Towards a Fault Detection Infrastructure for the European XFEL

Vom Promotionsausschuss der
Technischen Universität Hamburg

zur Erlangung des akademischen Grades

Doktor-Ingenieur (Dr.-Ing.)

genehmigte Dissertation

von
Gianluca Martino

aus
Filderstadt

2024

REVIEWERS

- Prof. Dr.-Ing. Görschwin Fey
- Dr. Holger Schlarb
- Prof. Dr. rer. nat. Volker Turau

ORAL EXAMINATION DATE

24.08.2023

IMPRINT

Towards a Fault Detection Infrastructure for the European XFEL

IDENTIFIERS

DOI: https://doi.org/10.15480/882.9087
Handle: https://hdl.handle.net/11420/45238
ORCID: https://orcid.org/0000-0002-7838-3844

COLOPHON

This thesis was typeset using LaTeX and the `memoir` documentclass. The initial template has been developed by Friedrich Wiemer[1]. It is based on Aaron Turon's thesis *Understanding and expressing scalable concurrency*[2], itself a mixture of `classicthesis`[3] by André Miede and `tufte-latex`[4], based on Edward Tufte's *Beautiful Evidence*.

The bibliography was processed by Biblatex.
All plots are made with PGF/Ti*k*Z.

The body text is set 10/14pt (long primer) on a 26pc measure. The margin text is set 8/9pt (brevier) on a 12pc measure. Matthew Carter's Charter acts as both the text and display typeface. Monospaced text uses Jim Lyles's `Bitstream Vera Mono` ("Bera Mono").

[1] https://github.com/pfasante/phd_thesis

[2] https://aturon.github.io/academic/turon-thesis.pdf

[3] https://bitbucket.org/amiede/classicthesis/

[4] https://github.com/Tufte-LaTeX/tufte-latex

What I learned on my own I still remember.
—Nassim Nicholas Taleb

Summary

Particle accelerators are instances of large cyber-physical systems where the impulse toward complete automation is growing. The vast number of interacting systems combined with the growing number of automated processes makes such a system difficult for an operator to oversee. The problem becomes even more complicated if the system operates in a hostile environment, e.g., a high-radiation environment. Fault detection allows further automation of the operation toward high autonomy levels.

The goal of this thesis is to devise techniques to improve the fault detection capabilities of the Low-Level Radio Frequency (LLRF) control system of the European X-ray Free Electron Laser (XFEL). We propose an architecture that uses both anomaly detection and runtime monitoring for the detection of faults. The thesis is divided into two parts. Part I describes the contributions toward the usage of anomaly detection for detecting faults. We illustrate and compare the performance of various anomaly detection techniques for finding anomalous time points in time series using real sensor data from the digital portion of the LLRF system at the European XFEL. We then use time series anomaly detection, i.e., the problem of finding anomalous time series among a set of time series, to tackle the quench detection problem. In this part, we also describe the work done for the hardware acceleration of some of the algorithms used. Part II describes the flow that, starting from an existing hardware digital design, automatically derives runtime monitors, i.e., hardware property checkers that ensure that the system satisfies a correctness specification during operation. We start by describing the technique that automatically extracts properties from an existing design. We then suggest two applications for the generated specification: using the enumeration principle to generate a circuit that implements the specification and the generation of runtime monitors using High-Level Synthesis (HLS).

The evaluations conducted in this thesis provide valuable insights into the effectiveness of the proposed techniques. The proposed techniques have significant potential for improving the fault detection capabilities of the LLRF control system at the European XFEL and other similar systems. The results of this thesis can be used to guide the development of more efficient and effective fault detection techniques for distributed systems, particularly those operating in hostile environments.

Abstract

This thesis aims to analyze techniques for improving the fault detection capabilities of large cyber-physical systems. The Low-Level Radio Frequency (LLRF) control system of the European X-ray Free Electron Laser (XFEL) is the main target system. In Part I, the thesis describes the contributions toward the usage of anomaly detection for detecting faults. In Part II, the thesis describes the flow that, starting from an existing hardware digital design, automatically derives runtime monitors, i.e., hardware property checkers that ensure that the system operates correctly.

Kurzzusammenfassung

Ziel dieser Arbeit ist es, Techniken zur Verbesserung der Fehlererkennung in großen Cyber-physischen Systemen zu analysieren. Das Low-Level Radio Frequency (LLRF) Kontrollsystem des European X-ray Free Electron Laser (XFEL) ist das Hauptzielsystem. In Teil I der Arbeit werden die Beiträge zur Nutzung der Anomalieerkennung für die Fehlererkennung beschrieben. In Teil II wird der Ablauf beschrieben, der ausgehend von einem bestehenden digitalen Hardware-Design automatisch Laufzeitmonitore ableitet, d.h. Hardware-Eigenschaftsprüfer, die sicherstellen, dass das System korrekt arbeitet.

Contents

List of Algorithms

List of Figures

List of Tables

Acknowledgments

After applying the final touches to this dissertation, it is time to express my gratitude to all the people who helped along the way.

First, I would like to express my deepest gratitude to Görschwin Fey, whose guidance, support, and wisdom have been invaluable throughout this journey. I could not have asked for a better supervisor and mentor. I am very grateful for the opportunities given to me, starting with being part of his research group. Moreover, I want to thank all the past and present Institute of Embedded Systems team members. It is, and it has been a pleasure working alongside each one of them.

I would also like to thank Holger Schlarb, Julien Branlard, and Lukasz Butkowsky for their invaluable support and direction. Furthermore, thanks to the rest of the MSK team at DESY. Many people in this group have contributed in various ways to my scientific contributions and, ultimately, to achieving my goal.

I am also grateful to the DASHH graduate school and all its members for the scientific exchanges and their introductions to a plethora of vastly different arguments, all under the umbrella of data science.

Most importantly, I want to thank Maria for her support, encouragement, love, and companionship throughout this journey. Words cannot express my gratitude for having her by my side.

Thanks to Carmelo and Concetta for their unchanging faith in my abilities and the light-hearted moments we shared. Thanks also to all my friends, whose support and love have enriched this experience.

Finally, I must thank my parents, Angelo and Margherita, for having planted the seed of curiosity in me and for the concrete support in all the phases of my studies. Without them, I'm not sure I would have reached this goal.

Part I

PROLOGUE

1

Introduction

"Reflect before you think."
—Umberto Eco

"Then why do you want to know? Because learning does not consist only of knowing what we must or we can do, but also of knowing what we could do and perhaps should not do."
—Umberto Eco

The complexity of digital systems grows faster than the verification capabilities. This escalating complexity is not merely a technical challenge; it poses significant practical concerns in terms of system dependability. As systems expand and interconnectivity increases, the potential for systemic vulnerabilities and unexpected failure modes grows exponentially. The intricacy of interactions between numerous subsystems and components can lead to emergent behaviors, which are exceedingly difficult to foresee or model accurately. This unpredictability is further compounded by the fact that VLSI-based systems, with their miniature and densely packed circuits, are inherently susceptible to physical phenomena leading to transient faults that can cause unpredictable system behavior.

Moreover, in distributed architectures, the communication and coordination between different subsystems introduce an additional layer of complexity. The heterogeneity of components, varying latencies, and the need for synchronized operation across different nodes demand robust and fault-tolerant communication protocols. Yet, ensuring such resilience in the face of transient faults and system failures becomes an increasingly difficult task as the system scales.

Traditional verification methods struggle to keep pace with the rapid advancement and growing intricacy of these systems. New methodologies and tools are required, ones that can intelligently predict potential faults and vulnerabilities, model the behavior of intricate systems under various scenarios, and provide insightful diagnostics to pinpoint the root cause of issues when they arise. Furthermore, there is a need for sophisticated fault-tolerance mechanisms that can not only detect and diagnose faults but also ensure that the system can continue to operate correctly or degrade gracefully in their presence.

Addressing the described complexities inherent to VLSI-based distributed systems necessitates a focus on dependability. Dependability, as a measure of system robustness and trustworthiness, plays a fundamental role in ensuring that these advanced digital systems align with operational standards and expectations, despite the complexities involved. The concept of dependabil-

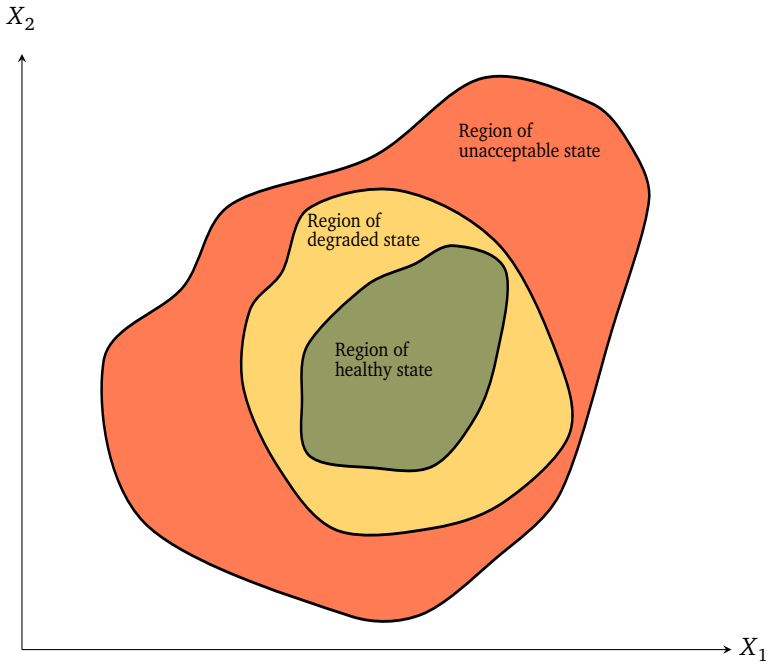


FIGURE 1.1: Representation of a system's state space at a specific point in time.

[Avi+04] Avizienis et al., "Basic Concepts and Taxonomy of Dependable and Secure Computing"

ity [Avi+04] of a system includes the following set of attributes:

- *reliability*: very few errors occur in the system;
- *availability*: downtime is minimized;
- *maintainability*: repair is easy and quick to carry out;
- *safety*: the system poses the minimum possible risk to users;
- *integrity*: no unintended system or information modification is possible;
- *confidentiality*: no unintended access to information is possible.

Each of those attributes is usually not clearly demarcated from the others, and the importance of each attribute in this list depends on the specific field of application. For example, a system that processes scientific data is less concerned with *confidentiality* than a banking system. This thesis concentrates only on the *reliability* and *availability* aspects.

A reliable and available system must possess the ability to, at least partially, self-adapt, i.e., the system can deliver service autonomously at varying environmental conditions and operating contexts. Self-healing was proposed to handle service interruptions by creating systems that can recover automatically from unexpected situations. The concept of self-healing was introduced by taking inspiration from biological organisms. One of the most interesting aspects of the self-healing mechanism in larger biological organisms is that damage repair occurs in a decentralized way and without conscious effort. Fundamentally, a self-healing system is a reliable, available, and survivable system.

The creation of a self-healing system poses a series of challenges that are not trivial to solve. The first question to answer is "what is a healthy or unhealthy state of a system?". In complex systems, the normal or healthy state cannot be defined precisely and varies depending on the specific user, application, and environment. Moreover, acceptable behavior in a complex system varies with time. Finally, depending on the application, a degraded

state in which some functions are not operational can be an acceptable state. Having established that a healthy state is a fuzzy operating zone, there remains the need to define when the system cannot be considered functioning anymore, and the system recovery should start [Gho+07]. In Figure 1.1, we represented a simplified health state space of a generic system. In a real system, the actual state space is composed of many dimensions, and each state can occupy multiple regions. A self-healing system needs to be able to monitor the system and detect deterioration of the health state, i.e., the crossing of a boundary in the state space.

[Gho+07] Ghosh et al., "Self-healing systems - survey and synthesis"

Monitoring complex systems can be done using multiple approaches. Healthy region boundaries are fixed threshold values of specific parameters, representing the normal operating region of a system, that can be used as triggers. Fixed thresholds can be refined by using models of the system under observation. Signal models [CP00] are a mathematical representation of one or more signals of a system. Signal models can be used to describe the characteristics of a signal, such as its frequency, amplitude, phase, and other features. They can also be used to predict how a signal will behave under different conditions. A system model [Ble+15; Naw+18] represents the interacting components that work together and can take many forms, e.g., mathematical models, physical prototypes, or conceptual diagrams. They can be used to understand the behavior of a system, predict its performance, optimize its design, or control its operation. Runtime monitoring is a method for ensuring the correctness of concurrent systems by observing and verifying the current execution of the system. Runtime monitors are synthesized from high-level correctness specifications and executed alongside the system to determine whether the specification is satisfied or violated. There are two main types of runtime monitoring: online, in which the system is directly monitored while it is executing, and offline, in which relevant system events are recorded and then inspected after the system has stopped executing. Online monitoring can be further divided into synchronous, in which the monitor is tightly coupled with the system and executes at the same time as the system, and asynchronous, in which the monitor executes independently of the system. The choice of the correct approach depends on the needs of the system to be monitored. Anomaly detection is the process of identifying unusual or unexpected patterns in data that may indicate a problem or deviation from the norm. It is often used to identify fraudulent activity and equipment failures, and for detecting and identifying markers that indicate the deterioration of the health state of a system. Anomaly detection can be performed using a variety of techniques, including statistical analysis, machine learning algorithms, and data visualization methods. These techniques can be applied to a wide range of data sources, including network traffic and sensor readings.

[CP00] Chan and Pang, "Fabric defect detection by Fourier analysis"

[Ble+15] Blesa et al., "An Interval NLPV Parity Equations Approach for Fault Detection and Isolation of a Wind Farm"

[Naw+18] Nawaz et al., "Anomaly Detection for the European XFEL using a Nonlinear Parity Space Method"

The choice of the correct monitoring approach depends on the system that needs to be monitored and on its observability, i.e., the ability to infer the internal state of a system from the external outputs, and diagnosability, i.e., the ability to provide a correct diagnosis or root cause of a fault.

A Large-scale Cyber-physical System

Cyber-physical systems (CPS) represent the integration of computation, net-

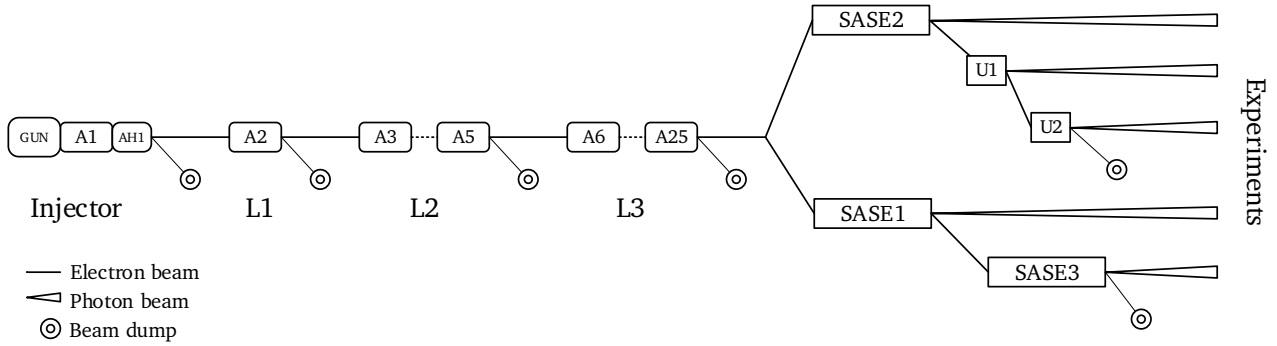


FIGURE 1.2: Simplified schematic of the European XFEL.

working, and physical processes. In the context of a particle accelerator, CPS can be understood as the intricate interplay between the physical components of the accelerator (like magnets, vacuum tubes, and detectors) and the computational algorithms that control these components. This integration is crucial for the precise operation, real-time monitoring, and adaptive control required in the complex environment of a particle accelerator. A particle accelerator relies on an array of sensors and actuators connected through computational networks. These networks process vast amounts of data in real time, adjusting physical processes based on complex algorithms and feedback loops. For instance, maintaining the beam stability in a particle accelerator requires real-time data processing and immediate feedback to adjust the vast number of parameters that can be manipulated.

In most cyber-physical systems, particularly in the context of a particle accelerator, the concept of distributed systems is central. While various definitions of distributed systems exist, often divergent and lacking consensus, a general understanding can be convened for our discussion: a distributed system is essentially a collection of autonomous computing elements that, to its users, manifests as a single coherent system [TS17]. This loose characterization matches very well with how a particle accelerator works as part of a cyber-physical system. In a particle accelerator, numerous autonomous components work cooperatively. Larger experiments are composed of thousands of components connected by high-speed data links [Abe+06; Sto11]. Each component, potentially a complex system in itself, operates under the orchestration of a control software. This distributed nature is not immediately apparent to the scientists and engineers who interact with the accelerator; to them, it presents itself as a single system.

The essence of a distributed system in this context is to abstract the complexity and autonomous nature of individual components, providing a seamless and integrated experience. This integration is achieved through a complex layer of networking, communication protocols, and data processing algorithms, ensuring that the individual actions of separate components synchronize harmoniously. Thus, in the cyber-physical landscape of a particle accelerator, the principles of distributed systems are not just an operational necessity but a strategic enabler. They allow the harmonious integration of physical processes with cyber capabilities, ensuring that the accelerator operates not as a disjointed array of independent components, but as a cohe-

[TS17] Tanenbaum and Steen, *Distributed Systems*

[Abe+06] Abela et al., *XFEL: The European X-Ray Free-Electron Laser - Technical Design Report*

[Sto11] Stohr, "Linac Coherent Light Source II (LCLS-II) Conceptual Design Report"

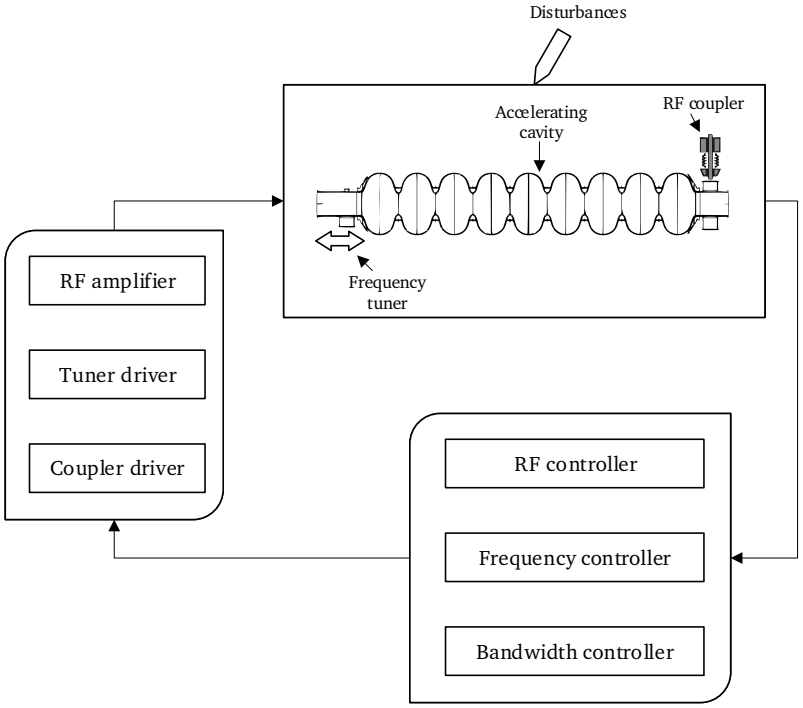


FIGURE 1.3: Conceptual view of the control loop of the European XFEL RF system.

sive, efficient, and purpose-driven system, pushing the frontiers of particle physics. In this thesis, the primary application is on particle accelerators; however, the concepts developed in the following sections can be applied to a wide variety of distributed systems.

The applications of particle accelerators range from defense and industrial use to medical and environmental [Mou22]. For many applications, the system's dependability is of primary importance, especially in the long term, where autonomous or partially autonomous accelerators will be a reality [Buo19; Eic+21]. An example of such a machine is the European X-ray Free Electron Laser (XFEL). The European XFEL is a machine composed of various types of strongly coupled systems, using cutting-edge technologies, and operating in a hostile environment. The machine needs to be able to reach very high performances, preserving flexibility and maintaining availability. The systems that operate the machine have millions of control parameters and generate more than 30 TB of data per day. A schematic of the European XFEL is represented in Figure 1.2. The figure shows the main components of the machine:

- the injector section, where bunched electrons are produced by a short-pulse laser impinging on a photocathode, accelerated to ultrarelativistic energies, and injected into the accelerating sections;
- the accelerating sections L1, L2, and L3 accelerate the electrons to their maximum energy;
- the SASE section, where the electrons stimulate the production of photons to be used in the experiments.

