Joint Computation and Communication Analysis of Hard Real-Time applications in Manycores

Anderson R. P. Domingues, Lucas Damo, Sergio J. Filho, and Fernando G. Moraes

{anderson.domingues, lucas.damo}@edu.pucrs.br; {sergio.filho, fernando.moraes}@pucrs.br

PUCRS – School of Technology, Porto Alegre, Brazil

Presenter: **Angelo Dal Zotto**

August 30, 2024

Overview

SBCCI2024

37th SYMPOSIUM ON INTEGRATED CIRCUITS AND SYSTEMS DESIGN

Introduction

- NoCs are attractive options for system interconnection in manycores due to their potential for massively parallel communication and scalability while inserting low overhead for area and energy consumption into the design.

- Routers provide the communication infrastructure in a typical NoC, allowing multiple data streams to simultaneously traverse the system [KJS+02, BDM02, JLV05].

- Some NoC designs acknowledge the implementation of NFR, e.g., security, safety [CM20, GHG+20], and real-time (*).

- The main concern of RT-NoCs is predictability — a particularly important property in control-theoretic, cyber-physical, and robotics domains due to the need for compliance with stringent safety constraints [AUT21, ISO20, VDA24].

Introduction (contd.)

- Designing hardware that accounts for multiple NFRs is challenging due to the implementation of conflicting NFRs, e.g., implementing a security feature may modify the timing* of the system. Similarly, predictable hardware may add energy and area overheads, due to additional circuitry.

- The lack of specialized literature aggravates the problem, as the techniques often manage NFRs as isolated.

- Also, studies on multiple NFRs, e.g., power and real-time [LLJ+23] propose techniques on non-publicly available platforms, making it unfeasible to replicate the techniques on industrial-grade projects.

SBCCI2024
37th SYMPOSIUM ON INTEGRATED CIRCUITS AND SYSTEMS DESIGN

Concerns of RT Analysis on Manycores

- The analysis of applications on NoC-based, private memory, and multitasking manycores must consider both the individual processing elements (PEs) and the underlying NoC. The analysis of PEs regards the CPU and I/O peripherals, while RT-NoCs partake in the RT guarantees for communication.

- Recent studies on RT-NoCs [DT18, CMMF20, PFHD19] do not approach computation, missing mechanisms for synchronizing processing and communication operations.

  - Considering commercial off-the-shelf (COTS) platforms, whose hardware cannot be changed, is it possible to run a given application and guarantee RT for tasks and traffic?
  - Assuming platforms where we can change the nominal frequency, what is the minimum frequency to address the RT requirements of an application? How can we address RT without tampering with other NFRs?

This paper...

- Addresses RT analysis on NoC-based, multitasking, private memory manycores whose hardware cannot be changed, e.g., COTS platforms, targeting manycores equipped with conventional, non-RT NoCs.

- As these platforms cannot take advantage of specific design-time optimizations [Pec20, BSP+16], we propose an off-line, *pre-runtime* framework for computation and communication RT analysis that can:
  1. determine if a given application meet its RT requirements, considering the architectural features and nominal operating frequency of the platform;
  2. find the minimum frequency to address an RT application in platforms whose frequency can be changed.

SBCCI2024

37th SYMPOSIUM ON INTEGRATED CIRCUITS AND SYSTEMS DESIGN

This paper... (contd.)

- At the computation scope, we combine a discrete-event simulation (DES) strategy with the critical-path analysis (CPA).

- The goal is to validate the CPU requirements for tasks at the PE level and the end-to-end application execution time at a system-wide level.

- We adopt a contention-free model for communication, assuming a pre-runtime optimization method.

- We assume the zero-load latency model of the NoC as a performance boundary.

- This assumption enables us to perform our analysis on *virtually any NoC*, adding RT support to systems without RT-NoCs.

- Our approach is *novel* because it supports COTS, is fully automated, and uses only open-source tools. We validate our approach in an open-hardware platform, favoring the reproducibility of our study.

Overview

1 Introduction

2 Baseline Platform

3 A Framework for Pre-Runtime RT Analysis

4 Proof-of-Concept and Results

5 Conclusions and Acknowledgements

Baseline Platform - Hardware

2D-mesh, credit-based, wormwhole, XY routing, manycore

(a)   (b)

Baseline Platform - Hardware (contd.)

