

Robust Feasible-Domain Modeling for Resource-Constrained Spaceborne Heterogeneous High-Performance Computing

Jianing Rao

National Space Science Center Chinese Academy of Sciences

Wenjie Zhao

zwjok123@126.com

National Space Science Center Chinese Academy of Sciences

Junshe An

National Space Science Center Chinese Academy of Sciences

Article

Keywords: spaceborne computing, heterogeneous scheduling, robust optimization, thermal constraint, reliability risk, task graph, heterogeneous interconnect, feasible-domain modeling

Posted Date: August 6th, 2026

DOI: <https://doi.org/10.21203/rs.3.rs-10426574/v1>

License: © ⓘ This work is licensed under a Creative Commons Attribution 4.0 International License.

[Read Full License](#)

Additional Declarations: No competing interests reported.

Version of Record: A version of this preprint was published at Scientific Reports on September 9th, 2026. See the published version at <https://doi.org/10.1038/s41598-026-70375-y>.

Robust Feasible-Domain Modeling for Resource-Constrained Spaceborne Heterogeneous High-Performance Computing

Jianing Rao^{1,2}, Wenjie Zhao^{1,2*} and Junshe An^{1,2*}

¹ National Space Science Center, Chinese Academy of Sciences, Beijing 101400, China

² University of Chinese Academy of Sciences, Beijing 100190, China

* Corresponding author: Wenjie Zhao and Junshe An

Email: zwjok123@126.com anjunshe@nssc.ac.cn

Abstract--A spaceborne heterogeneous computer must finish sensing, preprocessing, artificial-intelligence inference, compression, and data return within one mission window. Processor throughput alone is an incomplete guide because task placement simultaneously determines computation, transfer, energy, peak power, thermal accumulation, and reliability exposure. We developed a reproducible feasible-domain workflow that screens compatibility, transient memory, deadlines, period, peak power, repeated-cycle temperature, and segment reliability before multi-objective selection. Its executable model combines a directed task graph, task-dependent Roofline capability, mapping-induced communication, baseline-increment energy, a first-order RC recurrence, and bounded perturbations. Direct enumeration of a synthetic six-task case reduced 4096 CPU/GPGPU/NPU/FPGA mappings to 18 after timing and capacity checks and to 15 after thermal screening. Compared with the nominal energy-oriented mapping, robust selection used 1.51% more energy and shortened makespan by 122.701 ms. No deadline violation was observed in 1000 seeded replay samples at the declared perturbation strength; the Wilson 95% upper bound was 0.383%. Because the inputs are theoretical model parameters without hardware measurements, the study supports reproducible design screening and makes no claim about flight-system performance.

Index Terms--spaceborne computing, heterogeneous scheduling, robust optimization, thermal constraint, reliability risk, task graph, heterogeneous interconnect, feasible-domain modeling.

I. INTRODUCTION

Modern spacecraft are moving from isolated payload controllers to distributed heterogeneous computing platforms. High-resolution remote sensing, radar preprocessing, autonomous navigation, target recognition, compression and health management increasingly share CPUs, GPGPUs, NPUs, FPGAs and local high-speed links. In this setting, a mission product is completed only when sensing, transfer, computation and return paths all finish within the same deadline.

Peak throughput cannot express this coupled requirement. An accelerator may shorten an FFT or neural-network kernel yet add memory copies, PCIe traffic, RoCEv2 transfers, buffer pressure, and heat. A lower-power resource can create the opposite problem: favorable energy accounting with little timing margin for disturbance or recovery.

Task placement couples every limiting factor in the chain. It fixes computation time and dependency traffic, which in turn set communication delay, buffer occupancy, active power, and temperature. Sustained temperature and radiation-related exposure then affect segment reliability. Feasibility consequently belongs to the combined task graph, resource set, link model, and environment.

Existing work addresses complementary parts of this design problem. Heterogeneous DAG schedulers establish task-placement and makespan baselines [1]-[3], [27], [47], [50]; processor-level Roofline analysis bounds task throughput [4]; and startup-plus-bandwidth models describe communication cost [5]. Power, thermal, and dependability models supply additional constraints [6]-[10], [15], [28]-[36], [49], but these components do not by themselves define a mapping-level spacecraft feasibility workflow.

Equinox uses battery state, thermal headroom, and queue pressure for decentralized runtime scheduling [13]. When to Compute in Space compares execution locations through measurable metrics, feasibility constraints, and a unified utility function [14]. Recent spaceborne-computing studies further document heterogeneous accelerator platforms, fault-tolerant payload processing, and flight-software frameworks [21]-[25], [37]-[46], [48]. Our question is narrower. We examine how task-dependent capability, mapping-induced communication, repeated-cycle thermal response, segment reliability, and a declared perturbation box contract the offline mapping set before objective weights are applied.

We answer this question with an inspectable enumeration workflow for early architecture studies. Each candidate mapping is assigned a schedule and then evaluated by the same compatibility, transient-memory, deadline, period, peak-power, thermal, and reliability functions. Only mappings that pass this screen enter normalized time-energy-risk comparison. For larger graphs the same functions can be retained within MILP, decomposition, or heuristic search, although scalability is not evaluated here.

We make three methodological contributions without attaching them to a particular processor board. First, one executable evaluator connects task-dependent capability with mapping-induced communication. Second, reporting feasible-set contraction before ranking reveals which constraints are active. Third, a monotone box-uncertainty screen is checked by seeded replay with confidence intervals. The released supplementary package contains every synthetic input and output.

Evidence from the case study is limited to internal consistency, traceability, and sensitivity of the workflow. Absolute processor performance, radiation tolerance, and

enclosure-level thermal behavior require hardware or in-orbit measurements that were not available here.

II. REQUIREMENT NECESSITY AND RESEARCH POSITIONING

A. Mission-Level Feasibility Propagation

A representative processing chain begins with sensor preprocessing on an FPGA. Spectral processing can run on a GPGPU, and neural inference can run on an NPU. Compression may use either FPGA or GPGPU. Fusion or control can return to a CPU. Such accelerator combinations have been reported in spaceborne and small-satellite computing studies [21]–[25], [37]–[46]. Each transfer between resources consumes part of the same mission window as computation.

Dependency timing changes what resource capability means in practice. Two upstream completions can generate a short PCIe or RoCEv2 burst even when average traffic remains moderate. Nominal throughput may also be unusable when memory is insufficient or repeated execution drives the thermal recurrence past its limit.

B. Why Independent Optimization Is Insufficient

Time-only scheduling, energy-only scheduling and reliability-only selection each answer a useful but incomplete question. Time-only scheduling can select the fastest mapping while ignoring the energy and thermal cost of keeping high-power resources active [1]–[3]. Energy-only scheduling can select a low-energy mapping that remains close to the deadline and therefore fails under modest timing disturbance [6]–[8]. Reliability-only selection can prefer conservative resources without exposing whether the extra safety reduces schedulability [9], [10].

A common feasibility test replaces the separate time-only, energy-only, and reliability-only questions. Table 1 connects each research gap to a modeling extension. More importantly, every rejected mapping carries a specific cause, such as deadline, memory, peak power, accumulated temperature, or reliability, so a generic failure flag is unnecessary.

Hard-Constraint Screening and Pareto Set



Successive hard constraints reduce 4,096 candidate mappings to five Pareto-optimal solutions.

Fig. 1. Feasible-domain contraction in the canonical rerun. Hard constraints reduce 4096 candidate mappings to 15 feasible mappings; Pareto screening retains five.