The European XFEL's Low-Level Radio-Frequency (LLRF) system [Bra+12] is responsible for the control of all 26 stations and is an example of a complex distributed system. The goal of the LLRF system is to create and maintain a stable accelerating field in all the accelerating cavities during the traversal

[Mou22] Mounet, *European Strategy for Particle Physics - Accelerator R&D Roadmap*

[Buo19] Buongiorno, "The Future of Particle Accelerators May Be Autonomous"

[Eic+21] Eichler et al., "First Steps Toward an Autonomous Accelerator, a Common Project Between DESY and KIT"

[Bra+12] Branlard et al., "The European XFEL LLRF system"

of each particle bunch. Moreover, the LLRF system needs to be able to counteract a variety of disturbances that affect the field's stability. The conceptual structure of the control loop integrating the high-power RF system is represented in Figure 1.3. The main actuators of the system, i.e., the Radio Frequency (RF) source, the RF couplers, and the frequency tuners, are connected to amplifiers and drivers that regulate the accelerating RF field, resonance frequency, and bandwidth. The amount of energy delivered to the accelerating cavities by the RF amplifiers and the voltages generated by the tuner drivers and the coupler drivers are regulated by dedicated controllers, part of the LLRF system.

At the system level, the LLRF system is composed of analog, digital, and mixed-signal boards, interconnected using a variety of connection media and protocols. The control functionalities are distributed among the different devices. Moreover, the system is composed of multiple layers that stack on top of each other. The hardware is directly connected to the sensors, measuring the parameters of the particle accelerator used for the control, and the actuators. The analog components down- or up-convert the signals to a frequency that can be sampled and processed by the digital part or to drive the system. The hardware layer is connected to the software layer by the firmware layer. The firmware layer controls the hardware components that are used for sampling and processing the signals. The firmware implements all parts of the control algorithm that need a high processing speed. Going up one layer, the firmware is controlled using specifically developed drivers for communicating and controlling the firmware components. The task of this layer is to run slower control algorithms and communicate with the host CPU, where the operations software and automation scripts are running, at the top layer. The system is implemented using the μTCA.4 (MicroTCA.4)[1] standard [Mic], and all the components that compose a station are integrated into a modular rack. Most of the hardware components contain a Field Programmable Gate Array (FPGA) that hosts the firmware components.

Failures on the lower layers reflect on the upper layers in ways that are difficult to predict and manage. For example, Figure 1.4 show a signal trace that erroneously triggers the emergency switch off of the system. Box 1 and Box 2 of Figure 1.4 contain the glitches in the data that result in faulty behavior. For this reason, the first step to realizing a system that can operate autonomously is to establish the foundation for an integrated fault detection scheme that can catch faults at the lower layers. However, the system's lower layers operate at high speeds, produce large volumes of diagnostic data, and exchange information using high-speed communication channels. The fault detection system needs to be able to keep up with the performances of the systems to be monitored.

Such a fault detection system significantly increases the availability and reliability of a system by constantly monitoring its performance and identifying any irregularities or issues. Through this continuous monitoring, the system can catch and address faults early, often before they escalate into more severe problems. This immediate response not only prevents minor issues from causing major system failures but also drastically reduces downtime, as problems are swiftly managed and rectified. Moreover, the data gathered by the fault detection system can be analyzed to predict potential

[1] μTCA is a modular, open standard for building computer systems in a small form factor. μTCA.4 is a subset of the specification intended for use in large-scale scientific devices, such as particle accelerators or telescopes.

[Mic], *Micro Telecommunications Computing Architecture – Short Form Specification*

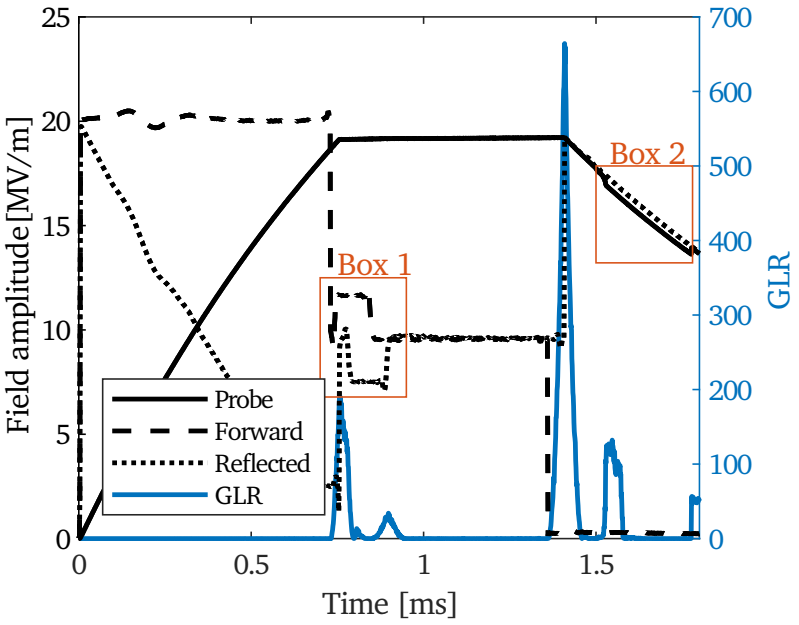


FIGURE 1.4: Example of a glitch in the signals processed by the LLRF system [EBT23].

failures. By repairing or replacing components before they fail, the system avoids unexpected breakdowns, further enhancing its availability. This predictive approach, coupled with the system's ability to maintain stability by ensuring that each component functions correctly, reduces the risk of cascading failures, especially in complex systems where one component's failure could affect many others. Consequently, a fault detection system is not just a tool for immediate problem-solving; it is a strategic component that contributes to the long-term efficiency, stability, and reliability of the system.

A FAULT DETECTION SYSTEM FOR THE LLRF SYSTEM

The concepts described and the problems investigated in the following are based on the design of a real-time fault detection system that is focused on the hardware and firmware level of the LLRF system. The overall idea for the design of a fault detection system that is integrated into a single μTCA.4 rack is depicted in Figure 1.5. The fault detection system is composed of a runtime monitor that ensures the correct execution of the system with respect to the specification. Additionally, anomaly detection servers collect signals from multiple systems to detect behaviors that do not conform with the defined health state. Depending on the speed of the signals and the severity of the possible fault, either the slow or fast anomaly detection server is used. This architecture is inspired by the ideas introduced for DiPS Monitor Architecture (DMonA) [Mic+02], a component-based architecture proposed for a self-adapting system of internet servers. This architecture introduces two distinct types of self-monitoring sensors:

1. STATE SENSOR, for monitoring internal system's states;
2. ANALYSIS SENSOR, for monitoring messages exchanged between systems.

We repurpose the concepts of STATE SENSOR and ANALYSIS SENSOR for our

[Mic+02] Michiels et al., "DistriNet: Self-adapting concurrency: the DMonA architecture"

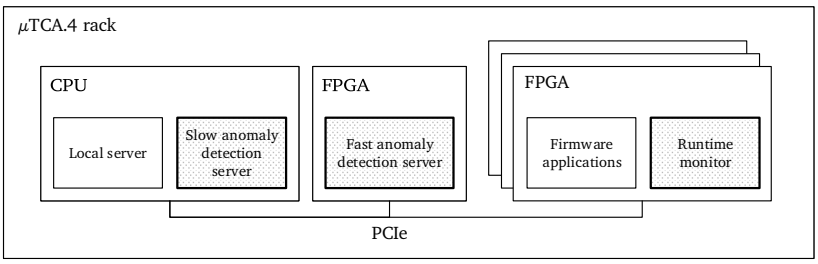


FIGURE 1.5: Rack-level design of the fault detection system. The highlighted elements are part of the fault detection system, while the non-highlighted ones are part of the initial system.

specific scenario: a STATE SENSOR is embedded in the FPGA fabric and has access to the internal states of the FPGA or of a specific IP core; an ANALYSIS SENSOR looks at the overall state of the system without having access to the full internal state.

The connection of neighboring fault detection systems permits the sharing of information. This gives a more exhaustive view of the state of the overall system at each station. This can be achieved with the existing infrastructure through the usage of the edge computing paradigm [Cao+20], only the strictly necessary data is shared, and most of the processing is done at the node where the data is produced. This makes it possible to reduce the data bandwidth pressure and improve the responsiveness of the fault detection system to faults. Such an interconnected network of fault detection systems resembles a computational immune system [FHS97]. Computational immune systems were proposed to face security problems in systems. However, we incorporate the same concepts for the creation of a fault detection architecture.

[Cao+20] Cao et al., "An Overview on Edge Computing Research"

[FHS97] Forrest et al., "Computer Immunology"

The realization of such a system requires the resolution of some open challenges in the area of fault detection, reliable systems, and self-healing systems. Initially, the approach begins with focusing on a limited set of target issues, which allows for a more concentrated and in-depth analysis of specific problems within the broader complex framework. Tackling these challenges to develop a fault detection system for such a complex setup is an exceedingly demanding task. It necessitates a multifaceted approach that combines expertise from various domains and typically requires a substantial investment of time, often amounting to many person-years. This intensive effort involves thoroughly understanding the intricacies of the system, devising effective monitoring strategies, and implementing robust self-healing mechanisms. The complexity of the endeavor is compounded by the need to ensure that the system not only detects and addresses faults in real-time but also adapts to evolving conditions and maintains a high standard of performance consistently.

1.1 GOALS AND OPEN CHALLENGES

The primary objective of this project is to design and test techniques that can be used for the implementation of an advanced fault detection system explicitly tailored for the Low-Level Radio Frequency (LLRF) digital systems at the European XFEL. The multifaceted nature of this project encompasses several specific goals aimed at enhancing anomaly detection within the LLRF infrastructure. The specific goals of the project include:

Goal 1 (Algorithm Selection and Adaptation)**.** *Identifying, adapting, and deploying algorithms suited for detecting anomalies within the digital systems governing the LLRF. This includes leveraging machine learning, statistical methods, or rule-based approaches tailored to the LLRF's specific characteristics.*

Goal 2 (Latency Optimization)**.** *Employing techniques such as pipelining, parallel processing, and optimizing data transmission across interconnected links to reduce processing time and enhance the overall fault detection responsiveness.*

Goal 3 (Automated Runtime Monitor Generation)**.** *Creating tools and frameworks for the automated generation of runtime monitors. This feature aims to streamline system deployment and maintenance, ensuring efficiency and adaptability.*

Goal 4 (Real-time Parameter Monitoring)**.** *Continuously monitoring critical parameters such as voltage, frequency, and phase in real-time to detect and respond to abnormal variations that surpass predefined thresholds.*

Goal 5 (Event Correlation Capability)**.** *Implementing systems capable of correlating multiple events or signals to uncover complex anomalies that might not be discernible through individual signal analysis alone.*

The successful achievement of these project objectives aims to significantly enhance the reliability, stability, and safety of operations within the European XFEL facility. The resulting fault detection system will play a critical role in early anomaly detection, ensuring uninterrupted operations and minimizing potential risks.

Fault detection, reliable systems, and self-healing systems have been topics of major interest in research and industry in the last 50 years. Despite the large body of knowledge in these research areas, there are still many open challenges due to the continuously evolving technology used for building systems. We can distinguish the open challenges depending on the methods used. In this thesis, we concentrate on fault detection using anomaly detection and fault detection using runtime monitoring.

Anomaly Detection

Anomaly detection techniques are designed to find anomalies in data. Anomalies are also known as outliers due to their characteristics of being often isolated from the rest of the data. As already mentioned, the field of anomaly detection is vast and extensively explored in very different fields such as intrusion detection [Por00; YD03; GT+09], fraud detection [BH02; Phu+10], medical applications [Lin+05], fault detection [DF96; DN00; Kin+02], and more [BGS08; Geb+13; HA04; CBK09]. However, due to the number of techniques developed, it is difficult to properly select the correct algorithm for the given problem. This becomes even more difficult when anomaly detection is applied in the field of fault detection for complex distributed systems. Anomaly detection algorithms are often designed to work well in a particular context, but they may not perform well when the context is different [Emm+16]. Moreover, anomaly detection algorithms often have difficulty handling imbalanced datasets where the number of normal instances far exceeds the number of anomalous instances.

[Por00] Portnoy, "Intrusion detection with unlabeled data using clustering"

[YD03] Yeung and Ding, "Host-based intrusion detection using dynamic and static behavioral models"

[GT+09] Garcia-Teodoro et al., "Anomaly-based network intrusion detection: Techniques, systems and challenges"

[BH02] Bolton and Hand, "Statistical Fraud Detection: A Review"

[Phu+10] Phua et al., "A Comprehensive Survey of Data Mining-based Fraud Detection Research"

[Lin+05] Lin et al., "Approximations to magic: finding unusual medical time series"

[DF96] Dasgupta and Forrest, "Novelty detection in time series data using ideas from immunology"

[DN00] Dasgupta and Nino, "A comparison of negative and positive selection algorithms in novel pattern detection"

[Kin+02] King et al., "The Use of Novelty Detection Techniques for Monitoring High-Integrity Plant"

[BGS08] Basharat et al., "Learning object motion patterns for anomaly detection and improved object detection"

[Geb+13] Gebhardt et al., "Document Authentication Using Printing Technique Features and Unsupervised Anomaly Detection"

[HA04] Hodge and Austin, "A Survey of Outlier Detection Methodologies"

[CBK09] Chandola et al., "Anomaly detection: A survey"

[Emm+16] Emmott et al., "A Meta-Analysis of the Anomaly Detection Problem"

In the context of systems in operation, we mainly deal with anomaly detection in time series. However, different anomaly types require different anomaly detection techniques. Time point anomalies, also time point outliers, are time points that stand out at a particular moment in time, either because they are significantly different from the other values in the time series (global outliers) or because they are significantly different from their neighboring points (local outliers). Time series anomalies, also time series outliers, are entire time series that differ from previous or future executions of the same system. They may or may not contain time points anomalies.

The following challenges arise:

Problem 1 (Time Point Outlier Anomaly Detection)**.** *Given a specific complex distributed system, select the appropriate time point anomaly detection algorithm for fault detection.*

Problem 2 (Time Series Outlier Anomaly Detection)**.** *Given a specific complex distributed system, select the appropriate time series anomaly detection algorithm for fault detection.*

Techniques based on Convolutional Neural Networks (CNNs) and convolutional layers have been proven to be successful not only in solving challenges related to time series anomaly detection [Zha+17; HTF09] but also in multiple other fields [Dai+20; Sch+21; Zhu+21]. However, the implementation of such algorithms in hardware requires an accurate selection of the correct platform. The choice of the correct platform is a challenge because it does not only depend on the optimization goal but also on the characteristics of the available hardware. Given the constantly evolving hardware and the new solutions tailored to the problem being made available by the hardware vendors, unbiased comparisons are needed.

[Zha+17] Zhao et al., "Convolutional neural networks for time series classification"

[HTF09] Hastie et al., *The Elements of Statistical Learning: Data Mining, Inference, and Prediction, 2nd Edition*

[Dai+20] Dai et al., "Fast and accurate cable detection using CNN"

[Sch+21] Scheinker et al., "An adaptive approach to machine learning for compact particle accelerators"

[Zhu+21] Zhu et al., "High-Fidelity Prediction of Megapixel Longitudinal Phase-Space Images of Electron Beams Using Encoder-Decoder Neural Networks"

Problem 3 (Hardware Implementation of CNNs)**.** *Given a specific convolutional neural network and a specific optimization goal, select the appropriate implementation platform for hardware acceleration.*

Addressing the initial two challenges, namely choosing a suitable algorithm for time point anomaly detection (Problem 1) and time series anomaly detection (Problem 2) for fault identification in a complex distributed system, is instrumental in fulfilling our objective of selecting and customizing algorithms (Goal 1), utilizing machine learning and statistical techniques. Furthermore, resolving the task of determining the optimal platform for implementing hardware-accelerated convolutional neural networks (CNN) (Problem 3) advances significantly our aim of reducing latency (Goal 2). By addressing these challenges, we also contribute to our objective of correlating multiple events (Goal 5), thereby facilitating the effective processing of event correlations.

▸ Runtime Monitoring

The application of runtime monitoring requires a set of formal specifications describing the expected behavior of each component of the system. One of the main challenges of runtime monitoring is to generate reliable, up-to-date

specifications. Manual creation requires a very large amount of work, and it is often unfeasible due to the associated costs.

Automated design understanding tools are able to analyze a design's components and provide insight into how it works. These tools use sophisticated algorithms to process the design's code and uncover patterns or relationships that would otherwise be difficult to detect. They can then be used as a prompt to make changes to the design to improve its performance or reliability, for understanding unknown or undocumented designs, and for the creation of runtime monitors if the generated set of assertions can be reviewed by a human designer.

Current automated assertion generation approaches for Register Transfer Level (RTL) designs [DeO+09; Vas+10] analyze and extract properties from simulation data. However, the effectiveness of the generated assertions is dependent on the quality and quantity of the simulation data used. In addition, the generated assertions tend to capture complex signal manipulations implemented in the RTL code, making them difficult to understand for human designers.

[DeO+09] DeOrio et al., "Inferno: Streamlining Verification With Inferred Semantics"

[Vas+10] Vasudevan et al., "GoldMine: Automatic assertion generation using data mining and static analysis"

Problem 4 (Automatic Specification Generation)**.** *Given an existing and operating large-scale system, automatically generate a readable set of assertions matching the specification.*

The next challenge is to implement the specification in runtime monitors that can ensure the correctness of the system. There are several monitoring frameworks and techniques available for software and hardware systems [Dru03; FS15; HR04; Kim+04], including the use of automata [II+20] and synthesis techniques for properties expressed in Timed Regular Expressions (TRE) and Signal Temporal Logic (STL) [Sel+17]. A framework for runtime monitoring of security properties using Bayesian networks and Metric Temporal Logic properties has also been proposed [MRS17]. However, none of the proposed frameworks is capable of generating LTL runtime monitors implementing the counting finitary semantics (c-LTL) [Bar+18], i.e., runtime monitors that can predict the future satisfaction or violation of one or more properties. The prediction of the future state of a runtime monitor is important in applications in which it is important to avoid the violation of the specification or for debugging.

[Dru03] Drusinsky, "Monitoring Temporal Rules Combined with Time Series"

[FS15] Francalanza and Seychell, "Synthesising correct concurrent runtime monitors"

[HR04] Havelund and Rosu, "An Overview of the Runtime Verification Tool Java PathExplorer"

[Kim+04] Kim et al., "Java-MaC: A Run-Time Assurance Approach for Java Programs"

[II+20] II et al., "Runtime Verification on FPGAs with LTLf Specifications"

[Sel+17] Selyunin et al., "Runtime Monitoring with Recovery of the SENT Communication Protocol"

[MRS17] Moosbrugger et al., "R2U2: monitoring and diagnosis of security threats for unmanned aerial systems"

[Bar+18] Bartocci et al., "A Counting Semantics for Monitoring LTL Specifications over Finite Traces"

Problem 5 (Runtime Monitor Generation)**.** *Given a specification of a system, generate a runtime monitor that can predict the future violation of the specification.*

By developing a readable set of assertions, tackling Problem 4, and devising a runtime monitor with the capability to forecast potential breaches of specifications, addressing Problem 5, we contribute to the achievement of Goal 3 and Goal 4.

The next section begins with a brief outline of the thesis, followed by a summary of the contributions in relation to the aforementioned challenges.

1.2 OUTLINE AND CONTRIBUTIONS

The remainder of the thesis is structured as follows. In Chapter 2, we discuss preliminary notions and give more details about the existing system. The technical parts of the thesis are then split in two.

In Part II, we first evaluate and compare the effectiveness, resource consumption, and runtime of various anomaly detection algorithms for time point outliers (see Chapter 3). For the analysis, we use historical sensor data. The goal of the analysis is to identify algorithms that let us automatically detect faulty behaviors of the LLRF system. Then Chapter 4 describes the application of anomaly detection for time series anomalies for the detection of quenches. Lastly, in Chapter 5, we evaluate the performance of the hardware implementation of convolutional neural networks.

Part III is split in three Chapters 6 to 8. We first discuss methods for the implementation of a design understanding tool and give more details about the property generation technique based on the syntax-guided enumeration method (see also [MRF18]). For the generated specification, we propose two applications: the usage of the same enumeration principle for the generation of a circuit implementing the specification and the generation of runtime monitors using High-Level Synthesis (HLS).