- Network interface (NI) allows for DMA (like [RCFM19, RLMM16])
- CPU get interrupted by the NI once a packet is entirely received
- Sending a packet follows a similar strategy, where the network device driver configures the NI to copy the packet from RAM into the local port of the router.
- At the kernel level, the network driver has a queue of packets, where the NI stores packets. Tasks can probe the driver for packets or perform a self-blocking operation until a packet is received. The driver also allows tasks in the same PE to communicate by moving packets between queues.

SBCCI2024
37th SYMPOSIUM ON INTEGRATED CIRCUITS AND SYSTEMS DESIGN

Baseline Platform - Software

- PEs run the UCX-OS [Joh24] kernel
- Priority-based round-robin (PBRR) technique [NI23]
- Accounts for dependency. The kernel keeps track of the number of iterations of tasks as new packets arrive, prioritizing (i) tasks in the oldest iterations and (ii) tasks meeting their data dependencies in case of a tie.

SBCCI2024
37th SYMPOSIUM ON INTEGRATED CIRCUITS AND SYSTEMS DESIGN

Overview

SBCCI2024

37th SYMPOSIUM ON INTEGRATED CIRCUITS AND SYSTEMS DESIGN

Overview

- We propose a framework to identify whether a NoC-based manycore can run a real-time application without deadline violations.

- We implemented a couple of tools to automate our approach, which relies on the *scheduling* of system resources — computation and communication.

- Scheduling tasks on a single-core CPU is not novel, as some existing solutions date back to the 80s [Gus84].

- However, incorporating the I/O components of the system, managing interruptions, and addressing the needs of NoCs requires additional modeling.

SBCCI2024

37th SYMPOSIUM ON INTEGRATED CIRCUITS AND SYSTEMS DESIGN

- Our framework concerns dataflow applications, where tasks have data dependency on sensors, the network, data off-loading, or other tasks.

- In addition, tasks must employ a variation of the predictable execution (PREM) [SBZL22] and the logical execution time (LET) [GKEQ21] models; we divide tasks in *receiving*, *processing*, and *sending* phases as the controlled execution of tasks is key for I/O access predictability.

- We expect applications to have real-time requirements on (i) the iteration time and (ii) the number of iterations per second. The former is the end-to-end application execution time, which is particularly important in sensor applications as it dictates the accuracy of sensors in time, i.e., how old the reading is. The latter is the application execution rate, which relates to the periodic execution of tasks.

SBCCI2024
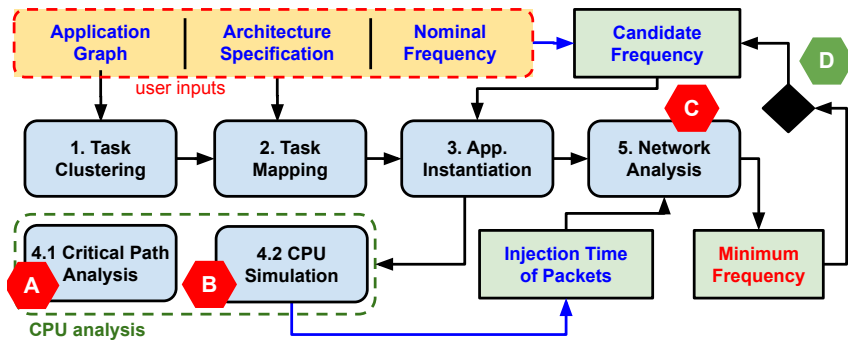37th SYMPOSIUM ON INTEGRATED CIRCUITS AND SYSTEMS DESIGN

Figure 1: A workflow depicting our approach. User inputs appear at the top. Activities appear as blue shapes. Outcomes appear as green shapes. Tags A, B, and C represent activities of the workflow that may fail due to the unavailability of system resources. Tag D represents the end of the workflow, where users can choose to abandon the workflow or further optimize the result.

The proposed framework (Figure 1) takes three inputs: (i) application graph, (ii) architecture specification, and (iii) nominal manycore frequency.

   i) We described applications using a directed, potentially cyclic, graph $G = (E, V)$, where V is the set of vertices and E is the set of edges. Vertices represent tasks, to which we tag the corresponding worst-case execution time (WCET), in cycles. Edges represent flows of data, i.e., a set of packets with periodic behavior, given by a 3-tuple $f = \langle p, c, d \rangle$, corresponding to the period (p), capacity (c), and deadline of flows (d), in clock cycles.

   ii) The architecture specification must describe a zero-load latency model of the network and a set of allowed operating frequencies for the manycore. Please note that the entire system runs at the same frequency.

   iii) The framework considers the nominal frequency as the start point for the frequency optimization.

