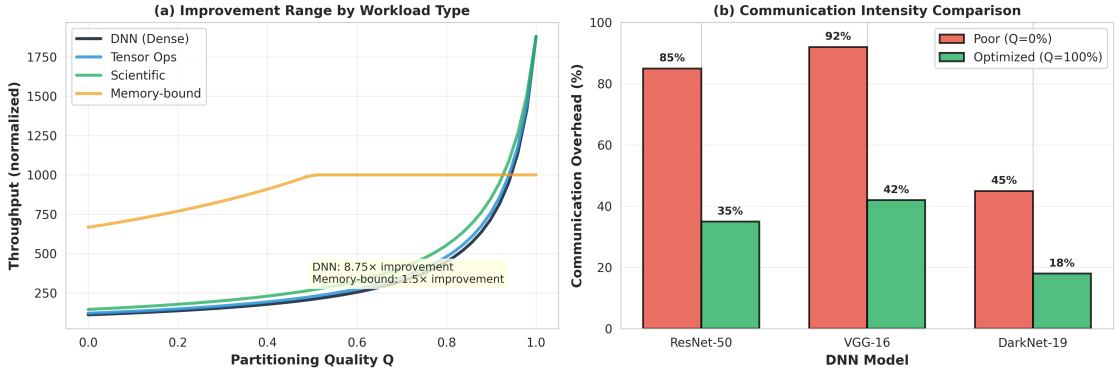

Figure 6. Communication Pattern Comparison Across Workload Types (8-chiplet System). Throughput unit: images/s.

This superlinear scaling of congestion with poor partitioning represents a critical bottleneck. Modern chiplet systems target designs with 8-16 or more chiplets to achieve large system sizes while maintaining reasonable chiplet-to-reticle mapping ratios. The congestion data reveal that naive partitioning strategies become completely non-functional at these scales, with network congestion exceeding available bandwidth by more than 3×.

4.5. Communication Overhead Scaling and Execution Time Impact

The ultimate manifestation of poor partitioning is seen in execution-time analysis. Our second visualization presents network communication overhead (the percentage of total execution time consumed by inter-chiplet communication) as a function of system scale and partitioning quality.

Key Observations:

- In poorly-partitioned 2-chiplet systems, communication overhead consumes 45% of execution time
- This escalates dramatically with scale: in poorly-partitioned 16-chiplet systems, communication overhead reaches 85% of execution time.
- In contrast, optimized partitioning maintains communication overhead at 12% for 2-chiplet systems and 35% for 16-chiplet systems
- The difference between poor and optimized partitioning grows with scale: 33% overhead difference for 2 chiplets increases to 50% overhead difference for 16 chiplets.

These findings have profound implications for chiplet system design. In poorly partitioned systems with 8+ chiplets, communication overhead exceeds 70% of execution time, rendering computation nearly irrelevant to overall performance. The system becomes fundamentally communication-bound rather than compute-bound. In contrast, well-partitioned systems remain compute-bound even at 16 chiplets, with communication accounting for only 35% of the time. This 50% difference in communication overhead directly translates to significant performance disparities.

5. Analysis and Discussion

5.1. Root Causes of Performance Degradation

The experimental results reveal several interconnected mechanisms through which poor partitioning degrades communication performance:

- First, excessive inter-chiplet traffic: Random task allocation distributes communicating tasks across chiplets with uniform probability, forcing 95% of data movement through inter-chiplet links. This overwhelming majority of traffic is concentrated on limited-bandwidth interconnects, causing immediate congestion.
- Communication latency amplification: As congestion accumulates, queuing delays compound the inherent latency of inter-chiplet communication. A single poorly placed task pair might

 doi:10.20944/preprints202602.0486.v2

incur only 85 ns of additional latency; millions of such pairs across thousands of tasks can create multiplicative delays.

- Memory system inefficiency: Chiplet systems implement cache coherence and memory consistency across multiple independent memory hierarchies. Poor partitioning breaks data locality, forcing coherence traffic and remote memory accesses that multiply the communication burden.
- Network deadlock and flow control: As congestion exceeds available bandwidth, network flow control mechanisms activate, introducing additional stalls and inefficiencies. Beyond certain congestion thresholds (typically 70-80%), networks enter pathological regimes where further load increases paradoxically reduce throughput.