[MRF18] Martino et al., "Coverage-Guided CTL Property Enumeration for Understanding Models of Reactive Systems"

Finally, Chapter 9 concludes the thesis.

Prior to proceeding to the preliminaries chapter, which provides a deeper understanding of the basic concepts applied across the thesis, we first outline our list of contributions.

1.2.1 *Monitoring of Distributed Digital Systems using Anomaly Detection*

This section details the contributions to the application of anomaly detection for fault detection.

▸ COMPARISON OF ANOMALY DETECTION ALGORITHMS FOR FAULT DETECTION

In some applications, traditional classification methods are not suitable due to the low number of abnormalities, such as in high-integrity systems. Anomaly detection solves the problem but requires testing to ensure that the correct anomalies are found. Previous work analyzed the performance of anomaly detection for fault detection in specific applications [DF96; DN00; Kin+02] and as comparative evaluations with standard datasets [GU16; Din+14; Dom+20]. However, due to the great complexity of the LLRF system, we need to compare the different algorithms using the data coming from the LLRF system to determine which algorithms to use.

[DF96] Dasgupta and Forrest, "Novelty detection in time series data using ideas from immunology"

[DN00] Dasgupta and Nino, "A comparison of negative and positive selection algorithms in novel pattern detection"

[Kin+02] King et al., "The Use of Novelty Detection Techniques for Monitoring High-Integrity Plant"

[GU16] Goldstein and Uchida, "A comparative evaluation of unsupervised anomaly detection algorithms for multivariate data"

[Din+14] Ding et al., "An experimental evaluation of novelty detection methods"

[Dom+20] Domingues et al., "A Comparative Evaluation of Novelty Detection Algorithms for Discrete Sequences"

In [Mar+21], we analyze the performance of various anomaly detection techniques for time point outliers using real sensor data from the digital portion of the LLRF system at the European XFEL. In addition, we conduct experiments using publicly available data from other high-integrity systems.

This chapter addresses Problem 1 (Time Point Outlier Anomaly Detection). Our contributions are twofold: 1. we provide a coherent description of the selected algorithms, which, to our knowledge, has not been done for semi-supervised anomaly detection; 2. our evaluation of state-of-the-art algorithms applied to high-integrity digital systems can assist in the design

[Mar+21] Martino et al., "Comparative Evaluation of Semi-Supervised Anomaly Detection Algorithms on High-Integrity Digital Systems"

phase by helping to choose the appropriate algorithm for a given set of requirements.

ANOMALY DETECTION BASED QUENCH DETECTION SYSTEM

Quenches are disruptive faults in Superconducting Radio Frequency (SRF) cavities. Loss of superconductivity in small areas of SRF cavities causes a sudden increase in temperature due to the high RF power used for accelerating. The additional heat dissipation causes the normal conducting area to grow rapidly until the RF field breaks down in the cavity. Associated large heat dissipation can trigger issues in related subsystems, e.g., the helium used for maintaining the cryogenic temperature can reach the boiling temperature and increase the pressure in the pipes of the cryogenic plant. Given the large disruptions that a quench can cause, traditional quench detection systems take a conservative approach. However, traditional quench detection systems can reach a false positive rate of up to 47% due to noisy signals and unexpected behaviors. A robust method for quench detection is an active research area [EBT23; Naw+18].

[EBT23] Eichler et al., "Anomaly detection at the European X-ray Free Electron Laser using a parity-space-based method"

[Naw+18] Nawaz et al., "Anomaly Detection for the European XFEL using a Nonlinear Parity Space Method"

In [Mar+22], we address the quench detection problem by using anomaly detection. Specifically, we evaluate two algorithms for detecting time-series anomalies. Additionally, we describe the work done at the SLAC National Accelerator Laboratory toward the creation of a fault display to be used for the visualization of past faults during the operation of the Linac Coherent Light Source (LCLS-II).

[Mar+22] Martino et al., "Anomaly Detection Based Quench Detection System for CW Operation of SRF Cavities"

This work addresses Problem 2 (Time Series Outlier Anomaly Detection). The contributions are: 1. the creation of a fault analysis interface for the Linac Coherent Light Source II (LCLS-II); 2. the usage of a novel technique based on time-series anomaly detection for the detection of anomalies at the machine level.

CNN IMPLEMENTATION AND ANALYSIS ON XILINX VERSAL ACAP AT EUROPEAN XFEL

Current general-purpose hardware cannot perform real-time inference for larger Convolutional Neural Networks (CNNs), e.g., the networks used for time series anomaly detection. Achieving real-time performances requires not only enough computational power but also a short latency in the computation. Although the GPU is well-suited for applications that demand high throughput, its low energy efficiency and prolonged higher latency prevent it from meeting the diverse needs of CNN applications [Lv+20; Mel+18].

[Lv+20] Lv et al., "Research on Dynamic Reconfiguration Technology of Neural Network Accelerator Based on Zynq"

[Mel+18] Meloni et al., "NEURAghe: Exploiting CPU-FPGA Synergies for Efficient and Flexible CNN Inference Acceleration on Zynq SoCs"

In [Al-+22], we implement and evaluate the performance of a customized CNN architecture at the European XFEL on the Xilinx Versal AI series. We compare the accuracy, throughput, and latency of our implementation to state-of-the-art GPUs, CPUs, and MPSoC FPGAs. We also analyze the results and draw conclusions about the performance of the Data Processing Unit (DPU) on the Xilinx Versal Adaptive Compute Acceleration Platform (ACAP) compared to the Xilinx Zynq UltraScale+ counterpart.

[Al-+22] Al-Zoubi et al., "CNN Implementation and Analysis on Xilinx Versal ACAP at European XFEL"

1.2.2 *Runtime Monitoring of Firmware Components*

This part consists of contributions to the automatic extraction of properties from an existing design and a subsequent generation of runtime monitors for the same designs.

▸ SYNTAX-GUIDED ENUMERATION OF TEMPORAL PROPERTIES

[Alu+15] Alur et al. "Syntax-Guided Synthesis"

[Fey+18] Fey et al., "Design Understanding: From Logic to Specification"

[MRF20] Martino et al., "Revisiting Explicit Enumeration for Exact Synthesis"

[MRF18] Martino et al., "Coverage-Guided CTL Property Enumeration for Understanding Models of Reactive Systems"

Alur et al. [Alu+15] proposed syntax-guided synthesis, a successful method for the synthesis of circuits starting from a correctness specification and a grammar that specifies a syntactic set of candidate implementations. In [Fey+18], we summarize the state-of-the-art research in design understanding with a focus on deriving human-understandable logic properties from an existing implementation. In [MRF20], we extend the idea of syntax-guided synthesis to temporal logic. We propose syntax-guided property enumeration, a technique for automatically obtaining a set of short and readable temporal logic properties from an existing design. The seminal idea for this paper was presented in [MRF18].

The set of properties found is complete up to a given length of the property, i.e., all temporal properties the block fulfills are found. Syntactic filtering removes trivial properties. The enumeration algorithm intelligently avoids obvious equivalences, e.g., due to the commutativity of operators. Structural hashing and simplification eliminate further simple equivalences. Trace checking uses random simulation to efficiently falsify properties that do not hold on the block.

This technique tackles Problem 4 (Automatic Specification Generation). Our approach is unique in being able to use the full grammar of the chosen language without any restrictions. No specific simulation traces are required to find meaningful properties, and only random simulation is used. By relying on enumeration, we generate human-readable specifications. Our approach tightly combines concepts from syntax-guided synthesis with model checking and powerful syntactic and semantic simplification procedures.

▸ RUNTIME MONITORING OF C-LTL SPECIFICATIONS ON FPGAS USING HLS

[LS09] Leucker and Schallhart, "A brief account of runtime verification"

[MF22] Martino and Fey, "Runtime Monitoring of c-LTL Specifications on FPGAs Using HLS"

[Bar+18] Bartocci et al., "A Counting Semantics for Monitoring LTL Specifications over Finite Traces"

As already introduced, runtime monitors are hardware property checkers that ensure that the system satisfies a correctness specification during operation [LS09]. Runtime monitoring is useful for digital systems because it allows for the detection and diagnosis of runtime errors and anomalies. It can also be used to gather performance metrics and log system activity for later analysis.

In [MF22], we address Problem 5 (Runtime Monitor Generation) with a system for the generation of runtime monitors implementing c-LTL [Bar+18] properties. Our contribution is the first framework that automatically generates synthesizable C++ monitors based on c-LTL semantics from an LTL property. The monitors generated using high-level synthesis (HLS) are technology-independent and take advantage of the features offered by different architectures. HLS introduces fine-grain optimizations that reduce area consumption and increase performance. Our framework reduces the generated code by using structural hashing and applying Boolean simplifications. The high-level resource sharing also leads to highly optimized C++ code;

for example, if multiple monitors share the same sub-formula, sub-monitors and evaluation sections are shared. Finally, the framework supports the use of HLS directives to guide the HLS synthesis process and achieve further optimizations.

▸ REVISITING EXPLICIT ENUMERATION FOR EXACT SYNTHESIS

The task of exact synthesis is to determine the most efficient logic design for a specific Boolean function based on certain cost considerations. Multiple possible approaches for solving the exact synthesis problem exist [Kar+61; RK62; FHS17; Ern09]. Exact synthesis is currently achieved through the use of SAT-based implicit enumeration algorithms, which are considered the most advanced methods. However, these algorithms have inherent difficulties, such as unpredictable runtimes that vary depending on the query and a difficult parallelization.

In [MRF20], the main contribution is an explicit enumeration algorithm for the generation of minimal-size circuits that can be effectively parallelized. Our algorithm allows for the search for multiple solutions using a single run, resulting in a runtime equal to that of finding only the last circuit in the search order. This makes our approach suitable for the creation of libraries of minimal-size circuits. To evaluate our approach, we implemented the framework in a prototype header-only library and compared it to the one available in the verification and synthesis software ABC [BM10] for generating minimal circuits using And-Inverter Graphs (AIGs) implementing all possible 3-input functions.

Before we proceed to the technical parts, we now give more details about the structure of the existing LLRF system at the European XFEL. Moreover, we recall the basic concepts for the formal description of systems' specifications.

[Kar+61] Karp et al., "A computer program for the synthesis of combinational switching circuits"

[RK62] Roth and Karp, "Minimization Over Boolean Graphs"

[FHS17] Fiser et al., "SAT -Based Generation of Optimum Function Implementations with XOR Gates"

[Ern09] Ernst, "Optimal combinational multi-level logic synthesis"

[BM10] Brayton and Mishchenko, "ABC: An Academic Industrial-Strength Verification Tool"

2

Preliminaries

"Art, like morality, consists of drawing the line somewhere."
—Gilbert Keith Chesterton

FAULT DETECTION using the scheme introduced in Chapter 1 relies on a wide range of research areas: from machine learning based techniques to traditional fault-tolerance and system safety, as well as more formal approaches such as runtime verification. In the next sections, we define the basic terms used, give an overview of the LLRF system of the European XFEL and its components, and give a brief description of temporal logic, defining the specification logic used throughout Part III. Most of the following chapters include sections that introduce chapter-specific concepts not introduced in this chapter.

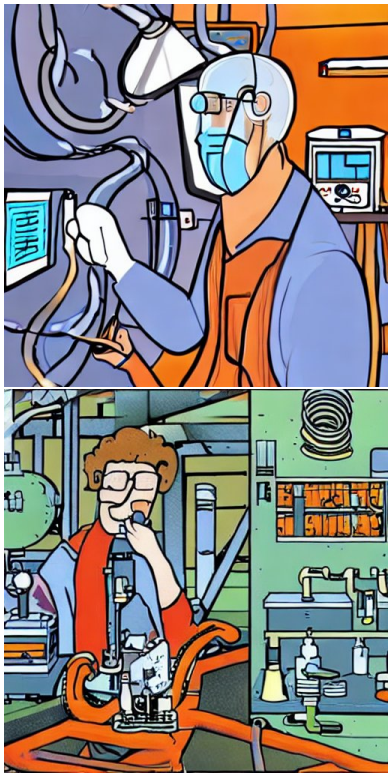
2.1 TERMINOLOGY

[LA90] Lee and Anderson, "System Structure and Dependability"

The terms `failure`, `fault`, and `error` are often used interchangeably in relation to digital systems, but they assume specific meanings [LA90].

The `failure` of a system occurs when the system does not perform its intended function. The definition of its intended function is described by the system's specification. The `failure` is always caused by either an erroneous transition of the system or a sequence of valid transitions that lead to an erroneous state. An `error` is the difference from a valid state that causes the erroneous state or transition. The difference between a `fault` and `error` lies in the system's structure: a `fault` in the system is an `error` in a component or a component's design. However, the distinction between `error` and `failure` does not merely reflect system structure. Rather, the difference is between a condition (or state) and an event.

To summarize, a `fault` is the cause of an `error`, and an `error` is the cause of a `failure`. A component `fault` can result in a component `failure`. The component `failures` can trigger an erroneous transition in the system, i.e., the manifestation of a `fault` in the system. The `fault` in the system will produce `errors` in the internal state of the system, leading to a system `failure`.



Artistic representations of a scientist operating a particle accelerator generated using a stable diffusion model [Rom+21].

2.2 THE LLRF SYSTEM OF THE EUROPEAN XFEL

In this section, we describe the European X-ray Free-Electron Laser (XFEL) functionalities and architecture.

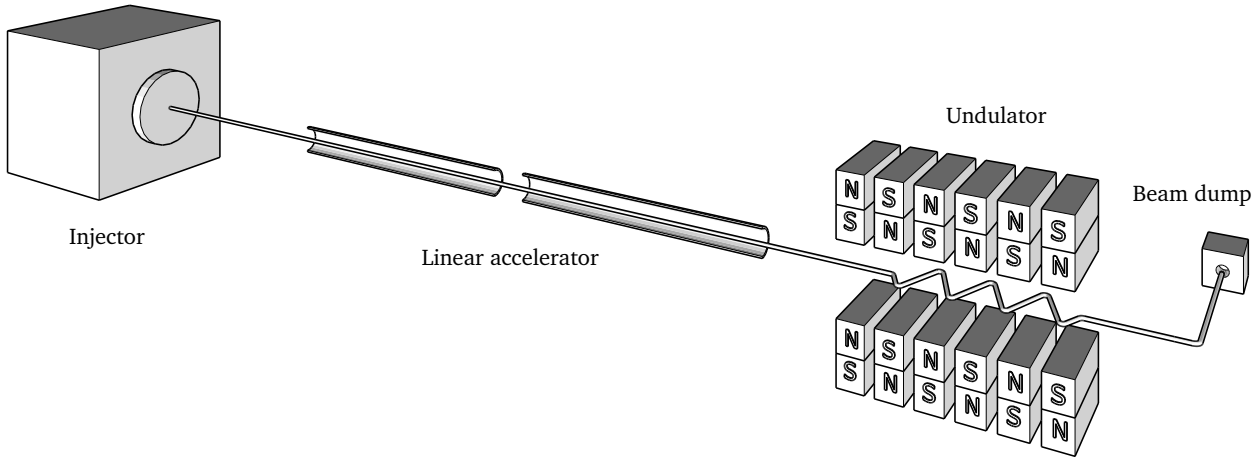


FIGURE 2.1: Representation of the setup of the electron system of a free-electron laser.

The European XFEL is a facility used for the production of ultrashort X-ray laser flashes. Since its commissioning in 2017, the European XFEL has been used to perform studies ranging from atomic physics and materials science to biology. The entire machine is 3.4 km long and mostly placed underground. The characteristics that make this machine unique are:

- the maximum energy levels that electrons can reach: electrons can be accelerated at an energy of up to 17.5 GeV;
- the high repetition rate: the European XFEL can produce 27000 flashes per second.

The European XFEL produces X-ray laser flashes through the process described in Figure 2.1. In the injector, a laser impinging a photocathode stimulates the emission of electrons. The electrons emitted, exposed to the accelerating field, are accelerated up to 6 MeV. After the injector, the superconducting LINear ACcelerator (LINAC) accelerates the electrons to the desired energy (up to 17.5 GeV). Finally, the electrons pass through the undulator, a set of magnets that force a sinusoidal path to their trajectory. This starts the photon emission through the Self-Amplified Spontaneous Emission (SASE) process [Kim86]. The photons emitted through this process are largely monochromatic and transverse coherent. After this point, the electrons are sent to the beam dump, and the photons are delivered to the experimental hall where the experiments are placed.

[Kim86] Kim, "An analysis of self-amplified spontaneous emission"

The accelerator part of the European XFEL is 2.2 km long, and it is composed of 24 Radio Frequency (RF) stations that control 784 superconducting cavities. The superconducting cavities need to be kept at cryogenic temperatures. For this reason, they are housed in 98 cryomodules. Each cryomodule houses 8 cavities.

The accelerator is controlled by the LLRF system of the European XFEL. The LLRF system is responsible for setting and maintaining stable voltage and phase for the individual RF stations. The required RMS field stability should be better than 0.01 % in amplitude and 0.01° in phase at 1.3 GHz, i.e., the cavity operating frequency. The LLRF system is distributed among the RF stations, and it is placed directly below the cryomodules, as shown

FIGURE 2.2: Picture of the physical setup of the cabinets under the cryomodules inside the European XFEL accelerator tunnel. The left cabinet contains the LLRF system.

in Figure 2.2. Concrete blocks are mounted above the racks to protect the electronics from radiation.

ARCHITECTURE AND COMPONENTS OF THE LLRF SYSTEM

The LLRF system is a distributed system that combines multiple different components. Each RF station controls a single klystron, i.e., an RF power amplifier based on vacuum tube technology, that injects power via waveguides into 32 cavities, distributed over 4 cryomodules. A single RF station is split into two separate crates, each collecting signals from 2 cryomodules (16 cavities). One of the two crates, the manager, is directly controlling the klystron, while the other one, the subordinate crate, reports the partial results to the manager crate. The setup is motivated by the need to minimize the cable lengths to the cavities and the processing elements that need to be housed within on MicroTCA crate. The sum of all cavity signals is used to compute a vector sum for controlling the RF input to the high-power klystron [Sch98]. A scheme of this setup is represented in Figure 2.3. Additionally, the figure shows that the system is composed of additional components:

[Sch98] Schilcher, "Vector Sum Control of Pulsed Accelerating Fields in Lorentz Forces Detuned Superconducting Cavities"

- a Drift Compensation Module (DCM), for compensating phase and amplitude drifts and providing an attenuated signal to the down-conversion signal processing chain;
- an optical REFerence Module (REFM), for providing a reference signal optically synchronized to the central laser oscillator (part of the synchronization system);
- a Local Oscillator Generation Module (LOGM), for the generation of the Local Oscillator (LO) signal (1354 MHz) used by the down-conversion signal stream and the external clock (CLK) signal (81.25 MHz) used by the digitizers.
- a Power Supply Module (PSM), for delivering power to the DCM, REFM, and LOGM;
- a PieZo driver Module (PZ16M), for driving the piezo tuners installed

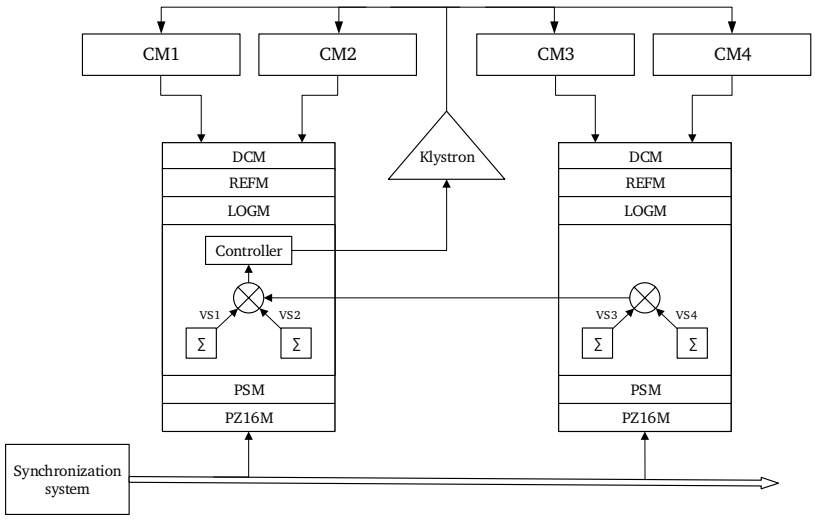


FIGURE 2.3: Scheme of the physical setup of the LLRF system inside the European XFEL accelerator tunnel.

in the cavities to compensate for external disturbances that can be counteracted by physically deforming the cavity [Pag+05].