SBCCI2024

37th SYMPOSIUM ON INTEGRATED CIRCUITS AND SYSTEMS DESIGN

- At the first iteration, the framework computes whether the nominal frequency suits the application.
- The framework runs in interactive or automated modes.
- In interactive mode, users must decide to increase the frequency (if the platform allows), try a lower frequency, or abandon the process at each iteration.
- In automated mode, the framework runs until it reaches a given number of iterations or cannot optimize the frequency further, assuming a tolerance factor.
- The framework evaluates the application at each iteration, using a 5-step analysis: (i) task clustering, (ii) task mapping, (iii) application instantiation, (iv) CPU analysis, and (v) network analysis. CPU analysis splits in critical path analysis (CPA) and CPU simulation.

SBCCI2024
37th SYMPOSIUM ON INTEGRATED CIRCUITS AND SYSTEMS DESIGN

1 - Task Set Clustering and Mapping

- Task set clustering consists of splitting an application graph into multiple task sets to be assigned by the framework to a PE during the mapping step (Figure 1, step 1).
- The goal is to generate one cluster per PE. We created an algorithm to cluster graphs that performs $O(n)$ on the number of vertices of the application task graph.
- The algorithm iteratively removes edges from the graph, combining vertices until the resulting graph has $|V| \leq |PEs|$, based on one of two criteria.
- The MIN-COMM criterion reduces the overall network load, eliminating edges with the most communication load groups communication-intensive tasks in the same cluster.
- Oppositely, the MAX-PROC criterion balances the CPU usage between PEs by grouping tasks with the least CPU load. Once the algorithm computes the

2 - Application Instantiation and Framework Loop

- As the framework explores different frequencies during its execution, it must adjust the application parameters to match the frequency at each iteration (Figure 1, step 3).
- The framework carries the WCET value of tasks along the multiple iterations, as the CPU architecture does not change. However, the framework must consider the periodicity of flows, adjusting the period and deadline of flows accordingly.
- Iterations occur as follows. At the first iteration, the framework takes the nominal frequency as the candidate frequency, employing no adjustments to the application.
- Based on the results of CPU analysis (Figure 1, step 4) and network analysis (Figure 1, step 5), the framework either increases or decreases the frequency. If the framework identifies violations of deadlines (tasks or flows), the candidate frequency will increase at the next iteration.

3.a - Critical Path Analysis (CPA)

- The CPU analysis is a 2-step process to assert that the manycore meets the CPU needs of an application.
- The framework uses the CPA method [KW59] to evaluate the end-to-end processing time of an application iteration (Figure 1, step 4.1).
- As PEs carry only subsets of the application task set, we must account for internal communication (task-to-task communication through memory spaces) and external communication (NoC).
- The CPA method proceeds as Djkistra's algorithm for computing the shortest path between two vertices [Dij59], except that we multiply the weights of edges (communication load) by -1.
- Although we cannot apply topological sort to cyclic graphs, we use depth-first search (DFS) to find and remove cycles, adding the weight of a removed edge to its origin vertex.
- The result of this step is the number of nodes that the application task set

3.b - Discrete-Event Simulation (DES) of PEs

- The execution time of some kernel routines, e.g., dynamic memory allocation (*malloc*) and interruption handling, is hard to predict. Our framework uses a DES engine to simulate PEs while assuming a worst-case characterization of kernel operations (Figure 1, step 4.2).
- It simulates PEs individually to find whether they can run the assigned task cluster. The simulation considers the WCET of the task scheduler, I/O interrupts, and other minor system-specific programmable interrupts.
- If the framework detects that one of the PEs cannot run the assigned task set, the framework discards the candidate frequency and moves to the next iteration.
- Due to the employment of the PREM and LET models, the framework can estimate the time in which tasks inject packets into the network. Packets enter the network only at the *sending* phase, triggering an I/O event in the kernel.
- The framework uses these values during the network analysis step (Figure 1, step

4 - Network Analysis

- The network analysis occurs after the CPU analysis, considering that the framework kept the candidate frequency.