Table 1. Positioning against representative lines of work

Representative line	Established scope	Boundary addressed here
DAG scheduling [1]–[3]	Task placement and makespan/energy heuristics	Adds mapping-induced links and spacecraft constraint screen
Processor/link models [4], [5]	Compute and communication bounds	Embeds both bounds in task mapping
Satellite schedulers [11]–[14]	Dynamic, orbital, or location-level constraints	Offline intra-platform feasible-domain contraction
Thermal MPSoc allocation [15]	Power/thermal-aware heterogeneous allocation	Adds segment reliability and declared perturbation replay

III. SYSTEM MODEL AND ASSUMPTIONS

A. Task Graph and Mapping Variables

We represent the application by a directed acyclic graph $G=(V,E)$. Task i carries computation demand O_i , memory traffic Q_i , transient working-set memory m_i^{work} , runtime buffer m_i^{buf} , release time r_i , deadline D_i , class-dependent vulnerability, and an operator class. Edge (i,j) carries data volume $d_{i,j}$ whenever its endpoints use different resources.

A binary variable $x_{i,k}$ assigns task i to resource k . Compatibility is determined by a positive task-class efficiency in the canonical configuration. Figure 2 reports the exact task deadlines, edge volumes, and resource envelope used for the rerun; the full matrices are supplied as machine-readable CSV files.

Mission timing enters through task outputs. Downstream stages cannot consume an output until all predecessor transfers finish, so attaching the deadline to an isolated kernel would be optimistic. A processor can report local success while the end-to-end chain still misses its frame or control deadline.

Selecting $x_{i,k}$ both assigns task i and determines communication on its outgoing edges. A shared resource or memory domain incurs local movement and scheduling overhead. Separation across devices or nodes invokes PCIe, AXI, RoCEv2, or another board-level transfer. Induced traffic therefore remains inside the mapping decision.

$$\sum_k R_i x_{i,k} = 1, x_{i,k} \in \{0,1\}, \forall i \in V \quad (1)$$

Equation (1) assigns each task to exactly one resource. In the released evaluator, non-preemptive tasks are serialized and their working sets and buffers are transient. Equation (2) is thus checked for each selected task-resource pair; simultaneous residency of every task is not assumed.

$$x_{i,k}(m_i^{\text{work}} + m_i^{\text{buf}}) \leq M_k, \forall i \in V, k \in K. \quad (2)$$

Transient capacity matches the released implementation. Persistent model weights, operating-system reservations, or concurrent buffer pools would add a resident-memory term and could shrink the feasible set. Those effects lie outside the synthetic case examined here.

Table 2. Task and resource envelope used in the model-level case

Element	Values	Role in feasibility
Task graph	6 tasks and 7 dependency edges	Defines precedence, data transfer and deadline propagation
Resources	CPU, GPGPU, NPU and FPGA	Defines compatible execution sites and resident memory limits
Mapping space	4 ⁶ –4096 assignments	Enumerated before feasibility screening
Period	1.1 s	Defines repeated thermal and energy accounting window
Declared limits	175 W; 45 °C; reliability ≥ 0.999990	Defines final pass/fail constraints

B. Task-Dependent Effective Capability

Each task-resource pair is characterized by task-dependent effective capability, not nominal peak throughput. C denotes computational capability and P denotes electrical power. Compute throughput, memory bandwidth, software efficiency, parallel efficiency, and startup overhead jointly bound the attainable rate. Consequently, the same GPGPU, NPU, or FPGA can suit one task and perform poorly on another.

Execution time follows from task demand and effective capability. A hardware study should calibrate the capability terms with microbenchmarks, performance counters, task

traces, and repeated timings across representative input sizes. This measurement requirement is consistent with experimental onboard-computing and accelerator studies [24], [37]-[41], [46]. In this sense, the model specifies the measurements needed for deployment and does not treat vendor peak values as observations.

Conservative credit is given only when a task has enough parallelism, memory locality, and operator support to approach a resource's laboratory throughput. Such qualification matters because preprocessing, transforms, detection, compression, and control differ substantially in arithmetic intensity and data movement.

Calibration can be reused across design sweeps. After profiling a task family on one resource over a defined input range, its effective rate can support studies with different deadlines, thermal limits, or links. New evidence of software overhead or memory contention requires updates to the calibrated rate and startup term, while the feasibility logic remains intact.

$$C_{i,k}^{\text{eff}} = \eta_{i,k} \min(C_k^{\text{peak}}, I_i B_k^{\text{mem}}) \quad (3)$$

$$T_{i,k}^{\text{exec}} = O_i / C_{i,k}^{\text{eff}} + \tau_{i,k}^{\text{start}} \quad (4)$$

Equations (3) and (4) compute effective capability and execution time from arithmetic intensity, memory bandwidth, software efficiency, and startup overhead. Both $C_{i,k}^{\text{eff}}$ and $T_{i,k}^{\text{exec}}$ can be measured directly; nominal data-sheet values alone cannot supply them.

C. Inter-Resource Communication

When dependent tasks are mapped to different resources, their data dependency becomes a communication stage. The communication time combines a fixed startup term, effective bandwidth and copy or protocol overhead. The effective bandwidth may represent PCIe, AXI, RoCEv2 or a board-level interconnect, depending on the placement boundary.

Application-visible completion time is the criterion for valuing a high-speed interconnect. Link speed helps only when power, buffer, and fault assumptions are held fixed. Communication is therefore screened together with computation within each candidate mapping.

A generic directed link supplies latency, effective bandwidth, energy per byte, and failure rate to the inter-resource term. PCIe, AXI, or RoCEv2 can provide those parameters. Our synthetic configuration omits protocol-specific congestion control, PFC/ECN behavior, and recovery. Kernel bypass and zero-copy can reduce software overhead, although descriptor preparation, DMA arbitration, NIC processing, and completion handling still contribute [18]-[20].

For early-stage architecture comparison, the Hockney-style decomposition is useful because it separates the startup term from the data-size term [5]. The startup term dominates small control and metadata transfers, while the bandwidth term dominates large image, tensor or compressed-block transfers. This separation enables sensitivity analysis: improving the link rate cannot compensate for large startup overhead when the dependency consists of many small messages.

$$T_{ij}^{\text{comm}} = \sum_k \sum_l x_{i,k} x_{j,l} \delta_{k,l} (\alpha_{k,l} + d_{ij} / B_{k,l}^{\text{eff}} + \tau_{k,l}^{\text{copy}}) \quad (5)$$

$$S_j \geq \max_{i: (i,j) \in E} (F_i + T_{ij}^{\text{comm}}) \quad (6)$$

Equation (5) activates communication through the selected mappings $x_{i,k}$ and $x_{j,l}$. The indicator $\chi_{k,l}$ is zero when the two tasks remain in the same local resource domain and one otherwise. Data size $d_{i,j}$, startup latency $\alpha_{k,l}$, effective bandwidth $B_{k,l}^{\text{eff}}$ and copy overhead $\tau_{k,l}^{\text{copy}}$ determine T_{ij}^{comm} . Equation (6) then propagates predecessor completion F_i and transfer delay into the earliest start time S_j of the successor task.

D. Energy, Thermal and Reliability Terms

We decompose energy into idle baseline, activity-increment and communication components. Activity energy $e_{i,k}^{\text{act}}$ is defined as the task activity-power increment integrated over execution time, and communication energy $e_{i,j}^{\text{comm}}$ is defined as the transfer energy associated with the mapped dependency edge. This avoids double-counting idle power and traces energy reduction to processor selection, task ordering, link optimization or idle-resource residency.