[Pag+05] Pagani et al., *The fast piezo-blade tuner for SCRF resonators*

μTCA.4 ARCHITECTURE

The μTCA.4 standard describes a modular architecture specifically targeting large-scale scientific devices. The architecture is derived from the Advanced Telecom Computing Architecture (AdvancedTCA or ATCA), a standard architecture mainly used for telecommunication purposes. The main characteristic of systems based on this class of standards is given by the composability of different devices built into pluggable boards.

The μTCA.4 crate houses a backplane and the boards composing the particular system. The backplane provides a set of programmable communication lines for point-to-point communication and power distribution. The boards are differentiated between Advanced Mezzanine Card (AMC) and Rear Transition Module (RTM) depending on whether the board can be plugged into the front or rear side, respectively. The AMCs are directly connected to the backplane's communication lines, while the RTMs are connected to the respective AMC cards.

A wide variety of boards can be used as AMCs. The list includes but is not limited to:

- Full computing systems (referred to as CPUs);
- Storage carriers including multiple storage drives;
- Digitizer boards equipped with high-precision Analog-to-Digital Converters (ADCs) and Digital-to-Analog Converters (DACs);
- Reconfigurable boards, called FPGA Mezzanine Cards (FMCs), that are equipped with function-specific electronics and can integrate various combinations of connectors;
- Clocking and timing boards, for systems synchronization;
- Signal processing boards.

Likewise, also RTMs can be of multiple types. However, since there exist incompatible combinations of RTMs and AMCs, a physical pin mechanically prevents incorrect pairings. An incomplete list of possible RTM types includes:

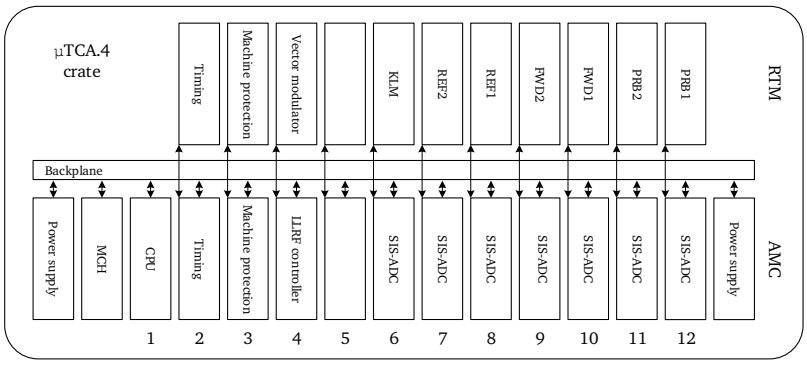


FIGURE 2.4: Scheme of a manager μTCA.4 crate in the European XFEL.

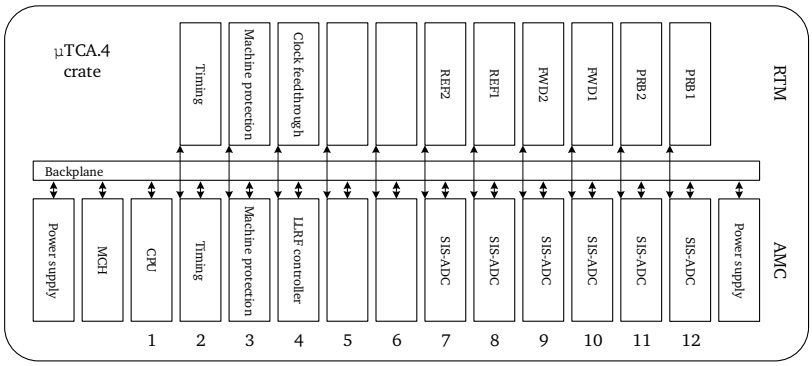


FIGURE 2.5: Scheme of a subordinate μTCA.4 crate in the European XFEL.

- Signal pre- and post-processing (both analog and digital);
- Clock distribution devices;
- Storage systems.

► THE LLRF SYSTEM OF THE EUROPEAN XFEL uses the configuration represented in Figure 2.4 for the manager crate and the configuration in Figure 2.5 for the subordinate crate. The slots containing the power supplies and the μTCA.4 Carrier Hub (MCH) are predefined in the μTCA.4 crates. The power supplies deliver power to all the systems installed in the crates. Since μTCA.4 supports redundant power supplies, two power supplies are installed. The MCH manages the boards connected, and it is responsible for the clock distribution and data line switching. The sampling of signals is synchronized by the timing boards (in slot 2), connected to the central synchronization system. The Machine Protection System (MPS) is installed in slot 3. The board in slot 4 in the subordinate μTCA.4 crate calculates the partial vector sum and reports it to the board in slot 4 in the manager μTCA.4 crate. The controller box represented in the left rack Figure 2.3 is implemented in the manager μTCA.4 crate in the boards in slot 4. The AMC board calculates the total vector sum and performs the regulation using a 2^{nd} order Multiple Inputs Multiple Outputs (MIMO) controller scheme [Sch+08] in its FPGA. The boards from slot 7 to slot 12 are responsible for the attenuation, down-conversion, sampling, analog-to-digital conversion, and partial vector sum of the signals coming from the cavities. Each slot is assigned to a specific signal of a cryomodule:

- boards in slots 7 and 8 collect the reflected signals of the cavities in

[Sch+08] Schmidt et al., "Parameter estimation and tuning of a multivariable RF controller with FPGA technique for the Free Electron Laser FLASH"

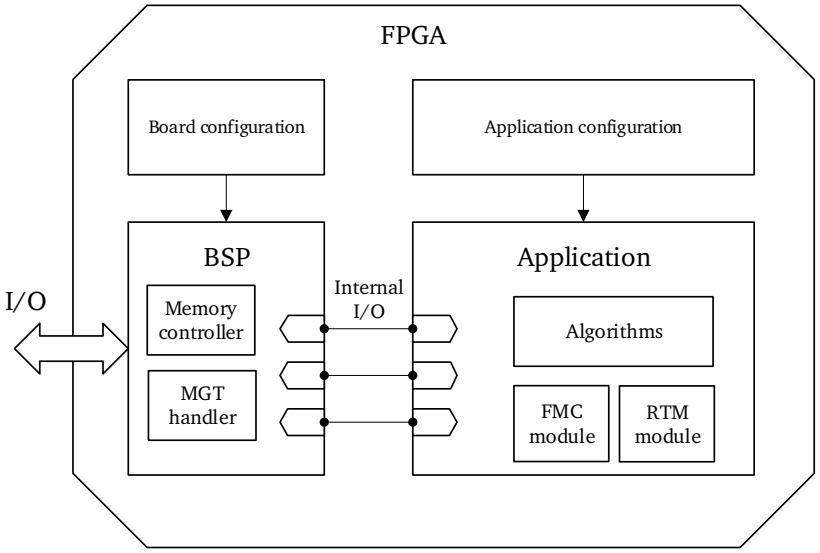


FIGURE 2.6: Representation of the typical FPGA firmware structure in FWK.

the two cryomodules;

- boards in slots 9 and 10 collect the forward signals of the cavities in the two cryomodules;
- boards in slots 11 and 12 collect the probe signals of the cavities in the two cryomodules.

Additionally, the FPGA included in the AMC board performs supplementary tasks, e.g., filtering, drift compensation, and IQ detection and rotation. In the manager μTCA.4 crate, we have an additional board in slot 6. This board implements the klystron lifetime management system (KLM) and is responsible for the protection of the klystron from the high-amplitude signals that could be reflected in certain situations. Finally, the CPU, installed in slot 3, runs the local servers for programming and managing the SIS-ADC boards. Additionally, the CPU runs servers that control external components, e.g., DCM, MPS, or PZ16M, perform data acquisition, and detect cavity problems.

All boards containing an FPGA are equipped with different firmware depending on the intended functionality.

[But+15] Butkowski et al., "FPGA firmware framework for MTCA.4 AMC modules"

► THE FIRMWARE used in the FPGAs of the LLRF system at the European XFEL is based on FWK [But+15], the FPGA firmware framework developed at DESY. FWK simplifies the development of new components by offering a set of reusable standard components and a unified building methodology for different platforms and vendors. Moreover, FWK gives a standard way to manage all communication channels. The same firmware framework is used in all projects supported by the Accelerator Beam Control (Maschine Strahlkontrollen or MSK, in short) group.

The structure of the firmware is represented in Figure 2.6. The Board Support Package (BSP) manages the physical interfaces and offers standard internal I/O interfaces to the applications. Additionally, the BSP manages access to the external memory through the memory controller. The BSP is configured with the specific board parameters. The components that are part of the application side use the internal I/O interfaces to communicate externally. The application side is composed of the algorithms that perform

the various operations required, e.g., vector sum, and the modules for controlling the RTM and, if present, the FMC. The applications are configured with machine-specific settings. Both sides are connected by internal communication lines based on either IBUS, a protocol developed internally, or AXI [Axi].

[Axi], *AMBA AXI and ACE – Protocol Specification*

After introducing the infrastructure for which the fault detection is designed, we introduce the theoretical concepts used throughout Part III.

2.3 TEMPORAL LOGIC

In the realm of computer science and system engineering, temporal logic stands as a cornerstone, offering a powerful and expressive framework for representing and reasoning about time-dependent behaviors of systems. Over the past 70 years, multiple formal representations of time have been introduced. In computer science, the introduction of both a linear and a branching representation of time has had a great influence on the development of model checking of reactive systems. Later, the same formal representation of linear and branching time has been used for runtime monitoring.

In Part III, we will make extensive use of various concepts of temporal logic. In this section, we introduce the main notions. We use Linear Temporal Logic (LTL) to represent linear time and Computation Tree Logic (CTL) to represent branching time, both modeled on Kripke structures as infinite-time specification languages.

Generated using a stable diffusion model [Rom+21].

▸ KRIPKE STRUCTURE

Kripke structures are used in model checking to represent a system's behavior. We use a Kripke structure $M = (S, s_0, T, A, L)$ with states S, initial state $s_0 \in S$, left-total transition relation $T \subseteq S \times S$, atomic propositions A, and labeling function $L : S \to 2^A$. We denote a path in M starting in state s_0 by $\pi = s_0, s_1, \ldots$, for every $i < |S|$, then $(s_i, s_{i+1}) \in T$. We use π_i to denote the suffix of π starting at s_i.

▸ LINEAR TEMPORAL LOGIC

Linear Temporal Logic (LTL) was introduced by Pnueli [Pnu77] and has become one of the most popular and widely used temporal logic in computer science. The syntax of LTL is defined as follows: $\top$, $\bot$ and every p for $p \in A$ are LTL formulæ; if φ_1 and φ_2 are LTL formulæ, then so are $\neg\varphi_1$, $\varphi_1 \wedge \varphi_2$, $\mathbf{X}\varphi_1$, $\mathbf{F}\varphi_1$, $\mathbf{G}\varphi_1$, $\varphi_1 \mathbf{U} \varphi_2$. All LTL formulæ are path formulæ.

[Pnu77] Pnueli "The Temporal Logic of Programs"

We write $\pi \models \varphi$, as a simplified notation for $M, \pi \models \varphi$, to denote that a path π of the Kripke structure M satisfies an LTL formula φ. The semantics of LTL are inductively defined as follows:

- $\pi \models \top$,
- $\pi \not\models \bot$,
- $\pi \models p$ iff $p \in L(s_0)$,
- $\pi \models \neg\varphi$ iff $\pi \not\models \varphi$,

- $\pi \models \varphi_1 \wedge \varphi_2$ iff $\pi \models \varphi_1$ and $\pi \models \varphi_2$,
- $\pi \models \mathbf{X}\varphi$ iff $\pi_1 \models \varphi$,
- $\pi \models \mathbf{F}\varphi$ iff $\exists j \geq 0.\pi_j \models \varphi$,
- $\pi \models \mathbf{G}\varphi$ iff $\forall j \geq 0.\pi_j \models \varphi$,
- $\pi \models \varphi_1 \mathbf{U} \varphi_2$ iff $\exists j \geq 0.\pi_j \models \varphi_2$ and $\forall k, 0 \leq k \leq j.\pi_k \models \varphi_1$.

The temporal operator **X**, if applied to the property φ, specifies that φ holds in the next state of the path, starting from the initial state. The property $\mathbf{F}\varphi$ specifies that φ eventually holds in the path. The property $\mathbf{G}\varphi$ specifies that φ holds for the entirety of the path. The property $\varphi_1 \mathbf{U} \varphi_2$ specifies that φ_1 holds for all states of the path until φ_2 holds.

▸ Computation Tree Logic

Branching time logics represent computation trees, i.e., all possible computations of a system obtained from unfolding the Kripke structure. Computational Tree Logic (CTL) was introduced by Clarke and Emerson [CE81].

[CE81] Clarke and Emerson "Design and Synthesis of Synchronization Skeletons Using Branching-Time Temporal Logic"

CTL formulæ are composed according to the following rules: $\top$, $\bot$ and every p for $p \in A$ are CTL formulæ; if φ_1 and φ_2 are CTL formulæ, then so are $\neg\varphi_1$, $\varphi_1 \wedge \varphi_2$, $\mathbf{AX}\varphi_1$, $\mathbf{AF}\varphi_1$, $\mathbf{AG}\varphi_1$, $\mathbf{A}\varphi_1\mathbf{U}\varphi_2$, $\mathbf{EX}\varphi_1$, $\mathbf{EF}\varphi_1$, $\mathbf{EG}\varphi_1$, $\mathbf{E}\varphi_1\mathbf{U}\varphi_2$. CTL formulæ reason about states of a Kripke structure M.

We write $\pi \models \varphi$, as a simplified notation for $M, \pi \models \varphi$, to denote that a path π of the Kripke structure M satisfies a CTL formula φ. A CTL path formula Φ is inductively defined as follows:

- $\pi \models \mathbf{X}\varphi$ iff $\pi_1 \models \varphi$,
- $\pi \models \mathbf{F}\varphi$ iff $\exists j \geq 0.\pi_j \models \varphi$,
- $\pi \models \mathbf{G}\varphi$ iff $\forall j \geq 0.\pi_j \models \varphi$,
- $\pi \models \varphi_1 \mathbf{U} \varphi_2$ iff $\exists j \geq 0.\pi_j \models \varphi_2$ and $\forall k, 0 \leq k \leq j.\pi_k \models \varphi_1$.

We write $s \models \varphi$, as a simplified notation for $M, s \models \varphi$, to denote that a state $s \in S$ of the Kripke structure M satisfies a CTL formula φ. We write $Paths(s)$ to designate the set of all possible paths of the Kripke structure M starting in s. We write $\pi \models \Phi$, as a simplified notation for $M, \pi \models \Phi$, to denote that a path π of the Kripke structure M satisfies a CTL path formula Φ. A CTL formula φ is inductively defined as follows:

- $s \models \top$,
- $s \not\models \bot$,
- $s \models p$ iff $p \in L(s)$,
- $s \models \neg\varphi$ iff $s \not\models \varphi$,
- $s \models \varphi_1 \wedge \varphi_2$ iff $s \models \varphi_1$ and $s \models \varphi_2$,

- $s \models \mathbf{E}\Phi$ iff $\exists \pi \in Paths(s).\pi \models \Phi$,
- $s \models \mathbf{A}\Phi$ iff $\forall \pi \in Paths(s).\pi \models \Phi$.

The path quantifier **E** specifies that such properties hold for at least one possible computation path. The path quantifier **A** specifies that such properties hold for all possible computation paths. That is, $\mathbf{EX}\varphi$ denotes that there is an execution trace starting in s that satisfies φ in the next step, $\mathbf{EF}\varphi$ denotes that there is an execution trace starting in s that eventually satisfies φ, $\mathbf{EG}\varphi$ denotes that there is an execution trace starting in s that always satisfies φ, and $\mathbf{E}\varphi_1\mathbf{U}\varphi_2$ that there is an execution trace starting in s that satisfies φ_1 until φ_2 holds (and eventually φ_2 has to hold). The property $\mathbf{AX}\varphi$ denotes that all execution traces starting in s satisfy φ in the next step, $\mathbf{AF}\varphi$ denotes that all execution traces starting in s eventually satisfy φ, $\mathbf{AG}\varphi$ denotes that all execution traces starting in s always satisfy φ, and $\mathbf{A}\varphi_1\mathbf{U}\varphi_2$ that all execution traces starting in s satisfy φ_1 until φ_2 holds (and eventually φ_2 has to hold). A formula is in negation normal form if the negation operator occurs only directly in front of atomic propositions.

Additionally, we define the usual Boolean and temporal logic abbreviations [CGP01]:

[CGP01] Clarke et al., *Model checking, 1st Edition*

- $\varphi_1 \vee \varphi_2 = \neg(\neg\varphi_1 \wedge \neg\varphi_2)$,
- $\varphi_1 \rightarrow \varphi_2 = \neg\varphi_1 \vee \varphi_2$,
- $\mathbf{AX}\varphi = \neg\mathbf{EX}\neg\varphi$,
- $\mathbf{EF}\varphi = \mathbf{E}(\top\mathbf{U}\varphi)$,
- $\mathbf{AG}\varphi = \neg\mathbf{EF}\neg\varphi$,
- $\mathbf{AF}\varphi = \neg\mathbf{EG}\neg\varphi$, and
- $\mathbf{A}(\varphi_1\mathbf{U}\varphi_2) = \neg\mathbf{E}(\neg\varphi_2\mathbf{U}(\varphi_1 \wedge \varphi_2)) \wedge \neg\mathbf{EG}(\varphi_2)$.

In the chapters of the next parts, the results of our research on fault detection are presented. Where necessary, additional chapter-specific concepts are introduced in the relative *Background* section.

Part II

MONITORING OF DISTRIBUTED DIGITAL SYSTEMS USING ANOMALY DETECTION

3

Comparison of Anomaly Detection Algorithms for Fault Detection

"Truth suffers from too much analysis."
—Frank Herbert

► SYNOPSIS In this chapter, we give a cohesive description and comparison of anomaly detection algorithms in the context of distributed high-integrity digital systems. This chapter is based on the papers

> Gianluca Martino, Arne Grünhagen, Julien Branlard, Annika Eichler, Görschwin Fey, and Holger Schlarb. "Comparative Evaluation of Semi-Supervised Anomaly Detection Algorithms on High-Integrity Digital Systems". In: *24th Euromicro Conference on Digital System Design, DSD 2021, Palermo, Italy, September 1-3, 2021*. Ed. by Francesco Leporati, Salvatore Vitabile, and Amund Skavhaug. IEEE, 2021, pp. 123–130. DOI: 10.1109/DSD53832.2021.00028.

and

> Arne Grünhagen, Julien Branlard, Annika Eichler, Gianluca Martino, Görschwin Fey, and Marina Tropmann-Frick. "Fault Analysis of the Beam Acceleration Control System at the European XFEL using Data Mining". In: *30th IEEE Asian Test Symposium, ATS 2021, Matsuyama, Ehime, Japan, November 22-25, 2021*. IEEE, 2021, pp. 61–66. DOI: 10.1109/ATS52891.2021.00023.

In Part II, we focus on developing techniques for anomaly detection on both historical data and real-time data. It is important to test different anomaly detection algorithms on real data to verify the accuracy of the algorithms for fault identification. Therefore, this chapter begins with a review of some existing anomaly detection algorithms, followed by a series of experiments to compare their performances.

This chapter aims at providing a cohesive overview of the selected algorithms for semi-supervised anomaly detection, as we are unaware of any previous comprehensive descriptions of these algorithms. This organized presentation of the algorithms may help others understand their strengths and limitations for other applications. In addition, this evaluation gives us a better understanding of the algorithms and their performances on our specific system for building a fast and reliable fault detection system.

3.1 INTRODUCTION

Complex systems such as the European XFEL often experience unexpected malfunctions due to the complex interactions between the various components. Detecting these faults can be time-consuming, expensive, and critical for safety. The European XFEL relies on a number of complex subsystems, including the Low-Level Radio Frequency (LLRF) system, which is responsible for measuring and controlling the electromagnetic fields used to accelerate electron bunches [Bra+12]. Every point in time when the accelerator cannot be operated is referred to as a machine trip. Faults need to be detected as early as possible to achieve high availability of the LLRF system. Traditionally, monitored values describing the status of the LLRF system are investigated manually after a machine trip occurs. This very time-consuming approach is aimed at reacting appropriately in time, preventing machine trips. The purpose of this work is to give insights into how faulty behaviors of an embedded system like the LLRF system can be detected automatically by analyzing historical sensor data provided by that system.