- In this step, the framework asserts whether the network flows can traverse the NoC without deadline violations for a static set of flows.

- Our flow model reassembles the Job-Shop model [MRG79], where we replace machines with links (channels) and jobs with flits (data units). The goal of the network analysis step is to find a schedule for packets. The network analysis has 3 steps:

SBCCI2024
37th SYMPOSIUM ON INTEGRATED CIRCUITS AND SYSTEMS DESIGN

4 - Network Analysis (contd.)

1. Unwrap flows to packets $p = \langle mrt, size, ad \rangle$, having a minimum release time (mrt), data size ($size$), and absolute deadline (ad) each.

2. Discover the path of packets in the NoC to compute the occupancy of links (L), i.e., a relation $O : P \times L \times T$, matching packets to the links they occupy in the discrete time domain (cycles), where $\forall p \in P$.

3. Find a schedule where the following constraints hold:
   - (c1) $prt \geq mrt$, where ptr is the release time of packets in the found schedule;
   - (c2) $ad \geq prt + size/lw$, where lw is the link width;
   - (c3) packets cannot overlap (single channel constraint);
   - (c4) flits of a same packet occupy the data bus one after another (wormhole constraint).

SBCCI2024

37th SYMPOSIUM ON INTEGRATED CIRCUITS AND SYSTEMS DESIGN

Beware!

- Generating a network schedule is NP-complete (similarly to job-shop).

- Instead, we use the *prt* values collected during the CPU analysis, reducing the scheduling to a decision problem.

- The framework indicates whether the schedule is feasible or not. If the schedule is feasible, the framework *memorizes* the minimum frequency found so far.

- Then, the framework can either try to find a lower frequency or abandon the process.

SBCCI2024

37th SYMPOSIUM ON INTEGRATED CIRCUITS AND SYSTEMS DESIGN

Overview

1 Introduction

2 Baseline Platform

3 A Framework for Pre-Runtime RT Analysis

4 Proof-of-Concept and Results

5 Conclusions and Acknowledgements

Experimentation Setup

- We experiment on the SAE (synthetic) application whose minimum frequency to meet its RT requirements is known (500MHz).
- We map SAE to a 2x2 instance of our manycore
- We perform clustering using the *max-proc* criterion and $n = 4$.
- The nominal frequency of our platform is 2.5GHz, which is also given as input to our framework.
- By the last iteration, our framework found 500.4MHz as the minimum frequency to meet the RT constraints on the platform. Since we cannot display the results for all iterations, we couple to show the last step and demonstrate how the found frequency meets the requirement of the application (the last frequency).

SBCCI2024
37th SYMPOSIUM ON INTEGRATED CIRCUITS AND SYSTEMS DESIGN

- The CPA method estimated the end-to-end iteration execution time of SAE based on the application graph and the WCET of the kernel operations.

- Our scheduler has a WCET of 110 kcycles, and our network driver has a WCET of 20 kcycles. Tasks are scheduled once per iteration due to our PBRR implementation. The total iteration execution time, as pointed out by the CPA method, is 8.464.895 cycles.

- The actual RTL execution is 6.799.200 cycles (80%). The CPA method overestimates $\simeq 26\%$ of the actual CPU usage. However, the overestimation is *acceptable* ($< 30\%$) as the actual CPU usage stays below 70%; a CPU usage over 70% is often considered questionable [LO11].

SBCCI2024

37th SYMPOSIUM ON INTEGRATED CIRCUITS AND SYSTEMS DESIGN

Table 1: Comparison of Execution time (CPU).

Cluster	Execution[1]	Estimation[2]	Difference	Diff. (%)[3]	Usage[4]
ABDF	$1,439,000$	$1,819,300$	$+380,300$	$+26.428\%$	72.722%
C	$801,300$	$963,300$	$+162.000$	$+20.217\%$	38.532%
EHI	$2,174,500$	$2,667,895$	$+493,395$	$+22.690\%$	71.616%
GJ	$2,384,400$	$3,014,400$	$+630.000$	$+26.421\%$	86.980%

[1]Execution time (RTL, cycles); [2]CPA estimation; [3]Execution time is 100%; [4]DES.

SBCCI2024
37th SYMPOSIUM ON INTEGRATED CIRCUITS AND SYSTEMS DESIGN

- The framework captured the time that tasks finished during the DES simulation. As tasks implement the PREM and LET models, we assume that packets are injected at the same time they leave the CPU.

