

## D4.3 FRACTAL AI-Based Algorithms for energy-efficient and safe temporal resource allocation

Deliverable Id:	<b>D4.3</b>
Deliverable name:	<b>FRACTAL AI-Based Algorithms for energy-efficient and safe temporal resource allocation</b>
Status:	<b>Final</b>
Dissemination level:	<b>PUBLIC</b>
Due date of deliverable:	<b>2022-10-31 (M26)</b>
Actual submission date:	<b>2020-10-21</b>
Work package:	<b>WP4 "Safety, Security &amp; Low Power Techniques"</b>
Organization name of lead contractor for this deliverable:	<b>SIEG</b>
Authors:	Carlos Lua, University of Siegen Daniel Onwuchekwa, University of Siegen Pascal Muoka, University of Siegen Alexander Flick, PLC2 Luca Bertaccini, ETHZ
Reviewers:	Artur Kaufmann (BEEA), Alexander Flick (PLC2)

**Abstract:**

**This deliverable aims to report the results and implementations of T4.2 FRACTAL AI-Based Algorithms for energy-efficient systems and safe temporal resource allocation. The deliverable reports the development of the needed support for AI capabilities by PULP and VERSAL platforms, the AI scheduling components for the space and temporal allocation of resources and the adaptation provided by a Hierarchical Metascheduler.**



This project has received funding from the ECSEL Joint Undertaking (JU) under grant agreement No 877056.



Co-funded by the Horizon 2020 Programme of the European Union under grant agreement No 877056.



Project	<b>FRACTAL</b>		
Title	<b>Safety, Security &amp; Low Power Techniques</b>		
Del. Code	<b>D4.3</b>		

## Content

1	History .....	4
2	Summary .....	5
3	Introduction.....	6
3.1	Document Organization.....	10
4	High Level Picture.....	11
4.1	Capabilities for AI supported adaptability in PULP .....	12
4.2	Versal RPU access to AI acceleration.....	12
4.3	AI Scheduling .....	13
5	Capabilities for AI supported adaptability in PULP – WP4T42-01 - ETHZ.....	14
5.1	Component description .....	14
5.2	Design and implementation .....	14
5.3	Testing and evaluation .....	16
5.3.1	Use-Case Integration .....	17
6	Versal RPU access to AI acceleration - WP4T42-02 – PLC2.....	18
6.1	Component description .....	18
6.2	Design and implementation .....	18
6.3	Testing and evaluation .....	20
6.3.1	Use-Case Integration .....	20
7	AI Scheduling - SIEG .....	21
7.1	Component description .....	21
7.2	Design and implementation .....	21
7.2.1	Scenario Generator – WP4T42-03 - SIEG .....	22
7.2.2	GA-Scheduler – WP4T42-04 - SIEG .....	23
7.2.3	AI-Based Scheduler – WP4T42-05 - SIEG.....	24
7.2.4	Schedule Verifier – WP4T42-06 - SIEG.....	27
7.3	Testing and evaluation .....	28
7.3.1	Use-Case Integration .....	34
7.4	Research on link prediction.....	35
8	Hierarchical Metascheduler – WP4T42-07 - SIEG .....	39
8.1	Component description .....	39
8.2	Design and Implementation.....	40



Project	<b>FRACTAL</b>		
Title	<b>Safety, Security &amp; Low Power Techniques</b>		
Del. Code	<b>D4.3</b>		


8.2.1	Platform Model (PM) .....	40
8.2.2	Application Model (AM) .....	41
8.2.3	Context Model (CM) .....	41
8.2.4	Schedule Model (SM) .....	42
8.2.5	Multi-schedule Graph (MSG) .....	42
8.2.6	AI Metascheduling .....	44
8.3	Testing and evaluation .....	45
9	Conclusions .....	50
10	List of figures .....	51
11	List of tables .....	52
12	List of abbreviations .....	53



Project	<b>FRACTAL</b>		
Title	<b>Safety, Security &amp; Low Power Techniques</b>		
Del. Code	<b>D4.3</b>		

# 1 History

Version	Date	Modification reason	Modified by
0.1	2022-05-31	First contributions	Authors
0.2	2022-07-15	Refine contributions and outline tests	Authors
1.0	2022-07-31	Draft version	SIEGEN
2.0	2022-08-30	Refine text and figures	SIEGEN
3.0	2022-10-18	Addressed comments from the reviewers.	SIEGEN


	Project	<b>FRACTAL</b>		
	Title	<b>Safety, Security &amp; Low Power Techniques</b>		
	Del. Code	<b>D4.3</b>		

## 2 Summary

---

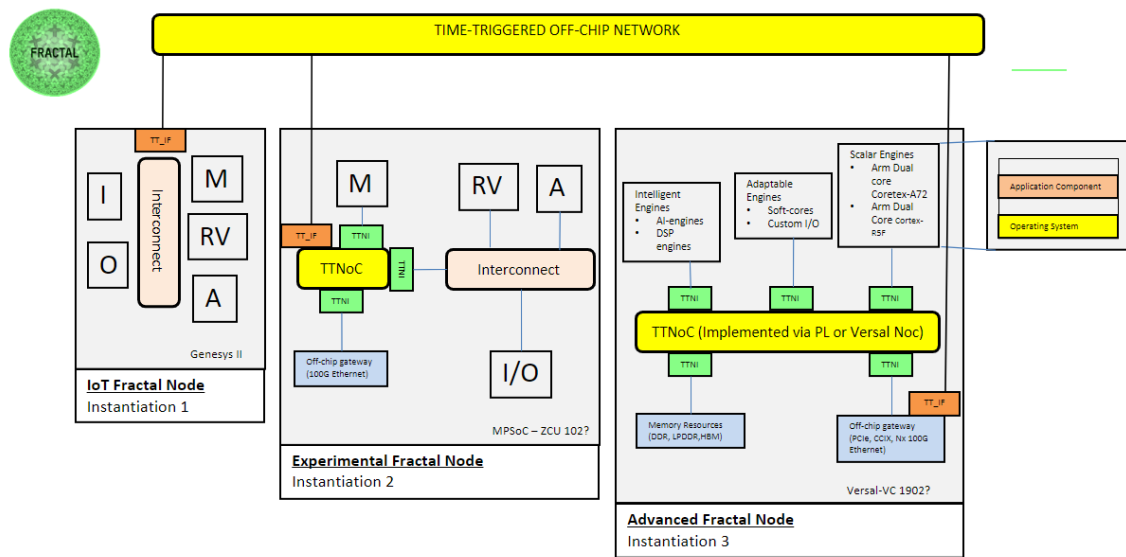
This deliverable aims to report the outcomes of T4.2 on FRACTAL AI-Based Algorithms for energy-efficient and safe temporal resource allocation. The results of the implementations carried out in the task are presented according to the components developed, which reflect the task's objectives.

The work in T4.2 was focused on implementing AI models that can contribute to energy-efficient systems and the allocation of resources at the node and system level. The necessary support for AI capabilities by PULP and VERSAL platforms is described in chapters 5 (ETH) and 6 (PLC2), respectively. The AI scheduling components developed are described in chapter 7 (SIEG). The task focused on the space and temporal allocation of resources. This allocation was extended to a hierarchical level in chapter 8, where adaptation and energy-efficient challenges were tackled.

	Project	<b>FRACTAL</b>		
	Title	<b>Safety, Security &amp; Low Power Techniques</b>		
	Del. Code	<b>D4.3</b>		

### 3 Introduction

The goal of WP7 is to develop safety, security and low-power services for individual FRACTAL nodes. In this document, we will shed light upon AI-based algorithms for energy-efficient and safe temporal resource allocation for FRACTAL systems extending the preliminary implementation reported in the previous deliverable D4.1. The development will include both the node-level (i.e., individual FRACTAL nodes) as well as the system level (i.e., distributed systems comprised of FRACTAL nodes) in accordance with the Fractal system architecture depicted in Figure 1. The resource allocation strategies at both levels seek to increase a system's dependability and energy efficiency while meeting the scheduling limitations. Besides general-purpose computational resources and communication networks, AI resources, such as tensor units, GPUs, and programmable logic, will be supported as the resource allocation target. Moreover, we are addressing runtime changes within a system by developing a semi-static time-triggered resource manager which is invoked after any context events, such as changes within a system at runtime.



I – Input, O – output, RV – Risc V, A – Accelerator, M – Memory, TTNi – Time-triggered Network Interface, TTNoC – Time-triggered Network-on-chip

Figure 1 Fractal system architecture

This task's strategic objective is to guarantee the FRACTAL system's energy efficiency and resource allocation. To accomplish this objective, the task was realized by several building blocks/ components contributing to fulfilling the T4.2 objectives, which are reusable and could be demodulated by any use case (UC). A brief description of each component and how they contribute to fulfilling the T4.2 objectives are reported in Table 1.


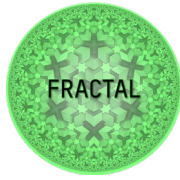
	Project	<b>FRACTAL</b>		
	Title	<b>Safety, Security &amp; Low Power Techniques</b>		
	Del. Code	<b>D4.3</b>		

Table 1 Components brief description and their contribution to fulfilling the T4.2 objectives

Capabilities for AI supported adaptability in PULP	
Description	The component aims to enhance existing PULP platforms to support resource allocation in the context of AI workloads. One of the main applications for PULP-based systems is ML inference on the edge. Further, computing capabilities for inference on the edge are continuously being added to the system (e.g., hardware accelerators, instruction set architecture extensions). Moreover, new features in the software stack are investigated to efficiently map a specific application to the resources available at the hardware level.
Contribution to achieving the T4.2 objectives	The component satisfies the objective of the task through: <ul style="list-style-type: none"> <li>• Exploring resource allocation optimizations on the PULP platform.</li> <li>• Extending computing capabilities for inference on the edge.</li> </ul>
Versal RPU access to AI acceleration	
Description	<p>Enhance RPU libraries to (1) access APU-based AI as a service and (2) enable local AI [acceleration] deployment from RPU.</p> <p>The RPU is running in the context of a time-triggered multicore architecture and needs to respond under real-time conditions for proper scheduling. Therefore, the proposed adaptive scheduling mechanisms rely on ML model inference to decide on actual adaptation.</p> <p>The RPU triggers an offloaded (accelerated) AI model in the Versal AI Engines to respond with low latency inference results. To coordinate the larger-scale task of setting up the ML model (loading), the RPU deploys the APU in Versal as a service provider. This component derives the required setups and protocols for setting up the model and handshake the execution.</p>
Contribution to achieving the T4.2 objectives	This structure is a platform-level support component that <ul style="list-style-type: none"> <li>• Allows ML model deployment on the infrastructure side of a FRACTAL node (leverage from mission mode AI in WP5).</li> <li>• Allows the FRACTAL edge node to compute the schedule predictors.</li> <li>• Provides temporal and spatial resource allocation strategy (if AI-based scheduling is turned on/ off).</li> </ul>
Scenario Generator	
Description	The scenario generator uses the Stanford Network Analysis Platform (SNAP) to generate multiple scenarios/ graphs, each



Project	<b>FRACTAL</b>		
Title	<b>Safety, Security &amp; Low Power Techniques</b>		
Del. Code	<b>D4.3</b>		

	<p>consisting of a particular scheduling problem. This component takes as input the characteristics of an initial model.</p> <p>Although the generated scenarios share some characteristics, they all differ in the precedent constraints and the numeric values, making it very unlikely to have repetitive samples. It is also possible to modify the system's topology by choosing between different options or designing a unique one. This topology can be fixed for every sample generated or randomly assigned.</p>
--	--

Contribution to achieving the T4.2 objectives	The dataset created by this component is sufficiently diverse to emulate different scheduling problems caused by context events. The AI model will utilize the dataset for the learning process, acquiring the adaptation properties that the system needs in order to react to any possible event.
---	---

### GA-Scheduler

Description	A Genetic Algorithm (GA) was chosen as the scheduling tool used for generating the solutions for the training process of the machine learning model. The GA receives all the scheduling graphs generated by the Scenario Generator (WP4T42-03). The chromosomes of the GA are set to optimize the processor allocation and the job order. The objective function evaluates the makespan to find the best feasible schedule.
-------------	---

Contribution to achieving the T4.2 objectives	Part of the Task 4.2 objectives is to provide spatial and temporal resource allocation. For example, some applications in real time-triggered systems require schedules to meet specific deadlines. The Genetic Algorithm is one of the best techniques to find the best solutions for scheduling problems by guaranteeing low makespans.
---	---

### AI Scheduler Model


Description	An Artificial Neural Network (ANN) model is implemented to predict task priorities by learning the behavior of the GA Scheduler. The dataset used to train the ANN scheduler consists of input features extracted from the scenarios created by the Scenario Generator (WP4T42-03) and the outputs, which are the time priorities of the jobs obtained from the GA solutions (WP4T42-04). The input features are selected to capture node characteristics, importance and connection with the rest of the nodes in the graph. The output features are obtained by applying an algorithm that converts the time priorities from the GA schedule into a multi-binary format that is used to train the ANN model. This format re-orders the priorities by comparing the priority values of every job with
-------------	--






Project	<b>FRACTAL</b>		
Title	<b>Safety, Security &amp; Low Power Techniques</b>		
Del. Code	<b>D4.3</b>		

	each other on a one-to-one basis. The neural networks' training parameters were chosen based on the binary accuracy of the predictions, keeping in mind that low accuracy values do not necessarily indicate an invalid schedule, as the predictions can still lead to alternative solutions with different makespans.
Contribution to achieving the T4.2 objectives	The component satisfies the Task 4.2 objectives by providing temporal and spatial resource allocation for a single FRACTAL node, enhancing the system's dependability while fulfilling the timing constraints. It also contributes by decreasing the runtime of scheduling with AI predictions.
<b>Schedule Verifier</b>	
Description	The Schedule verifier component is a tool that allows us to recompute a new schedule using the job priorities learned by the ANN model (WP4T42-05). Predicting a whole real time-triggered schedule solution is not a task that an AI model can do by itself since any slight inaccuracy can lead to an incorrect schedule and, therefore, to a possible accident. The schedule verifier detects any possible error and ensures a correct schedule.
Contribution to achieving the T4.2 objectives	The component satisfies the Task 4.2 objectives by providing the correctness and safety of the resource allocations at development time.
<b>Hierarchical Metascheduler</b>	
Description	The Hierarchical Metascheduler iteratively calls a scheduler (GA metascheduler) to generate modified schedules for each context event in the context model. The inputs to the metascheduler are the application model (AM) describing the computational jobs and communication messages for a given application. The platform model (PM) describes the system architecture on which the application is run. The context models (CM) describe all events relevant to the application and platform model.
Contribution to achieving the T4.2 objectives	Runtime context events within a system, either at the FRACTAL node level or in a set of FRACTAL nodes, are adapted for energy efficiency and fault recovery by optimizing the system resource allocation (scheduling). The computed schedules allow for runtime adaptation of the system to context events.

	Project	<b>FRACTAL</b>		
	Title	<b>Safety, Security &amp; Low Power Techniques</b>		
	Del. Code	<b>D4.3</b>		

### 3.1 Document Organization

The document's structure is as follows: in Section 4, we present a high-level picture of the Fractal solutions developed in WP4. Then, in sections 6 and 7, we describe the components that provide the necessary support to the VERSAL and PULP platforms in the context of the AI functionalities. Next, sections 8 and 9 are dedicated to AI scheduling, composed of the AI-based scheduler and the Hierarchical Metascheduler. The components within this section are designed to deploy time-critical schedules and provide adaptation at run-time. Finally, in section 9, we draw conclusions and our future plans.