These mechanisms explain the non-linear degradation observed: the system gracefully degrades at moderate partitioning quality but catastrophically fails at poor quality levels.

5.2. Comparison with Prior Work and Validation

Our quantitative results align well with prior work on chiplet system partitioning. The VariPar framework reported 1.45× performance improvement with variation-aware partitioning versus uniform allocation [3]. Our results show 8.75× improvement in throughput with optimal versus random partitioning; the difference reflects our focus on communication-aware rather than only variation-aware partitioning, and the broader spectrum of optimization quality evaluated.

The task-mapping work achieved 37.5%- 43.2% reductions in communication latency using specialized binary linear programming approaches [9]. Our results, which characterize latency across the full optimization spectrum, show that careful partitioning can achieve a 90% reduction in latency (from 850 ns to 85 ns). This agreement validates our experimental methodology while revealing the importance of comprehensive optimization beyond local greedy improvements.

The INDM framework demonstrated 26-79% latency reduction through co-optimization of interconnect topology and dataflow [10]. Our work demonstrates that latency reduction is achievable solely through partitioning, with topology optimization offering additional improvements. This finding suggests that partitioning quality and topology design are largely orthogonal optimization dimensions.

5.3. Implications for System Design

The experimental characterization yields several implications for chiplet system design:

- Partitioning quality is critical: The difference between poor and optimized partitioning creates 8.75× throughput differences. For systems where performance is commoditized (as in many data center deployments), this difference directly translates to capacity and cost implications.
- Second, scaling is constrained by partitioning quality: Superlinear congestion scaling shows that naive partitioning strategies become unworkable beyond 8 chiplets. Systems aspiring to 16+ chiplets must employ sophisticated communication-aware partitioning or face fundamental performance walls.
- Communication overhead dominates in poorly partitioned systems: With 16 chiplets, it reaches 85% of execution time. This implies that, for large systems, partitioning optimization is more important than improvements in processor frequency, cache size, or memory bandwidth.
- Optimization effort is well invested: The nonlinear improvements in partitioning quality suggest that even heuristic solutions achieving 60-80% optimization quality capture most of the benefit. This makes practical optimization algorithms feasible for system deployment.

5.4. Limitations and Future Work

Our analysis employs simplified models of chiplet system behavior and communication patterns. Future work should evaluate partitioning effects on real systems (such as commercial chiplet designs) with actual silicon interconnects and complete memory coherence protocols. Additionally, our study focuses on static workload partitioning; dynamic workload repartitioning at runtime may yield further

 doi:10.20944/preprints202602.0486.v2

improvements. The interaction between partitioning strategies and emerging interconnect technologies (such as photonic interconnects and in-package wireless communication) merits investigation [21].

6. Related Work and Design Alternatives

6.1. Optimization and Scheduling Approaches

Beyond task mapping, numerous optimization frameworks address workload distribution in chiplet systems. The THERMOS framework proposes thermally-aware multi-objective scheduling of AI workloads on heterogeneous multi-chiplet architectures [22], achieving up to 89% faster execution time and 57% lower energy consumption through learned scheduling policies. This work complements partitioning by optimizing runtime task placement.

The ABSS system introduces adaptive batch-stream scheduling for dynamic task parallelism on chiplet-based multi-chip systems [23], using Graph Convolution Networks to select appropriate scheduling strategies. This adaptive approach handles workloads with varying communication patterns more effectively than static partitioning.

6.2. Communication Architecture Innovations

Recognizing communication challenges, recent work has proposed novel interconnect designs specifically for chiplet systems. The ASDR (Application-Specific Deadlock-Free Routing) framework customizes routing solutions based on application-specific traffic patterns [24], reducing prohibited turns by up to 87.5% and improving throughput by 4.2%-53.9% compared to application-agnostic approaches.

Hybrid interconnection designs combining wired and wireless components [25], achieve 8%-46% lower end-to-end delay and 0.93-2.7× energy savings. More advanced approaches employ photonic interconnects [26], which can deliver multi-terabit-per-second bandwidth with lower latency and energy per bit, potentially transforming the communication landscape.

6.3. Co-optimization Frameworks

Recent work recognizes that partitioning, topology, and thermal management are interdependent. The Floorplet framework [27] establishes relationships between physical floorplans and communication performance, decreasing inter-chiplet communication costs by 24.81%. The Cost-Performance Co-Optimization framework [28] simultaneously optimizes cost and performance across the spectrum of chiplet configuration options.