- Table 2 shows the characterization of flows, where ph is the phase time (the period of scheduling interrupts), pv is the *phase* of the task, and $\psi = (ph \times pv)$.

- A phase corresponds to the alignment of application execution to the scheduler tick. For instance, the SAE application cannot finish before the 4th phase due to the critical path traverses 4 tasks in the application graph. This is a necessary step, as the DES simulation does not account for task dependency. Deadlines match the end of the phase of the receiving task.

SBCCI2024

37th SYMPOSIUM ON INTEGRATED CIRCUITS AND SYSTEMS DESIGN

Table 2: Characterization of flows for the SAE application.

Flow	MRT[1]	S	D	Volume	Deadline
F_1	$285300 + \psi$	A	C	12300	$110000 + \psi$
F_2	$1819300 + \psi$	F	G	67800	$110000 + \psi$
F_3	$963300 + \psi$	C	E	12000	$110000 + \psi$
F_4	$1280700 + \psi$	I	J	56700	$1208800 + \psi$
F_5	$1733700 + \psi$	H	J	89000	$1208800 + \psi$

(S) source task, (D) destination task, [1]Minimum release time.

SBCCI2024
37th SYMPOSIUM ON INTEGRATED CIRCUITS AND SYSTEMS DESIGN

- In Figure 2, we present the RTL simulation of our manycore running the SAE application at the minimum frequency of 500.4MHz, obtained from our framework. The goal is to demonstrate that the application meets its RT requirements.

- The hyperperiod of SAE is 20ms, corresponding to 4 phases of 5ms each, meeting the expected iteration time and iterations per second requirements. SAE receives stimuli from outside the system once per 5ms, triggering 4 instances of the application per $20ms$ (Task A). SAE returns results to outside the system via Task J.

- The 3rd instance of the application (pink) has additional 1.457ms in its iteration time, related to the scheduling of Task I; the task executes two iterations at once due to two packets received in the phase time, delaying Task J. Nevertheless, Task J meets its deadlines for all instances of SAE, for an iteration time of $16,820$ms for the 1st, 2nd and 3th instances, and 18.277ms for the 3rd instance.
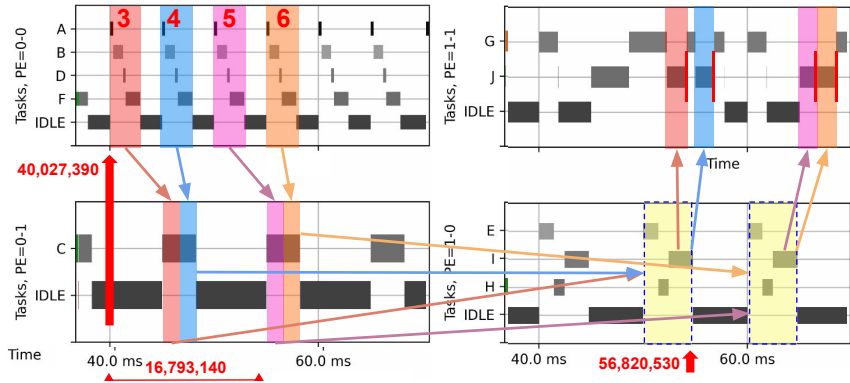
Figure 2: Interaction of tasks of the SAE application during RTL simulation. Rectangles with the same color represent tasks in the same iteration (instances). Arrows indicate the direction of the communication. The yellow rectangle represents a phase overlapping.

Overview

1 Introduction

2 Baseline Platform

3 A Framework for Pre-Runtime RT Analysis

4 Proof-of-Concept and Results

5 Conclusions and Acknowledgements

SBCCI2024
37th SYMPOSIUM ON INTEGRATED CIRCUITS AND SYSTEMS DESIGN

Conclusion

- This paper proposes a framework for asserting the RT properties of applications running on NoC-based, multitasking, private-memory manycores.

- Our framework observes the computation and communication aspects of the application, adjusting the frequency of the target platform to the minimum while allowing the application to meet its RT requirements.

- We demonstrate our framework on the SAE application, running on our manycore. Our framework could reduce the manycore frequency without compromising the RT requirements of SAE.

SBCCI2024

37th SYMPOSIUM ON INTEGRATED CIRCUITS AND SYSTEMS DESIGN

Conclusion (contd.)