	Project	FRACTAL		
	Title	Safety, Security & Low Power Techniques		
	Del. Code	D4.3		

## 4 High Level Picture

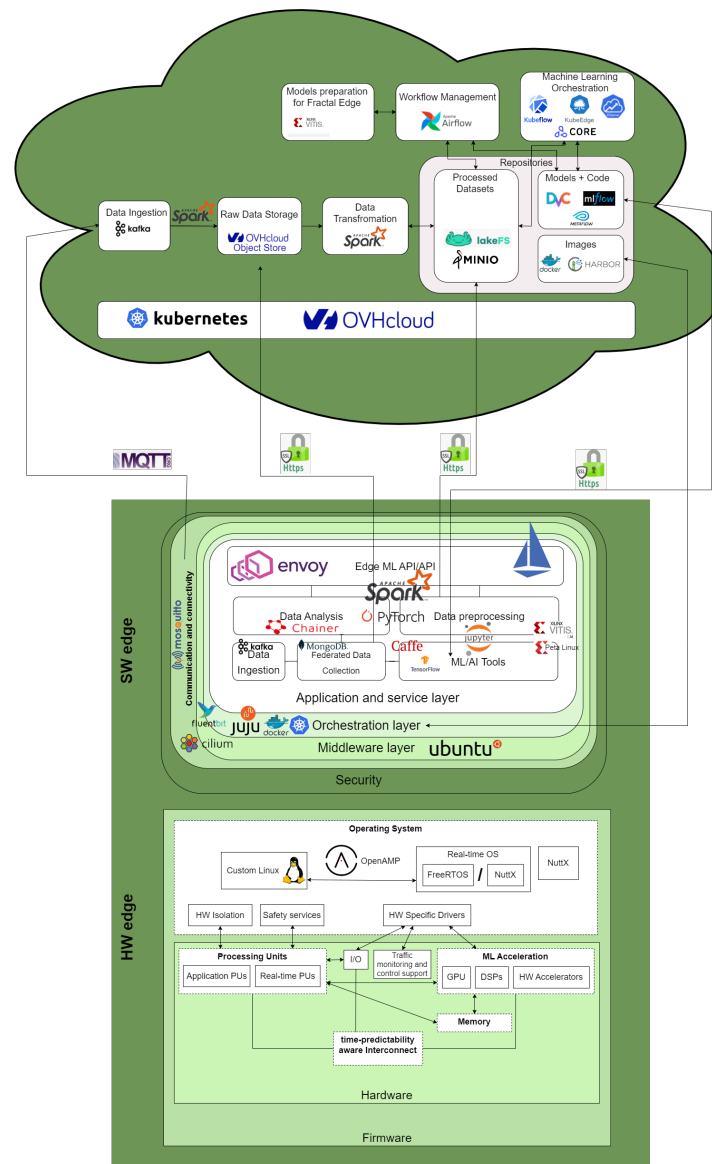



Figure 2 Big picture of the FRACTAL project

The big picture of the project, illustrated in Figure 2, is a holistic representation of the FRACTAL solution. It provides an answer to the use case requirements, which are the functional and non-functional needs captured by FRACTAL use cases at the beginning of the project. Based on these requirements, a set of features has been established to give a technical notion to the requirements.

The components mentioned above, developed in WP4 made of software or hardware, participate in fulfilling some of the FRACTAL features. In the following subsections, we report how each component is integrated into the big picture.

	Project	FRACTAL		
	Title	Safety, Security & Low Power Techniques		
	Del. Code	D4.3		

## 4.1 Capabilities for AI supported adaptability in PULP

One of the main applications for PULP-based systems is ML inference on the edge. This component aims to enhance the resource allocation and computing capabilities of PULP-based systems in the context of AI workloads, so it is located on the hardware side of the big picture, as shown in Figure 3. Use cases built upon PULP-based IoT systems will benefit from the enhanced features, reaching higher energy efficiencies when running ML inference on the edge.

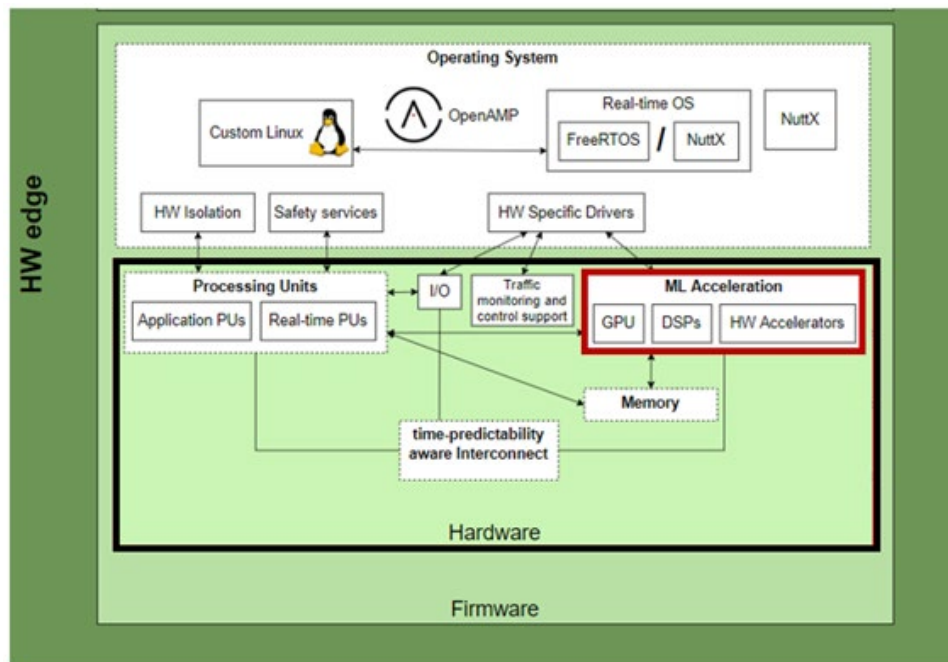



Figure 3 Capabilities for AI supported adaptability in PULP integrated in the FRACTAL big picture

## 4.2 Versal RPU access to AI acceleration

Versal RPU, even if used in safety-centric designs, may require access to AI inference accelerators to enhance context awareness and autonomous planning of the FRACTAL node. The Xilinx Vitis AI-based inference acceleration approaches for Versal are typically supported by Linux-based APU applications. This component creates an RPU-side interface to expose the APU-based AI applications to the RPU application. Figure 4 shows where this component is located in the big picture of the project.

	Project	FRACTAL		
	Title	Safety, Security & Low Power Techniques		
	Del. Code	D4.3		

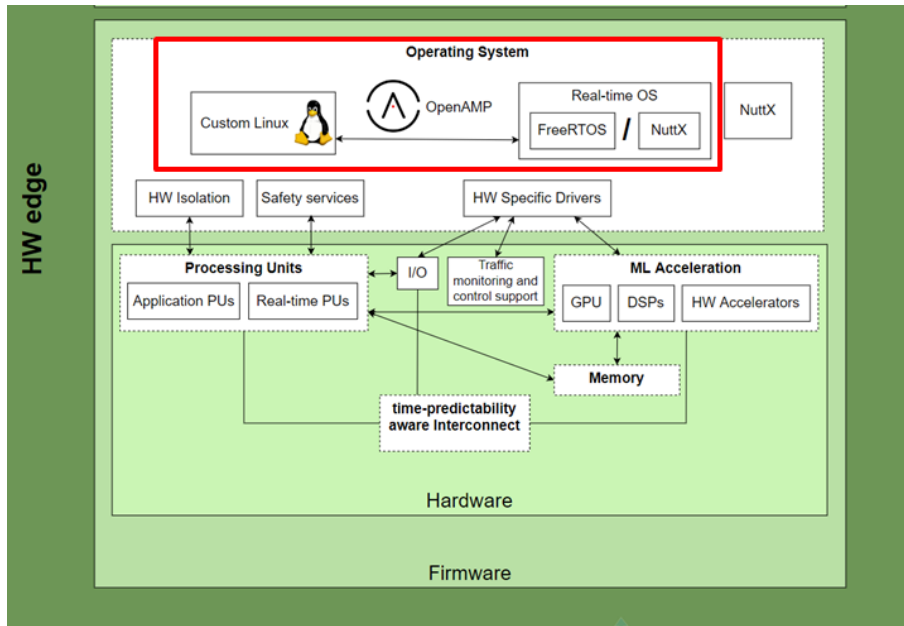


Figure 4 Integration of Versal RPU access to AI acceleration in the big picture

### 4.3 AI Scheduling

The AI scheduling components will be developed at the software level; thus, it could be regarded as a part of the software in the edge node of the big picture shown in Figure 5. Specifically, the AI models implemented at runtime will use the ML/Tools provided by the application and service layer.

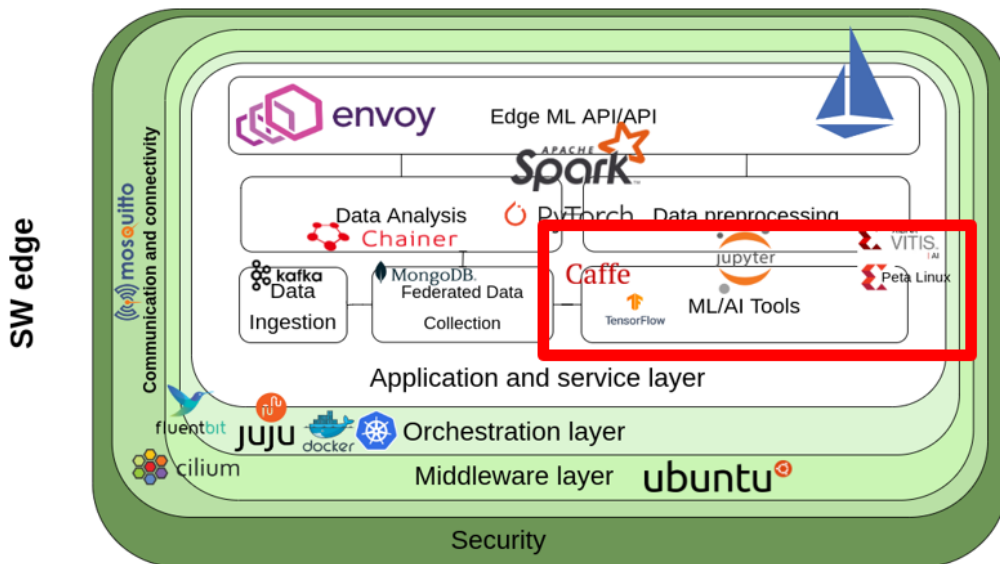


Figure 5 Integration of AI Scheduling in the big picture

	Project	FRACTAL		
	Title	Safety, Security & Low Power Techniques		
	Del. Code	D4.3		

## 5 Capabilities for AI supported adaptability in PULP – WP4T42-01 - ETHZ

### 5.1 Component description

PULP-based IoT end nodes have been optimized for energy efficiency. One of the main applications targeted by such devices is ML inference directly on the edge. In this component, novel microarchitectural modifications, instruction set architecture (ISA) extensions and specialized hardware accelerators have been implemented to unleash ML capabilities on PULP-based systems. In addition, the software stack has been extended to optimize resource allocation.

### 5.2 Design and implementation

To increase the compute capabilities of PULP-based systems, PULPissimo can be extended with an acceleration cluster where eight RISC-V cores share a scratchpad memory, as shown in Figure 6. The accelerator additionally contains a Direct Memory Access (DMA) engine to efficiently move data from PULPissimo to the accelerator and vice versa and an event unit for fast wakeup/sleep of the cluster cores.

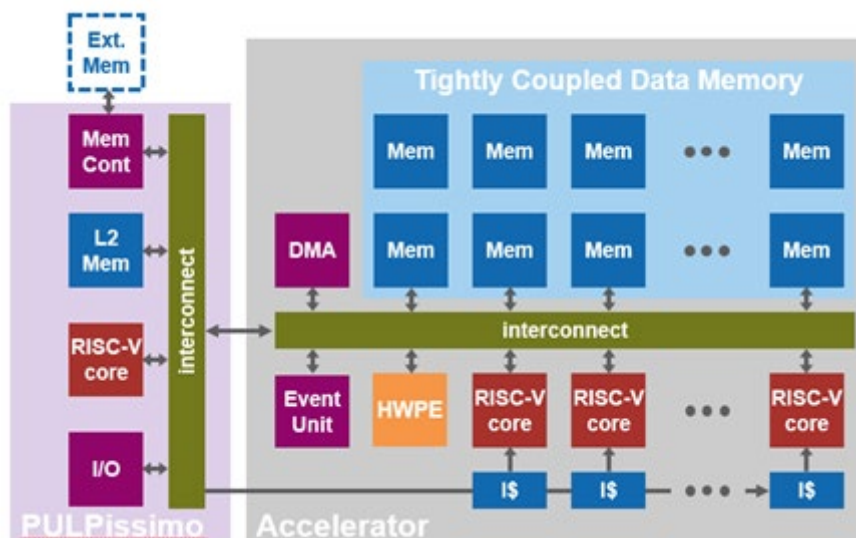



Figure 6 PULPissimo + Accelerator

The new accelerated PULPissimo system is conceived to work as follows: (i) PULPissimo can be coupled to sensors and manages the communication with the outside world, (ii) whenever new data is available for computation, one of the cores programs the DMA engine to move the data from PULPissimo to the accelerator, (iii) once the data is in the accelerator memory, the cores are woken up by the event unit and start computing, (iv) once the computation is completed, the result is moved

	Project	FRACTAL		
	Title	Safety, Security & Low Power Techniques		
	Del. Code	D4.3		

back from the accelerator to PULPissimo. During normal operation, all these phases can happen in parallel. For example, as soon as the first set of data is moved to the accelerator, the cores can start computing while the DMA continues moving the rest of the inputs. The same can happen towards the end of the computation when the cores are still calculating, but the first set of results is already available.

The standard 32-bit RISC-V ISA has also been extended to further increase the performance of the system. Load and store instructions with post increment, hardware loops, and SIMD dot product instructions have been introduced to reduce overheads, allowing to spend most of the time and energy on the real computation. The PULP ISA extension allows for a 10x speedup with respect to vanilla RV32IMC cores (see Figure 7).

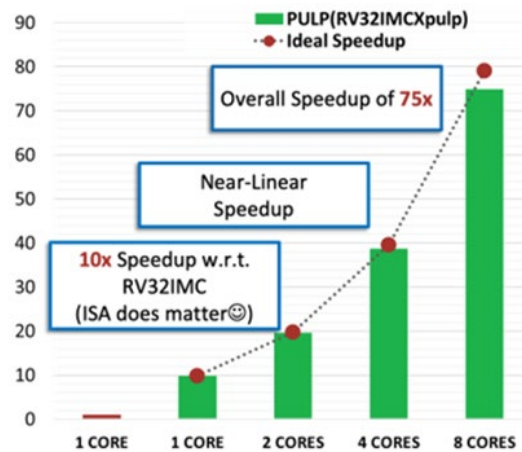


Figure 7 ISA extension for ML inference

We explored further enhancements by adding a specialized hardware accelerator called HWPE (see Figure 8). HWPEs share the scratchpad memory with the cluster core, allowing for efficient data sharing, and are software programmed by the cluster cores. Such hardware blocks further specialize the architecture, thus increasing its energy efficiency for specific workloads. For example, a convolutional accelerator can be added to the system to speed up ML inference.