[Bra+12] Branlard et al., "The European XFEL LLRF system"

Anomaly detection deals with the problem of identifying unexpected values in data sets. Anomaly detection is commonly used in very different fields such as intrusion detection [Por00; YD03; GT+09], fraud detection [BH02; Phu+10], medical applications [Lin+05], fault detection [DF96; DN00; Kin+02], and more [BGS08; Geb+13; HA04; CBK09]. Since very different fields use anomaly detection, a variety of different techniques have been developed over the course of the years. However, not all the techniques can be used for all possible data sets and application fields. This is because the nature and characteristics of the data vary significantly depending on the application field. As highlighted in [Emm+16], the algorithm's ability to detect anomalies depends on the characteristics of the data analyzed, which depend on the problem to be studied.

[Por00] Portnoy, "Intrusion detection with unlabeled data using clustering"

[YD03] Yeung and Ding, "Host-based intrusion detection using dynamic and static behavioral models"

[GT+09] Garcia-Teodoro et al., "Anomaly-based network intrusion detection: Techniques, systems and challenges"

[BH02] Bolton and Hand, "Statistical Fraud Detection: A Review"

[Phu+10] Phua et al., "A Comprehensive Survey of Data Mining-based Fraud Detection Research"

[Lin+05] Lin et al., "Approximations to magic: finding unusual medical time series"

[DF96] Dasgupta and Forrest, "Novelty detection in time series data using ideas from immunology"

[DN00] Dasgupta and Nino, "A comparison of negative and positive selection algorithms in novel pattern detection"

[Kin+02] King et al., "The Use of Novelty Detection Techniques for Monitoring High-Integrity Plant"

[BGS08] Basharat et al., "Learning object motion patterns for anomaly detection and improved object detection"

[Geb+13] Gebhardt et al., "Document Authentication Using Printing Technique Features and Unsupervised Anomaly Detection"

[HA04] Hodge and Austin, "A Survey of Outlier Detection Methodologies"

[CBK09] Chandola et al., "Anomaly detection: A survey"

[Emm+16] Emmott et al., "A Meta-Analysis of the Anomaly Detection Problem"

In high-integrity systems, traditional classification methods are not suited due to the low number of abnormalities occurring in such systems. The low number of anomalies, in addition to the difficulty of retrieving data describing all possible fault modes of such systems, makes anomaly detection the technique of choice for building self-adaptive high-integrity systems.

In this chapter, we focus first on analyzing the performance of several anomaly detection techniques for time point outliers. We use real sensor data from the European XFEL for this analysis. In addition, the experiments are conducted using publicly available data describing other high-integrity systems. The chapter is structured as follows: in Section 3.2, we give some details about related work; in Section 3.3, we describe the main concepts used in the following sections; in Section 3.4, we illustrate all models used for the experiments; in Section 3.5, we report the experimental results for the unsupervised anomaly detection models; in Section 3.6, we report the experimental results for the semi-supervised anomaly detection models; finally, in Section 3.7, we put the results in perspective in the overall context of the fault detection system.

3.2 RELATED WORK

Despite extensive literature about anomaly detection, existing literature gives no clear indication of the performance of existing algorithms when the application is monitoring high-integrity distributed digital systems.

In [GU16], the authors thoroughly compare and evaluate a wide range of algorithms using a set of publicly available data sets. The evaluation method used makes it easy to make new comparisons when new algorithms or new implementations are proposed. However, they consider only unsupervised anomaly detection methods. In [Din+14], the authors perform an experimental evaluation of semi-supervised anomaly detection methods to deduct general guidelines on semi-supervised anomaly detection. In the experiments, they use a wide variety of data sets consisting of publicly available data. However, they perform the evaluation using only a few models representing whole classes of algorithms: a Gaussian Mixture model, an example of the density estimation method; two k-nearest neighbor models; and a Support Vector Data Description (SVDD) model, an example of the Support Vector Machine (SVM) method. Also, the data sets chosen are not related to hardware systems. In [Dom+20], the authors conduct an experimental evaluation of novelty detection methods for discrete sequences. They consider a different set of algorithms with respect to the previous works. However, the main focus of their analysis is on stable discrete sequences. The use cases considered are protein identification for genomics, fraud and intrusion detection, and user behavior analysis. In [Gup+14; Pim+14; MS03a; MS03b], the authors surveyed state-of-the-art methods for outlier detection for temporal data and novelty detection. However, they do not evaluate the algorithms. Other comparisons can be found in other papers presenting specific algorithms, but the results give only partial indications.

Regarding the application of anomaly detection in the context of a particle accelerator, [Fol+20] concentrates on applying different anomaly detection techniques to detect faulty behavior of the beam position monitors at the Large Hadron Collider (LHC) at CERN. Most of the faulty behavior is detected by traditional approaches, such as comparing the current data with predefined thresholds or applying advanced signal improvement techniques that are based on singular value decomposition [CT04].

There are different groups working at the German Electron Synchrotron (DESY) with the aim of detecting and preventing faulty behavior in the early stages. An approach also applied to the LLRF system is identifying the root causes of machine trips at European XFEL by analyzing the RF signals of the different RF stations. The mechanism consists of a collection of past machine trips that are categorized by human LLRF system experts. Whenever a new machine trip occurs, the mechanism compares the faulty data to the data of past machine trips. If the new machine trip cannot be assigned to any of the categories, human experts usually identify the root causes and add the type of machine trip to the collection. So the actual root cause analysis is still done by humans. A second approach that is applied to the optical synchronization system (LbSync) of the European XFEL consists of training a K-means clustering model. The model is trained with only healthy data, so the model represents a healthy system. This approach is able to decide

[GU16] Goldstein and Uchida, "A comparative evaluation of unsupervised anomaly detection algorithms for multivariate data"

[Din+14] Ding et al., "An experimental evaluation of novelty detection methods"

[Dom+20] Domingues et al., "A Comparative Evaluation of Novelty Detection Algorithms for Discrete Sequences"

[Gup+14] Gupta et al., "Outlier Detection for Temporal Data: A Survey"

[Pim+14] Pimentel et al., "A review of novelty detection"

[MS03a] Markou and Singh, "Novelty detection: a review–part 1: statistical approaches"

[MS03b] Markou and Singh, "Novelty detection: a review–part 2:: neural network based approaches"

[Fol+20] Fol et al., "Detection of Faulty Beam Position Monitors using Unsupervised Learning"

[CT04] Calaga and Tomás, "Statistical Analysis of RHIC Beam Position Monitors Performance"

whether incoming data is healthy or faulty by comparing new data to the created clusters. This approach is still a work in progress.

3.3 BACKGROUND

Anomaly detection techniques use data sets containing vastly varying information, e.g., temperature (real number), distance (non-negative number), or activation status (binary class). Additionally, the same type of information could be represented in different ways, e.g., by measuring temperatures using different measurement units. An important step for anomaly detection algorithms is to represent all the different dimensions of the data set in a common reference frame. This preprocessing step is called normalization. In the experiments, we use Z-normalization, calculating the mean value and standard deviation on the training set.

[GU16] Goldstein and Uchida, "A comparative evaluation of unsupervised anomaly detection algorithms for multivariate data"

Anomaly detection is usually categorized [GU16] as follows:

- supervised anomaly detection, when the algorithm uses a set of labeled training data for the detector's initial training. In this category, we can find algorithms traditionally used for pattern recognition;
- semi-supervised anomaly detection or novelty detection, when the algorithm requires a training phase using a data set containing only positive examples, i.e., correct behavior, or negative examples, i.e., anomalous behavior;
- unsupervised anomaly detection, when the data available are not labeled. In this case, the algorithm will use the same data for both the training phase and the inference phase.

In the following, we refer to a data set as an $m \times n$-matrix where m is the number of dimensions of the data, i.e., the values coming from all the different signals at each time step, and n is the number of time steps. A sample p is the vector of size m containing the values of all the dimensions in a single time step.

3.4 SELECTED MODELS

In the next sections, we give a description of the algorithms and their implementation for both unsupervised and semi-supervised anomaly detection.

3.4.1 *Unsupervised Algorithms*

[Llo82] Lloyd, "Least squares quantization in PCM"

▶ K-MEANS CLUSTERING

K-means clustering [Llo82] is an unsupervised learning algorithm that divides a set of data points into K distinct clusters. The goal of K-means clustering is to identify clusters of similar data points and assign them to the same cluster while minimizing the within-cluster sum of squares. The algorithm works by first selecting K initial centroids, which are representative points within the data set. The data points are then assigned to the cluster corresponding to the nearest centroid. The centroids are then updated to be the mean of

all data points assigned to that cluster, and the process is repeated until convergence, i.e., the centroids do not change significantly from one iteration to the next. It is a simple and efficient method for grouping data points into clusters, but it has some limitations, such as the assumption of spherical cluster shapes and the need to specify the number of clusters upfront.

- GAUSSIAN MIXTURE MODELS (GMM)
  Also the Gaussian Mixture Model (GMM) [Rey15] partitions the data points into different clusters. However, the data is not assigned to a single group but assumes a probability of belonging to a certain group based on the degree of similarity between the data points. GMM represents the distribution of a set of data points as a weighted sum of several multivariate Gaussian distributions, where the weights represent the probabilities of the data points belonging to each distribution. The method assumes that the data is a mixture of different Gaussian-distributed populations.

[Rey15] Reynolds, "Gaussian Mixture Models"

- ISOLATION FOREST (IF)
  An Isolation Forest (IF) [LTZ08] is an ensemble learning method consisting of multiple isolation trees. It is based on the concept of isolation, where an outlier is a data point that is more isolated from the rest of the data points. The isolation trees are created independently from each other, each of which is trained on a random subset of the data. For each tree, the algorithm splits the data points into smaller and smaller subgroups based on a randomly selected feature until each data point is in its own subgroup. The number of splits required to isolate a data point is referred to as the path length. The outlier score of a data point is calculated as the average path length of the data point across all of the trees. Data points with shorter path lengths are considered more anomalous, as they are more isolated from the rest of the data points.

[LTZ08] Liu et al., "Isolation Forest"

3.4.2 *Semi-supervised Algorithms*

- *K*-NEAREST NEIGHBORS (KNN/AKNN)
  K-Nearest Neighbors (KNN) [AP02], not to be confused with k-nearest neighbor classification, uses either the distance to the *k*th-Nearest Neighbor (KNN) or the average of the distances with the *k*-Nearest Neighbors (aKNN) to calculate a score. The aKNN score for a sample p is calculated as follows:

$$aKNN(p) = \frac{\sum_{\forall o \in N_k} Dist(p, o)}{k}, \tag{3.1}$$

  where k is the number of neighbors, N_k is the set of k-nearest neighbors, and $Dist(p, o)$ is the distance between the data point p and the data point o.

[AP02] Angiulli and Pizzuti, "Fast Outlier Detection in High Dimensional Spaces"

- LOCAL OUTLIER FACTOR (LOF)
  The local outlier factor (LOF) [Bre+00] is an anomaly detection algorithm that gives a score based on the local densities of both the sample and the k-nearest neighbors. The LOF score for each sample p is defined as:

$$LOF(p) = \frac{\sum_{\forall o \in N_k(p)} \frac{LRD_k(o)}{LRD_k(p)}}{|N_k(p)|}, \tag{3.2}$$

[Bre+00] Breunig et al., "LOF: Identifying Density-Based Local Outliers"

with:

$$LRD_k(p) = 1/\frac{\sum_{\forall o \in N_k(p)} ReachDist_k(p,o)}{|N_k(p)|}, \tag{3.3}$$

where $|N_k(p)|$ is the number of k-nearest-neighbors to p, $ReachDist_k(p,o)$ is the reachability distance between p and o, and LRD is the local reachability density. The implementation used in this paper is available as part of the framework Scikit-learn [Ped+11]. The usage as a semi-supervised anomaly detection algorithm involves a training phase that produces a threshold and a prediction phase where the threshold is used for the classification. The threshold is selected as the value of the cth percentile of the LOF score for the training set, where c is the estimated proportion of outliers in the data set.

[Ped+11] Pedregosa et al., "Scikit-learn: Machine Learning in Python"

- CLUSTER-BASED LOCAL OUTLIER FACTOR (CBLOF)

[HXD03] He et al., "Discovering cluster-based local outliers"

The Cluster-Based Local Outlier Factor (CBLOF) [HXD03] was proposed to solve the problem of not properly considering both clustering and outlier discovery in the data set with previous algorithms. The algorithm gives a score based on the clusters' size and on the distances between the clusters and the samples. The CBLOF distinguishes between small and large clusters using the parameters α and β. In particular, the authors use the numeric parameters α and β to construct a formula that defines the boundary between small and large clusters.

The CBLOF is defined as:

$$CBLOF(p) = \begin{cases} \min_{C_j \in \boldsymbol{C}} Dist(p, C_j) & \text{if } p \in SC \\ Dist(p, C_i) & \text{if } p \in LC \end{cases}, \tag{3.4}$$

where $p \in C_i$. The set $\boldsymbol{C}$ is the set of all clusters, SC and LC are the set of small clusters and large clusters, respectively, $Dist(p,C)$ is the distance between the data point p and the center of the cluster C.

[HXD02] He et al., "Squeezer: An Efficient Algorithm for Clustering Categorical Data"

For clustering, the authors propose the *Squeezer* algorithm [HXD02], but any clustering algorithm can be used in practice.

- HISTOGRAM-BASED OUTLIER SCORE (HBOS)

[GD12] Goldstein and Dengel, "Histogram-based outlier score (HBOS): A fast unsupervised anomaly detection algorithm"

The Histogram-Based Outlier Score (HBOS) [GD12] is a statistical anomaly detection algorithm. This algorithm's main characteristic is that each dimension of the data is considered independent of the others. The HBOS is defined as:

$$HBOS(p) = \sum_{i=0}^{d} \log\left(\frac{1}{Hist_i(p)}\right), \tag{3.5}$$

where d is the number of dimensions of the data and $Hist(p)$ is the height of the bin of p after the normalization of the histograms, a density estimation of the data point p. The creation of the bins of a histogram should use the dynamic bins method, which means that the number of elements in a single bin depends on the element's values, such that larger values form smaller bins.

- ANGLE-BASED OUTLIER DETECTOR (ABOD/FASTABOD)

[KSZ08] Kriegel et al., "Angle-Based Outlier Detection in High-Dimensional Data"

Angle-based outlier detector (ABOD) [KSZ08] was designed to solve the

issues that other algorithms have with high-dimensional data. The main idea is to consider the directions of the distance vectors between the points of the data set. If the spectrum of the observed directions by a point is wide, the point is inside (or close to) a cluster. Otherwise, it is an outlier.

The angle-based outlier factor (ABOF) is calculated considering each triplet of points and calculating:

$$ABOF(p) = Var_{\forall o', o'' \in \mathcal{D}} \left(\frac{\langle \overline{po'}, \overline{po''} \rangle}{\left\| \overline{po'} \right\|^2 \cdot \left\| \overline{po''} \right\|^2} \right), \tag{3.6}$$

where $\mathcal{D}$ is the data set, $\overline{po'}$ and $\overline{po''}$ represent the distance vector from p to o' and from p to o'', respectively, $\|\cdot\|$ is the norm of a vector, and $\langle \cdot, \cdot \rangle$ is the scalar product of two vectors.

To use this algorithm as a semi-supervised anomaly detection algorithm, the data points to which each sample is compared are only the ones of the training set.

Since the complexity is $O(n^3)$ due to the need to consider each triplet in the data set, FastABOD solves this problem by restricting the set of points to be considered for each point to the k-nearest neighbors.

- MINIMUM COVARIANCE DETERMINANT (MCD)

The minimum covariance determinant (MCD) estimator [RD99] is a robust estimator of a data set's covariance. MCD is used to estimate the covariance matrix of a portion of the training data such that the determinant results minimal. The idea is that the covariance matrix having the minimal determinant represents the set of data closer to each other.

[RD99] Rousseeuw and Driessen, "A Fast Algorithm for the Minimum Covariance Determinant Estimator"

The score is then the Mahalanobis distance between the sample considered and the distribution represented by the covariance matrix:

$$D(p) = \sqrt{(p-m)^T \cdot C^{-1} \cdot (p-m)}, \tag{3.7}$$

where m is the average of the subset of the training data used to construct the covariance matrix, and C^{-1} is the inverse of the covariance matrix.

- SUPPORT VECTOR MACHINE (SVM)

The Support Vector Machine (SVM) [Abe05] separates data points of different categories by a multidimensional hyperplane. The hyperplane is built by maximizing the distances between the hyperplane and the nearest points of each category. The SVM works especially well if the faulty data points have a high degree of dissimilarity to the healthy data points.

[Abe05] Abe, *Support Vector Machines for Pattern Classification*

- ONE-CLASS SUPPORT VECTOR MACHINE (OCSVM)

A classifier that is based on a One-Class Support Vector Machine (OCSVM) [Sch+99] constructs a hyperplane and finds a linear boundary on the hyperplane. The classifier separates anomalous data and non-anomalous data using the boundary defined. The hyperplane constructed, also known as kernel space, is commonly obtained from the training data using a transformation based on the radial basis function (RBF) kernel [Cha+10].

[Sch+99] Schölkopf et al., "Support Vector Method for Novelty Detection"

After having a defined transformation function, the decision function gives a classification score based on the sample's position on the hyperplane with respect to the linear boundary defined.

[Cha+10] Chang et al., "Training and Testing Low-degree Polynomial Data Mappings via Linear SVM"

- PRINCIPAL COMPONENT ANALYSIS (PCA)

Principal component analysis (PCA) [Pea01] is commonly used for the reduction of the data dimensionality by identifying the principal components and then using the main ones discarding the others. The same technique can be used for anomaly detection [Shy+03]. In this case, the entity of deviation from the components identified can be used as an anomaly score.

[Pea01] Pearson, "On lines and planes of closest fit to systems of points in space"

[Shy+03] Shyu et al., "A Novel Anomaly Detection Scheme Based on Principal Component Classifier"

For calculating the set of principal components, which is the set of new variables obtained by linearly combining the random variables of the data, an eigenanalysis of the covariance matrix is sufficient. The new set of variables will be uncorrelated and ordered from the component with the highest variance to the component with the least variance.

The usage of semi-supervised anomaly detection requires a training phase where the set of principal components is identified. The score is then calculated using the new data as follows:

$$PCA(p) = \sum_{\forall_{c \in C}} \frac{Dist(p,c)}{c_w}, \tag{3.8}$$

where $c \in C$ is a component, part of the set of components C, identified during the training phase, and c_w is the component weighted by the eigenvalue.

3.5 EVALUATION OF UNSUPERVISED MODELS

We created two data sets by injecting anomalies into the real system. The data acquisition phase comprises the recording of the experiment data that is provided by the LLRF system. In the preprocessing stage, we cleaned the recorded data and labeled the data according to the experiments with either healthy or faulty. After preprocessing, the data is directly used for fault analysis. Additionally, the preprocessed data is transformed into a feature-extracted data set using the Pearson correlation coefficient, a Fourier analysis, and the actual feature extraction. This feature-extracted data set serves as a second data set for the fault analysis. In Figure 3.1, we show the experimental data preparation and experimentation steps.

For the evaluation, we created the two following data sets by extracting data from the European XFEL.