- Our framework enables the exploration of RT at the pre-runtime, alleviating the effects of stacked NFRs in the design. We avoid modeling design-specific features in our framework, e.g., buffer, allowing our framework to suit virtually any NoC that provides a zero-load latency model.
- Future works include
    1. evaluating our approach while considering multiple NFRs, e.g. area and energy requirements;
    2. experiment with more complex applications [SBI10];
    3. make use of dynamic voltage-frequency scaling (DVFS) to improve energy efficiency without compromising real-time performance, and (iv) improve the automation of the approach, e.g., integration with ModelSim.

SBCCI2024
37th SYMPOSIUM ON INTEGRATED CIRCUITS AND SYSTEMS DESIGN

Acknowledgement

SBCCI2024
37th SYMPOSIUM ON INTEGRATED CIRCUITS AND SYSTEMS DESIGN

Joint Computation and Communication Analysis of Hard Real-Time applications in Manycores

Anderson R. P. Domingues, Lucas Damo, Sergio J. Filho, and Fernando G. Moraes

{anderson.domingues, lucas.damo}@edu.pucrs.br; {sergio.filho, fernando.moraes}@pucrs.br

PUCRS – School of Technology, Porto Alegre, Brazil

Presenter: **Angelo Dal Zotto**

August 30, 2024

References I

[AUT21]   AUTOSAR. Standards – AUTOSAR, 2021. Online. Visited in February 2021. https://www.autosar.org/standards/.

[BDM02]   L. Benini and G De Micheli. Networks on chip: a new paradigm for systems on chip design. In *Design, Automation Test in Europe Conference (DATE)*, pages 418–419, 2002. https://doi.org/10.1109/DATE.2002.998307.

[BSP+16]   B. Bohnenstiehl, A. Stillmaker, J. Pimentel, T. Andreas, B. Liu, A. Tran, E. Adeagbo, and B. Bass. A 5.8 pJ/Op 115 billion ops/sec, to 1.78 trillion ops/sec 32nm 1000-processor array. In *IEEE Symposium on VLSI Circuits (VLSIC)*, pages 1–2, 2016. https://doi.org/10.1109/vlsic.2016.7573511.

[CM20]   Subodha Charles and Prabhat Mishra. Securing Network-on-Chip Using Incremental Cryptography. In *IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, pages 168–175, 2020. https://doi.org/10.1109/ISVLSI49217.2020.00039.

[CMMF20]   Yong Chen, Emil Matus, Sadia Moriam, and Gerhard P. Fettweis. High performance dynamic resource allocation for guaranteed service in network-on-chips. *IEEE Transactions on Emerging Topics in Computing*, 8(2):503–516, 2020.

[Dij59]   E. W Dijkstra. A Note on Two Problems in Connexion with Graphs. *Numerische Mathematik*, 1:269–271, 1959. http://eudml.org/doc/131436.

[DT18]   Stanisław Deniziak and Robert Tomaszewski. Codesign of energy and resource efficient contention-free Network-on Chip for real-time embedded systems. In *International Workshop on Network on Chip Architectures (NoCArc)*, pages 1–6, 2018. https://doi.org/10.1109/NOCARC.2018.8541199.

[GHG+20]   Pengxing Guo, Weigang Hou, Lei Guo, Zizheng Cao, and Zhaolong Ning. Potential Threats and Possible Countermeasures for Photonic Network-on-Chip. *IEEE Communications Magazine*, 58(9):48–53, 2020. https://doi.org/10.1109/MCOM.001.2000029.

[GKEQ21]   Kai-Björn Gemlau, Leonie KÖHLER, Rolf Ernst, and Sophie Quinton. System-level Logical Execution Time: Augmenting the Logical Execution Time Paradigm for Distributed Real-time Automotive Software. *ACM Transactions on Cyber-Physical Systems*, 5(2):1–27, 2021. https://doi.org/10.1145/3381847.

References II

[Gus84]   Dan Gusfield. Bounds for naive multiple machine scheduling with release times and deadlines. *Journal of Algorithms*, 5(1):1–6, 1984.

[ISO20]   ISO. ISO 26262:9 — Automative Safety Integrity Level (ASIL), 2020. Online. Visited in February 2021. https://www.iso.org/obp/ui/#iso:std:iso:26262:-9:ed-2:v1:en.