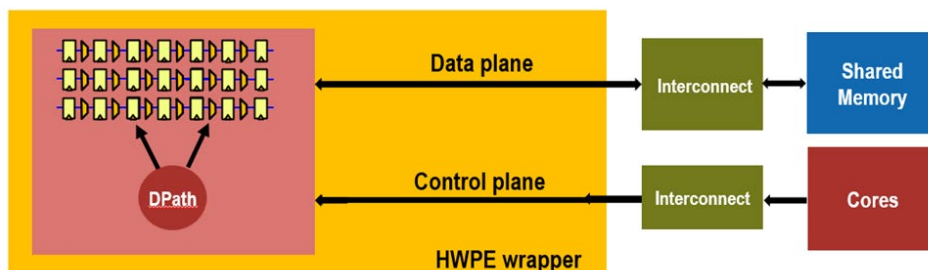



Figure 8 HWPE

To efficiently exploit the new features and improve the final user experience, a full open-source SW stack is provided (see Figure 9). A neural network model can be



	Project	FRACTAL		
	Title	Safety, Security & Low Power Techniques		
	Del. Code	D4.3		

trained and quantized through the QuantLab tool (<https://github.com/pulp-platform/quantlab>), Dory (<https://github.com/pulp-platform/dory>) is used as a deployment framework, taking care of the tiling, optimized libraries are provided to exploit the new instructions, and finally, the PULP SDK (<https://github.com/pulp-platform/pulp-sdk>) takes care of the low-level software routine to program the accelerator.

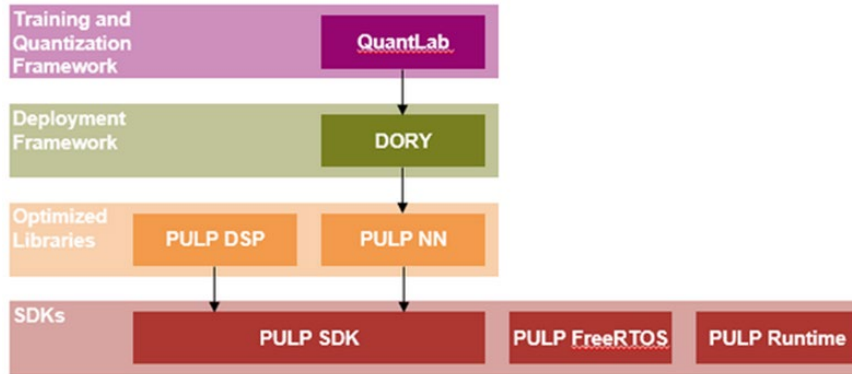


Figure 9 SW Stack for ML inference

Finally, we investigated ISA extensions for low-precision floating-point computing by adding dot product instructions that accumulate in a larger format. A SIMD unit containing such dot product modules has been integrated into a double-precision FPU, enabling up to 7.2x performance increase when computing on FP8 data and accumulating on FP16 precision with respect to employing double-precision (see Figure 10).

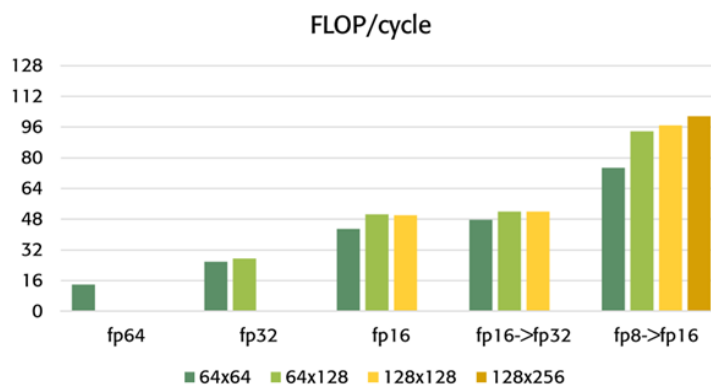



Figure 10 Low-Precision FP dot product extension – Performance results

### 5.3 Testing and evaluation

Multiple architectures based on the PULPissimo platform plus acceleration cluster have been tested and evaluated on FPGA. Deployment scripts for the Xilinx ZCU102 FPGA are open-source on the PULP GitHub page (<https://github.com/pulp->




	Project	<b>FRACTAL</b>		
	Title	<b>Safety, Security &amp; Low Power Techniques</b>		
	Del. Code	<b>D4.3</b>		

[platform/pulp](#)). Partners interested in prototyping their PULP-based use case can use such scripts to speed up the testing process. Furthermore, multiple chips have been taped out and tested at ETH.

### 5.3.1 Use-Case Integration

A PULP-based architecture will be used in UC3 (Smart meters for everyone), where a smart meter prototype will be designed. A PULP-based IoT system will be connected to a camera to take a picture of the display of a mechanical meter, process it to extract the information displayed by the meter, and finally send the data over the cellular network. A neural network model will be used to extract the information from the picture. The ML features provided by PULP will allow for efficient ML inference on the edge.

	Project	FRACTAL		
	Title	Safety, Security & Low Power Techniques		
	Del. Code	D4.3		

## 6 Versal RPU access to AI acceleration - WP4T42-02 – PLC2

---

### 6.1 Component description

In FRACTAL nodes based on Versal the safety related hardware isolation is provided by physical access separation. This is available in the platform designs and provides two compute domains, the real-time cores (RPUs) and the Linux system that typically runs on the A72 ARM cores (APU). In such node setups the infrastructure service and scheduling part runs on the RPU domain while microservice middleware and applications run on the Linux level.

With the context-aware and autonomous scheduling as derived in WP4 the RPU domain will require AI model support to feed decisions on system state changes and more. In the Versal ecosystem AI inference applications are accelerated by the Deep Learning Processing Unit (DPU) and are typically supported by a runtime layer on Linux to setup the model processing and flow control. For larger AI models the DPU processing may involve interleaved CPU computation.


This supporting layer is not readily available for RPU and the CPU computations would not map efficiently to RPU as well. This component therefore enables the RPU side to utilize the APU as a service to access AI inference accelerators.

The RPU application in the safety domain is enabled to setup an AI target through the APU and trigger inference runs afterwards and retrieve results. RPU gets even access to the node orchestration level to support model exchange through the FRACTAL system level.

The corresponding component in the APU system level is the Versal Model Deployment Layer (WP3T34-03).

### 6.2 Design and implementation

The block level of the component is displayed in Figure 11. The split between the RPU and APU domains also shows the separation between the safety channel and the rest of the node design.

	Project	FRACTAL		
	Title	Safety, Security & Low Power Techniques		
	Del. Code	D4.3		

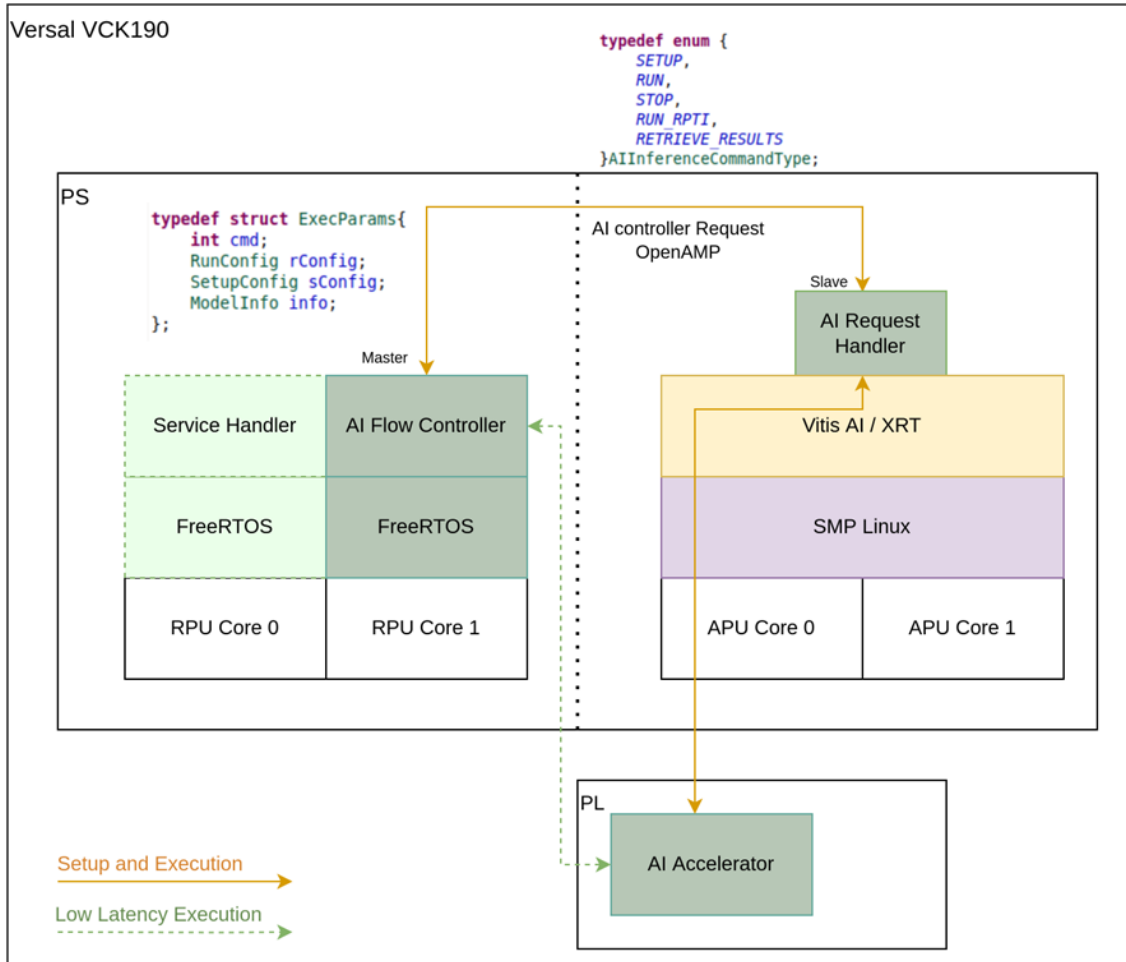



Figure 11 RPU Access to APU AI Inference Application

The communication between RPU and APU is established through the OpenAMP framework. To achieve a potentially certifiable node the feature set of this framework must be restricted.

The RPU and APU application layer exchange messages on a protocol that allows an extensible instruction list, which to date holds the minimally required commands:

- SETUP: Load a DPU based model
- RUN: Single execution of an inference run
- RUN\_RPTI: Continuous inference execution
- STOP: Stop a continuous execution
- RETRIEVE\_RESULTS: Retrieve results

The AI inference is either triggered by the command level or by hardware access that bypasses the overhead of the communication loop. This secondary trigger path requires specific support in hardware setups that needs to be made available during the UC integration of the component.

	Project	FRACTAL		
	Title	Safety, Security & Low Power Techniques		
	Del. Code	D4.3		

## 6.3 Testing and evaluation


The setup is tested and based on the Versal FRACTAL reference platform (WP3) that is enhanced by the additional components WP3T34-03 Versal Model deployment layer, WP4T41-06 Versal Isolation Design to build an augmented platform that needs to use the cross-domain communication as described here.

This augmented platform comprises a standalone FRACTAL node that is used to load and trigger a Yolov3 model on the APU by pushing the commands from the RPU code side. This model comes with well understood accuracy but the main focus on assessing this APU proxy method is to optimize for the worst-case latency.

The limit of this for a given model needs to satisfy the real-time condition of the RPU side processing. The Versal device implementation can generate fast inference frame rates with repeatable latency depending on the actual model, while the overhead for the communication will contribute a dynamic latency on top that needs to be evaluated along the integration with specific UC requirements.

### 6.3.1 Use-Case Integration

The augmented setup shall be added to the Versal based FRACTAL node in UC8. In that context the system level connections can be tested to retrieve AI models dynamically.

	Project	FRACTAL		
	Title	Safety, Security & Low Power Techniques		
	Del. Code	D4.3		

## 7 AI Scheduling - SIEG

---


### 7.1 Component description

In safety-critical applications, essential properties of time-triggered systems must be preserved. These exhibited properties include avoidance of resource contention without dynamic resource arbitration, implicit synchronization, guaranteeing of timing constraints, implicit flow control, and fault containment **[SIEG-AI1]**. Scheduling in time-triggered systems is traditionally carried out offline. The scheduling strategies for time-triggered systems include mathematical techniques, heuristics, and neighborhood search. Adaptation in time-triggered systems is motivated by the need to provide energy-efficient operation, fault recovery, and adaptation to any change in the system.

We address a safe and adaptive artificial intelligence (AI) quasi-static scheduling approach for time-triggered scheduling problems. The priority of jobs is predicted using an AI-based model trained offline. With recent technological advancements toward deploying AI accelerators in hardware, AI models can easily be deployed. At runtime, adaptation can be triggered upon a context event, such as processing element failures. The adaptation is attained by accurately predicting the job priority, which is then used to perform an online computation of a new schedule.

### 7.2 Design and implementation

A complete description of the AI scheduling implementation and their components is described in **[SIEG-AI2]**. Four components are developed for implementing AI scheduling in a FRACTAL node, as shown in Figure 12: *Scenario Generator*, *GA-Scheduler*, *AI-Based Scheduler* and *Schedule Verifier/ Reconstructor*. The scenario generator is responsible for generating diverse application models, which are then fed into the GA scheduler to obtain schedules associated with each given platform and application model. The priorities are extracted from the schedules and used with the corresponding application and platform model outputs from the scenario generator to create a dataset. This dataset is then used to train a machine learning model.

	Project	FRACTAL		
	Title	Safety, Security & Low Power Techniques		
	Del. Code	D4.3		

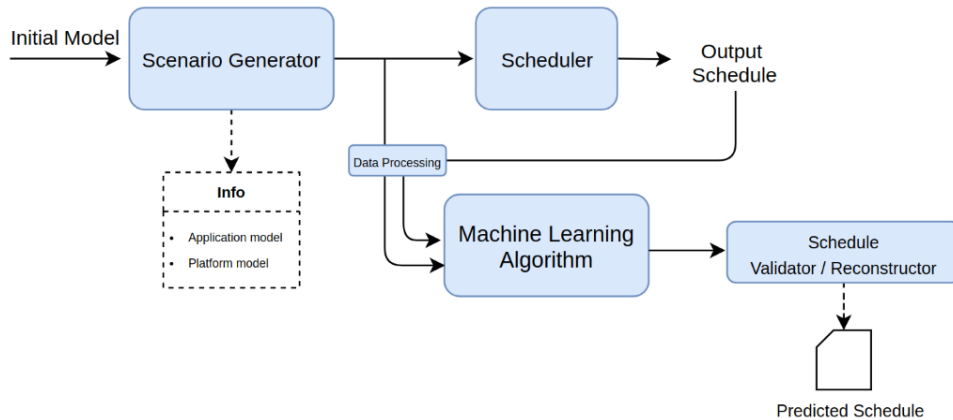


Figure 12 AI scheduling components

### 7.2.1 Scenario Generator – WP4T42-03 - SIEG

The scenario generator, shown in Figure 12, is composed of the Stanford Network Analysis Platform (SNAP). SNAP is a general-purpose network analysis and graph mining library inspired by the work in [SIEG-AI3], which is used in this work to generate multiple scenarios. It makes use of a network theory (or graph theory) based approach for developing functions that can be used in the analysis and manipulation of large networks. The library takes the inputs from an initial model specifying parameters such as the number of jobs, number of platform model participants (switches and cores), and the in-degree and out-degree of each job to create a graph for the application and the platform model, as shown in Figure 13. Each scheduling problem, represented by the mentioned graphs, is expressed using JavaScript Object Notation (JSON) files to make data processing easier. The information contained inside the files upholds the division between the platform or physical model, which specifies the hardware topology in which the system runs, and the application or logical model, which contains data regarding the pre-constraints of the schedule.