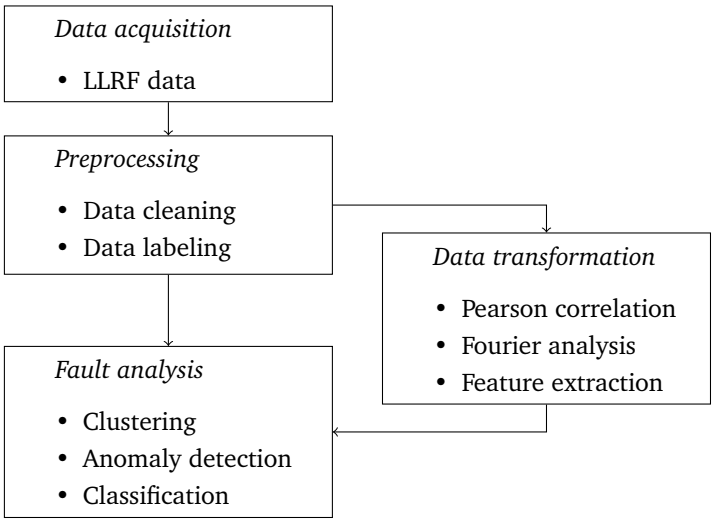
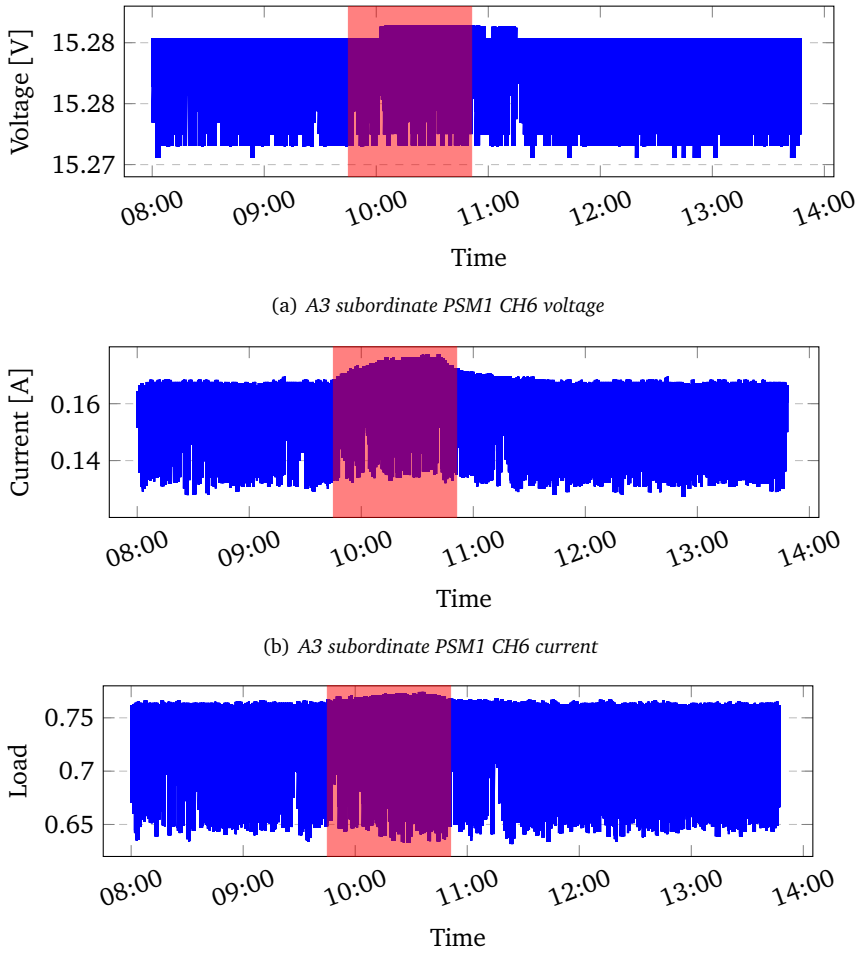


FIGURE 3.1: Experimental data preprocessing workflow.



(a) *A3 subordinate PSM1 CH6 voltage*

(b) *A3 subordinate PSM1 CH6 current*

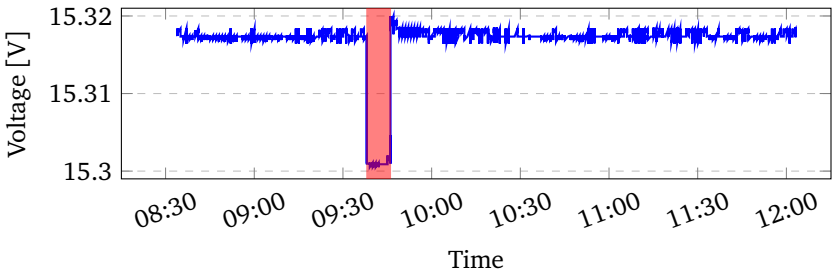
(c) *A3 subordinate PSM1 CH6 load*

FIGURE 3.2: A3 subordinate PSM1 CH6 voltage, current, and load during the door open experiment. The red area indicates the time when doors were opened.

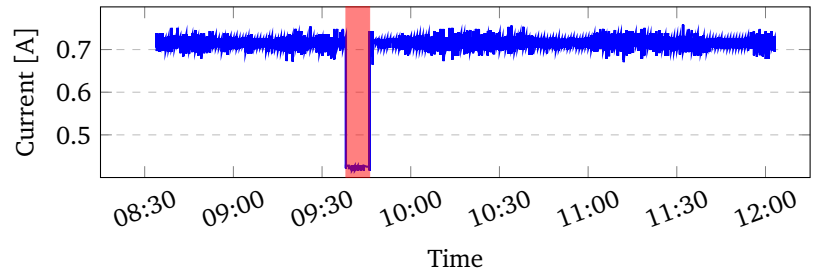
xfel-door was created by opening a door of the housing of the LLRF subordinate system at RF station A3. The resulting data set contains data of approximately 6 hours with a sampling frequency of 10 Hz. For this experiment, we recorded the signals provided by the Power Supply modules (PSMs): voltage, current, temperature, load, is_connected. Additionally, we recorded temperature signals provided by the housing of the LLRF system, as this experiment consists of changing the temperature by opening the door. In total, 116 signals are recorded and used for the fault analysis. Figure 3.2 illustrates the behavior of some signals during the *xfel-door* experiment.

xfel-dcm was created by turning off the Drift Compensation Module (DCM) at the LLRF manager of the RF station A2SP. It comprises data of approximately 3.5 hours, again at a sampling frequency of 10 Hz. We recorded the same signals as in the first experiment except for the temperature signals of the LLRF housing. In total, this data set comprises the recording of 102 different PSM signals. Figure 3.3 shows the behavior of example signals during the *xfel-dcm* experiment.

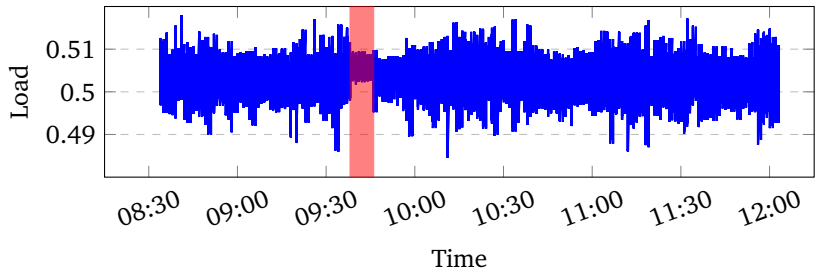
Each experiment focuses only on a specific RF station. Therefore, the data sets that are used by the fault detection algorithms contain just data coming from the experiment-specific LLRF system.



(a) *A2SP manager PSM1 CH2 voltage*



(b) *A2SP manager PSM1 CH2 current*



(c) *A2SP manager PSM1 CH2 load*

FIGURE 3.3: A2SP manager PSM1 CH2 voltage, current, and load during the DCM off experiment. The red area indicates the time when the DCM was turned off.

3.5.1 *Preprocessing and Data Transformation*

The preprocessing stage comprises mainly two steps. First, we cleaned the recorded data containing approximately 1 % missing values. Due to the low proportion of missing values, the data points containing missing values are simply dropped. Then, we manually labeled the data points with either healthy (negative) or faulty (positive).

We used the preprocessed signals as a basis and transformed the data with a feature extraction method. Before the actual feature extraction step, we calculated the correlations between the recorded signals and analyzed the frequencies of the recorded signals. The results of the Fourier analysis are required for the feature extraction as the signals are split at full periods.

▶ CORRELATIONS

[Ben+09] Benesty et al., "Pearson Correlation Coefficient"

Pearson's correlation [Ben+09] is a statistical measure that gives information about the strength of the linear relationship between two samples. The Pearson's correlation coefficient between to samples X and Y is defined as $\rho_{X,Y} = \frac{\mathrm{Cov}(X,Y)}{\sigma_X \sigma_Y}$. The value range of the Pearson's correlation coefficient $\rho_{X,Y}$ goes from -1 to 1, where a correlation coefficient of 1 indicates a perfect linear relationship and a correlation coefficient of -1 indicates that the

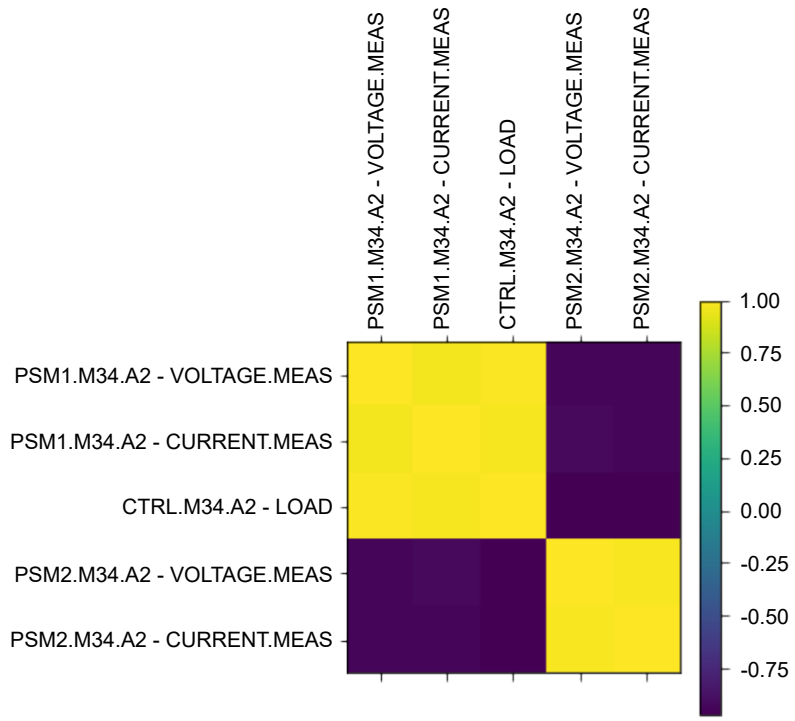
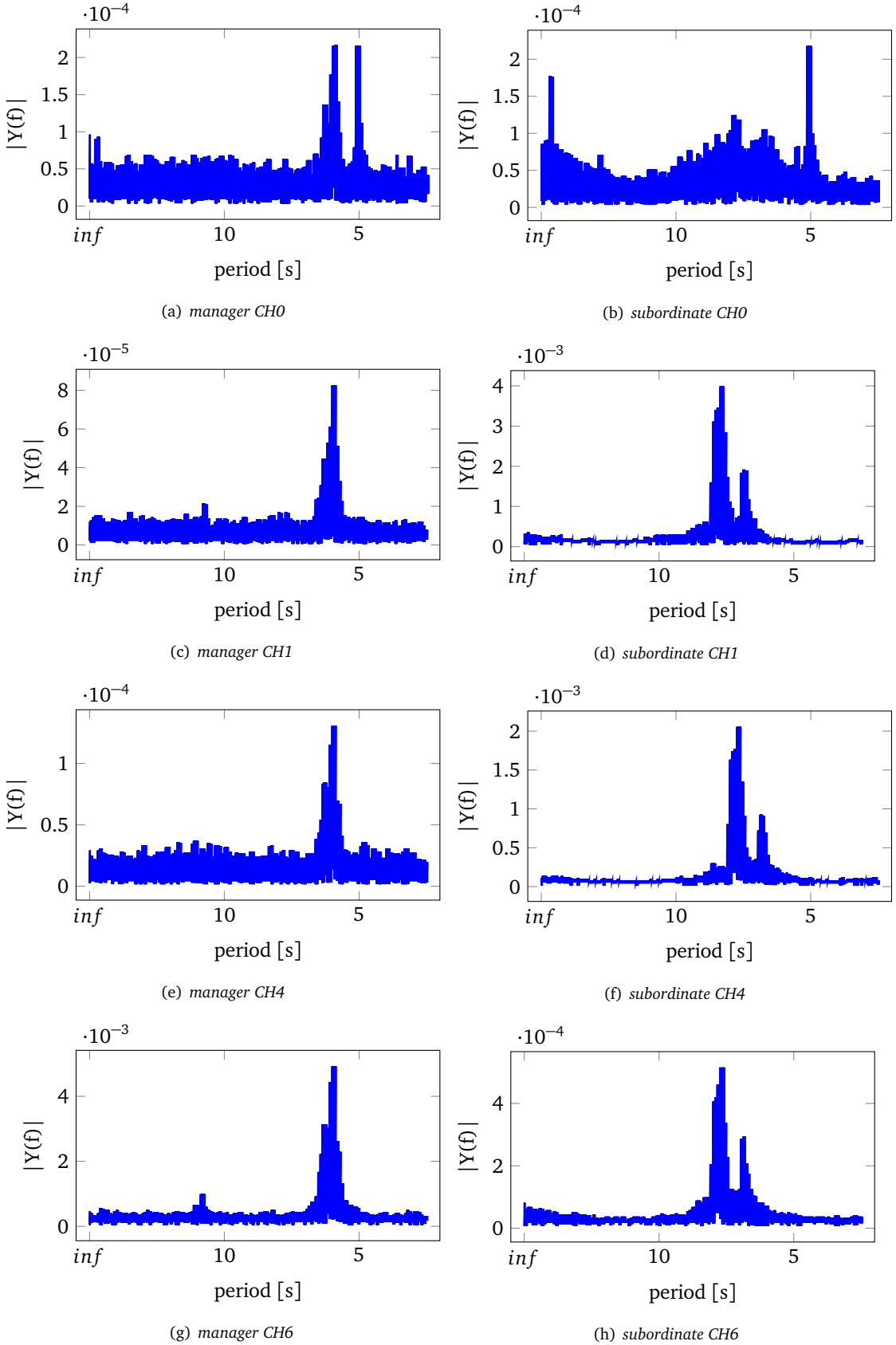


FIGURE 3.4: Correlation matrix of signals from PSM1 and PSM2 channel 2 of RF station A2.

observed samples move in opposite directions. A correlation coefficient of 0 means that the two observed samples do not have a linear relationship. Figure 3.4 shows five different signals of the same voltage channel, measured on PSM1 and PSM2. It shows that the signals coming from the same PSM (PSM1 or PSM2) have a high linear relationship. Signals of PSM1 have a high anti-correlation to the signals coming from PSM2 and vice versa. That is expected since PSM1 and PSM2 are redundant and they share the total workload. We observed the same phenomenon on all the other channels and for all the PSMs.

▸ FOURIER ANALYSIS

The frequencies of signals coming from the LLRF manager and LLRF subordinate are depicted in Figure 3.5. As described in Section 3.5.1, the signals (current, voltage, load) coming from the same PSM do have a linear relationship to each other. Thus, the spectra of the remaining signals are similar. The shown LLRF manager signals tend to have a peak period at 5.5 s, and the LLRF subordinate signals have a peak period at 7 s. The background of these specific values as well as the reason for the differing peak periods is so far unknown and needs to be further investigated in future work. The main finding of the Fourier analysis is that the signals of both manager and subordinate depend on their respective frequencies. The feature extraction makes use of this information by splitting the signals only at multiples of their full periods.



(a) *manager CH0*

(b) *subordinate CH0*

(c) *manager CH1*

(d) *subordinate CH1*

(e) *manager CH4*

(f) *subordinate CH4*

(g) *manager CH6*

(h) *subordinate CH6*

FIGURE 3.5: Frequencies of the selected LLRF manager signals (upper signals) and LLRF subordinate signals (lower signals). The signals belong to the corresponding current channels at RF station A3.

▸ FEATURE EXTRACTION

The feature extraction uses the preprocessed signals to build a new data set. The feature extraction consists of three steps:

1. Build segments of a specified length from the recorded signals
2. Extract different features from these segments (i.e., mean, minimum)
3. Combine extracted features to a new data set

Since the signal behavior depends on its frequency, the segments are built as multiples of their full periods (see Section 3.5.1). Each segment has a length of 77 s. This is the minimal common integer multiple of the analyzed periods (manager: 5.5 s, subordinate: 7 s). We used three basic features, namely the minimum, the maximum, and the mean value. The structure of the original data set and the feature extracted data sets are depicted in Table 3.1.

3.5.2 *Results*

All the selected algorithms are trained and tested on the four data sets. We used k-fold cross validation [Bro20] with 15 folds each for evaluating the model accuracies. The models were trained using the Python library Scikit-learn [Ped+11]. For both K-means clustering and GMM, the number of clusters was set to 2. Since some of the models use a distance-based measure, the data is scaled to a mean of 0 and a variance of 1.

[Bro20] Brownlee, *A Gentle Introduction to K-Fold Cross-Validation*

[Ped+11] Pedregosa et al., "Scikit-learn: Machine Learning in Python"

New data sets were created at another date to validate the trained models. The *xfel-door-v* contains only healthy data. Therefore, the models trained using the *xfel-door* data set are just checked for predicting healthy data points (negatives) correctly. For validating the models trained with the *xfel-dcm* data set, we created a new *xfel-dcm-v* data set by power cycling the DCM again at another date.

The resulting accuracy of training and testing the recorded experiment data sets are depicted in Table 3.2. The table contains the proportions of true positives and true negatives for each trained model. In Table 3.2, we can see that the algorithms perform better when facing the *xfel-dcm* data set compared to the *xfel-door* data set. Also, the algorithms perform better when using the feature data set compared to the original data set.

When using the *xfel-dcm* data set, all algorithms have a proportion of true positives and true negatives of 99 % or above for the original data set. All

	xfel-door		*xfel-dcm*	
	Original	Feature extracted	Original	Feature extracted
Dimensions	116	348	102	306
Length	206840	268	143997	163
Outliers	40718	52	5181	6
% Outliers	19.68 %	19.4 %	3.7 %	4.13 %

TABLE 3.1: Overview of original and feature extracted data sets.

TABLE 3.2: True Positives (TP) and True Negatives (TN) of training the models using k-fold cross-validation.

			K-means	GMM	IF
xfel-dcm	O*	TP	99.99 %	93.83 %	99.98 %
		TN	99.71 %	99.90 %	99.98 %
	F†	TP	100.00 %	100.00 %	100.00 %
		TN	100.00 %	100.00 %	100.00 %
xfel-door	O*	TP	60.45 %	57.17 %	48.13 %
		TN	41.77 %	60.70 %	86.89 %
	F†	TP	86.54 %	84.61 %	80.76 %
		TN	100.00 %	100.00 %	94.44 %

* Original data set.
† Feature extracted data set.

TABLE 3.3: True Positives (TP) and True Negatives (TN) of applying the trained models to the validation data set.

			K-means	GMM	IF
xfel-dcm-v	O*	TP	99.71 %	100.00 %	99.61 %
		TN	99.99 %	55.10 %	99.98 %
	F†	TP	100.00 %	100.00 %	100.00 %
		TN	100.00 %	100.00 %	100.00 %
xfel-door-v	O*	TP	65.05 %	0.00 %	41.83 %
	F†	TP	100.00 %	100.00 %	65.05 %

* Original data set.
† Feature extracted data set.

algorithms predict all positives and negatives correctly when being trained with the feature *xfel-dcm* data set.

When using the *xfel-door* data set, all algorithms have very poor results on the original data set. However, the algorithms have very good accuracies for predicting negatives when using the feature data set. Also, the proportions of the true positives are much higher.

Table 3.3 shows proportions of true positives and true negatives for applying the trained models to the validation data sets. We can see that all models with good training accuracy also perform very well on the validation data sets. All models that are trained with the *xfel-dcm* data set perform very well and show an accuracy of 99 % and above. Only the GMM performs poorly at predicting negatives correctly. The models trained with the original *xfel-door* data set again have poor results, especially compared to the results of the models trained with the feature data set.

3.6 EVALUATION OF SEMI-SUPERVISED ALGORITHMS

We also performed a new set of experiments to evaluate the semi-supervised anomaly detection algorithms. In this case, the experimental data included the two original data sets and two additional data sets. The experiments were performed using the Python libraries PyOD [ZNL19] and Scikit-learn [Ped+11].

[ZNL19] Zhao et al., "PyOD: A Python Toolbox for Scalable Outlier Detection"

[Ped+11] Pedregosa et al., "Scikit-learn: Machine Learning in Python"

Data set	Dim*	Training	Testing	Outliers	%Outliers
xfel-dcm	70	40000	100000	5188	5.19 %
xfel-door	83	40000	166837	40617	24.35 %
xfel-pcie	12	43308	57743	15731	27.24 %
shuttle	9	34108	14500	3022	20.84 %

* Number of dimensions.

TABLE 3.4: Data sets summary.

The runtimes were measured on a system running Python 3.7 on Windows 10 with a processor Intel(R) Core(TM) i7-8750H running at 2.20 GHz and 32 GB of RAM.