[JLV05]   A. Jantsch, R. Lauter, and A Vitkowski. Power analysis of link level and end-to-end data protection in networks on chip. In *IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1770–1773, 2005. https://doi.org/10.1109/ISCAS.2005.1464951.

[Joh24]   Sergio Johann. UCX/OS - Microcontroller Executive / OS, 2024. Online. Visited in January 2024. https://github.com/sjohann81/ucx-os.

[KJS+02]   S. Kumar, A. Jantsch, J.-P. Soininen, M. Forsell, M. Millberg, J. Oberg, K. Tiensyrja, and A Hemani. A network on chip architecture and design methodology. In *IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, pages 117–124, 2002. https://doi.org/10.1109/ISVLSI.2002.1016885.

[KW59]   James E. Kelley and Morgan R Walker. Critical-path planning and scheduling. In *Eastern Joint IRE-AIEE-ACM Computer Conference*, 1959. https://doi.org/10.1145/1460299.1460318.

[LLJ+23]   Xin Li, Zhi Li, Yaqi Ju, Xiaofei Zhang, Rongyao Wang, and Wei Zhou. Cop: A combinational optimization power budgeting method for manycore systems in dark silicon. *IEEE Transactions on Computers*, 72(5):1356–1370, 2023.

[LO11]   Phillip A. Laplante and Seppo J. Ovaska. *Real-Time Operating Systems*. Wiley-IEEE Press, 4th edition, 2011.

[MRG79]   David S. Johnson Michael R. Garey. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Freeman, 1st edition, 1979.

[NI23]   N. Nithya and Srikanth Itapu. Design Of Low Area Interconnect Architecture for CPU-GPU Network-On-Chips (NoCs). In *IEEE International Conference on Electronics, Computing and Communication Technologies (CONECCT)*, pages 1–5, 2023. https://doi.org/10.1109/CONECCT57959.2023.10234778.

References III

[Pec20]   O. Peckham. Esperanto Unveils ML Chip with Nearly 1,100 RISC-V Cores, 2020.
           https://www.hpcwire.com/2020/12/08/esperanto-unveils-ml-chip-with-nearly-1100-risc-v-cores.

[PFHD19]   T. Picornell, J. Flich, C. Hernández, and J. Duato. DCFNoC: A Delayed Conflict-Free Time Division Multiplexing Network on Chip. In
           ACM/IEEE Design Automation Conference (DAC), pages 1–6, 2019. https://doi.org/10.1145/3316781.3317794.

[RCFM19]   Marcelo Ruaro, Luciano L. Caimi, Vinicius Fochi, and Fernando G Moraes. Memphis: a framework for heterogeneous many-core SoCs
           generation and validation. *Design Automation for Embedded Systems*, 23(4):103–122, 2019.
           https://doi.org/10.1007/s10617-019-09223-4.

[RLMM16]   Marcelo Ruaro, Felipe Lazzarotto, César Marcon, and Fernando Gehm Moraes. DMNI: A specialized network interface for NoC-based
           MPSoCs. In *IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1202–1205, 2016.
           https://doi.org/10.1109/ISCAS.2016.7527462.

[SBI10]   Zheng Shi, Alan Burns, and Leandro Indrusiak. Schedulability Analysis for Real Time On-Chip Communication with Wormhole Switching.
           International Journal of Embedded and Real-Time Communication Systems, 1:1–22, 2010.
           https://doi.org/10.4018/jertcs.2010040101.

[SBZL22]   Ikram Senoussaoui, Mohammed Kamel Benhaoua, Houssam-Eddine Zahaf, and Giuseppe Lipari. Toward memory-centric scheduling for
           PREM task on multicore platforms, when processor assignments are specified. In *International Conference on Embedded & Distributed
           Systems (EDiS)*, pages 11–15, 2022. https://doi.org/10.1109/EDiS57230.2022.9996534.

[SSKH13]   Amit Kumar Singh, Muhammad Shafique, Akash Kumar, and Jörg Henkel. Mapping on multi/many-core systems: survey of current and
           emerging trends. In *ACM/IEEE Design Automation Conference (DAC)*, pages 1–10, 2013. https://doi.org/10.1145/2463209.2488734.

[VDA24]   VDA. Verband der Automobilindustrie e.V. (VDA) – Automotive SPICE, 2024. Online. Visited in march 2024.
           http://www.automotivespice.com/.