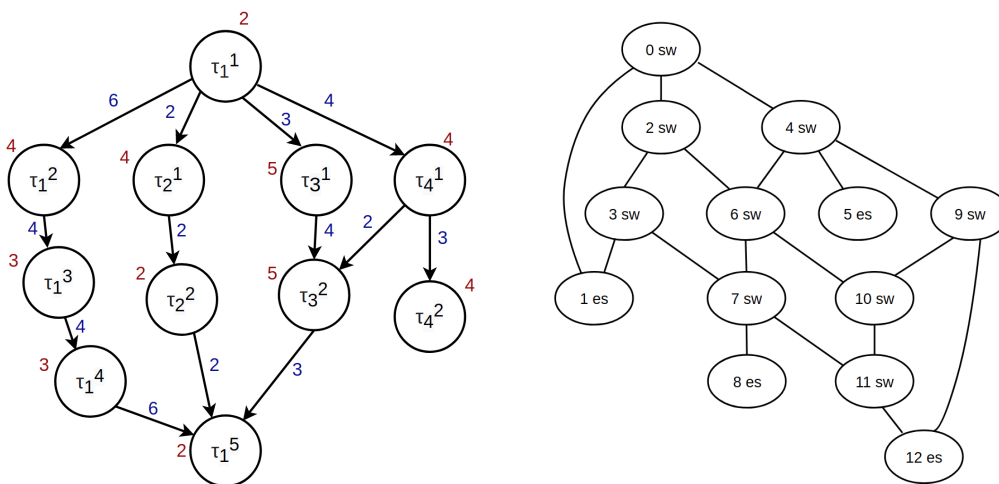



Figure 13 Application and Platform model

	Project	FRACTAL		
	Title	Safety, Security & Low Power Techniques		
	Del. Code	D4.3		

We required this component to generate synthetic data for the model since we did not have any real-world data that we could use for that purpose. The initial parameters of the synthetic data can be modified to make it look closer to an actual application. For example, it allows changing the system's topology to bus, mesh, or manually creating a predefined topology. Depending on the application, parameters like jobs' execution time and the messages' size can be modified. Also, the predefined constraints can be adjusted by limiting the number of out and in degrees of freedom a job will have. The intention is that it can be used by different applications depending on the actual use case in which the synthetic data is needed.

### 7.2.2 GA-Scheduler – WP4T42-04 - SIEG

The scheduler block takes the output platform and application models from the scenario generator and finds a feasible schedule for each example. The scheduling problem is a well-defined optimization problem that can be tackled with different approaches such as mathematical techniques, scheduling heuristics, metaheuristics and neighborhood search. In this work, the genetic algorithm (GA), a metaheuristics method, is used to compute the schedules for each example. Nevertheless, other scheduling approaches, such as list scheduling and Ant Colony Optimization (ACO), can be utilized in the scheduler block. In this work, the chromosomes of the GA are set to optimize the processor allocation and the job order. The objective function evaluates the makespan to find the best feasible schedule. The output schedule is obtained from the GA, but only the job priority is fed into the machine learning algorithm.

Three sections form each genome inside the genetic algorithm (see Figure 14):

- Allocation cells: The allocation cells indicate the end system in which a task will be executed. The selection inside these cells is restricted to the available end systems that are able to execute that particular task.
- Priority cells: The priority cells indicate the order in which the tasks are executed. In conjunction with the application model's pre-constraints, these cells determine the schedule sequence.
- Routing cells: The routing cells select one of the available paths each message will take to go from one end system to another. The number of available paths can differ depending on the platform model.

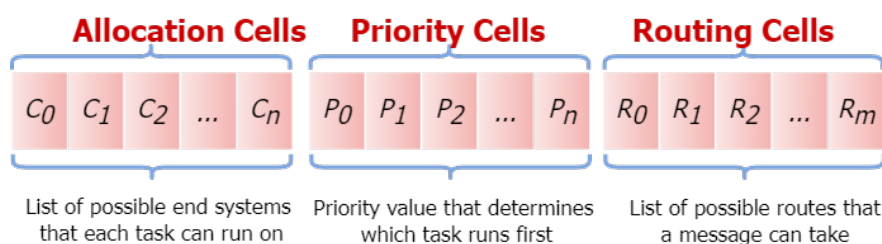



Figure 14 Genome cells

	Project	<b>FRACTAL</b>		
	Title	<b>Safety, Security &amp; Low Power Techniques</b>		
	Del. Code	<b>D4.3</b>		

The optimal GA's behavior varies depending on the model's complexity and size. The best performance is obtained by adjusting the following parameters:

- Population
- Generations
- Crossover
- Mutation
- Replacement population

The GA's output contains the job execution sequence as well as the end systems to which they are allocated. The output also includes the job starting time, messages injection time, and the route the messages must take. All this information is captured in JSON files.

- Since the time it takes for the GA to compute one schedule is high, a whole dataset must be executed in parallel. Depending on the server where the GA will run, it is possible to adjust the parallelization parameters to the server's capabilities, ensuring optimal utilization of the available resources.

### 7.2.3 AI-Based Scheduler – WP4T42-05 - SIEG

Due to safety considerations, we focused on learning only the priorities of the jobs. Relying on a machine learning algorithm to directly predict schedules for a safety-critical application is not certifiable. Machine learning models are generalized approximation models, and time-triggered applications require an exact schedule in the event of online schedule adaptation. Therefore, the priorities are predicted, and a schedule verification algorithm is enabled to ensure that only correct schedules are generated from the prediction. There is no restriction on the machine learning algorithm to be deployed in this block. However, a feed-forward artificial neural network (ANN) is used in this work to learn the priorities of the tasks. The ANN learns the scheduler, which is subsequently used to predict job priorities, after which the "schedule verifier" component is deployed to obtain a schedule for the application.

#### ***Input Features***

Inputs and outputs are saved in files in JSON format to simplify the management of data. The application and platform models are included in the input files; this information is extracted and kept in tables to compute relevant features later. According to the directed acyclic graph, the extracted features shown in Table 2 attempt to describe the job's attributes and relationships with its neighbors.




	Project	FRACTAL		
	Title	Safety, Security & Low Power Techniques		
	Del. Code	D4.3		

Table 2 Job features

Feature	Description	Formula
$f_{1,j}$	Job id / Number of jobs	$j/n$
$f_{2,j}$	Job execution time / Total execution time	$e_j/e_{Total}$
$f_{3,j}$	Max. size - msgs sent / Maximum size	$ms_{j,out}/ms_{max}$
$f_{4,j}$	Max. size - msgs received / Maximum size	$ms_{j,in}/ms_{max}$
$f_{5,j}$	Number of msgs sent / Total number of msgs	$m_{j,out}/m_{Total}$
$f_{6,j}$	Number of msgs received / Total number of msgs	$m_{j,in}/m_{Total}$
$f_{7,j}$	Top level value / Maximum top level value	$tl_j/tl_{max}$
$f_{8,j}$	Bottom level value / Maximum button level value	$bl_j/bl_{max}$

### Output Features


Only the job priorities are collected from the scheduling solutions so that the machine learning model can use them as targets. However, the priorities must be transformed first because the model anticipates binary labels. These labels are obtained by comparing the job priority values with each other on a one-to-one basis, resulting in a vector of size  $n*(n-1)/2$ , where  $n$  is the total number of jobs. Figure 15 shows this transformation into a scheduling problem with ten jobs.

Jobs	$j_0$	$j_1$	$j_2$	$j_3$	$j_4$	$j_5$	$j_6$	$j_7$	$j_8$	$j_9$
$j_0$	x	1	0	0	0	1	1	0	0	0
$j_1$	x	x	1	0	0	0	1	0	0	1
$j_2$	x	x	x	0	1	0	1	0	0	1
$j_3$	x	x	x	x	0	1	1	0	1	0
$j_4$	x	x	x	x	x	1	0	0	1	0
$j_5$	x	x	x	x	x	x	0	0	0	1
$j_6$	x	x	x	x	x	x	x	0	1	0
$j_7$	x	x	x	x	x	x	x	x	1	0
$j_8$	x	x	x	x	x	x	x	x	x	1
$j_9$	x	x	x	x	x	x	x	x	x	x

$$[j_0/j_1, j_0/j_2, \dots, j_0/j_9, \dots, j_1/j_2, j_1/j_3, \dots, j_1/j_9, \dots, j_8/j_9]$$

Figure 15 Job priorities transformation

S2P algorithm (see Figure 16) shows the process of converting the priorities from the GA schedule into a multi-binary format that is used to train the ANN scheduler. This format reorders the priorities in a way that directly compares each job's priority to the others on a one-to-one basis.

	Project	FRACTAL		
	Title	Safety, Security & Low Power Techniques		
	Del. Code	D4.3		

---

**Algorithm 1:** S2P algorithm

---

```

Input: size
Output: binary_labels
1 Function S2P (size):
2   for s in range(size) do
3     for each i in num_jobs - 1 do
4       for each j in range(i + 1, num_jobs) do
5         if position[i] > position[j] : then
6           Append '1' to vector label_vector
7         else
8           Append '0' to vector label_vector
9         end if
10        end for
11      end for
12      Append label_vector to binary_labels
13    end for
14    return binary_labels

```

---

Figure 16 S2P algorithm


An Artificial Neural Network is chosen as the machine learning model to predict the job priorities, which requires predicting multiple mutually non-exclusive classes/labels. The implemented ANN is composed of three layers:

- Input Layer: The number of nodes in this layer is dictated by the number of jobs and features extracted.
- Hidden Layer: The number of nodes in the hidden layer varies according to the size of the scheduling problems. It can go from 100 to 1000 nodes.
- Output Layer: The number of nodes in this layer is determined by the formula:

$$n*(n-1)/2,$$

where  $n$  is the total number of jobs.

For the activation function, we are using ReLU (Rectifier Linear Unit) after the input layer and the Sigmoid function before the output nodes. Since we deal with multi-label classification, we need to compute the loss from all the labels as a whole. The Binary Cross-Entropy Loss (BCE) function is usually employed for binary labels, but since our data is not balanced, we have to use a modified version of the BCE called 'weighted balanced cross entropy'. This function adjusts the BCE by adding weighting. The weights are determined dynamically for every batch by identifying how many positive and negative classes are present and modifying them accordingly. The learning rate of the ANN is 0.001. The number of learning epochs is set to 300. The ANN architecture is shown in Figure 17.

	Project	FRACTAL		
	Title	Safety, Security & Low Power Techniques		
	Del. Code	D4.3		

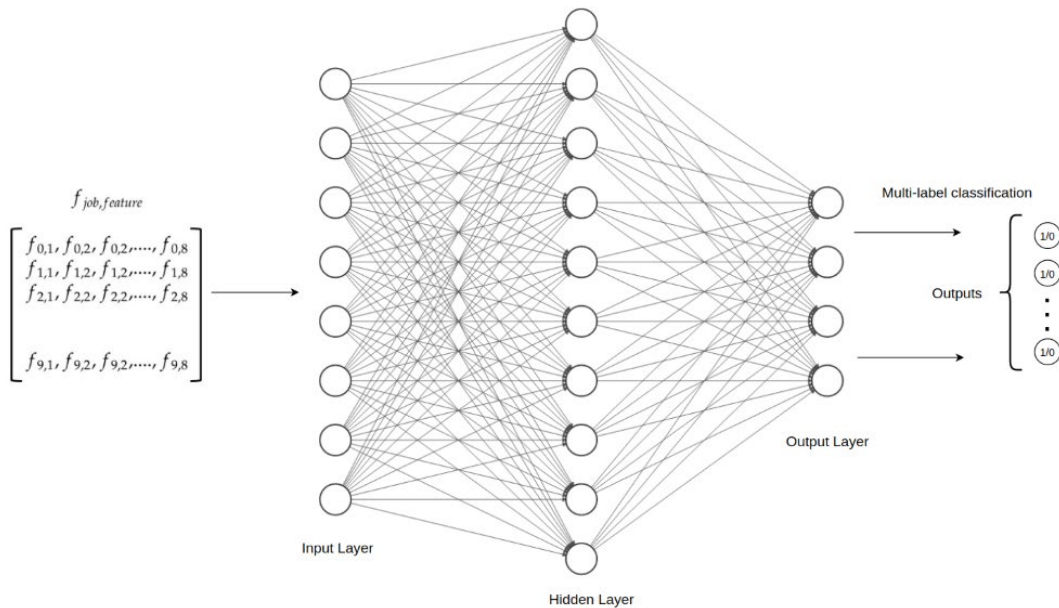


Figure 17 ANN architecture

### 7.2.4 Schedule Verifier – WP4T42-06 - SIEG

The ANN does not provide an entire schedule; instead, the output predictions purely indicate the importance of each job at the moment of the allocation. Since the schedules are generated for safety-critical systems, they must always be correct. The schedules are validated and reconstructed by using the predicted values. In the allocation step, the end systems in which the jobs are executed are selected based on the earliest start time. After obtaining the ANN predictions and inserting them in the schedule reconstructor, we can visualize the complete schedules (see Figure 18), extract the average makespan and compare it with the Genetic Algorithm and other scheduling techniques. This time we chose List Scheduling as one of the fastest methods to compare computation times.

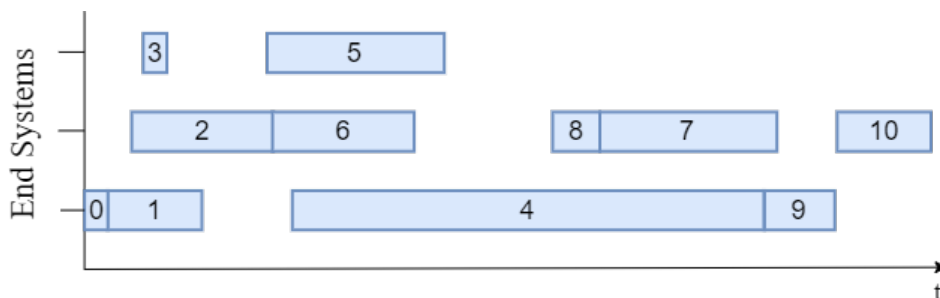



Figure 18 Schedule visualization

The schedule verifier must solve possible collisions between messages. Figure 19, for instance, depicts a scenario in which three messages are transmitted simultaneously. The verifier's task is to detect the collisions and adjust the messages by delaying them depending on the job priorities.