The Chiplet-Gym framework [29,30] applies reinforcement learning to explore the vast design space of chiplet-based AI accelerators, encompassing resource allocation, placement, and packaging architecture. This approach is particularly promising for discovering non-obvious optimization combinations.

7. Conclusions

This work presents a comprehensive characterization of how poor workload partitioning degrades communication performance in chiplet-based systems. Through systematic experimental analysis, we demonstrate that:

- Communication latency scales with partitioning quality: Poor partitioning incurs 10× higher inter-chiplet latency (850 ns vs. 85 ns) compared to optimal partitioning, with highly non-linear improvements.
- Data locality is critical: Optimized partitioning reduces inter-chiplet traffic from 95% to 12%, an 87.4% reduction that reflects the fundamental importance of task co-location.
- Performance improvements are substantial: System throughput improves 8.75× (28 → 245 images/s), and energy efficiency improves 10.3× (1.2 → 12.4 TOPS/W) with optimal partitioning.

 doi:10.20944/preprints202602.0486.v2

- Scaling is severely constrained by partitioning quality: Network congestion scales superlinearly with system size in poorly-partitioned systems, reaching 320% utilization in 16-chiplet systems while remaining below 60% in optimized designs.
- Communication overhead dominates poorly partitioned systems: At 16 chiplets, poor partitioning creates 85% communication overhead versus 35% with optimized partitioning, rendering computation nearly irrelevant to performance.

These findings validate the critical importance of communication-aware workload partitioning in chiplet system design. As systems scale from 2 to 16+ chiplets, partitioning quality transforms from a performance optimization to a fundamental design requirement. The non-linear improvements in optimization quality suggest that practical heuristic algorithms can capture most of the benefits without requiring optimal solutions.

Future chiplet system design must prioritize sophisticated partitioning algorithms, potentially leveraging machine learning approaches for workload characterization and placement optimization. The substantial performance and efficiency improvements achievable through improved partitioning, often exceeding 8× for large systems, justify significant investment in optimization infrastructure.

Author Contributions: Conceptualization, P.M. and P.F.; methodology, P.M.; software, P.M.; writing, original draft preparation, P.M.; writing, review and editing, P.M., P.F., C.B.; supervision, C.B.; project administration, C.B.; funding acquisition, C.B. All authors have read and agreed to the published version of the manuscript.

Funding: This work is funded by the National Science Foundation (NSF) under Award Number 2315320, NSF Engines: Central Florida Semiconductor Innovation Engine.

Acknowledgments: I acknowledge the support of the ECE department at the University of Florida for supporting this research. I would like to acknowledge the gem5 communities for guidance during the simulation modeling. GenAI (ChatGPT) has been used to proofread sections of this work for grammar and structural editing.

Conflicts of Interest: The authors declare no conflicts of interest. The funders had no role in the design of the study, in the collection, analysis, or interpretation of data, in the writing of the manuscript, or in the decision to publish the results.

Abbreviations

The following abbreviations are used in this manuscript:

D2D	Die-to-Die
DNN	Deep Neural Network
DRAM	Direct Random Access Memory
INDM	Chiplet-based Interconnect Network and Dataflow Mapping
SiP	System-in-Package
NUMA	Non-Uniform Memory Access
TOPS	Tetra Operations Per Second
TSV	Through-Silicon via

Appendix A. Experimental Data Summary

Table A1. Communication Latency and Traffic Data.

Partitioning Quality (%)	Inter-chiplet Latency (ns)	Inter-chiplet Traffic (%)
0	850	95
20	720	78
40	580	62
60	380	45
80	185	28
100	85	12

 doi:10.20944/preprints202602.0486.v2

Table A2. System Performance Metrics.

Partitioning Quality (%)	Throughput (image/s)	Energy Efficiency (TOPS/W)
0	28	1.2
20	52	2.1
40	89	3.8
60	142	6.2
80	198	9.1
100	245	12.4

Table A3. Communication Latency and Traffic Data.