We model thermal behavior with a first-order resistance-capacitance recurrence over repeated task periods [7], [8]. In Equation (8), $a = e^{-\Delta t / RC}$ is the thermal memory coefficient and $b = R(1-a)$ converts period power into temperature rise under the compact RC approximation. The recurrence cannot replace board-level thermal simulation, yet it suffices to reveal when a mapping with acceptable energy still accumulates too much temperature over sustained operation.

Segment reliability is computed from resource and communication exposure. Here it serves as one screening boundary during mapping and replay; no radiation-qualification claim follows from the calculated probability.

Peak power and accumulated temperature remain separate constraints. A mapping can meet the instant power budget yet heat gradually across repeated periods. Another can satisfy average temperature while briefly exceeding a bus or power-supply envelope.

Reliability enters the diagnosis in the same way as deadline, memory, and temperature. For each mapping, the evaluator asks whether longer exposure or additional communication pushes segment reliability below its declared threshold. Joint timing, energy, thermal, and reliability constraints have also been examined for heterogeneous embedded platforms [28]-[36]. Designers can later tighten or relax that measurable boundary as calibration improves.

$$E = \sum_k P_k^{\text{idle}} T^{\text{period}} + \sum_i \sum_k x_{i,k} P_{i,k}^{\text{act}} T_{i,k}^{\text{exec}} + \sum_{(i,j)} e_{ij}^{\text{comm}} \quad (7)$$

$$\theta_{n+1} = \theta_{\text{amb}} + a(\theta_n - \theta_{\text{amb}}) + b P_n, \quad a = e^{-\Delta t / RC}, \quad b = R(1-a) \quad (8)$$

$$R^{\text{seg}} = \prod_i \prod_k \exp(-x_{i,k} \lambda_{i,k} T_{i,k}^{\text{exec}}) \prod_{(i,j)} \exp(-\lambda_{ij} T_{ij}^{\text{comm}}) \quad (9)$$

Equations (7)-(9) keep the non-timing constraints explicit. Energy E is split into idle power P_k^{idle} , activity energy $e_{i,k}^{\text{act}}$ and communication energy $e_{i,j}^{\text{comm}}$. Temperature θ_n accumulates across repeated periods, and segment reliability R^{seg} decreases with resource exposure rates $\lambda_{i,k}$ and communication exposure rates $\lambda_{i,j}$.

IV. OPTIMIZATION FORMULATION AND ROBUST FEASIBILITY

A. Feasible-Domain Construction

Feasibility requires a sequence of explicit checks. Every task must be compatible with its assigned resource, and memory capacity must be respected. Precedence and communication must meet the deadline. Peak power must remain within budget, thermal recurrence below its limit, and segment reliability above its threshold.

Order matters: weighted objectives are meaningful only among mappings that satisfy the engineering constraints. The objective in the example uses normalized makespan, energy and reliability risk with weights 0.1, 0.8 and 0.1. The weights express an energy-oriented design preference after feasibility has already been established.

Feasible-set contraction changes how an optimum is read. A single-objective winner may sit close to a hard boundary, yet its score does not reveal that fragility. Counting survivors after each screen exposes the active limits and the margin of the selected mapping. Figure 1 and the ablation table therefore connect architecture-space reduction to an engineering decision, while still reporting the best schedule.

Exhaustive enumeration is practical for the small graph studied here and makes every rejection reproducible. Each candidate is checked for precedence, memory, resource non-overlap, power, temperature, and reliability. Larger graphs could retain the same tests within mixed-integer, heuristic, or decomposition solvers. We have not evaluated that scaling step, so no claim of algorithmic scalability is made.

$$F_i = S_i + \sum_k \in K x_{i,k} T_{i,k}^{\text{exec}}, F_i \leq D_i, \forall i \in V \quad (10)$$

$$P^{\text{peak}}(x) \leq P^{\text{max}}, \max_n \theta_n(x) \leq \theta^{\text{max}}, R^{\text{seg}}(x) \geq R^{\text{min}} \quad (11)$$

$$P(t;x) = \sum_k P_k^{\text{idle}} + \sum_i \sum_k x_{i,k} (P_k^{\text{act}} - P_k^{\text{idle}}) 1[S_i, F_i](t), \\ P^{\text{peak}}(x) = \max_t P(t;x). \quad (11a)$$

$$\min_x \in \mathcal{F} w_t T^{\text{norm}}(x) + w_e E^{\text{norm}}(x) + w_r \text{Risk}^{\text{norm}}(x) \quad (12)$$

$$S_j \geq F_i - M(1 - y_{i,j,k}), \quad S_i \geq F_j - M y_{i,j,k} \\ S_j \geq F_i - M(3 - x_{i,k} - x_{j,k} - z_{ij,k}), \quad S_i \geq F_j - M(2 - x_{i,k} - x_{j,k} + z_{ij,k}). \quad (13)$$

For each unordered task pair $\{i,j\}$ and resource k , $z_{i,j,k}$ is a binary ordering variable. When $x_{i,k} = x_{j,k} = 1$, $z_{i,j,k}$ selects one of the two non-overlap inequalities; otherwise the Big-M terms deactivate both inequalities. The enumeration code implements the same non-preemptive serialization by maintaining the next available time of each resource.

Equations (10)-(13) define completion, resource non-overlap, and the hard constraints. Equation (11a) specifies the event-based peak-power trace used by the released evaluator. Thermal recurrence is applied to the scheduled active and idle intervals for 180 repeated cycles. Normalized makespan, energy, and mission risk are compared only within the feasible set.

B. Robust Perturbation Model

Nominal parameters may vary with input content, temperature, contention, and environment. We use the independent box $U = \{\rho_T \in [0, 0.15], \rho_B \in [0, 0.20], \rho_\lambda \in [0, 0.20]\}$, where execution time is multiplied by $1 + \rho_T$, bandwidth by $1 - \rho_B$, and resource/link failure rates by $1 + \rho_\lambda$. For the monotone deadline, period, thermal, and reliability

functions used here, the corner (0.15, 0.20, 0.20) is the deterministic worst case.

Robust screening defines a bounded engineering contract. It provides no distribution-free guarantee [16], [17]. Separately, stochastic replay samples scaled uniform ranges and estimates deadline-violation frequency. Related work has used stochastic, reliability-aware, or energy-aware scheduling formulations for heterogeneous platforms [30], [34], [35], [49]. Every distribution, sample count, seed, and Wilson 95% interval is recorded in the supplement.

Table 3 reports the constraint screens and robust budgets used in the case study. Figure 1 shows the corresponding survivor counts. We interpret those counts as successive contraction of the feasible domain, not as independent optimization results.

Worst-corner feasibility is checked before the final weighted objective is applied. Nominally efficient mappings can disappear at this stage when they lack perturbation margin. A slightly more energy-intensive mapping may survive and then win within the robust subset. That ordering accounts for the difference between robust and nominal energy-oriented results in Fig. 3.

Figure 6 reports the replay boundary. A zero observed count is described as an observation, not a zero failure probability; with 1000 samples, the Wilson 95% upper bound is 0.383%. Protection outside the declared box is not claimed.

$$T_{i,k}^{\text{exec}} = T_{i,k}^{\text{exec}}(1 + \rho_T), B_{k,l}^{\text{eff}} = B_{k,l}^{\text{eff}}(1 - \rho_B) \quad (14)$$

$$\lambda_{i,k} = \lambda_{i,k}(1 + \rho_\lambda), \lambda_{i,j} = \lambda_{i,j}(1 + \rho_\lambda) \quad (15)$$

$$x \in F^{\text{rob}} \Leftrightarrow g_m(x; T, B, \lambda) \leq 0, m = 1, \dots, M_c \quad (16)$$

Equations (14)-(16) apply the perturbation factors and define the robust feasible subset. Robust selection checks the deterministic worst corner before applying the same normalized objective within that subset. This choice is valid for the monotone evaluator released with the paper; models with non-monotone thermal control, retries, or correlated faults would require a different uncertainty analysis.