	Project	FRACTAL		
	Title	Safety, Security & Low Power Techniques		
	Del. Code	D4.3		

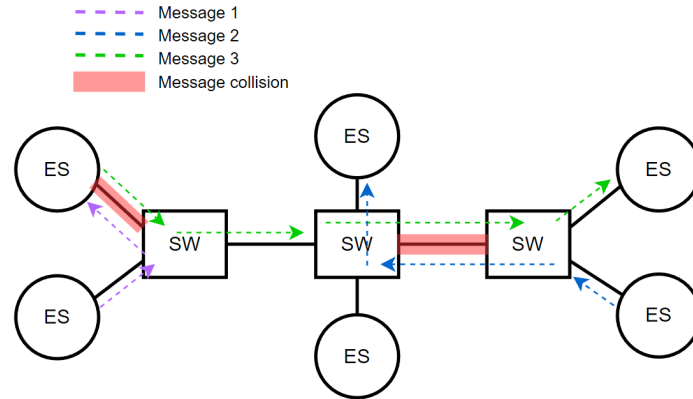


Figure 19 Message collision

### 7.3 Testing and evaluation

The deployment components are available on the FRACTAL GitHub page (<https://github.com/project-fractal/WP4>). Partners interested in utilizing them can download them and follow the README files before testing.

The background of this experiment is the use of AI algorithms in time-triggered systems to produce schedules when the system changes. Deploying AI algorithms in hardware can significantly improve efficiency and latency. Xilinx’s Vitis AI is available to translate such AI models to specific IP (DPU) hardware accelerators to help researchers to quickly perform inference with these highly evolved algorithms, minimizing time and cost. The model deployed in this work was an Artificial Neural Network, the framework was Tensorflow2, and the Keras model was imported.

Since the training process is done offline, the only part that requires deployment is the network structure and the weights tuned during the training process. After the training process, it is necessary to save the model in the Hierarchical Data Format (HDF) or H5 file. The general H5 file includes the structure and weights of the neural network. In the subsequent deployment process, the H5 file will be used as the input file for quantitative processing.

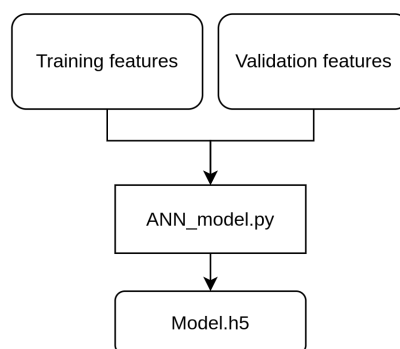



Figure 20 H5 file generation

	Project	FRACTAL		
	Title	Safety, Security & Low Power Techniques		
	Del. Code	D4.3		

The following python statements are written in the ANN\_model.py file to generate and save the model.

```

model = Sequential()
model.add(Input(80))
model.add(Flatten())
model.add(Dense(100, input_dim=80))
model.add(ReLU())
model.add(Dense(45, input_dim=100))
model.add(Activation('sigmoid'))
model.save('model.h5')

```

The H5 file generated by the code above can be imported into the netron.app website to visualize the full structure of the model in a clear way (see Figure 21).

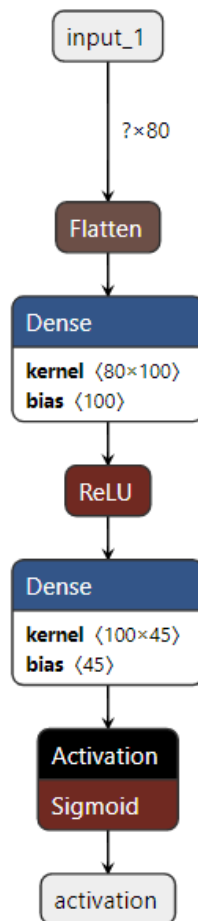



Figure 21 ANN layer structure

The following steps explain the deployment process of the ANN model on the VCK190 ES1 board.

	Project	FRACTAL		
	Title	Safety, Security & Low Power Techniques		
	Del. Code	D4.3		

### AI model compilation:

#### 1. Installing a Board Image.

- Download the SD card system image files from the following links:  
<https://www.xilinx.com/member/forms/download/design-license-xef.html?filename=xilinxvck190-dpu-v2020.2-v1.4.0.img.gz>

Note: The version of the board image should be 2020.2 and the image is only for VCK190 ES1 board.

- Use Etcher software to burn the image file onto the SD card.

#### 2. Download the Vitis AI library

- Run Linux command:

```
$git clone --recurse-submodules https://github.com/Xilinx/Vitis-AI/tree/1.4
```

Or download directly from <https://github.com/Xilinx/Vitis-AI/tree/1.4>

- Go to the Vitis AI folder (workplace) under the download file path

```
$cd Vitis-AI
```

In this experiment, Vitis-AI-v1.4 is the library used. But the latest version is Vitis-AI2.0. The Vitis-AI-v1.4 version is selected, because the FPGA used in the experiment is the VCK190 ES1 board. The Pre-image provided by Xilinx for the VCK190 ES1 board is `xilinx-vck190-dpuv2020.2-v1.4.0.img.v1.4.0` and corresponds to the 1.4.0 version of Vitis-AI. If another version is selected an error will occur.

#### 3. Docker needs to be downloaded for the compilation process. Docker is a working environment specially configured by Xilinx for users, which can be considered as a virtual machine. After downloading the Docker container, follow the instructions below to download the latest Docker, which runs on the CPU.

- Pull Vitis AI Image


```
$docker pull xilinx/vitis-ai-cpu:1.4.916
```

- Launch the Docker Image:

```
$/docker_run.sh xilinx/vitis-ai-cpu:1.4.916
```

- Activate framework tensorflow2:

```
$conda activate vitis-ai-tensorflow2
```

	Project	FRACTAL		
	Title	Safety, Security & Low Power Techniques		
	Del. Code	D4.3		

4. The compilation starts and the xmodel file is generated. Before compiling, it is necessary to organize the data and quantize the model. Run "bash -x compile.sh" to compile the quantized model. The compile.sh file includes the following:

```

1  ARCH=/opt/vitis_ai/compiler/arch/DPUCVDX8G/VCK190/arch.json
2  OUTDIR=./output
3  NET_NAME=FRACTAL
4  MODEL=./output/quantized_model.h5
5
6  echo "-----"
7  echo "COMPILING MODEL FOR VCK190.."
8  echo "-----"
9
10 compile() {
11     vai_c_tensorflow2 \
12     --model      $MODEL \
13     --arch       $ARCH \
14     --output_dir $OUTDIR \
15     --net_name   $NET_NAME
16 }
17
18
19 compile 2>&1 | tee compile.log
20
21
22 echo "-----"
23 echo "MODEL COMPILED"
24 echo "-----"

```

The input ARCH is the hardware structure. The VCK190 ES1 board is used in this experiment, and the DPU is DPUCVDX8G. The input Model is the model trained on the host and the quantized h5-file is the weight and bias file of the neural network. After the execution of the script file the FRACTAL.xmodel is generated. This file contains hardware information and frame structure.

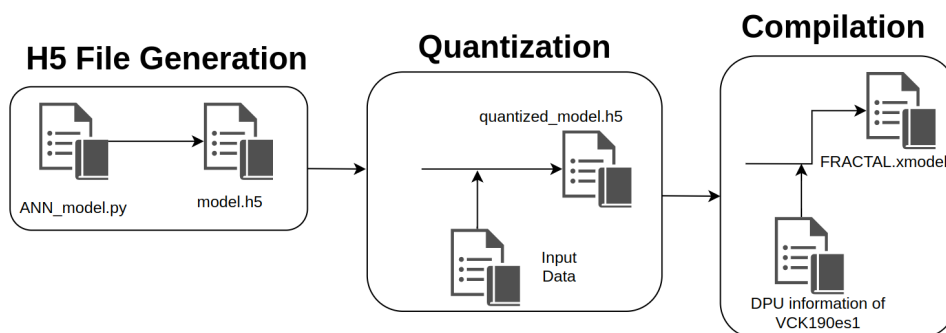



Figure 22 Compilation process of AI models

*Quantization process:* The parameters of a general neural network are 32-bit floating point numbers. However, deploying deep learning models on edge devices requires consideration of device storage space, memory size, operating power consumption, latency, and other issues. The AI quantizer can reduce computational complexity without compromising prediction accuracy by converting 32-bit floating-point weights

	Project	FRACTAL		
	Title	Safety, Security & Low Power Techniques		
	Del. Code	D4.3		

and activation functions to fixed-point such as INT8. Fixed-point network models require less memory bandwidth and are, therefore, faster and more power-efficient than floating-point network models. The disadvantage is that there will be a loss of precision, but it is generally within an acceptable range. If the actual accuracy is not good, fine-tuning within the quantization tool flow can improve accuracy.

5. Docker exit. The compiled FRACTAL.xmodel needs to be copied to the /root/home path of the SD card.

### AI application

After the quantified fractal models and the data to be processed are ready, they need an application to invoke them and make the predictions. This application is called AI application.

The figure below depicts the process:

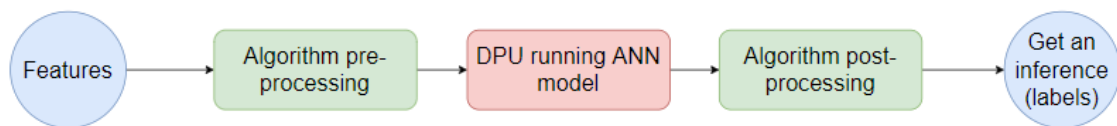


Figure 23 AI application


The most important block above is the intermediate one that involves the execution of the ANN model by the DPU (DPUCVDX8G). The FRACTAL.xmodel file already integrates the ANN model, the VCK190 board and the corresponding DPUCVDX8G information.

If a layer cannot be quantized, such as the softmax layer, the developer needs to add an extra piece of code after the DPU running model section to make it run on the CPU. In addition to the DPU processing model, there is an algorithm for pre-and post-processing. The preprocessing function is to normalize the data and input it to the DPU; Post-processing can be used as a test to check the accuracy of DPU inference.

Since the python program is used in this experiment, the developer can directly use the python app\_mt.py command on the arm core of VCK190 to compile the AI application. If it is a C program, it can be cross-compiled on an x86 system first and then put the generated executable file on the SD card and run it directly.

The input data should also be placed in the /root/home path of the SD card (the specific location should be coordinated with the executable file).



	Project	FRACTAL		
	Title	Safety, Security & Low Power Techniques		
	Del. Code	D4.3		

## Start board

Board setup is quick and easy, the figure below shows the instructions and diagrams for setup. Both SW11 and SW1 are [4:1]0001.

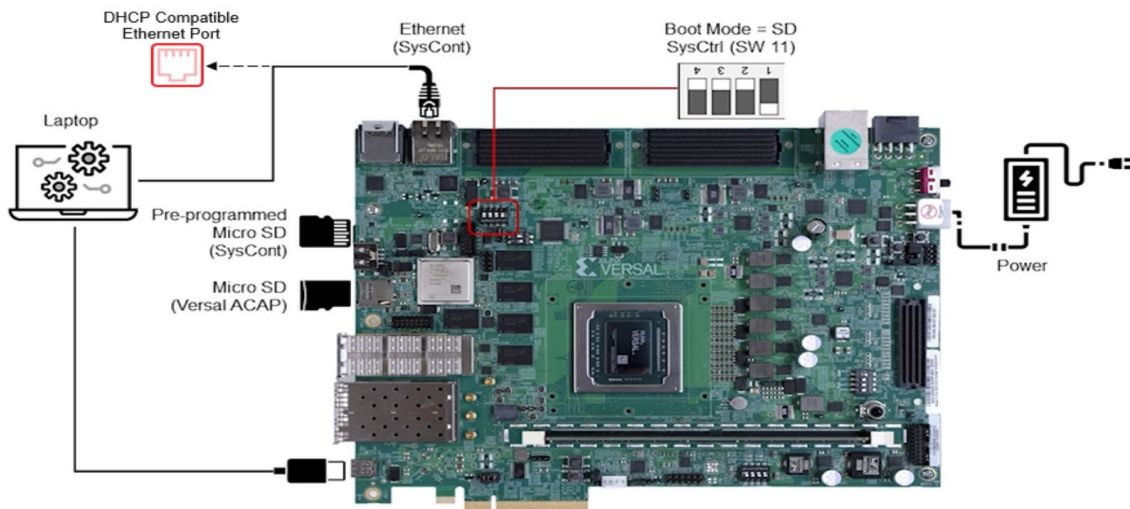



Figure 24 SD Launch

VCK190 ES1 board's DPU Pre-image download link:

<https://www.xilinx.com/member/forms/download/design-license-xef.html?filename=xilinxvck190-dpu-v2020.2-v1.4.0.img.gz>

## Testing

We conducted three classes of experiments for our evaluation. The three classes include experiments using the list scheduler, ANN-based scheduler, and GA-based scheduler. For each class, the 4000 scheduling problems from the testing data consisting of 10, 40, 70, and 100 jobs are evaluated. Figure 25 displays a plot of the makespans against the number of jobs for each experiment class. It can be seen that the schedules computed by the GA-based approach have better makespans compared to the list scheduler and the ANN model. Figure 25 also shows that the list scheduler has lower makespans than the ANN-based approach for a job size of 10 and 40. However, as the number of jobs increases, the ANN produces makespans lower than the list scheduler, as seen in the case of 70 and 100 jobs. In real time-triggered systems, we can expect a higher number of jobs, which will increase the gap of the makespans between the list scheduling and the ANN-based approach.

	Project	<b>FRACTAL</b>		
	Title	<b>Safety, Security &amp; Low Power Techniques</b>		
	Del. Code	<b>D4.3</b>		

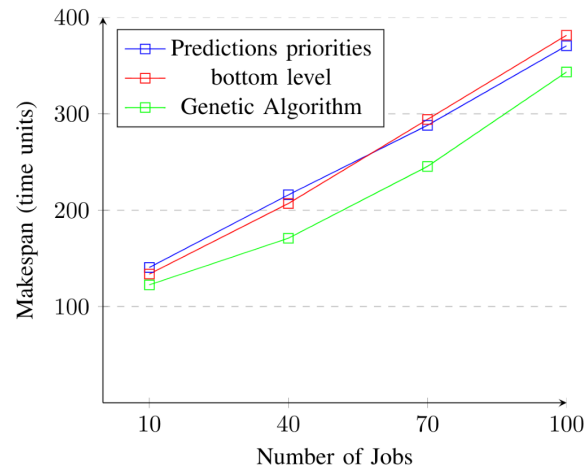


Figure 25 Makespan comparison

The implication of the contribution is the applicability of the ANN-based approach for dynamic reconfiguration of safety criticality application in response to context events. A GA-based scheduler is not desirable for real-time systems with stringent computational time requirements. In Figure 26, we show the computational times for each of the evaluations carried out. The computation time of the GA-based approach is much more than that of the list scheduler and the ANN-based scheduler. It can be seen that the computation time of the GA-based approach is much more than that of the ANN-based scheduler.