3.6.1 *Data Sets Summary*

The evaluation of semi-supervised anomaly detection algorithms requires two sets of data:

- a training set used for training the detector, containing only positive samples, i.e., non-anomalous data;
- a testing set used for the inference phase, including both normal data and anomalies;

In the following paragraphs, we give a brief description of the additional data sets used for the evaluation.

xfel-pcie In this case, the data describes a failure in the PCI Express interface that stops all communication between a specific board and the host system. The signals used include temperature, current, and voltage of the failing board.

shuttle This data set containing data from the shuttle statlog was obtained from the UCI Machine Learning Repository [DG17]. The 9 dimensions of this data set describe the radiator position during flight. For the experiments, all anomalous data points were removed from the training set.

[DG17] Dua and Graff, *UCI Machine Learning Repository*

Table 3.4 shows a summary of the properties of the data sets used. For all the experiments, the data sets were normalized using Z-normalization [GK95].

[GK95] Goldin and Kanellakis, "On Similarity Queries for Time-Series Data: Constraint Specification and Implementation"

3.6.2 *Algorithms Parameters*

All the algorithms selected contain a controllable parameter. The value of the parameter influences different aspects of the calculations, both during testing and inference. A summary of all the parameters used is presented in Table 3.5.

For AKNN and KNN, the parameter chosen is the number of neighbors used during the clustering phase. The interval is between 5 and 50 since a value in that range is the typical choice. The step size selected is 5, i.e., the interval between each selected experimental value. For evaluating ABOD, we use the FastABOD implementation with the same parameters and intervals used for the previous algorithms. We use the same parameters and intervals for LOF. In this case, we fix the parameter c to 0.1. The parameter chosen for CBLOF is the number of clusters formed during the execution of the clustering step. Additionally, we use k-means for clustering, and we fix the parameters α and β to 0.9 and 5, respectively, which means that the

Algorithm	Parameter	Interval	Step Size
ABOD	#neighbors	5 - 50	5
AKNN	#neighbors	5 - 50	5
CBLOF	#clusters	5 - 50	5
HBOS	#bins	5 - 50	5
KNN	#neighbors	5 - 50	5
LOF	#neighbors	5 - 50	5
MCD	support fraction	0.1 - 0.9	0.2
OCSVM	ν	0.1 - 0.9	0.2
PCA	#components	1 - 11[1]	1

TABLE 3.5: Summary of the algorithms' parameters.

[1] In the data set *shuttle*, the interval is 1 - 8 due to the lower number of dimensions available.

large clusters will contain 90 % of the data and that the large clusters are a least five times bigger than the small clusters. For HBOS, the parameter chosen is the number of bins composing the histograms. Also in these cases, the interval chosen is between 5 and 50, and the step size selected is 5. In the case of MCD, the parameter chosen is the support fraction, i.e., the proportion of points to include in the MCD estimation support. The parameter values are chosen in the interval between 0.1 and 0.9. The step size is 0.2. For OCSVM, the parameter ν is both an upper bound on the fraction of outliers in the training set and a lower bound on the fraction of examples used as support vectors. The values for the parameter are chosen in the interval between 0.1 and 0.9, and the step size is 0.2 also in this case. Finally, the parameter used for PCA is the number of components selected from the training phase. In this case, all cases between 1 and the number of dimensions of the data sets are considered.

3.6.3 *Performance metric*

In order to be able to communicate the result of the evaluation in a synthetic way, we use a score that gives an idea of the performances of each algorithm. For this reason, the Area Under the Receiver Operating Characteristic (AUROC) score was adopted. The AUROC score takes into account both the capability of the algorithms to detect anomalies, i.e., distinguishing between classes and the scores assigned to each datapoint [HT01]. It is defined as the area underneath the ROC curve and ranges between 0 and 1. An AUROC score of 1 represents a predictor whose predictions are 100 % correct. An AUROC score of 0 represents a predictor whose predictions are 100 % wrong, i.e., opposite predictions. Finally, an AUROC score of 0.5 represents a predictor whose predictions are random guesses.

The AUROC score is the most used performance metric for evaluating classifiers when the data to be analyzed present an imbalance between the classes [Hai+17]. This is because the metric can cope with the imbalances of the data without showing a bias. However, the main factor to be considered for the choice of the performance metric is the focus on classification success shown by this metric [JCDLT13; Luq+19]. Like the one presented in this

[HT01] Hand and Till, "A Simple Generalisation of the Area Under the ROC Curve for Multiple Class Classification Problems"

[Hai+17] Haixiang et al., "Learning from class-imbalanced data: Review of methods and applications"

[JCDLT13] Jeni et al., "Facing Imbalanced Data-Recommendations for the Use of Performance Metrics"

[Luq+19] Luque et al., "The impact of class imbalance in classification performance metrics based on the binary confusion matrix"

paper, some applications can tolerate classification errors, and the score does not mask poor performances. Different applications, i.e., medical diagnostics, should instead use different metrics.

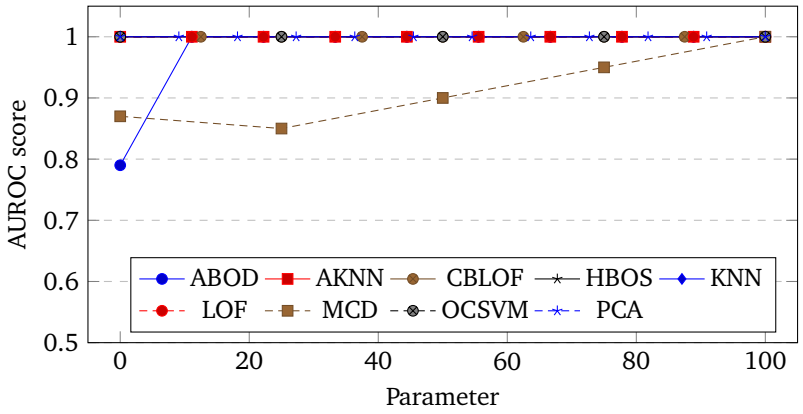
3.6.4 *Results*

The experimental results were obtained by running each algorithm once for each value of the parameter. We show the results for all data sets in Figure 3.6. We plot the value of the AUROC score for each value of the parameter. Each plot is then rescaled in the X-axis to fit in the interval 0–100. For example, for KNN, the value 5 of the parameter is represented as 0, and the value 50 is represented as 100; for MCD, the value 0.1 of the parameter is represented as 0, and the value 0.9 is represented as 100. This is done to show comparably the trend created by the variation of the value, regardless of its meaning.

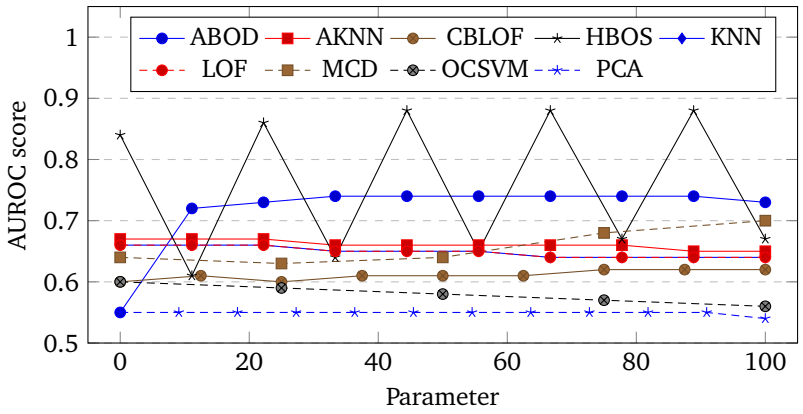
In Figure 3.6(a), we can see that for the *xfel-dcm* data set, all algorithms but ABOD and MCD have a score of 0.99 or above for all values of the parameter. ABOD has a lower score only when using 5 as the parameter for the number of neighbors to consider. In MCD's case, the AUROC score stays lower but reaches a maximum value of 0.99 for a value of the support fraction of 0.9. In Figure 3.6(b), we see that the data set *xfel-door* is more challenging for all the algorithms. In this case, all algorithms perform much worse than the other data sets. We also notice that the AUROC score of HBOS has large swings depending on the value of the parameter. In Figure 3.6(c), we can see that for the *xfel-pcie* data set, all algorithms but ABOD perform very well for all values of the parameters. In ABOD's case, selecting only 5 neighbors, the AUROC score is 0.50, i.e., a random guess. In Figure 3.6(d), we can see that most of the algorithms tend to perform the same or better, using a higher value of the parameter. The only exception to this is OCSVM, for which the AUROC score decreases linearly from 0.97 to 0.92. Also in this case, we notice the swinging behavior of HBOS, although with lower variability than the *xfel-door* case.

From Figure 3.6, we deduce that the effect of the parameter is not always noticeable in the AUROC score. Another general takeaway point is that choosing a higher value of the parameter returns better results for the interval that we considered. Especially when using ABOD, a value of the parameter that is too low results in a decreased AUROC score. However, the parameter sometimes also influences the runtime of some algorithms. Table 3.6 shows the average runtimes for both the training and the inference phases at each value of the parameter over all data sets. Here, we can see that the runtimes of ABOD, CBLOF, OCSVM depend on the parameter. We also notice that using only 5 neighbors with HBOS, the training phase runtime increases by almost 20 times. This behavior is consistent across all considered data sets.

Table 3.7 reports the mean value and the standard deviation over all the runs for each data set. The last two columns give a summative result for all data sets: in the first column, we report the average of the scores reported in the previous columns for all the data sets; in the second column, we report the average variance's square root. The last row gives a summative result

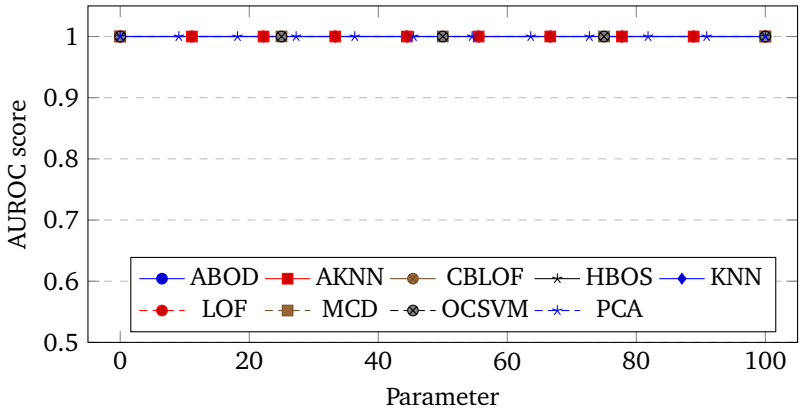


(a) *xfel-dcm*

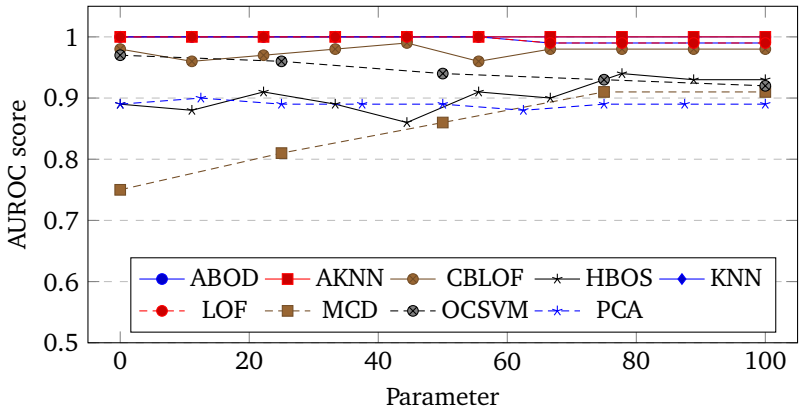


(b) *xfel-door*

FIGURE 3.6: AUROC scores for the data sets at different values of the selected parameter.



(c) *xfel-pcie*



(d) *shuttle*

TABLE 3.6: Training and inference runtime results for each value of the parameters averaged over all data sets in Table 3.4.

		Runtime [s]									
Neighbors		5	10	15	20	25	30	35	40	45	50
ABOD	Train	61.77	106.87	124.17	133.45	181.94	239.39	293.39	364.96	438.77	516.53
	Infer	298.51	349.96	425.24	536.54	574.97	743.49	895.43	1010.83	1210.83	1448.30
AKNN	Train	83.04	123.91	126.31	131.54	152.40	138.91	146.94	143.61	145.61	133.76
	Infer	424.03	410.10	424.95	418.65	423.98	442.24	438.31	451.07	456.11	457.28
KNN	Train	88.03	125.65	127.86	130.07	131.07	132.76	134.99	134.72	135.49	137.12
	Infer	418.20	448.85	457.03	464.61	463.07	465.92	472.52	468.53	485.15	498.52
LOF	Train	92.16	134.31	136.82	145.97	145.05	139.35	140.06	136.82	142.16	143.56
	Infer	424.94	436.48	432.71	457.12	449.68	452.32	453.97	453.71	493.48	450.25
Clusters		5	10	15	20	25	30	35	40	45	50
CBLOF	Train	1.84	2.52	2.78	3.36	4.33	4.75	5.48	6.33	6.88	7.46
	Infer	0.01	0.17	0.19	0.19	0.21	0.21	0.21	0.21	0.21	0.24
Bins		5	10	15	20	25	30	35	40	45	50
HBOS	Train	1.97	0.11	0.09	0.10	0.09	0.09	0.09	0.09	0.10	0.11
	Infer	0.22	0.21	0.21	0.21	0.21	0.22	0.22	0.22	0.22	0.22
Support Fraction		0.1	0.3	0.5	0.7	0.9					
MCD	Train	23.26	24.79	27.28	27.76	26.71					
	Infer	0.39	0.37	0.37	0.39	0.38					
ν		0.1	0.3	0.5	0.7	0.9					
OCSVM	Train	45.01	120.80	174.80	211.21	225.74					
	Infer	31.13	102.50	167.10	233.24	296.44					
Components		1	2	3	4	5	6	7	8	9	10	11
PCA	Train	0.16	0.17	0.18	0.18	0.19	0.19	0.18	0.18	0.23	0.20	0.21
	Infer	0.05	0.05	0.06	0.06	0.07	0.07	0.08	0.08	0.12	0.12	0.13

TABLE 3.7: AUROC results.

Algorithm	*xfel-dcm*		*xfel-door*		*xfel-pcie*		*shuttle*		*Summary*	
	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD
ABOD	0.9786	0.0639	0.7158	0.0545	0.9234	0.1420	0.9993	0.0002	0.9043	0.0825
AKNN	1.0000	<0.0001	0.6604	0.0052	0.9992	0.0001	0.9976	0.0004	0.9143	0.0026
CBLOF	1.0000	<0.0001	0.6107	0.0080	0.9991	<0.0001	0.9766	0.0087	0.8966	0.0059
HBOS	0.9999	0.0001	0.7568	0.1119	1.0000	<0.0001	0.9033	0.0249	0.9150	0.0573
KNN	1.0000	<0.0001	0.6497	0.0076	0.9991	0.0001	0.9958	0.0015	0.9112	0.0039
LOF	1.0000	0.0001	0.7248	0.0471	0.9885	0.0053	0.9988	0.0004	0.9280	0.0237
MCD	0.9143	0.0522	0.6577	0.0277	0.9997	0.0004	0.8489	0.0599	0.8552	0.0421
OCSVM	0.9999	<0.0001	0.5776	0.0154	0.9996	<0.0001	0.9463	0.0192	0.8809	0.0123
PCA	1.0000	<0.0001	0.5490	0.0040	0.9991	<0.0001	0.8904	0.0068	0.8596	0.0040
Mean	0.9881	0.0275	0.6558	0.0458	0.9898	0.0474	0.9508	0.0229	0.8961	0.0375

for all algorithms instead. Also in this case, the first column contains the average of the scores reported in the previous rows for all the data sets, and the second column contains the average variance's square root.

The table shows that almost all algorithms have a very good AUROC score for the data set *xfel-dcm*. Only MCD and, to a lower extent, ABOD have a worse score. In the data set *xfel-door*, all algorithms have a lower score. The algorithms PCA and OCSVM perform only slightly better than a random guess. In this case, only HBOS, and only for some values of the parameter, achieves a maximum AUROC score of 0.88. The table shows very well also the minor differences in the *xfel-pcie* data set. ABOD is penalized in this table by the low score obtained with a lower number of neighbors. For this algorithm, the mean is 0.9804, and the standard deviation is 0.0042, removing the first two values obtained (0.50 and 0.88, respectively). For the *shuttle* data set, Table 3.7 shows that ABOD has the best overall score. In this case, the AUROC score remains above 0.99 for all values of the parameter. From the last two columns, we can see that LOF has the best overall score, and MCD has the worst overall score.

Table 3.8 and Table 3.9 report the runtime values for the training and testing phases, respectively. From the two tables, we can immediately understand that the number of samples considered for each phase is the main factor for the differences. In particular, in Table 3.9, we can see that, since the number of samples of the testing set of the *xfel-pcie* data set is 4 times larger than the *shuttle* training set, the runtime numbers are 4 times larger as well. Another difference that affects the runtimes is the number of dimensions. Considering this time the training time for the data sets *xfel-door* and *xfel-pcie*, we have very similar ratios in the number of dimensions of the two training sets and average training times.

The algorithm summaries show that the most noticeable result is the poor runtime of ABOD with respect to the other algorithms. From the same columns, we can also see that the best-performing algorithms in the training

TABLE 3.8: Training time results [s].

Algorithm	*xfel-dcm*		*xfel-door*		*xfel-pcie*		*shuttle*		*Summary*	
	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD
ABOD	181.470	139.418	468.874	182.835	132.797	97.622	201.377	168.923	246.129	150.774
AKNN	238.707	41.041	257.993	36.334	26.554	1.173	7.181	1.914	132.609	27.430
CBLOF	8.062	2.689	8.317	2.767	1.009	0.415	2.192	0.935	4.895	1.996
HBOS	0.362	0.594	0.380	0.551	0.203	0.540	0.204	0.565	0.287	0.563
KNN	202.689	28.534	274.871	24.593	26.605	1.597	6.952	1.588	127.780	18.869
LOF	235.795	34.984	273.423	25.760	26.265	1.184	7.041	1.566	135.631	21.745
MCD	38.841	2.460	52.768	2.499	2.834	0.113	9.412	2.949	25.964	2.292
OCSVM	218.806	93.180	309.065	129.763	53.946	23.788	40.248	17.817	155.516	81.247
PCA	0.288	0.068	0.336	0.090	0.092	0.026	0.056	0.015	0.193	0.058
Mean	125.002	38.108	182.892	45.021	30.034	14.051	30.518	21.808	92.112	58.618

TABLE 3.9: Inference time results [s].

Algorithm	*xfel-dcm*		*xfel-door*		*xfel-pcie*		*shuttle*		*Summary*	
	Mean	SD	Mean	SD	Mean	SD	Mean	SD	Mean	SD
ABOD	844.195	397.339	1868.200	855.200	203.637	144.922	81.626	66.400	749.414	19.130
AKNN	563.947	35.635	1141.651	88.584	28.124	2.503	4.990	0.703	434.678	5.644
CBLOF	0.268	0.027	0.543	0.045	0.022	0.003	0.006	0.001	0.210	0.137
HBOS	0.282	0.026	0.578	0.026	0.022	0.002	0.003	0.001	0.221	0.116
KNN	603.362	41.275	1221.880	39.791	26.970	2.187	4.772	0.685	464.246	4.581
LOF	587.019	33.565	1190.931	50.398	20.865	2.299	3.064	0.673	450.470	4.662
MCD	0.379	0.015	1.149	0.019	0.010	0.000	0.002	0.000	0.385	0.094
OCSVM	186.148	99.540	447.118	257.089	26.195	14.963	4.889	2.790	166.087	9.674
PCA	0.130	0.112	0.272	0.257	0.015	0.004	0.003	0.001	0.105	0.306
Mean	309.525	67.504	652.480	143.490	33.984	18.543	11.039	7.917	251.757	7.705

phase are HBOS and PCA. Since, as seen in Table 3.6, HBOS requires a much larger time for training using 5 bins. We calculated that the mean value is 0.1 s removing those values. The best runtimes for the testing phase are obtained using CBLOF, HBOS, MCD, and PCA.

3.7 SUMMARY

In this chapter, we gave a unified description of various anomaly detection algorithms, performed a comprehensive evaluation, and studied their performance and scalability when applied to the data produced by the LLRF

system of the European XFEL. Additionally, we tested the anomaly detection algorithms on a standard data set acquired from the public UCI Machine Learning Repository [DG17]. The data sets were selected to show the distinct characteristics of each algorithm.