Table 3. Robust feasibility settings and screening stages

Item	Value	Interpretation
Compute latency perturbation	+15%	Models execution-time uncertainty
Bandwidth perturbation	-20%	Models link and transfer degradation
Failure-rate perturbation	+20%	Models reliability-risk inflation
Timing and capacity feasible mappings	18	Mappings that pass deadline and memory checks
After thermal screening	15	Mappings remaining after the 45 C limit
Pareto nondominated mappings	5	Trade-off set used for final comparison
Main replay	N=500; seed 20260615	Paired common-random-number comparison
Boundary replay	N=1000 per point	Per-strength seeds listed in supplementary CSV

Task Graph, Deadline Ladder, and Resource Envelope

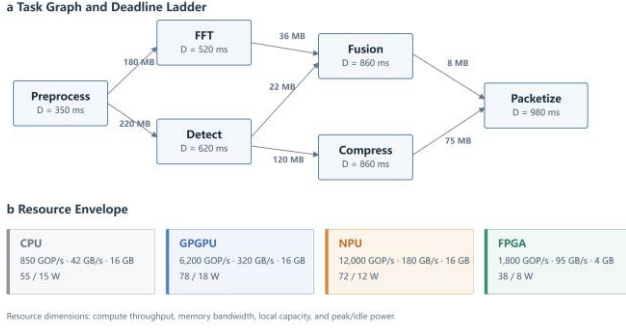


Fig. 2. Canonical task graph, deadlines, edge volumes, and resource envelope. Values are synthetic model inputs and are fully listed in the supplementary package.

V. REPRODUCIBLE MODEL-LEVEL EVALUATION

A. Questions, Configuration and Evidence Boundary

Evaluation focused on three questions. How much does each constraint shrink the candidate space? How do time-, energy-, risk-, and robustness-oriented choices differ in makespan, energy, and reliability? Finally, how far beyond its design box can the robust mapping be perturbed before violations appear?

Six tasks and seven dependency edges define the canonical case, with edge volumes of 8-220 MB and deadlines of 350, 520, 620, 860, 860, and 980 ms. CPU, GPGPU, NPU, and FPGA roles form the resource set[21]-[26]. A 1.10 s period is repeated for 180 thermal cycles from a 28 °C coolant boundary. Hard limits are 175 W, 45 °C, and segment reliability of at least 0.999990. Figure 2 and Appendix A summarize these values, and the supplementary CSV files retain the complete inputs.

All inputs are theoretical model-level values retained from the executable Chapter-2 simulation record. They are not hardware telemetry or vendor specifications. The rerun reported here used the released Python evaluator without changing the canonical configuration; SHA-256 hashes and environment details are recorded in run_manifest.json.

Five selection rules are compared within one evaluator: makespan, energy, mission risk, a nominal weighted score (0.10/0.80/0.10), and worst-corner robust selection. They function as objective and robustness ablations. Because external schedulers such as HEFT, CAE, and Equinox were not rerun under the same configuration, Table 4 does not rank the proposed workflow against them.

Identical scheduling and constraint functions evaluate every mapping. Objective weights and robustness rules are the only strategy-level changes, so workload, resource, and link assumptions remain paired. Table 4 should therefore be read as a controlled comparison within the model.

Table 4. Representative strategy results

Strategy	Makespan (ms)	Energy (J/cycle)	Reliability	Peak power
Time objective	763.272	117.461	0.999994360	173 W
Energy / nominal	963.555	109.679	0.999994281	173 W
Risk objective	840.854	111.332	0.999994519	173 W
Robust trade-off	840.854	111.332	0.999994519	173 W

B. Feasible-Domain Contraction

Of 4096 candidate mappings, 18 passed timing and capacity checks. The 175 W peak-power limit removed none of them in the nominal case. Applying the 45 °C limit reduced the count to 15, whereas the chosen reliability threshold caused no additional rejection. Five mappings remained after Pareto screening.

Figure 1 attributes most contraction to deadline and capacity, followed by repeated-cycle temperature. A one-cycle energy metric would have retained the three thermally rejected schedules. Table 3 gives the engineering meaning of each screen.

No mapping was rejected solely by reliability at the selected threshold. Under these synthetic parameters, all thermally feasible mappings also met segment reliability. A different radiation environment, processor technology, or topology could reverse that ordering. Reporting both active and inactive constraints makes such dependence visible.

Five nondominated mappings survive as alternatives with different makespan, energy, and reliability risk. Mission policy can then choose among them using required margin or resource-reservation rules, without enumerating the architecture space again.

C. Strategy-Level Trade-Off

Makespan-oriented selection produced 763.272 ms at 117.461 J per cycle. Energy-oriented selection reduced consumption to 109.679 J but extended makespan to 963.555 ms, leaving little deadline margin. Risk-oriented and robust selection chose the same mapping in this case, at 840.854 ms and 111.332 J per cycle.

Figure 3 and Table 4 report the nominal metrics. The robust mapping increased energy from 109.679 to 111.332 J per cycle (+1.51%) and reduced makespan from 963.555 to 840.854 ms (−122.701 ms). The risk-optimal and robust selections coincided in this configuration; this is an observed instance property, not evidence that the two objectives are generally equivalent.

Within this instance, 763.272 ms is the observed lower makespan bound and costs 117.461 J per cycle. Only 16.445 ms separates the nominal energy mapping from the 980 ms final deadline. Robust selection increases that margin to 139.146 ms, which explains the sharply different replay behavior.

A weighted score alone would hide why one mapping was selected. Table 4 reports makespan, energy, reliability, and peak power separately, leaving the underlying engineering trade-off open to inspection.

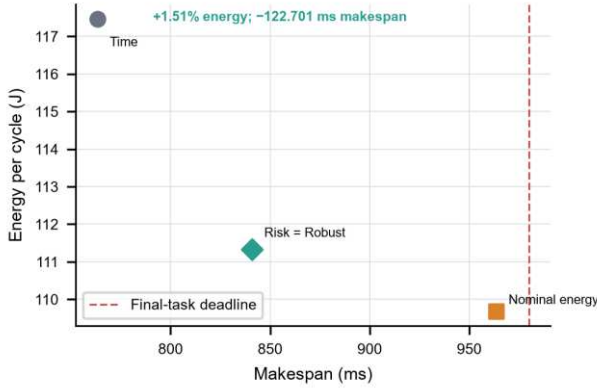


Fig. 3. Nominal makespan-energy comparison. The risk-optimal and robust selections coincide in this configuration. The dashed line marks the 980 ms final-task deadline.

D. Mapping Shift and Mechanism

Only compression changes resource between the nominal energy and robust mappings, moving from FPGA to GPGPU (Fig. 4). Its position between detection and packetization places both execution time and cross-resource transfer on the final-task path.

That single reassignment follows from the available timing margin. FPGA is attractive under nominal energy accounting, but its communication and execution path leaves little room for delay. Robust screening accepts the higher active energy of GPGPU in exchange for lower timing exposure. Table 5 lists every assignment so the change can be traced directly.

All other assignments remain fixed: FPGA handles preprocessing and packetization, GPGPU runs FFT, NPU runs detection, and CPU runs fusion. Their stability depends on the released efficiency, setup, and link matrices. Hardware calibration could produce another pattern.