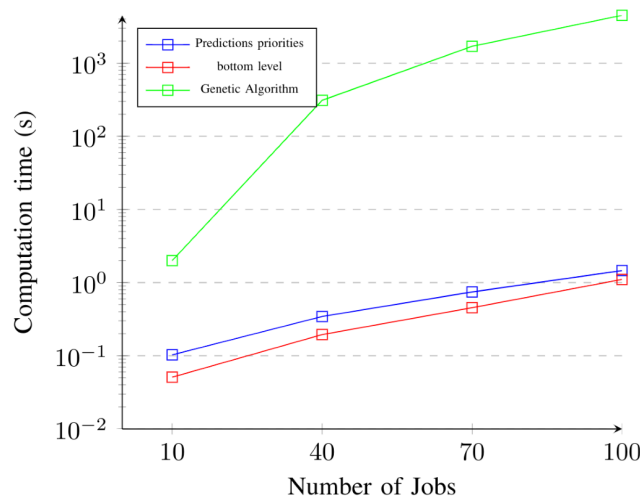



Figure 26 Computation time comparison

### 7.3.1 Use-Case Integration

AI scheduling will be used in UC8 (Autonomous warehouse shuttles), where a FRACTAL concept will be implemented as an automated storage and retrieval solution to increase adaptability and reliability. The handling, storage and retrieval of warehouse goods by automated shuttles will be optimized using the components developed in the current work package by adapting the components to the use case.

	Project	FRACTAL		
	Title	Safety, Security & Low Power Techniques		
	Del. Code	D4.3		

In addition, an AI scheduler will organize and analyze the generated data sets optimally to improve warehouse throughput.

## 7.4 Research on link prediction


Alternative approaches with the potential to solve the time-triggered scheduling problems have been addressed. Link prediction using GNNs is one approach that caught our attention but is still in its early stages.

Link prediction objective is to predict whether an edge exists between two particular edge nodes in a graph/ network. It has many applications, such as network reconstruction [SIEG-AI4], recommender systems [SIEG-AI5], and spam mail detection [SIEG-AI6].

We took as reference the work carried out in [SIEG-AI7]. In this paper, M. Zhang and Y. Chen proposed a link prediction framework called "SEAL" to simultaneously learn from local enclosing subgraphs, embeddings and attributes based on graph neural networks. Furthermore, they demonstrated the framework's potential in link prediction problems by comparing it to other heuristics and network embedding algorithms.

Original implementation extracts local enclosing subgraphs around links from a big graph as input, and outputs how probable it is that the connections exist. A GNN network is trained over the enclosing subgraphs around the missing links. In the training process, there must be a differentiation between links, treating the edges in the graph as positive samples and non-existent edges as negative samples.

In this work, we modelled the scheduling solution provided by the GA as a heterogeneous graph (see Figure 27), by combining the application and the platform models. This graph consists of three different node sets, one for the jobs, the other for the processors, and the third represents the routers/ switches. Directed and undirected edges satisfy specific constraints based on the type of nodes they connect.

	Project	FRACTAL		
	Title	Safety, Security & Low Power Techniques		
	Del. Code	D4.3		

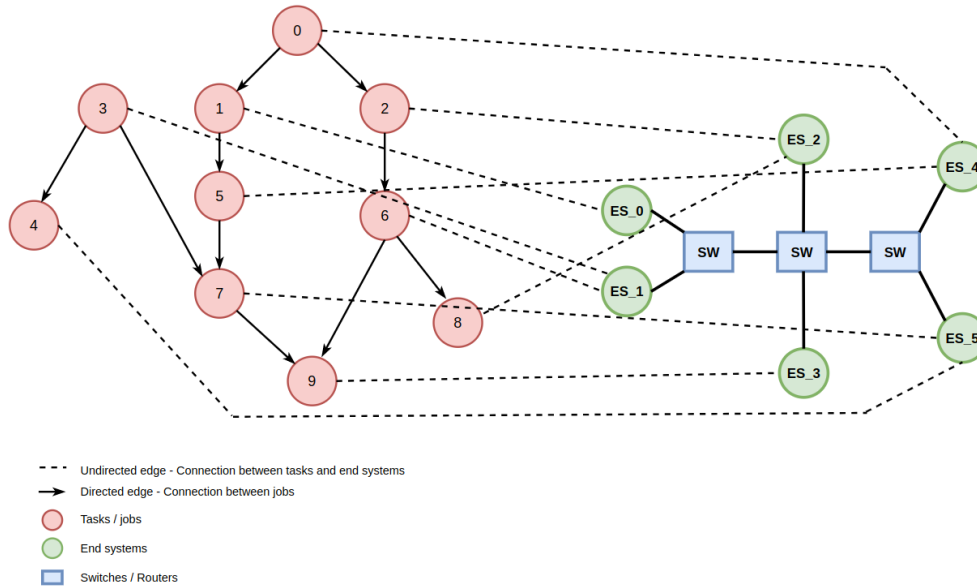


Figure 27 Scheduling solution modeled as a heterogeneous graph

The undirected dotted lines join each job with a processor, meaning this job will be executed there. For example, job four should run on end system five based on the scheduling solution. The heterogeneous graphs are used as input to our GNN link prediction model. The objective after training the model is to see how effective the model is in predicting partially or the totality of the undirected edges between the tasks and the end systems.

The dataset for this experiment consists of 40,000 heterogeneous graphs representing the scheduling solutions obtained from the GA algorithm for each of the 10-job and 40-job application models. For model training, 39,000 heterogeneous graphs were used whereas the remaining 1000 are used to evaluate the model performance.

The predicted results are compared with the original deleted value to get the accuracy of the prediction of the model. The accuracy is shown in the following tables.

Table 3 10-Job Dataset results

No.	Learning rate	Model setting			Training Result		
		Type of model	No. of Layers	No of edges	Loss	Training accuracy	Test accuracy
1	0.001	SAGEConv	2	1	1.5854	0.6372	0.3750
2	0.01	SAGEConv	2	1	1.8324	0.3628	0.2969
3	0.001	SAGEConv	2	3	1.4963	0.7587	0.3125
4	0.001	SAGEConv	2	6	1.5209	0.7899	0.3281
5	0.001	SAGEConv	2	10	1.4187	1	0.2031
6	0.001	SAGEConv	2	1	1.4682	0.6852	0.1956
7	0.001	SAGEConv	3	1	1.5012	1	0.2016


	Project	FRACTAL		
	Title	Safety, Security & Low Power Techniques		
	Del. Code	D4.3		

Table 4 40-Job Dataset results

No.	Learning rate	Model setting			Training Result		
		Type of model	No. of Layers	No of edges	Loss	Training accuracy	Test accuracy
1	0.001	SAGEConv	2	1	1.4463	0.885	0.3008
2	0.01	SAGEConv	2	1	1.858	0.2704	0.1992
3	0.001	SAGEConv	2	3	1.4396	0.9783	0.2852
4	0.001	SAGEConv	2	6	1.4307	0.9987	0.2500
5	0.001	SAGEConv	2	10	1.4251	1	0.2852
6	0.001	GraphConv	2	1	1.4056	0.7512	0.1658
7	0.001	SAGEConv	3	1	1.4546	1	0.2315

Table 5 Prediction accuracies

Dataset	Predicted job	No. of models						
		1	2	3	4	5	6	7
10 jobs	job 4	0.333	0.356	0.31	0.3433	0.3233	0.3133	0.31
	job 8	0.3233	0.3366	0.3566	0.3166	0.31	0.3	0.3033
	5 jobs	0.3506	0.35	0.3586	0.3606	0.3293	0.3228	0.3266
	10 jobs	0.356	0.362	0.343	0.319	0.301	0.2866	0.29
40 jobs	job 12	0.33	0.3333	0.3133	0.33	0.29	0.2866	0.29
	job 24	0.3	0.41	0.35	0.3266	0.3033	0.29	0.31
	20 jobs	0.3228	0.3553	0.3476	0.331	0.3251	0.3033	0.319
	40 jobs	0.3282	0.3505	0.3582	0.3392	0.3312	0.3251	0.3282

We can observe from the results above that the prediction accuracy is much lower than the training accuracy but more consistent with the testing accuracy due to a certain degree of overfitting in the training process. However, the testing accuracy can more accurately reflect the results of the model operation. The more edges masked during the training process, the less accurate the final prediction result will be. The highest recorded prediction accuracy was 0.41.


Several factors contributed to the obtained results, but there is clear potential in the link prediction approach applied to scheduling time-triggered systems. Future tests and research will determine the possibility of applying this alternative method with a specific use case of the FRACTAL project.

**[SIEG-AI1]** Obermaisser, R., Ahmadian, H., Maleki, A., Bebawy, Y., Lenz, A., & Sorkhpour, B. (2019). Adaptive time-triggered multi-core architecture. *Designs*, 3(1), 7.

**[SIEG-AI2]** Lua, C., Onwuchekwa, D., & Obermaisser, R. (2022, June). AI-Based Scheduling for Adaptive Time-Triggered Networks. In *2022 11th Mediterranean Conference on Embedded Computing (MECO)* (pp. 1-7). IEEE.

**[SIEG-AI3]** Leskovec, J., & Sosič, R. (2016). Snap: A general-purpose network analysis and graph-mining library. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 8(1), 1-20.


**[SIEG-AI4]** Koren, Y., Bell, R., & Volinsky, C. (2009). Matrix factorization techniques for recommender systems. *Computer*, 42(8), 30-37.

	Project	<b>FRACTAL</b>		
	Title	<b>Safety, Security &amp; Low Power Techniques</b>		
	Del. Code	<b>D4.3</b>		

**[SIEG-AI5]** Nickel, M., Murphy, K., Tresp, V., & Gabrilovich, E. (2015). A review of relational machine learning for knowledge graphs. *Proceedings of the IEEE*, 104(1), 11-33.

**[SIEG-AI6]** Oyetunde, T., Zhang, M., Chen, Y., Tang, Y., & Lo, C. (2017). BoostGAPFILL: improving the fidelity of metabolic network reconstructions through integrated constraint and pattern-based methods. *Bioinformatics*, 33(4), 608-611.

**[SIEG-AI7]** Zhang, M., & Chen, Y. (2018). Link prediction based on graph neural networks. *Advances in neural information processing systems*, 31.

	Project	FRACTAL		
	Title	Safety, Security & Low Power Techniques		
	Del. Code	D4.3		

## 8 Hierarchical Metascheduler – WP4T42-07 - SIEG

---

### 8.1 Component description

Time-triggered systems require a schedule to avoid resource contention, error propagation, and deadline misses when executing a system application. While scheduling time-triggered systems allow for temporal predictability, adaptation to runtime events is challenging since scheduling decisions are decided offline, and the static schedule is deployed online. A static schedule defines such global allocations of the start times of tasks, execution time and resource allocation of tasks. Similarly, the communication traffic of the system is described, such as the injection time of messages into the network-on-chip (NoC) and its routing information.


Battery-operated devices could benefit from runtime context events such as a dynamic slack in task execution where the task is completed before its scheduled worst-case execution time (WCET). The system's dependability must be maintained where a crash event leads to hardware resources not being accessible. Task reallocation could be motivated by changing application priorities or thermal management, in which high-intensity tasks are moved to cooler cores to prevent system overheating. For each scenario, context-specific adaptation is accomplished by switching the system schedule to an aligned schedule adapted to the context to avoid system failure brought on by a schedule change.

The Hierarchical Metascheduler (HM) introduced in D4.1 and presented in [SIEG-HM1] is an offline tool to compute a Multi-schedule Graph (MSG) for hierarchical time-triggered systems. The MSG facilitates adaptation at runtime through traversing the graph from one node to the next. It is also a directed acyclic graph where each node in the graph is a system schedule, and each directed edge is the occurrence of a context event during the execution of the schedule.

In addition, an optimization algorithm is implemented to manage the MSG size as the number of context events increases. In computing schedules adapted for slack events, for instance, a re-convergence horizon assures that the new schedule may only deviate from its predecessor during a time window in which slack events result in energy savings. Furthermore, the re-convergence horizon permits paths re-convergence in the MSG, reducing the number of generated schedules.

The generated MSG suffers from state explosion issues where the MSG's size increases exponentially, rendering it unusable in online mode. Our proposed model uses the MSG generated by the metascheduler to train different AI models that will be deployed at runtime. The models explored are Graph Neural Network (GNN) based model, Artificial Neural Network, Encoder/Decoder Neural Network and Random Forest Classifier. The metascheduler generates an MSG at design time, which is then used to train the AI models to predict the temporal priorities of the following schedule.



	Project	FRACTAL		
	Title	Safety, Security & Low Power Techniques		
	Del. Code	D4.3		

## 8.2 Design and Implementation

The HM requires an application (AM), platform (PM), and context model (CM) to generate the MSG. A new schedule is computed for every combination of context events in the context model, resulting in a tree of schedules linked by those events, referred to as the multi-schedule graph (MSG). An adapted schedule, described by the schedule model (SM), is a synchronised computation and communication schedule for each context event.

### 8.2.1 Platform Model (PM)

The PM describes the computational and communication resources on which a system application is executed. The PM represents the runtime hardware architecture of the FRACTAL node consisting of processing elements (PEs) for the execution of application tasks. Communication resources facilitate data communication between application tasks from one PE to the next. The PM also describes hardware communication resources such as gateways to enable communication between nodes, a set of routers which make up the time-triggered (TT) network-on-chip (TTNoC) and network switches representing the TT off-chip communication network. The bi-directional physical links interconnecting the various components of the PM further contribute to describing the hardware architecture, such as in the topology of the TTNoC, which is a 2D mesh. Each PE accesses the TTNoC through a local router. Messages between nodes are communicated through the TT off-chip communication network, where gateways within each node provide a bridge between the on-chip communication network (TTNoC) and the TT off-chip communication network, as illustrated in Figure 28.

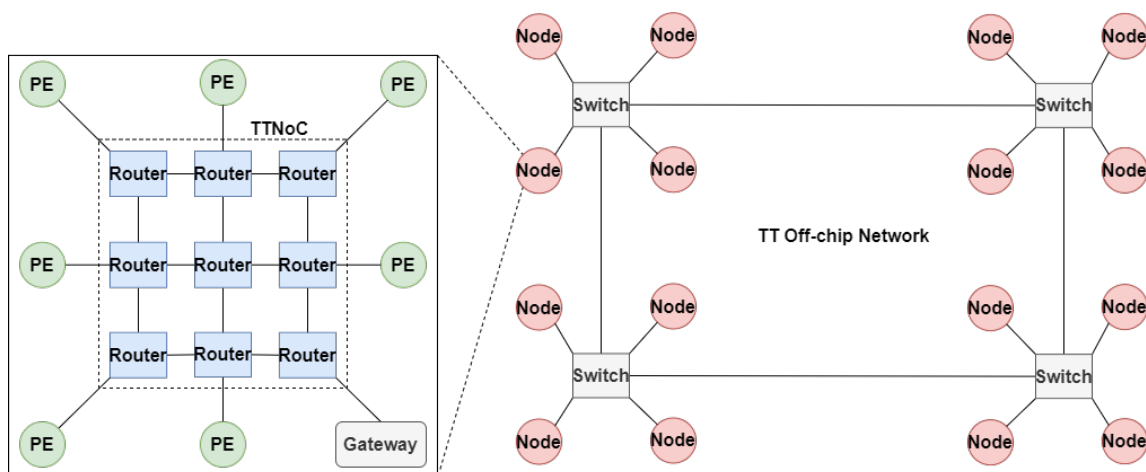



Figure 28 Hierarchical Platform Model

Each communication resource in the PM is modelled with a constant delay factor that adds to the message hop time over the resource. In addition, each link has a link speed and bandwidth, contributing to the message routing rate between communication resources.