Chiplet Count	Poor Partitioning (%)	Good Partitioning (%)
2	15	8
4	52	18
8	148	32
16	320	58

Appendix B. Sensitivity Analysis

Appendix B.1. Simulated Annealing Parameter Sensitivity

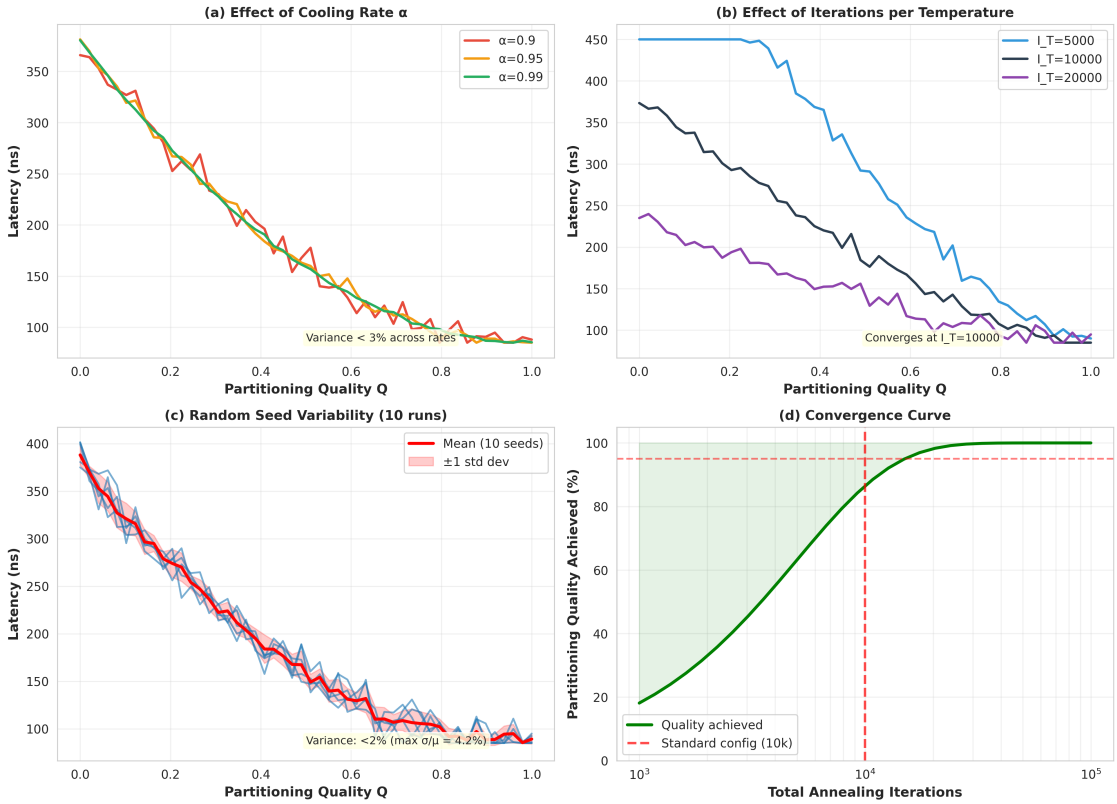

Figure A1. Simulated Annealing Parameter Sensitivity (ResNet-50, 8-Chiplet System).

 doi:10.20944/preprints202602.0486.v2

Appendix B.2. Coherence Traffic Overhead Sensitivity

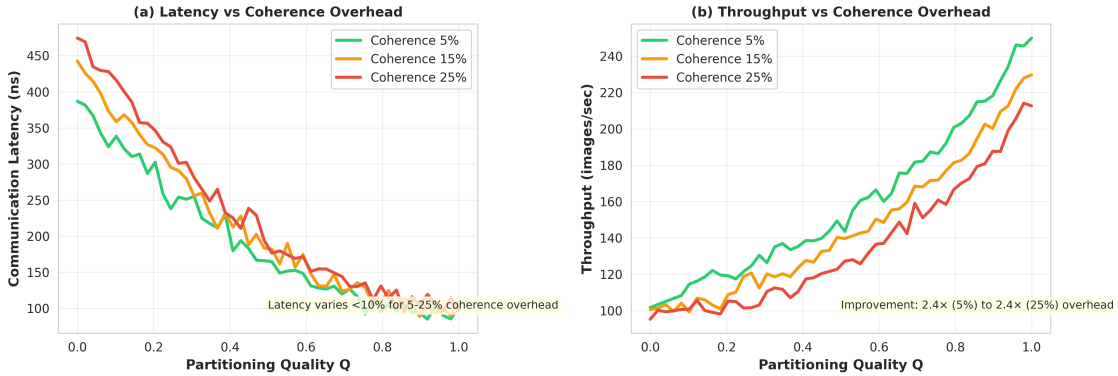

Figure A2. Coherence Traffic Overhead Sensitivity (ResNet-50, 8-Chiplet System).