Engineers can test this explanation by varying input size, compression ratio, memory bandwidth, or link startup time. If compression no longer benefits from GPGPU after calibration, robust selection should move it elsewhere. The model supplies a traceable reason for the recommendation and does not prescribe a permanent resource choice.

Nominal vs. Robust Task Mapping

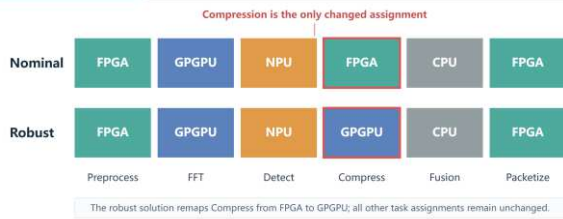


Fig. 4. Nominal and robust task-resource assignments. Compression is the only changed assignment in the canonical case.

Table 5. Nominal and robust task-resource mapping

Task	Nominal trade-off	Robust trade-off	Design implication
Preprocessing	FPGA	FPGA	Streaming I/O remains on FPGA
FFT	GPGPU	GPGPU	Data-parallel transform remains on GPGPU
Detection	NPU	NPU	Neural inference remains on NPU
Compression	FPGA	GPGPU	Robust mapping trades energy for timing margin
Fusion	CPU	CPU	Control-heavy logic remains on CPU

Task	Nominal trade-off	Robust trade-off	Design implication
Packetization	FPGA	FPGA	I/O formatting remains on FPGA

E. Sensitivity and Robustness Replay

Sensitivity sweeps produced three distinct signatures (Fig. 5). Reducing bandwidth scale from 1.0 to 0.6 reduced feasible mappings from 15 to 9 and increased the best makespan from 763.272 to 786.050 ms. The power screen changed from zero feasible mappings at 165 W to 15 at 175 W. Raising the thermal limit from 42 to 48 °C increased the feasible count from 12 to 18, after which the count saturated in the tested range.

Each one-factor sweep used five bandwidth scales, five power budgets, or five temperature limits. These deterministic sweeps locate active boundaries in the synthetic case; uncertainty in the parameter values is addressed separately by replay.

Replay used 1000 independent samples for every strategy-strength pair. At strength 0.25, the nominal mapping missed the deadline in 758 samples (75.8%; Wilson 95% CI, 73.05%-78.35%). Its count rose to 969 at strength 1.0 (96.9%; 95.63%-97.81%). Robust selection had no observed violation through strength 1.0, with a Wilson 95% upper bound of 0.383%.

Outside the declared set, the robust violation count rose to 112/1000, 272/1000, and 477/1000 at strengths 1.25, 1.50, and 2.00, respectively. Corresponding Wilson intervals are included in robustness_replay.csv. The nominal and robust series use explicitly recorded deterministic seeds at every point.

Narrow final-task margin accounts for the nominal mapping's high violation rate under the specified sampling model. Operational spacecraft failure probability cannot be inferred because the input distributions are synthetic and independent.

Sensitivity and replay answer different engineering questions. The former locates a parameter that constrains the feasible set; the latter tests a chosen mapping under declared uncertainty. Read together, Figs. 5 and 6 and Table 6 suggest a practical sequence: find the active constraint, calibrate its variation, and retest the mapping.

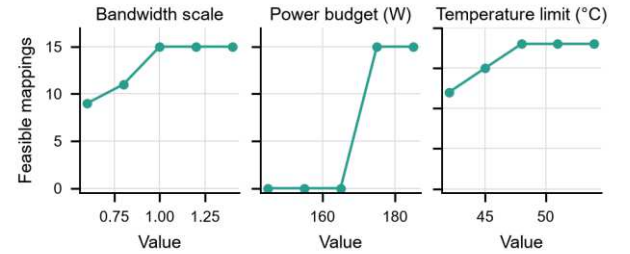


Fig. 5. One-factor sensitivity of feasible-mapping count to bandwidth scale, peak-power budget, and temperature limit.

Table 6. Sensitivity and robustness-replay summary

Diagnostic	Condition	Observed result	Interpretation
Bandwidth sensitivity	Scale 0.6 $\rightarrow$ 1.0 $\rightarrow$ 1.4	9 $\rightarrow$ 15 $\rightarrow$ 15 feasible mappings	Link improvement helps until another constraint dominates
Power sensitivity	165 W $\rightarrow$ 175 W	0 $\rightarrow$ 15 feasible mappings	Power behaves as a hard threshold
Thermal sensitivity	42 C $\rightarrow$ 45 C $\rightarrow$ 48 C	12 $\rightarrow$ 15 $\rightarrow$ 18 feasible mappings	Thermal relaxation gradually expands the domain

Diagnostic	Condition	Observed result	Interpretation
Nominal replay	Disturbance scale 1.0	969/1000 (96.9%; 95% CI 95.63%-97.81%)	Nominal margin is insufficient
Robust replay	Disturbance scale 1.0	0/1000 (95% upper bound 0.383%)	No violations were observed within the declared box
Robust boundary	Disturbance scale 2.0	477/1000 (47.7%; 95% CI 44.62%-50.80%)	Protection does not extend beyond calibration

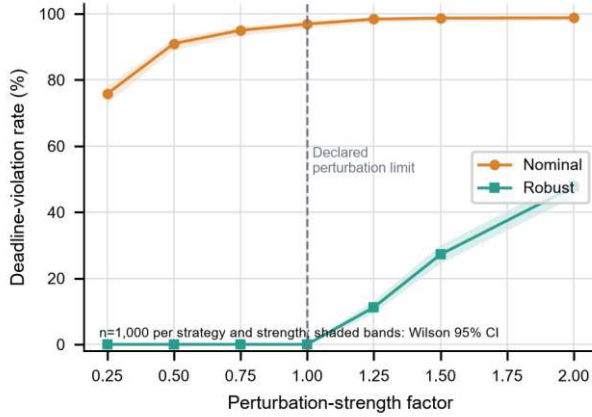


Fig. 6. Deadline-violation frequency under seeded perturbation replay. Shaded bands are Wilson 95% intervals; $n=1000$ per strategy and strength.

VI. DISCUSSION

A. Main Design Implication

Peak capability failed to rank mappings consistently once task class, transfer placement, power overlap, and repeated thermal response shared one deadline. More useful was the diagnostic record: which checks removed mappings, and why did robust selection change one assignment?

In this configuration, moving compression from FPGA to GPGPU exchanged 1.51% more nominal energy for 122.701 ms less makespan. Figure 4 identifies the responsible task, keeping the result local to the tested configuration and away from a universal resource recommendation.

Feasibility should be inspected through its margins. A mapping that clears every constraint comfortably is a plausible candidate for hardware validation. One that survives on a single narrow margin remains calibration-sensitive. Exposing that distinction before board integration is the main practical value of the workflow.

Changed requirements can be applied to the same mapping space. A tighter temperature limit, lower power budget, or larger data edge may preserve the selected mapping, favor another placement, or reveal the need for an added resource or faster link.

B. Distinction from Prior Analytical Boundaries

Our scope is offline intra-platform task-resource mapping with executable link, power, thermal, and reliability screens. Objective-only heterogeneous schedulers rank candidates without first reporting this feasible set, while satellite schedulers and space-computing frameworks [11]-[14], [21]-[25], [37]-[46], [48] address dynamic, cross-location, platform, or flight-software decisions. These distinctions concern workflow and scope; the six-task enumerator is not presented as a replacement for those systems.