	Project	FRACTAL		
	Title	Safety, Security & Low Power Techniques		
	Del. Code	D4.3		

### 8.2.2 Application Model (AM)

Similar to the PM, the application model (AM) describes the application to be executed on the platform comprising tasks and messages. As illustrated in Figure 29, a directed message from one task to another indicates dependencies between tasks such that successor tasks are performed after parent tasks. In addition, the AM describes the characteristics of tasks pertinent to the TT system, such as the worst-case execution time (WCET) of tasks, representing the computational cost of executing the task on a PE. Tasks are also modelled with deadlines, a property of TT applications indicating when a task must be completed. Tasks can also be constrained to a node, such as in the case of data collection tasks constrained to sensor nodes. Finally, each message in the AM is characterised by a source (sender task), destination (receiver task) and message size.

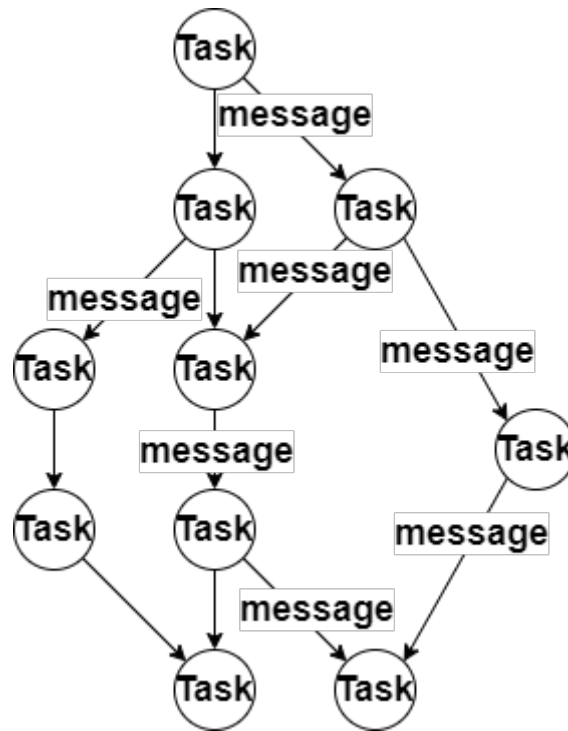



Figure 29 Application Model (AM)

### 8.2.3 Context Model (CM)

The context model (CM) describes all context events adaptable to at runtime by the platform. At runtime, where a task is completed earlier than its WCET, a slack event is described in the CM, which is the difference between the WCET and runtime execution time. A failure event in the CM describes a hardware resource permanently unavailable to execute a system service. For example, a PE crash or a broken link in the communication resource. A thermal event in the CM describes a scenario where the temperature threshold of a resource has been exceeded.

	Project	FRACTAL		
	Title	Safety, Security & Low Power Techniques		
	Del. Code	D4.3		

### 8.2.4 Schedule Model (SM)

The schedule model (SM) is a temporal and spatial mapping of the AM to the PM. Each application task in the AM is mapped to a PE in the PM (spatial allocation) at guaranteed timeslots (temporal allocation) for the execution of the application. Messages in the AM are also mapped on a source-to-destination path through the TTNoc and TT off-chip communication network (spatial allocation), as illustrated in Figure 30. For each message, an injection time is specified to avoid resource contention and collisions in routing (temporal allocation). Routing messages in the SM also ensures that precedence constraints between sender and receiver are preserved.

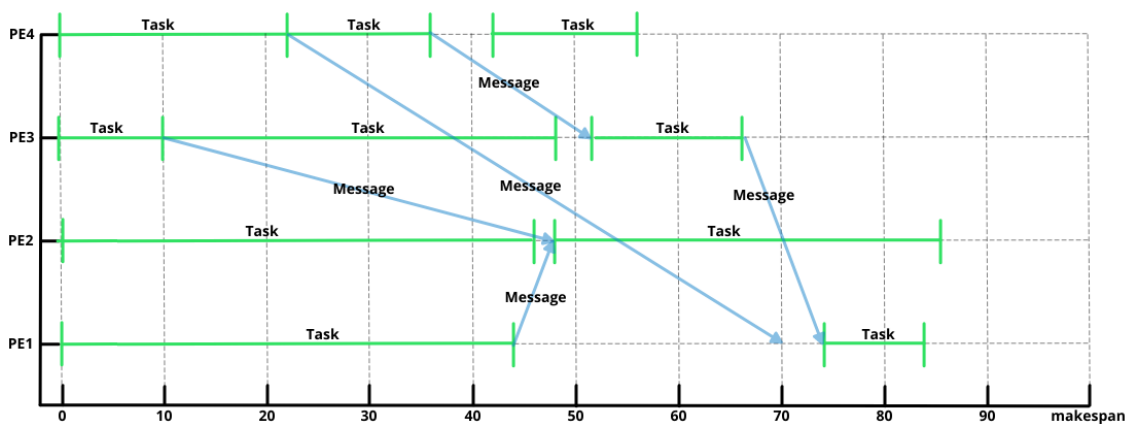


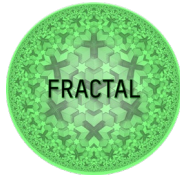
Figure 30 Schedule Model (SM)

Each SM is validated against some conditions to prevent runtime system failure due to the deployment of schedules containing errors. Conditions checked against each SM include:

- for all jobs in the AM, there is a corresponding temporal and spatial allocation to the PM
- for all jobs allocated to any PE in the PM, there are no temporal overlaps in the guaranteed allocation slot
- for all messages in the AM, there is a corresponding temporal and spatial allocation to the PM
- for all message allocations, injection times of messages are after the finish times of sender tasks
- all constrained tasks are allocated after the arrival of a preceding message
- for all communication resources in the PM, there are no temporal collisions in the routing of messages

### 8.2.5 Multi-schedule Graph (MSG)

The Hierarchical Metascheduler (HM) generates an MSG for every CM. The creation of the MSG is described in Figure 31. Each event in the CM is applied to the AM or PM, and a new schedule is created, representing an event-specific adaptation of the system application. Each new schedule is added to the MSG.



Project	FRACTAL		
Title	Safety, Security & Low Power Techniques		
Del. Code	D4.3		

---

**Algorithm 1:** GA-based meta-scheduler
 

---

**Input:** Application, Platform and Context Models

**Output:** MSG


```

1 Initiate Application model as  $AM$ ;
  Initiate Platform model as  $PM$ ;
  Initiate Context model as  $Cal$ ;
  Initiate Multi Schedule Graph as  $MSG$ ;
  Function  $AM, PM\{$ 
2   Construct initial population ( $AM$ );
   Selection;
   Crossover;
   Mutation;
   Fitness evaluation (Genome in population,  $PM$ );
3   return Schedule with best makespan
4 }
5 Function meta-scheduler ( $AM, PM, Cal,$ 
   $Schedule\{$ 
6   Take earliest event in  $e$  in  $Cal$ .
    $cal' =$  Remove  $e$  from  $cal$ .
    $AM', PM' =$  Modify  $AM, PM$  according to  $e$ ;
   New schedule = GA-scheduler( $AM', PM'$ )
   Add New schedule to MSG as node ( $S_n$ );
   Add  $e$  to MSG as edge ( $e$ );
   If  $cal$  is not empty :
   meta-scheduler( $AM, PM, Cal, New\ schedule$ );
   end if;
   return  $MSG$ 
7 }
8 begin
9   base schedule = GA-scheduler( $AM, PM$ );
   add base schedule to MSG as node ( $S_0$ );
   meta-scheduler( $AM, PM, Cal, base\ schedule$ )
10 end
  
```

---

Figure 31 GA-based metascheduler algorithm

A link between two schedules is defined by a context event signifying a transition from a current schedule to an adapted schedule, as illustrated in Figure 32. The MSG is a graph of schedules and events used by the FRACTAL node at runtime for adaptation to runtime events by switching the current system schedule from one node to the next in the MSG.

	Project	FRACTAL		
	Title	Safety, Security & Low Power Techniques		
	Del. Code	D4.3		

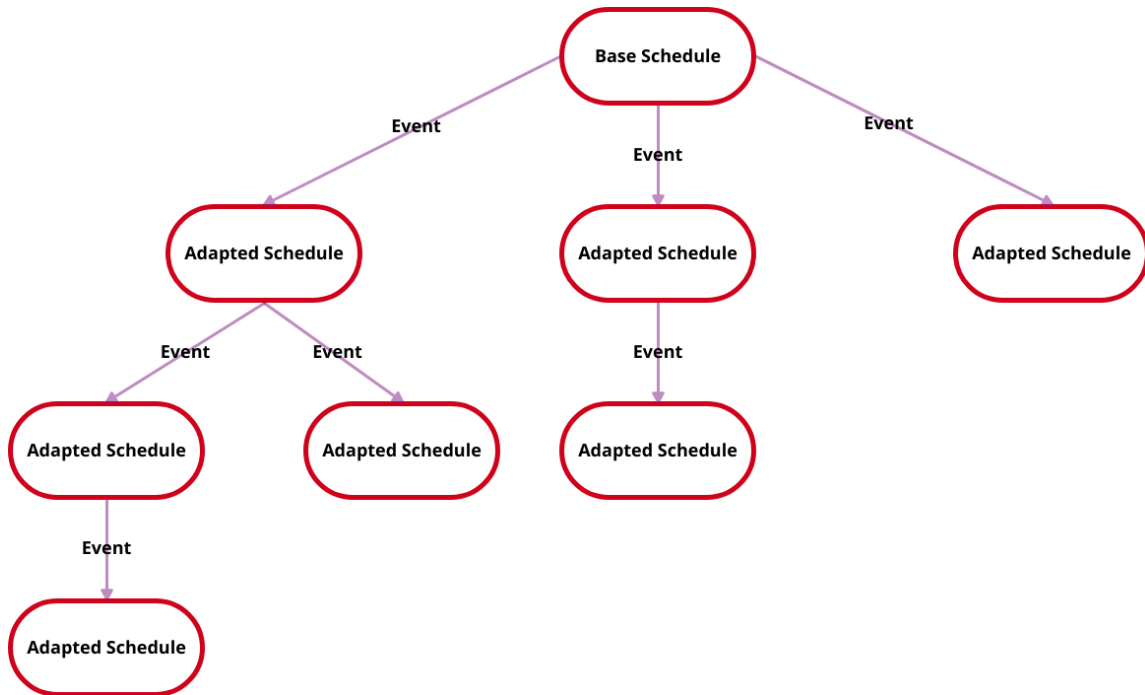


Figure 32 Multi-schedule Graph (MSG)


### 8.2.6 AI Metascheduling

After the generation of the MSG described in the previous chapter, an AI model needs to be developed. The AI Metascheduling extracts the data from the MSG at design time for the training process to subsequently make the predictions at runtime.

AI Metascheduling consists of two sections:

- **AI Model:** The fundamental idea of an AI model is to capture the structure of a scheduling graph, the relation of a schedule with another schedule as the result of a context event. The AI model is trained to predict the output temporal priorities. After the temporal priorities of the valid schedule are generated, they are sent to the reconstruction Model.
- **Reconstruction and Safety check:** This component takes predicted temporal priorities and performs a schedule reconstruction. In addition, it performs safety checks to make sure precedence constraints are not violated. The predicted temporal priorities decide the order in which the tasks are allocated. The spatial allocation is done based on the communication costs and the earliest start time of the jobs. Finally, a second safety check is performed to avoid message collisions.

Figure 33 shows the flow diagram of the Hierarchical Metascheduler by dividing the implementation into design time and runtime.

	Project	FRACTAL		
	Title	Safety, Security & Low Power Techniques		
	Del. Code	D4.3		

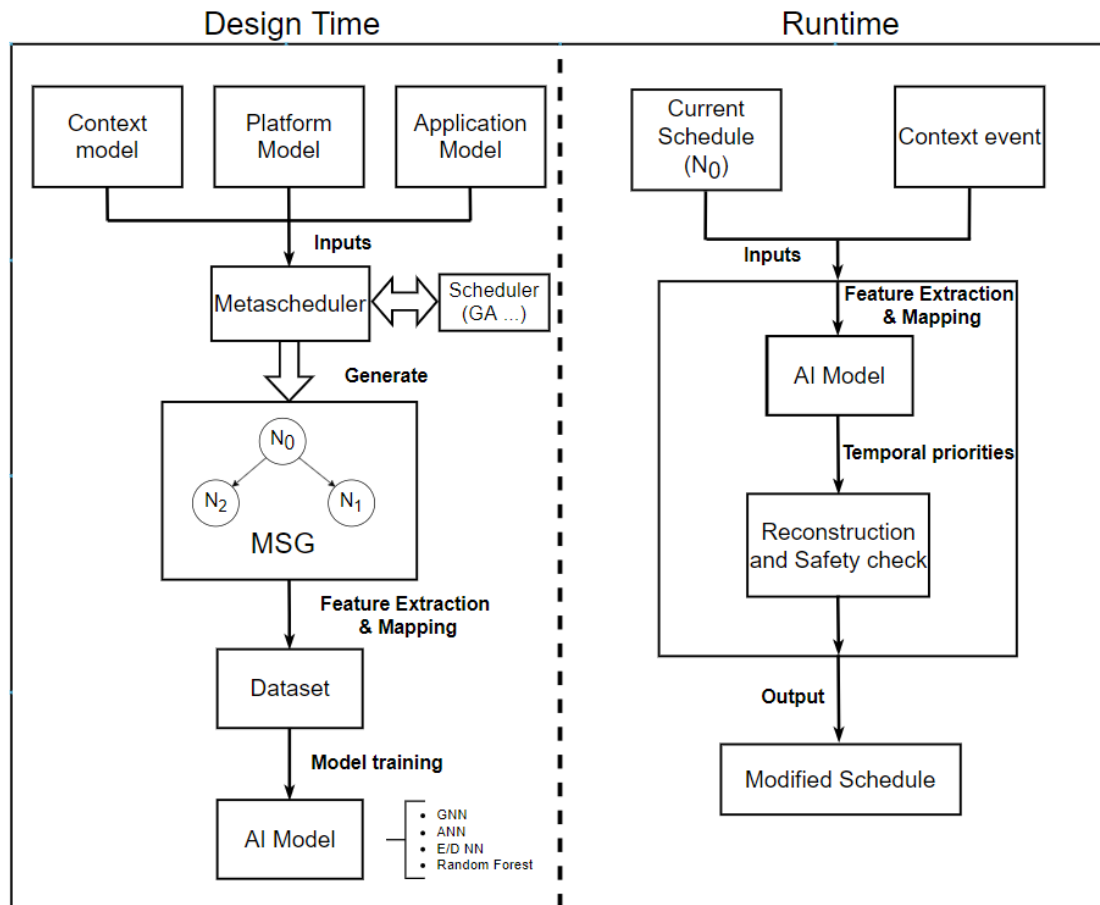



Figure 33 Hierarchical Metascheduler implementation diagram

### 8.3 Testing and evaluation

The deployment component is available on the FRACTAL GitHub page (<https://github.com/project-fractal/WP4>). Partners interested in utilizing them can download them and follow the README files before testing.