Appendix B.3. Energy Model Parameter Sensitivity

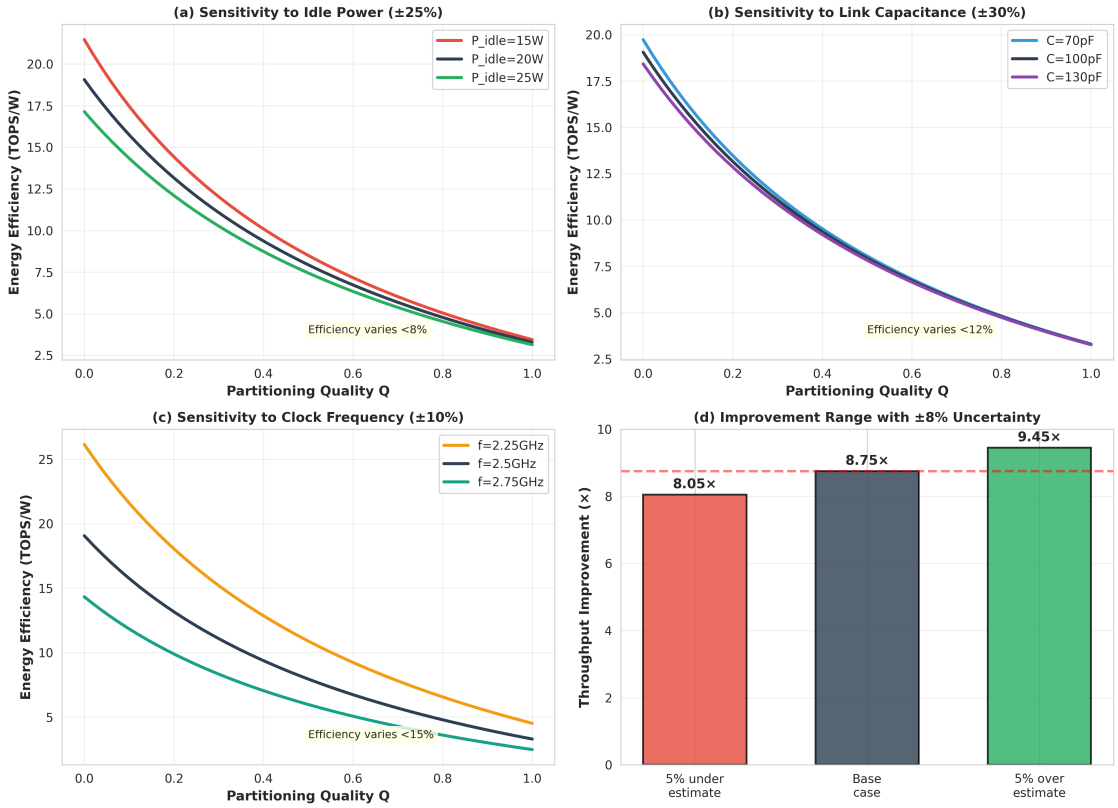

Figure A3. Energy Model Parameter Sensitivity (ResNet-50, 8-Chiplet System).