Communication is induced by task placement and evaluated with each mapping (Fig. 2). The released model uses a generic

startup-plus-bandwidth link and therefore supports PCIe-, AXI-, or RoCEv2-like parameterization, but it does not validate any one protocol implementation.

Reliability remained inactive at the selected threshold: it removed no additional mappings, although mission risk affected ranking. The present experiment therefore validates the reliability calculation path, not a reliability advantage. A harsher or hardware-calibrated exposure case is needed to evaluate reliability-driven mapping changes.

Compared with purely robust optimization, the paper emphasizes interpretability. The robust budget is tied to compute latency, bandwidth and failure-rate perturbations, and its effect is checked by replay. This avoids a black-box conservative result whose cost is hard to explain. The designer can see which task moved, how much energy changed, how much margin was gained and where the protection boundary ends.

C. Calibration and Verification Pathway

Hardware calibration would begin with task capability measured from microbenchmarks and traces. The deployed interconnect would supply startup and bandwidth measurements, while idle and activity power would be separated experimentally. Board-level temperature traces would then determine the RC recurrence. In that role, the model acts as a checklist for design screening.

Validation should proceed in stages. Software-in-the-loop experiments can explore feasible-domain structure under controlled parameter sweeps. Hardware-in-the-loop replay can then test the exact board, bus and interconnect behavior. Finally, environmental testing can update the reliability and thermal parameters that determine whether the robust set remains valid.

Calibration stage one targets task families, not isolated kernels. For each task family, the team measures execution time over representative input sizes, record memory traffic where counters are available and identify the resource modes that change startup overhead. These measurements populate the effective-capability matrix used in the model.

Stage two covers communication edges. For each dependency class, the team measures startup latency, sustained bandwidth and copy or protocol overhead under the same software stack expected in deployment. RoCEv2, PCIe and local-memory movement should not be collapsed into one average value because they respond differently to data size and contention.

Stage three connects power and temperature. Idle baseline power is measured separately from task increments so that a schedule does not receive an artificial energy benefit from hiding idle residency. Thermal recurrence parameters are fitted under repeated task periods, because a single pass through the workload may miss accumulation that appears over many cycles.

D. Limitations

All evidence reported here is synthetic and model-level. It establishes executable consistency and comparative behavior for one canonical configuration. Processor-board accuracy, interconnect behavior, operating-system overhead, enclosure

thermals, and radiation response remain untested because no hardware, in-orbit, or fault-injection records were used.

Several assumptions delimit the evaluator: independent perturbations, non-preemptive execution, transient task memory, first-order per-resource RC dynamics, and independent rare-event reliability. Congestion, persistent model residency, shared-memory interference, active thermal control, retries, checkpoints, redundancy, and common-cause failures could alter both feasibility and the worst corner.

Exhaustive enumeration provides a transparent optimum only for the six-task, four-resource case. Algorithmic scaling beyond 4096 mappings was not evaluated. Larger graphs require MILP, decomposition, pruning, or heuristics, followed by the same constraint checks.

Finally, the paper does not include a physical fault-injection campaign. Robust replay perturbs modeled latency, bandwidth and failure rate, which is appropriate for checking the internal robust-design contract. It cannot replace experiments that disturb clocks, links, processor frequency, memory arbitration or thermal boundary conditions on real hardware.

VII. CONCLUSION

Evaluation begins with task-dependent capability, mapping-induced communication, event-based power, repeated-cycle thermal response, and segment reliability. Time-energy-risk weights are applied only after these checks. A monotone box uncertainty set is screened at its worst corner.

Compared with nominal energy-oriented selection, robust selection increased energy by 1.51% and reduced makespan by 122.701 ms. No deadline violation was observed in 1000 replay samples through the declared perturbation strength.

Within the released synthetic configuration, the results support reproducible early-stage design screening. Extending the conclusions to a flight system requires task profiling, link microbenchmarks, board-level power and thermal identification, and radiation or fault-injection data. The corresponding hardware and software validation burden is illustrated by current spaceborne-computing platforms and research infrastructure [37]-[46].

APPENDIX A. REPRODUCIBILITY PARAMETERS AND PROTOCOL

One canonical configuration supplies every reported parameter. Values were retained from the executable simulation record and are identified as synthetic model inputs. Appendix A and the supplementary_simulation package provide the matrices, mapping screens, scripts, hashes, and figure sources needed to audit the rerun.

A. Task and Resource Matrices

Table A1. Canonical task-requirement matrix

Task/class	GOP	Traffic (GB)	Work+buf (GB)	Release/deadline (ms)	Vulnerability
preprocess / stream	120	18	1.2+0.4	0/350	0.8
fft / fft	320	28	2.0+0.5	0/520	1.0
detect / ai	900	34	3.0+0.8	0/620	1.2
compress / codec	210	20	1.5+0.4	0/860	0.9
fusion / control	80	6	0.8+0.2	0/860	1.1
packetize / stream	60	8	0.6+0.2	0/980	0.8

Table A2. Canonical resource-performance and failure-rate matrix

Resource	Peak compute	Memory BW	Memory	Active/idle	Failure rate (s ⁻¹)
CPU	850 GOP/s	42 GB/s	16 GB	55/15 W	2.00e-08
GPGPU	6200 GOP/s	320 GB/s	16 GB	78/18 W	3.50e-08
NPU	12000 GOP/s	180 GB/s	16 GB	72/12 W	3.00e-08
FPGA	1800 GOP/s	95 GB/s	4 GB	38/8 W	2.50e-08

B. Directed Link Matrix

Table A3. Complete directed inter-resource link-parameter matrix

Directed link	Latency (ms)	BW (GB/s)	Energy (nJ/B)	Failure rate (s ⁻¹)	Vulnerability
CPU->GPGPU	0.020	12.0	1.8	1.00e-09	1.0
GPGPU->CPU	0.020	12.0	1.8	1.00e-09	1.0
CPU->NPU	0.025	10.0	1.7	1.00e-09	1.0
NPU->CPU	0.025	10.0	1.7	1.00e-09	1.0
CPU->FPGA	0.018	8.0	1.4	1.00e-09	0.9
FPGA->CPU	0.018	8.0	1.4	1.00e-09	0.9
GPGPU->NPU	0.055	7.5	2.4	1.50e-09	1.1
NPU->GPGPU	0.055	7.5	2.4	1.50e-09	1.1
GPGPU->FPGA	0.050	7.0	2.2	1.50e-09	1.0
FPGA->GPGPU	0.050	7.0	2.2	1.50e-09	1.0
NPU->FPGA	0.060	6.5	2.1	1.50e-09	1.0
FPGA->NPU	0.060	6.5	2.1	1.50e-09	1.0

C. Thermal Model Parameters

Table A4. First-order per-resource RC thermal parameters

Resource	Rth (K/W)	Cth (J/K)	RthCth (s)	a	b (K/W)
CPU	0.34	110	37.4	0.971017	0.009854
GPGPU	0.30	145	43.5	0.975030	0.007491
NPU	0.32	125	40.0	0.972875	0.008680
FPGA	0.40	95	38.0	0.971468	0.011413

For each resource, $a = \exp[-T_{\text{period}}/(R_{\text{th}}C_{\text{th}})]$ and $b = R_{\text{th}}(1-a)$. The recurrence uses $T_{\text{period}} = 1.10$ s, 180 cycles, a 28 °C coolant temperature, and a 45 °C upper limit.