The Genetic Algorithm (GA) metascheduler was used to generate the samples for an application model containing the tasks and context events. The AI models evaluated were RFC, ANN, E/D NN and GNN. A more extended explanation of the development of these models can be found in [SIEG-HM2] and [SIEG-HM3].

In [SIEG-HM2] there were investigated the performance of RFC, ANN and E/D NN. An application model was created for five different job sizes: 15 jobs, 20 jobs, 30 jobs, 40 jobs and 60 jobs, all of them with diverse input feature sizes. The metascheduler was deployed using one platform model. The TensorFlow library is used to implement the ANN and the E/D NN. The Sequential class is used to create a fully connected (dense) ANN. Three hidden layers are configured with 35, 18, and 10 neurons for the first, second, and third layer, respectively. Based on the outcomes of numerous trials, these neuron sizes and the number of layers were chosen. The input and output sizes

	Project	FRACTAL		
	Title	Safety, Security & Low Power Techniques		
	Del. Code	D4.3		

are based on the number of features of each dataset, as explained above. A learning rate of 0.001 is configured, and the batch size is set to 128. For the E/D NN, an encoding dimension of 64 is selected as optimal after multiple trials.

The Scikit-Learn (Sklearn) python library is used for the Random Forest Classifier model, with a maximum depth of 10, a minimum sample leaf of 5, a max feature of 150, and 50 trees. These parameters were attained using the "GridSearchCV" class.

Table 6 Detailed performance evaluation of the models (RFC, ANN and E/D NN)


Model	# Jobs	Input feature size	Output feature size	Accuracy
Random Forest Classifier	15	339	180	95.33%
	20	513	300	80.39%
	30	609	690	77.25%
	40	873	1320	82.34%
	60	1233	3120	60.94%
Artificial Neural Network	15	339	180	86.86%
	20	513	300	85.94%
	30	609	690	83.60%
	40	873	1320	81.23%
	60	1233	3120	81.05%
Encoder/Decoder Neural Network	15	339	180	98.39%
	20	513	300	96.36%
	30	609	690	94.84%
	40	873	1320	92.47%
	60	1233	3120	87.67%

Table 6 shows the model accuracies against the number of jobs. E/D NN is seen to perform better than RFC and ANN. The highest accuracy was attained across all job sizes. Nevertheless, accuracy declines as the number of jobs increases. The decreasing accuracy is due to the fixed model structure and the increasing complexity model.

Similarly, the accuracy of the ANN model also reduces with the increasing number of jobs. The decrease in accuracy is not as much as both E/D NN and RFC. The decreasing accuracy for ANN ranges from 86.86% to 81.05%. In contrast, the decreasing accuracy for E/D NN ranges from 98.39% to 87.67% and RFC from 95.33% to 60.94%. The RFC performed more than the ANN for small job sizes but is the most unstable model with varying job sizes. For this reason, it can be drawn that the RFC does not provide a desirable generalized model for the metascheduling application when the model structure is fixed.

## GNN

[SIEG-HM3] deployed a Graph Neural Network to learn the multi schedules from the metascheduling algorithm required for adaptation. Py-Torch geometric library was

	Project	FRACTAL		
	Title	Safety, Security & Low Power Techniques		
	Del. Code	D4.3		

used to perform classification on the MSG’s irregular data structure and train a continuous kernel-based convolutional operator GNN based model. The model utilizes different Multi-Layer Perceptrons (MLPs) that contains 2 layers for edge features, and 4 layers for node features, having a total of 192,957 adjustable weights. Schedules generated from the GNN model were compared to the schedules from the GA metascheduler to determine the quality of schedules produced in terms of makespan. In addition, we compared our approach to schedules generated by the List scheduling (LS) technique. The LS uses the bottom level to determine task temporal priorities while the GNN based scheduler uses priorities learned from the GA. The schedules are categorized according to the number of tasks that occurred after the context event time in each schedule of the MSG.

Figure 34 represents a schedule example in which a context event occurred during task id 4. In this example, we considered as the context event a slack of 50%. Tasks that started before the event occurrence remain the same. However, to take advantage of the slack event, we invoke the metascheduler to recompute a new schedule on the remaining tasks to adapt to the scenario. By so doing, we get a better makespan which can then be utilized for energy saving.

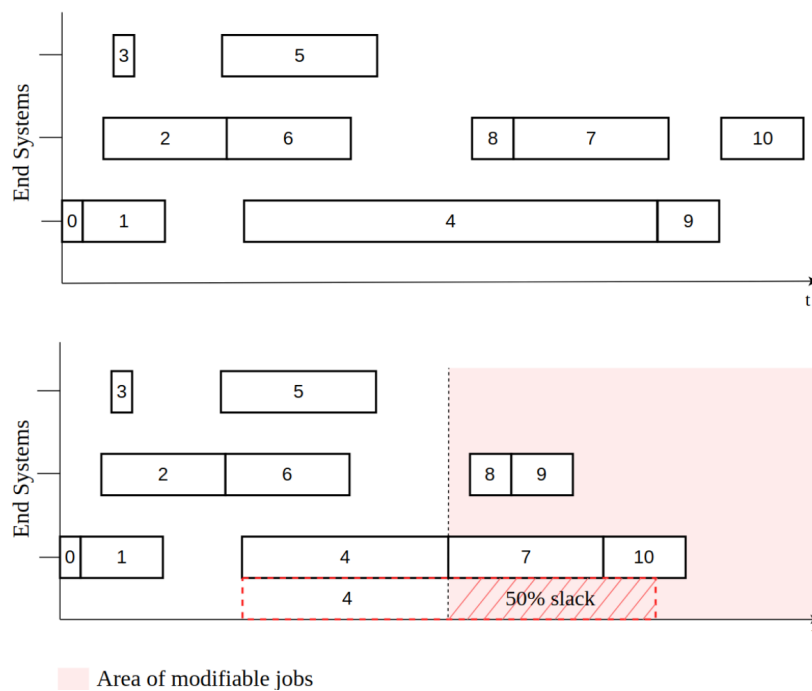



Figure 34 Schedule reconfiguration due a slack event

Each sample in the generated 16,384 samples is categorized according to the number of modifiable tasks. And each category is tested separately on all the metascheduling algorithms.

Throughout the training process, the accuracy of the GNN inference model training operation is calculated by validating 10% of the training dataset for each epoch. The results show a training accuracy of 97% at the end of training at 180 epochs. In

	Project	FRACTAL		
	Title	Safety, Security & Low Power Techniques		
	Del. Code	D4.3		

addition, 20% of the entire dataset was used for accuracy validation testing, an accuracy of 76% was obtained.

Figure 35 shows the output results for the makespans for each schedule generated by the GA, GNN and LS.

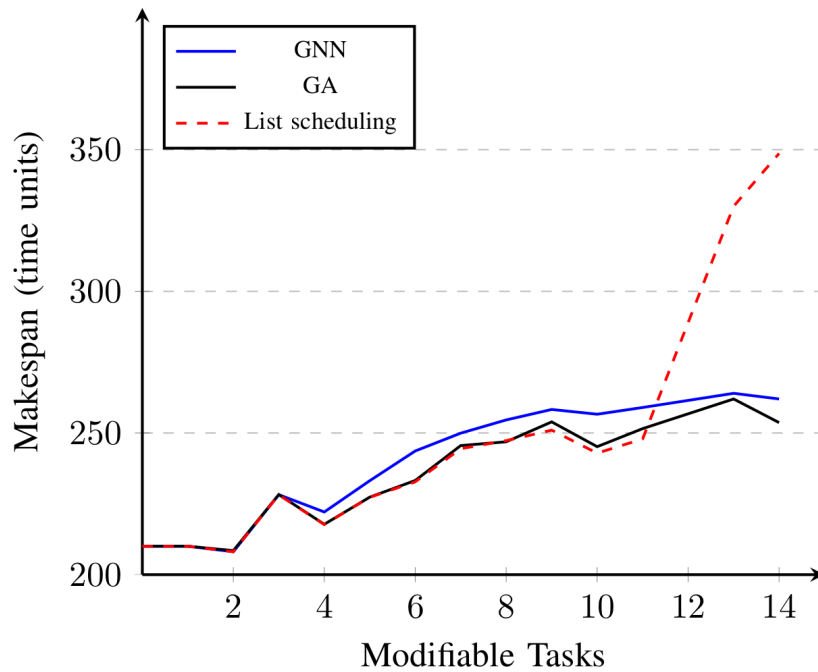



Figure 35 Makespan comparison

Throughout the dataset, each schedule in the MSG is given a label that would define the number of modifiable tasks in the schedule. This is determined by the occurrence time of the context event itself. Schedules with the same labels are grouped together and compared with different meta-scheduling algorithms. The results show that when the scheduling problem is simple with less than ten modifiable tasks, all meta-scheduling algorithms perform almost the same, and the GNN performs the worst with an average ranging from 10 to 20-time units' difference from the GA and LS. However, when the complexity of the scheduling problem is increased with a more significant number of tasks, the quality of makespan for the list scheduling falls behind GNN and GA techniques. The results show that in terms of makespan quality, the GA ranks first, followed by the GNN at rank 2, and finally, the LS. GA is unsuitable for hard real-time systems due to its unpredictable high timing and computational requirement. On the other hand, the LS is desirable for hard real-time applications due to its computational speed compared to the GA. Nevertheless, since the makespan quality is nowhere compared to the GA, the proposed GNN technique provides a way to utilize the benefits of the GA in application with stringent timing requirements.



	Project	<b>FRACTAL</b>		
	Title	<b>Safety, Security &amp; Low Power Techniques</b>		
	Del. Code	<b>D4.3</b>		

The Vitis AI of the Versal board cannot support the structure of GNN models; however, the ANN model was implemented as a substitute, proven to have good results.

**[SIEG-HM1]** Muoka, P., Onwuchekwa, D., & Obermaisser, R. (2021). Adaptive Scheduling for Time-Triggered Network-on-Chip-Based Multi-Core Architecture Using Genetic Algorithm. *Electronics*, 11(1), 49.

**[SIEG-HM2]** Onwuchekwa, D., Dasandhi, M., Alshaer, S., & Obermaisser, R. (2022). Artificial Intelligence-Based Meta-Scheduling for Adaptive Time-Triggered Networks. In *2022 27th International Conference on Emerging Technologies and Factory Automation (ETFA)*, IEEE.

**[SIEG-HM3]** Alshaer, S., Lua, C., Muoka, P., Onwuchekwa, D., & Obermaisser, R. (2022). Graph Neural Networks Based Meta-scheduling in Adaptive Time-Triggered Networks [Unpublished manuscript]. Department of Embedded Systems, University of Siegen.




Project	<b>FRACTAL</b>		
Title	<b>Safety, Security &amp; Low Power Techniques</b>		
Del. Code	<b>D4.3</b>		

## 9 Conclusions

---

The preliminary implementations of the AI-based algorithms for safe and efficient temporal resource allocation that were presented in the previous deliverable D4.1 have been developed in this document. In addition, several building blocks and components have been developed to assist in accomplishing the T4.2 objectives and might be demodulated or used by any Use Case. Specifically, these are capabilities for AI supported adaptability in PULP, Versal RPU access to AI acceleration, and AI scheduling components. The design, implementation, testing, and evaluation of these components have been described in detail.

	Project	<b>FRACTAL</b>		
	Title	<b>Safety, Security &amp; Low Power Techniques</b>		
	Del. Code	<b>D4.3</b>		

## 10 List of figures

Figure 1 Fractal system architecture .....	6
Figure 2 Big picture of the FRACTAL project .....	11
Figure 3 Capabilities for AI supported adaptability in PULP integrated in the FRACTAL big picture .....	12
Figure 4 Integration of Versal RPU access to AI acceleration in the big picture.....	13
Figure 5 Integration of AI Scheduling in the big picture .....	13
Figure 6 PULPissimo + Accelerator.....	14
Figure 7 ISA extension for ML inference.....	15
Figure 8 HWPE .....	15
Figure 9 SW Stack for ML inference.....	16
Figure 10 Low-Precision FP dot product extension – Performance results .....	16
Figure 11 RPU Access to APU AI Inference Application .....	19
Figure 12 AI scheduling components .....	22
Figure 13 Application and Platform model .....	22
Figure 14 Genome cells .....	23
Figure 15 Job priorities transformation .....	25
Figure 16 S2P algorithm .....	26
Figure 17 ANN architecture .....	27
Figure 18 Schedule visualization .....	27
Figure 19 Message collision.....	28
Figure 20 H5 file generation.....	28
Figure 21 ANN layer structure .....	29
Figure 22 Compilation process of AI models .....	31
Figure 23 AI application.....	32
Figure 24 SD Launch.....	33
Figure 25 Makespan comparison.....	34
Figure 26 Computation time comparison.....	34
Figure 27 Scheduling solution modeled as a heterogeneous graph .....	36
Figure 28 Hierarchical Platform Model .....	40
Figure 29 Application Model (AM).....	41
Figure 30 Schedule Model (SM) .....	42
Figure 31 GA-based metascheduler algorithm .....	43
Figure 32 Multi-schedule Graph (MSG) .....	44
Figure 33 Hierarchical Metascheduler implementation diagram .....	45
Figure 34 Schedule reconfiguration due a slack event .....	47
Figure 35 Makespan comparison.....	48




Project	<b>FRACTAL</b>		
Title	<b>Safety, Security &amp; Low Power Techniques</b>		
Del. Code	<b>D4.3</b>		

## 11 List of tables

---

Table 1 Components brief description and their contribution to fulfilling the T4.2 objectives.....	7
Table 2 Job features.....	25
Table 3 10-Job Dataset results .....	36
Table 4 40-Job Dataset results .....	37
Table 5 Prediction accuracies .....	37
Table 6 Detailed performance evaluation of the models (RFC, ANN and E/D NN)...	46
Table 7 List of abbreviations .....	53

	Project	<b>FRACTAL</b>		
	Title	<b>Safety, Security &amp; Low Power Techniques</b>		
	Del. Code	<b>D4.3</b>		

## 12 List of abbreviations

Table 7 List of abbreviations

Acronym	Title
ACO	Ant Colony Optimization
AI	Artificial Intelligence
AM	Application Model
API	Application Programming Interface
APU	Accelerated Processing Unit
BCE	Binary Cross-Entropy
CM	Context Model
DMA	Direct Memory Access
DPU	Deep Learning Processing Unit
EDP	Energy-Delay Product
GA	Genetic Algorithm
GNN	Graph Neural Network
GPU	Graphics Processing Unit
HDF	Hierarchical Data Format
HM	Hierarchical Metascheduler
HW	Hardware
IoT	Internet of Things
ISA	Instruction Set Architecture
JSON	JavaScript Object Notation
LS	List Scheduling
ML	Machine Learning
MLP	Multi-layer Perceptron
MMU	Memory Management Unit
MSG	Multi-schedule Graph
NoC	Network on Chip
PE	Processing Element
PM	Platform Model
PULP	Parallel Ultra Low Power
QoS	Quality of Service
ReLU	Rectifier Linear Unit
RPU	Real time Processing Unit
SNAP	Stanford Network Analysis Platform
SoC	System on Chip
SW	Software
TT	Time-triggered
TTNoC	Time-triggered Network on Chip
UC	Use Case
WCET	Worst Case Execution Time