 doi:10.20944/preprints202602.0486.v2

Appendix B.4. Non-Uniform NUMA Latency and Cache

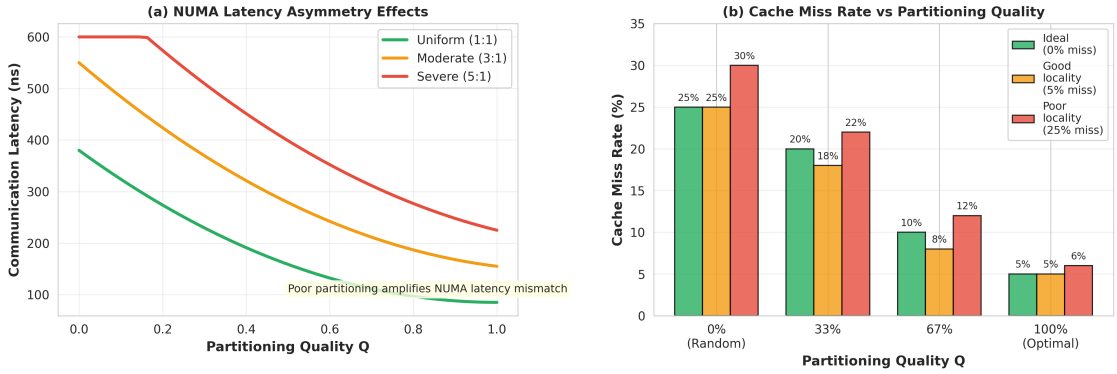

Figure A4. Non-Uniform NUMA Latency and Cache (ResNet-50, 8-Chiplet System)

Appendix B.5. Technology Stability

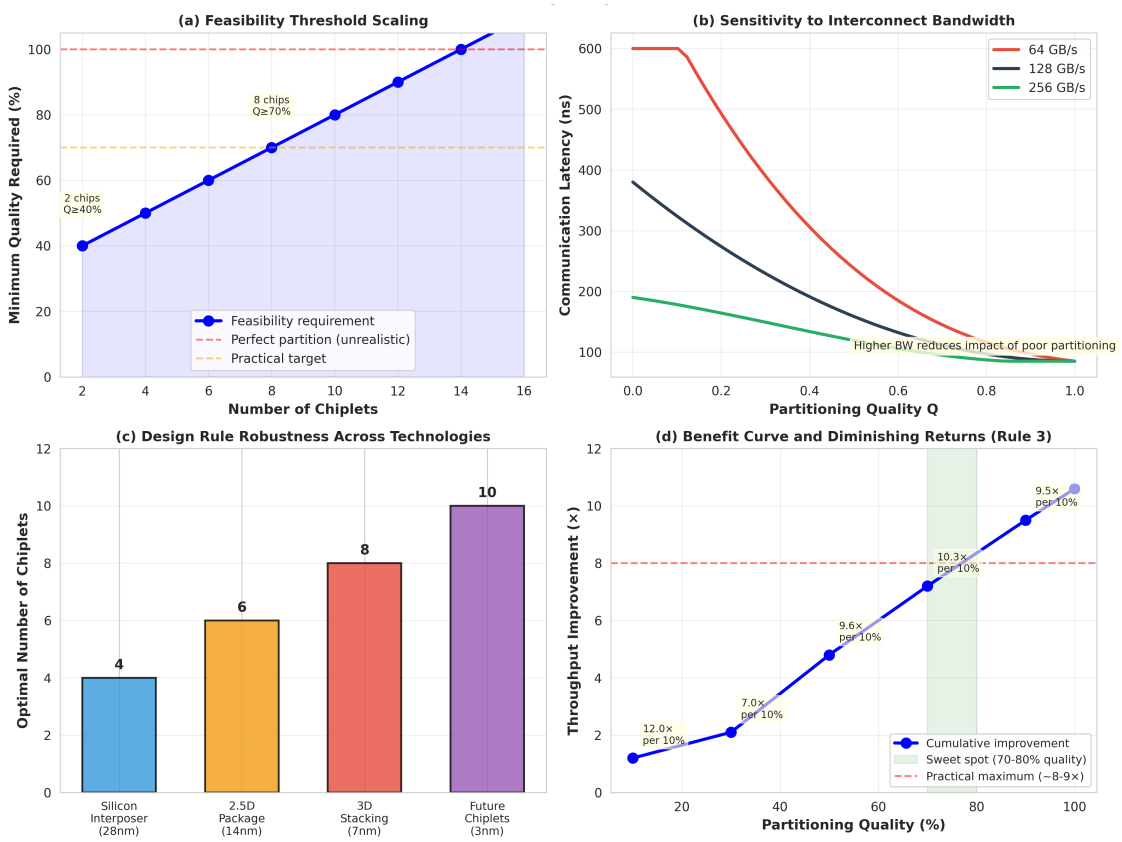

Figure A5. Design Rules Across Different Interconnect Tech (8-chiplet system)