D. Random Replay Samples and Seeds

Table A5. Random-replay sample counts and exact deterministic seeds

Replay	Strength s	N/strategy	Nominal seed	Robust seed
Main paired	declared box	500	20260615	20260615
Boundary	0.25	1000	20260642	20260643
Boundary	0.50	1000	20260667	20260668
Boundary	0.75	1000	20260692	20260693
Boundary	1.00	1000	20260717	20260718
Boundary	1.25	1000	20260742	20260743
Boundary	1.50	1000	20260767	20260768
Boundary	2.00	1000	20260817	20260818

Main paired replay multipliers are independent draws from $U[0.95,1.15]$ for execution, $U[0.80,1.05]$ for bandwidth, and $U[0.90,1.20]$ for failure rate; the same seed supplies common random numbers. At boundary strength s, the corresponding distributions are $U[1,1+0.15s]$, $U[\max(0.30,1-0.20s),1.05]$, and $U[1,1+0.20s]$. Wilson 95% intervals are reported for every binomial violation frequency.

E. Supplementary Package

The reviewer reproducibility package contains the canonical JSON configuration; task, edge, resource, efficiency, setup, link, and control CSV files; executable source code; complete mapping-screen outputs; feasible and Pareto subsets; strategy, sensitivity, and replay results; English reproduction instructions; and SHA-256 integrity records. It excludes Python caches, temporary files, hardware telemetry, and unverified external datasets.

DATA AND CODE AVAILABILITY

The supplementary file 'Spaceborne_Heterogeneous_HPC20260720_reviewer_reproducibility_package.zip' accompanies this submission. It

contains the canonical synthetic input configuration and matrices ('data/'), executable Python source ('src/'), and the complete mapping screen and derived results ('results/'). 'reproduce_all.py' regenerates the numerical outputs and figures; 'verify_results.py' checks the delivered screening counts and headline values; 'SHA256SUMS.txt' verifies the supplied files. The archive contains no hardware benchmark data, in-orbit telemetry, radiation-test measurements, proprietary operational data, or third-party restricted data. All reported results are deterministic or seeded outputs from the included source code and synthetic model inputs; the archive supports reproduction of the model-level claims only.

Funding

This paper received no external funding.