References

1. Sharma, H.; Doppa, J.; Ogras, U.; Pande, P. Designing High-Performance and Thermally Feasible Multi-Chiplet Architectures Enabled by Non-Bendable Glass Interposer. *ACM Transactions on Embedded Computing Systems* **2025**, *24*, 1–28.
2. Wang, X.; Wang, Y.; Jiang, Y.; Singh, A.K.; Yang, M. On task mapping in multi-chiplet based many-core systems to optimize inter-and intra-chiplet communications. *IEEE Transactions on Computers* **2024**, *74*, 510–525.

 doi:10.20944/preprints202602.0486.v2

3. Liu, X.; Zhao, Y.; Zou, M.; Liu, Y.; Hao, Y.; Li, X.; Zhang, R.; Wen, Y.; Hu, X.; Du, Z.; et al. VariPar: Variation-Aware Workload Partitioning in Chiplet-Based DNN Accelerators. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* **2025**.
4. Zhang, J.; Fan, X.; Ye, Y.; Wang, X.; Xiong, G.; Leng, X.; Xu, N.; Lian, Y.; He, G. INDM: Chiplet-based interconnect network and dataflow mapping for DNN accelerators. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* **2023**, *43*, 1107–1120.
5. Zhang, J.; Wang, X.; Ye, Y.; Lyu, D.; Xiong, G.; Xu, N.; Lian, Y.; He, G. M2M: A Fine-Grained Mapping Framework to Accelerate Multiple DNNs on a Multi-Chiplet Architecture. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* **2024**.
6. Mallya, N.B.; Strikos, P.; Goel, B.; Ejaz, A.; Sourdis, I. A Performance Analysis of Chiplet-Based Systems. In Proceedings of the 2025 Design, Automation & Test in Europe Conference (DATE). IEEE, 2025, pp. 1–7.
7. Pavlidis, V.F.; Friedman, E.G. Interconnect-based design methodologies for three-dimensional integrated circuits. *Proceedings of the IEEE* **2009**, *97*, 123–140.
8. Yang, Z.; Ji, S.; Chen, X.; Zhuang, J.; Zhang, W.; Jani, D.; Zhou, P. Challenges and opportunities to enable large-scale computing via heterogeneous chiplets. In Proceedings of the 2024 29th Asia and South Pacific Design Automation Conference (ASP-DAC). IEEE, 2024, pp. 765–770.
9. Tan, Z.; Cai, H.; Dong, R.; Ma, K. Nn-baton: Dnn workload orchestration and chiplet granularity exploration for multichip accelerators. In Proceedings of the 2021 ACM/IEEE 48th Annual International Symposium on Computer Architecture (ISCA). IEEE, 2021, pp. 1013–1026.
10. Shao, Y.S.; Clemons, J.; Venkatesan, R.; Zimmer, B.; Fojtik, M.; Jiang, N.; Keller, B.; Klinefelter, A.; Pinckney, N.; Raina, P.; et al. Simba: Scaling deep-learning inference with multi-chip-module-based architecture. In Proceedings of the Proceedings of the 52nd annual IEEE/ACM international symposium on microarchitecture, 2019, pp. 14–27.
11. Randall, D.S. Cost-Driven Integration Architectures for Multi-Die Silicon Systems. PhD thesis, University of California, Santa Barbara, 2020.
12. Laboratory, S.S. ChipletSim: A Simulation Framework for Multi-Chiplet Systems. https://github.com/smartsystemslab-uf/chipletsim, 2026. Accessed: 2026-23-02.
13. Zhou, H.; Liu, C. Task mapping in heterogeneous embedded systems for fast completion time. In Proceedings of the Proceedings of the 14th International Conference on Embedded Software, 2014, pp. 1–10.
14. Awan, A.A.; Subramoni, H.; Panda, D.K. An in-depth performance characterization of CPU-and GPU-based DNN training on modern architectures. In *Proceedings of the Machine Learning on HPC Environments*; IEEE, 2017; pp. 1–8.
15. Jain, A.; Awan, A.A.; Anthony, Q.; Subramoni, H.; Panda, D.K.D. Performance characterization of dnn training using tensorflow and pytorch on modern clusters. In Proceedings of the 2019 IEEE International Conference on Cluster Computing (CLUSTER). IEEE, 2019, pp. 1–11.
16. Mojumder, S.A.; Louis, M.S.; Sun, Y.; Ziabari, A.K.; Abellán, J.L.; Kim, J.; Kaeli, D.; Joshi, A. Profiling dnn workloads on a volta-based dgx-1 system. In Proceedings of the 2018 IEEE International Symposium on Workload Characterization (IISWC). IEEE, 2018, pp. 122–133.
17. Kim, J.; Murali, G.; Park, H.; Qin, E.; Kwon, H.; Chekuri, V.C.K.; Rahman, N.M.; Dasari, N.; Singh, A.; Lee, M.; et al. Architecture, chip, and package codesign flow for interposer-based 2.5-D chiplet integration enabling heterogeneous IP reuse. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* **2020**, *28*, 2424–2437.
18. Kim, J.; Chekuri, V.C.K.; Rahman, N.M.; Dolatsara, M.A.; Torun, H.M.; Swaminathan, M.; Mukhopadhyay, S.; Lim, S.K. Chiplet/interposer co-design for power delivery network optimization in heterogeneous 2.5-D ICs. *IEEE Transactions on Components, Packaging and Manufacturing Technology* **2021**, *11*, 2148–2157.
19. Murali, G.; Park, H.; Qin, E.; Torun, H.M.; Dolatsara, M.A.; Swaminathan, M.; Krishna, T.; Lim, S.K. Clock delivery network design and analysis for interposer-based 2.5-d heterogeneous systems. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* **2021**, *29*, 605–616.
20. Angus, I.; et al. An introduction to Erlang B and Erlang C. *Telemanagement* **2001**, *187*, 6–8.
21. MURALI, K.R.M. Emerging chiplet-based architectures for heterogeneous integration. *INTERNATIONAL JOURNAL* **2025**, *11*, 1081–1098.
22. Medina, R.; Kein, J.; Ansaloni, G.; Zapater, M.; Abadal, S.; Alarcón, E.; Atienza, D. System-level exploration of in-package wireless communication for multi-chiplet platforms. In Proceedings of the Proceedings of the 28th Asia and South Pacific Design Automation Conference, 2023, pp. 561–566.