REFERENCES

- [1] H. Topcuoglu, S. Hariri, and M.-Y. Wu, "Performance-effective and low-complexity task scheduling for heterogeneous computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 13, no. 3, pp. 260-274, 2002, doi: 10.1109/71.993206.
- [2] Y.-K. Kwok and I. Ahmad, "Static scheduling algorithms for allocating directed task graphs to multiprocessors," *ACM Comput. Surv.*, vol. 31, no. 4, pp. 406-471, 1999, doi: 10.1145/344588.344618.
- [3] T. D. Braun et al., "A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems," *J. Parallel Distrib. Comput.*, vol. 61, no. 6, pp. 810-837, 2001, doi: 10.1006/jpdc.2000.1714.
- [4] S. Williams, A. Waterman, and D. Patterson, "Roofline: An insightful visual performance model for multicore architectures," *Commun. ACM*, vol. 52, no. 4, pp. 65-76, 2009, doi: 10.1145/1498765.1498785.
- [5] R. W. Hockney, "The communication challenge for MPP: Intel Paragon and Meiko CS-2," *Parallel Comput.*, vol. 20, no. 3, pp. 389-398, 1994, doi: 10.1016/0167-8191(94)90021-3.
- [6] A. P. Chandrakasan, S. Sheng, and R. W. Brodersen, "Low-power CMOS digital design," *IEEE J. Solid-State Circuits*, vol. 27, no. 4, pp. 473-484, 1992, doi: 10.1109/4.126534.
- [7] W. Huang et al., "HotSpot: A compact thermal modeling methodology for early-stage VLSI design," *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 14, no. 5, pp. 501-513, 2006, doi: 10.1109/TVLSI.2006.876103.
- [8] K. Skadron et al., "Temperature-aware microarchitecture," in *Proc. ACM/IEEE Int. Symp. Comput. Archit.*, 2003, pp. 2-13, doi: 10.1145/859618.859620.
- [9] I. Koren and C. M. Krishna, *Fault-Tolerant Systems*. San Francisco, CA, USA: Morgan Kaufmann, 2020, ISBN: 978-0-12-818105-8.
- [10] A. Avizienis, J.-C. Laprie, B. Randell, and C. Landwehr, "Basic concepts and taxonomy of dependable and secure computing," *IEEE Trans. Dependable Secure Comput.*, vol. 1, no. 1, pp. 11-33, 2004, doi: 10.1109/TDSC.2004.2.
- [11] C. Wang, X. Ma, C. Chi, A. Zhou, R. Xing, and S. Wang, "Temperature- and energy-aware dynamic task scheduling and computing resource allocation for satellite computing," *IEEE Trans. Mobile Comput.*, vol. 25, no. 5, pp. 6245-6260, 2026, doi: 10.1109/TMC.2025.3637314.
- [12] S. Mitra, "Constraint-aware execution planning for hybrid space-ground compute workloads," *arXiv:2605.04052*, 2026, doi: 10.48550/arXiv.2605.04052.
- [13] A. K. Erol and D. Mahajan, "Equinox: Decentralized scheduling for hardware-aware orbital intelligence," *arXiv:2604.19958*, 2026, doi: 10.48550/arXiv.2604.19958.
- [14] R. Thummala and G. Falco, "When to compute in space," *arXiv:2512.17054*, 2025, doi: 10.48550/arXiv.2512.17054.
- [15] Z. Hanzálek, O. Benedikt, P. Šůcha, P. Zaykov, and M. Sojka, "Thermal modeling and optimal allocation of avionics safety-critical tasks on heterogeneous MPSoCs," *J. Parallel Distrib. Comput.*, vol. 203, Art. no. 105107, 2025, doi: 10.1016/j.jpdc.2025.105107.
- [16] D. Bertsimas and M. Sim, "The price of robustness," *Oper. Res.*, vol. 52, no. 1, pp. 35-53, 2004, doi: 10.1287/opre.1030.0065.
- [17] A. Ben-Tal, L. El Ghaoui, and A. Nemirovski, *Robust Optimization*. Princeton, NJ, USA: Princeton Univ. Press, 2009, ISBN: 9780691143682.
- [18] Y. Zhu et al., "Congestion control for large-scale RDMA deployments," in *Proc. ACM SIGCOMM*, 2015, pp. 523-536, doi: 10.1145/2785956.2787484.
- [19] R. Mittal et al., "Revisiting network support for RDMA," in *Proc. ACM SIGCOMM*, 2018, pp. 313-326, doi: 10.1145/3230543.3230557.
- [20] N. Hanford, V. Ahuja, M. Farrens, B. Tierney, and D. Ghosal, "A survey of end-system optimizations for high-speed networks," *ACM Comput. Surv.*, vol. 51, no. 3, Art. no. 54, 2018, doi: 10.1145/3184899.
- [21] V. Leon, G. Lentaris, E. Petrongonas, D. Soudris, G. Furano, A. Tavoularis, and D. Moloney, "Improving performance-power-programmability in space avionics with edge devices: VBN on Myriad2 SoC," *ACM Trans. Embedded Comput. Syst.*, vol. 20, no. 3, 2021, doi: 10.1145/3440885.
- [22] F. C. Bruhn, N. Tsog, F. Kunkel, O. Flordal, and I. Troxel, "Enabling radiation tolerant heterogeneous GPU-based onboard data processing in space," *CEAS Space J.*, vol. 12, pp. 551-564, 2020, doi: 10.1007/s12567-020-00321-9.
- [23] A. Mohammadjafari and P. Khajouie, "Optimizing task scheduling in heterogeneous computing environments: A comparative analysis of CPU, GPU, and ASIC platforms using E2C Simulator," *arXiv:2405.08187*, 2024, doi: 10.48550/arXiv.2405.08187.
- [24] A. D. George and C. M. Wilson, "Onboard processing with hybrid and reconfigurable computing on small satellites," *Proc. IEEE*, vol. 106, no. 3, pp. 458-470, 2018, doi: 10.1109/JPROC.2018.2802438.
- [25] V. Leon et al., "FPGA & VPU co-processing in space applications: Development and testing with DSP/AI benchmarks," in *Proc. 28th IEEE Int. Conf. Electronics, Circuits and Systems*, 2021, pp. 1-5, doi: 10.1109/ICECS53924.2021.9665462.
- [26] H. Esmailzadeh et al., "Dark silicon and the end of multicore scaling," in *Proc. ACM/IEEE Int. Symp. Comput. Archit.*, 2011, pp. 365-376, doi: 10.1145/2000064.2000108.
- [27] H. Arabnejad and J. G. Barbosa, "List scheduling algorithm for heterogeneous systems by an optimistic cost table," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 3, pp. 682-694, 2014, doi: 10.1109/TPDS.2013.57.
- [28] J. Zhou, K. Cao, P. Cong, T. Wei, M. Chen, G. Zhang, J. Yan, and Y. Ma, "Reliability and temperature constrained task scheduling for makespan minimization on heterogeneous multi-core platforms," *J. Syst. Softw.*, vol. 133, pp. 1-16, 2017, doi: 10.1016/j.jss.2017.07.032.
- [29] P. Mukherjee, K. Jain, and S. Chattopadhyay, "Thermal-aware task allocation and scheduling for periodic real-time applications in mesh-based heterogeneous NoCs," *Real-Time Syst.*, vol. 55, pp. 774-809, 2019, doi: 10.1007/s11241-019-09327-x.
- [30] A. Roy, H. Aydin, and D. Zhu, "Energy-aware primary/backup scheduling of periodic real-time tasks on heterogeneous multicore systems," *Sustain. Comput.: Inform. Syst.*, vol. 29, Art. no. 100474, 2021, doi: 10.1016/j.suscom.2020.100474.
- [31] B. Alahmad and S. Gopalakrishnan, "Energy efficient task partitioning and real-time scheduling on heterogeneous multiprocessor platforms with QoS requirements," *Sustain. Comput.: Inform. Syst.*, vol. 1, no. 4, pp. 314-328, 2011, doi: 10.1016/j.suscom.2011.09.001.
- [32] M. Thammawichai and E. C. Kerrigan, "Energy-efficient real-time scheduling for two-type heterogeneous multiprocessors," *Real-Time Syst.*, vol. 54, pp. 132-165, 2018, doi: 10.1007/s11241-017-9291-6.
- [33] T. Nakada, H. Yanagihashi, K. Imai, H. Ueki, T. Tsuchiya, M. Hayashikoshi, and H. Nakamura, "An energy-efficient task scheduling for near real-time systems on heterogeneous multicore processors," *IEICE Trans. Inf. Syst.*, vol. E103-D, no. 2, pp. 329-338, 2020, doi: 10.1587/transinf.2019EDP7101.
- [34] Q. Chen, Y. Han, J. Wu, and Y. Gan, "Energy-saving task scheduling based on hard reliability requirements: A novel approach with low energy consumption and high reliability," *Sustainability*, vol. 14, no. 11, Art. no. 6591, 2022, doi: 10.3390/su14116591.
- [35] H. Aydin, R. Melhem, D. Mossé, and P. Mejía-Alvarez, "Dynamic and aggressive scheduling techniques for power-aware real-time systems," in *Proc. IEEE Real-Time Systems Symp.*, 2001, pp. 95-105, doi: 10.1109/REAL.2001.990600.
- [36] J. Zhou, J. Yan, K. Cao, Y. Tan, T. Wei, M. Chen, G. Zhang, X. Chen, and S. Hu, "Thermal-aware correlated two-level scheduling of real-time tasks with reduced processor energy on heterogeneous MPSoCs," *J. Syst. Archit.*, vol. 82, pp. 1-11, 2018, doi: 10.1016/j.sysarc.2017.09.007.
- [37] F. C. Bruhn et al., "Introducing radiation tolerant heterogeneous computers for small satellites," in *Proc. IEEE Aerospace Conf.*, 2015, pp. 1-10, doi: 10.1109/AERO.2015.7119158.
- [38] L. Kosmidis, J. Lachaize, J. Abella, O. Notebaert, F. J. Cazorla, and D. Steenari, "GPU4S: Embedded GPUs in space," in *Proc. Euromicro Conf. Digital System Design*, 2019, pp. 399-405, doi: 10.1109/DSD.2019.00064.
- [39] N. Tsog, M. Behnam, M. Sjödin, and F. Bruhn, "Intelligent data processing using in-orbit advanced algorithms on heterogeneous system architecture," in *Proc. IEEE Aerospace Conf.*, 2018, pp. 1-8, doi: 10.1109/AERO.2018.8396536.
- [40] C. Adams et al., "Towards an integrated GPU accelerated SoC as a flight computer for small satellites," in *Proc. IEEE Aerospace Conf.*, 2019, pp. 1-7, doi: 10.1109/AERO.2019.8741765.
- [41] S. Chien et al., "Onboard autonomy on the Intelligent Payload EXperiment (IPEX) CubeSat mission," *J. Aerosp. Inf. Syst.*, 2016, 14(6):1-9, doi: 10.2514/1.1010386.
- [42] National Aeronautics and Space Administration, "High-Performance Spaceflight Computing (HPSC) project overview," *NASA Technical Reports Server*, NTRS 20180007636, 2018.
- [43] National Aeronautics and Space Administration, "High Performance Spacecraft Computing (HPSC) middleware update," *NASA Technical Reports Server*, NTRS 20190033894, 2019.
- [44] National Aeronautics and Space Administration, "SpaceCube Edge TPU SmallSat card for autonomous operations and onboard science-data analysis," *NASA Technical Reports Server*, NTRS 20210019764, 2021.

- [45] National Aeronautics and Space Administration, "NASA's vision for spaceflight computing," NASA Technical Reports Server, NTRS 20220015330, 2022.
- [46] E. Gizzi et al., "OnAIR: Applications of the NASA On-Board Artificial Intelligence Research Platform," *Proc. AAAI Conf. Artif. Intell.*, vol. 39, no. 28, pp. 28893-28899, 2025, doi: 10.1609/aaai.v39i28.35156.
- [47] G. Taheri, A. Khonsari, R. Entezari-Maleki, and L. Sousa, "A hybrid algorithm for task scheduling on heterogeneous multiprocessor embedded systems," *Appl. Soft Comput.*, vol. 91, Art. no. 106202, 2020, doi: 10.1016/j.asoc.2020.106202.
- [48] X. Fan and X. Li, "Thermal-aware scheduling of hybrid mission workflows in software-defined satellites," in *Proc. 8th Int. Conf. Algorithms, Computing and Artificial Intelligence*, 2025, doi: 10.1109/ACA168217.2025.11406370.
- [49] Q. Tan, W. Chen, D. Liu, and R. Murray, "Task scheduling for heterogeneous multi-core processors based on deep reinforcement learning," *Int. J. Intell. Syst.*, 2025, doi: 10.1155/int/7562400.
- [50] Z. Chen and D. Marculescu, "Task scheduling for heterogeneous multicore systems," *arXiv:1712.03209*, 2017, doi: 10.48550/arXiv.1712.03209.