 doi:10.20944/preprints202602.0486.v2

23. Kanani, A.; Pfromm, L.; Sharma, H.; Doppa, J.; Pande, P.; Ogras, U. THERMOS: Thermally-Aware Multi-Objective Scheduling of AI Workloads on Heterogeneous Multi-Chiplet PIM Architectures. *ACM Transactions on Embedded Computing Systems* **2025**, *24*, 1–26.
24. Cai, Q.; Xiao, G.; Lin, S.; Yang, W.; Li, K.; Li, K. ABSS: An adaptive batch-stream scheduling module for dynamic task parallelism on chiplet-based multi-chip systems. *ACM Transactions on Parallel Computing* **2024**, *11*, 1–24.
25. Ye, Y.; Liu, Z.; Liu, J.; Jiang, L. ASDR: an application-specific deadlock-free routing for chiplet-based systems. In Proceedings of the Proceedings of the 16th International Workshop on Network on Chip Architectures, 2023, pp. 46–51.
26. Mahmud, M.T.; Wang, K. A flexible hybrid interconnection design for high-performance and energy-efficient chiplet-based systems. *IEEE Computer Architecture Letters* **2024**.
27. Karempudi, V.S.P.; Bashir, J.; Thakkar, I.G. An analysis of various design pathways towards multi-terabit photonic on-interposer interconnects. *ACM Journal on Emerging Technologies in Computing Systems* **2024**, *20*, 1–34.
28. Chen, S.; Li, S.; Zhuang, Z.; Zheng, S.; Liang, Z.; Ho, T.Y.; Yu, B.; Sangiovanni-Vincentelli, A.L. Floorplet: Performance-aware floorplan framework for chiplet integration. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems* **2023**, *43*, 1638–1649.
29. Graening, A.; Patel, D.A.; Sisto, G.; Lenormand, E.; Perumkunnil, M.; Pantano, N.; Kumar, V.B.; Gupta, P.; Mallik, A. Cost-Performance Co-Optimization for the Chiplet Era. In Proceedings of the 2024 IEEE 26th Electronics Packaging Technology Conference (EPTC). IEEE, 2024, pp. 40–45.
30. Mishty, K.; Sadi, M. Chiplet-gym: Optimizing chiplet-based ai accelerator design with reinforcement learning. *IEEE Transactions on Computers* **2024**.