[DG17] Dua and Graff, *UCI Machine Learning Repository*

This analysis has a twofold utility: 1. it gives a precise representation of the algorithms to the researchers of the anomaly detection community; 2. it also gives us a better understanding of the algorithms and their performances on our specific system.

From the experimental results, we conclude that for semi-supervised anomaly detection, simple algorithms, such as HBOS or CBLOF, can perform satisfactorily. This makes it feasible to use such algorithms for real-time or near-real-time fault detection. Also, a careful selection of the signals is necessary since the prediction can be more or less accurate depending on the anomaly observed. This analysis of anomaly detection algorithms for time point outliers gives us an idea about the effective algorithms and data processing steps. In the next chapter, we use anomaly detection for time-series anomalies to tackle a specific problem of particle accelerators.

4

Anomaly Detection Based Quench Detection System of SRF Cavities

"I learned very early the difference between knowing the name of something and knowing something."
—Richard P. Feynman

▸ SYNOPSIS In this chapter, we discuss the application of anomaly detection for time-series outliers in the context of a specific real-world problem. This chapter is based on the paper

> Gianluca Martino, Andrea Bellandi, Julien Branlard, Annika Eichler, Holger Schlarb, Görschwin Fey, Lawrence Doolittle, Sebastian Aderhold, Andrew Benwell, Daniel Gonnella, Sonya Hoobler, Janice Nelson, Ryan Douglas Porter, Alessandro Ratti, and Lisa Zacarias. "Anomaly Detection Based Quench Detection System for CW Operation of SRF Cavities". In: *Proc. 31st International Linear Accelerator Conference (LINAC'22)* (Liverpool, UK). International Linear Accelerator Conference 31. JACoW Publishing, Geneva, Switzerland, Sept. 2022, THPOPA15, pp. 775–777. ISBN: 978-3-95450-215-8. DOI: 10.18429/JACoW-LINAC2022-THPOPA15.

We show that the anomaly detection techniques discussed can be applied to the problem of quench detection. We compare the performances of two techniques based on convolutional layers. Using such techniques in the context of machine-level faults has the advantage of being robust against noisy signals.

Moreover, in this chapter, we discuss the work carried out at the SLAC National Accelerator Laboratory toward the creation of a fault display for the upgraded Linac Coherent Light Source (LCLS-II). The fault display visualizes traces of previous trips to analyze and identify faults. Additionally, it allows labeling faults to be used for training anomaly detection algorithms.

4.1 INTRODUCTION

Modern linear accelerators use superconducting radio frequency (SRF) cavities as the main component for achieving high accelerating gradients. SRF cavities are used due to their superior energy efficiency for the same accelerating gradient and lower beam impedance. This means reaching higher particle energies than normally possible at a lower operating cost [Sch06]. However, a lot of care needs to be posed to the control system of SRF

[Sch06] Schmüser, "Basic principles of RF superconductivity and superconducting cavities"

[Bel+21] Bellandi et al., "Online Detuning Computation and Quench Detection for Superconducting Resonators"

[Cha+09] Champion et al., "Quench-Limited SRF Cavities: Failure at the Heat-Affected Zone"

cavities due to the high susceptibility to cavity resonance frequency perturbations, e.g., external microphonics and gradient-induced Lorentz force detuning [Bel+21]. One of the main limiting factors for SRF cavities is the disruption of the superconductivity in part or the entirety of the cavity. Such superconductivity disruption is also referred to as quench. Quenches happen mainly when a cavity containing defects or contamination of the material is driven by excessive RF power [Cha+09]. Quenches must be avoided since the disruption of the superconductivity leads to an increased heat load and subsequent lengthy disruptions in the cryogenic system.

The detection of quenches is usually performed by estimating the value of the unloaded quality factor Q_0. However, we can only measure the loaded quality factor Q_L, which relates to Q_0 and the external quality factor Q_{ext} as follows:

$$\frac{1}{Q_L} = \frac{1}{Q_{ext}} + \frac{1}{Q_0}, \tag{4.1}$$

Typical values for Q_{ext} and Q_0 in normal conditions are approximately 3×10^6 and 2×10^{10}, respectively. In the case of a quench, Q_0 can reach values as low as 10^7. With such values, from Eq. (4.1), it can be derived that it is necessary to detect minimal variations of the value of Q_L.

The decrease of Q_L, e.g., as a result of a quench, can be detected as an increase of the cavity half bandwidth $f_{(1/2)}$. In pulsed machines, the amplitude decay estimation can be used to calculate $f_{(1/2)}$ thanks to the following:

$$f_{(1/2)} = \frac{f_0}{2Q_L} = \frac{1}{2\pi\tau}, \tag{4.2}$$

where f_0 is the cavity resonance frequency and τ is the exponential time constant of the cavity gradient decay. In CW machines, since no amplitude decay is available during normal operation, it is necessary to use the cavity signals expressed as in-phase and quadrature (I&Q) components. For the half bandwidth calculation, starting from the cavity dynamics model, the following is used [Bel+21]:

$$f_{(1/2)} = \frac{I_p\left(KI_f + BI_B - \frac{\dot{I}_p}{2\pi}\right) + Q_p\left(KQ_f + BQ_B - \frac{\dot{Q}_p}{2\pi}\right)}{I_p^2 + Q_p^2}, \tag{4.3}$$

where $K = \frac{f_0}{Q_{ext}}$, $B = \frac{f_0}{2} r/Q$, r/Q is the geometric shunt impedance, and the subscripts p, f and b refer to the probe, forward, and beam current signals, respectively. A different possible approach for the detection of a quench can be derived from the estimation of the cavity power dissipation in the absence of beam:

$$P_{diss} = P_f - P_r - \dot{U}, \tag{4.4}$$

where P_f is the forward power, P_r is the reflected power, and $U = \frac{V^2}{f_0(r/Q)}$ is the cavity stored energy equation. In this case, an increase in the cavity power dissipation above a certain threshold indicates a cavity quench.

The approaches presented for CW machines require a precise calibration of the cavity signals and suffer noisy signals. For those reasons, it is possible to have false quenches with a consequent trip that results in unnecessary downtime.

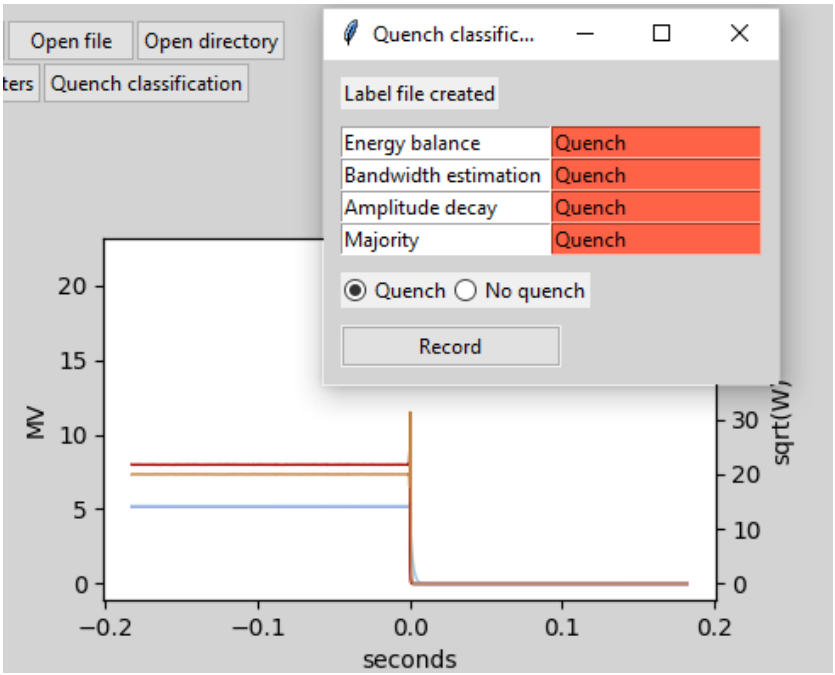


FIGURE 4.1: Fault display classification window.

In this chapter, we compare two different approaches for time-series anomaly detection: 1) a Convolutional Neural Network (CNN) classifier based on the work in [Zha+17]; 2) Arsenal, an ensemble of ROCKET classifiers [DPW20].

The recent usage of CNNs and in general of techniques based on convolutional kernels is motivated by the recent success of CNNs for image classification. In [Faw+20], the authors explain that such techniques should be effective also for time series classification since time series have the same topology as images, with one less dimension.

The dataset used has been obtained during LCLS-II commissioning. LCLS-II is a superconducting linear accelerator (LINAC) at SLAC that results from the collaboration between multiple United States Department of Energy (DOE) laboratories. LCLS-II operates in CW operation mode to accelerate an electron beam with a repetition rate of 1 MHz. Additionally, for this project, we built the LCLS-II fault display, a panel that helps identify, analyze, and categorize faults. The main component is a user interface that enables the data selection for the training of the quench detection scheme. The user interface is composed of plot displays for the visualization of cavity signals and processed data, e.g., the calculation of the quench detection system.

[Zha+17] Zhao et al., "Convolutional neural networks for time series classification"

[DPW20] Dempster et al., "ROCKET: exceptionally fast and accurate time series classification using random convolutional kernels"

[Faw+20] Fawaz et al., "InceptionTime: Finding AlexNet for time series classification"

4.2 LCLS-II FAULT DISPLAY

The main goal of a fault analysis interface is to provide the means for performing incident review and post-mortem analysis. Specifically, it needs to provide a way to review past events, identify the causes, and for debugging.

The LCLS-II fault display allows for visualizing previous faults, automatically saved by the control system. Since the initial focus of this project was on the quench detection system, the fault display also calculates and plots the cavity power dissipation. The calculations are performed independently

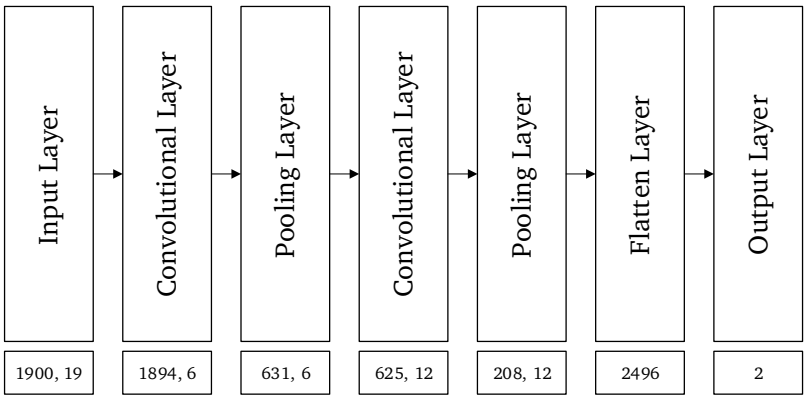


FIGURE 4.2: CNN network structure.

of the FPGA to exclude bugs in the implementation.

A quench classification panel has been added to introduce the human-in-the-loop paradigm. The goal of this panel is to aid the classification of the events. At this stage, the classification is relative only to quench events.

This quench classification panel gives an initial guess about whether or not the event is a quench based on the three different methods explained in Section 4.1. However, there are multiple ways for those methods to fail, e.g., uncalibrated signals or data corruption. For this reason, the expert needs to manually select whether the event is a quench or not. A screenshot of the quench labeling interface is shown in Figure 4.1.

4.3 ROBUST QUENCH DETECTION

[EBT23] Eichler et al., "Anomaly detection at the European X-ray Free Electron Laser using a parity-space-based method"

[Naw+18] Nawaz et al., "Anomaly Detection for the European XFEL using a Nonlinear Parity Space Method"

Robust methods for quench detection is an active research area [EBT23; Naw+18]. While classical methods use the cavity model to compute a signal that represents a residual, improved quench detection schemes require adding signals to distinguish variations given by the normal operation of the machine and faulty behaviors.

We compare two methods for time series classification:

- a Convolutional Neural Network (CNN);
- an ensemble of Random Convolutional Kernel Transform (ROCKET) classifiers [DPW20] known as Arsenal [Mid+21].

[DPW20] Dempster et al., "ROCKET: exceptionally fast and accurate time series classification using random convolutional kernels"

[Mid+21] Middlehurst et al., "HIVE-COTE 2.0: a new meta ensemble for time series classification"

The CNN classifier uses a sequence of convolutional layers interspersed with average pooling layers for filtering and down-sampling, respectively. The structure of the classifier is represented in Fig. 4.2. This CNN structure shows good classification accuracy with multivariate time series and good noise tolerance [Zha+17].

[Zha+17] Zhao et al., "Convolutional neural networks for time series classification"

The ROCKET classifiers, composing Arsenal, are composed of a large number of convolutional kernels to capture the features of the input data. However, in contrast to CNNs, the convolutional kernels are not learned but generated randomly. The ensemble classifier is composed of 5 ROCKET classifiers using a ridge classifier with built-in cross-validation [HTF09] for the generation of the final classification.

[HTF09] Hastie et al., *The Elements of Statistical Learning: Data Mining, Inference, and Prediction, 2nd Edition*

We use the waveforms of the amplitude, phase, power, I, and Q signals from the cavity, the forward, and the reverse probes. Additionally, each time series is enhanced with additional waveforms, calculated starting

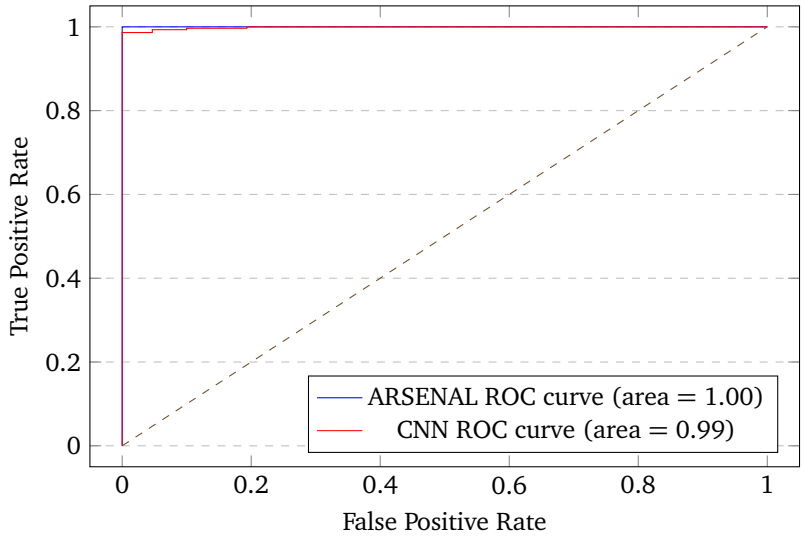


FIGURE 4.3: AUROC curves for the entire waveforms.

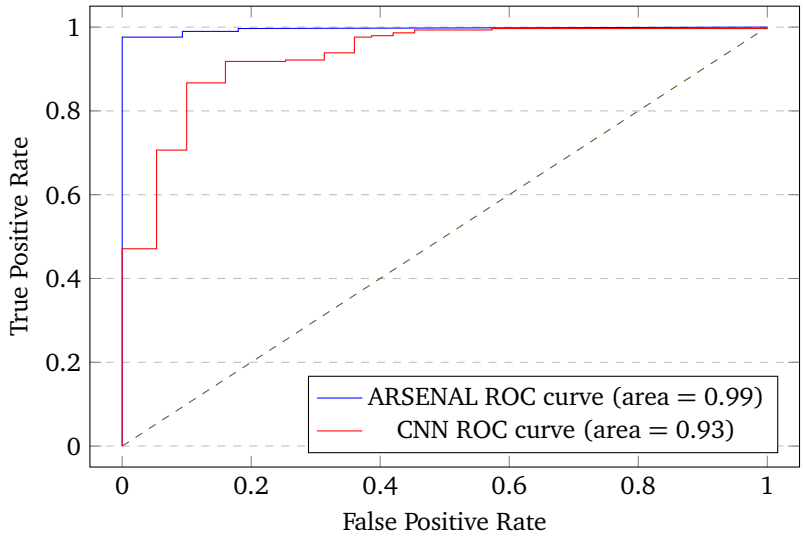


FIGURE 4.4: AUROC curves for the reduced waveforms.

from Eq. (4.4). Specifically, we add the calculated cavity stored energy, the system stored energy, and the waveguide energy estimations.

4.4 EXPERIMENTAL RESULTS

The data set used contains all fault traces stored during LCLS-II commissioning. The data is stored when a trip occurs and is tagged with the system that triggered the trip event. Each event generates a multidimensional time series composed of multiple waveforms and enhanced with additional waveforms, calculated starting from Eq. (4.4). The waveforms stored start 2 ms before the event, end 2 ms after the event, and contain 2048 time points. The data set is split equally between the training and testing sets.

We performed two experiments:

- in the first experiment, the data set includes the entire waveforms;
- in the second experiment, the waveforms are reduced to include all time points leading to the trip event but excluding the event itself and a variable number of time points preceding.

The results are evaluated by using the Receiver Operating Characteristic (ROC) curve (see Section 3.6.3).

In the first experiment, we test the detection capabilities of the algorithms. The result of the first experiment is shown in Figure 4.3. In the second experiment, we test the prediction capabilities of both algorithms instead. This is possible since the quenching threshold is never surpassed in both data sets. The result of the second experiment is shown in Figure 4.4.

The results for both experiments show that the ARSENAL approach performs better both in prediction and detection. The accuracy scores recorded for the second experiment are 0.96 for the ARSENAL classifier and 0.86 for the CNN classifier.

4.5 SUMMARY

In this chapter, we presented a fault display that allows performing post-mortem analysis and incident review for most quench-related faults and more. However, additional data representation will help further improve the analysis capabilities. Such plots can help identify FPGA-related issues that are not visible otherwise.

Additionally, the experiments show that it is feasible to predict whether the running conditions represented by a set of traces will lead to a quench before the value calculated based on the cavity model surpasses the quenching threshold. This chapter also shows the efficacy of anomaly detection for time-series outliers in the prediction of system faults.

Utilizing such a technique in a real machine requires fast reaction and prediction times. The next step will be to implement both techniques in FPGAs to reduce both the computation time and latency of the calculations. In the next chapter, we show a comparison of different CPU, GPU, and FPGA implementations for a specific CNN implementation.

5

Hardware Implementation of Convolutional Layers

"Premature optimization is the root of all evil."
—Donald Knuth

▸ SYNOPSIS This chapter discusses the performance of various hardware implementations of convolutional neural networks. It is based on the paper

> Ahmad Al-Zoubi, Gianluca Martino, Fin Hendrik Bahnsen, Jun Zhu, Holger Schlarb, and Görschwin Fey. "CNN Implementation and Analysis on Xilinx Versal ACAP at European XFEL". in: *35th IEEE International System-on-Chip Conference, SOCC 2022, Belfast, United Kingdom, September 5-8, 2022*. Ed. by Sakir Sezer, Thomas Büchner, Jürgen Becker, Andrew Marshall, Fahad Siddiqui, Tanja Harbaum, and Kieran McLaughlin. IEEE, 2022, pp. 1–6. DOI: 10.1109/SOCC56010.2022.9908101.

In this chapter, we introduce the field of hardware acceleration of Convolutional Neural Networks (CNNs). After presenting and discussing the related work, we give some details of the specific convolutional neural network to be implemented in hardware. The CNN used in this chapter has been trained to predict some parameters of the European XFEL injector section from the longitudinal phase-space image obtained at the end of the injector section.

The results of these experiments are useful for evaluating the optimal hardware implementation strategy for a wide class of CNNs, including the ones used for time-series anomaly detection.

5.1 INTRODUCTION

Convolutional Neural Networks (CNNs) have shown significant success in a range of applications, including computer vision, pattern recognition, and many others [Dai+20; Sch+21; Zhu+21]. However, as the CNN architectures are evolving, a general trend of increased network size and complexity [Ahm+19], poses a real challenge for the underlying hardware to keep up with the increased computational demands.

Researchers have devised and used numerous hardware accelerators to provide CNNs with high performance and efficiency. From General Purpose GPUs (GPGPUs), customized FPGA accelerators, and ASIC designs [SSR19; SS20; Qas+21]. Although these accelerators improve the CNN performance,

[Dai+20] Dai et al., "Fast and accurate cable detection using CNN"

[Sch+21] Scheinker et al., "An adaptive approach to machine learning for compact particle accelerators"

[Zhu+21] Zhu et al., "High-Fidelity Prediction of Megapixel Longitudinal Phase-Space Images of Electron Beams Using Encoder-Decoder Neural Networks"

[Ahm+19] Ahmad et al., "Xilinx First 7nm Device: Versal AI Core (VC1902)"

[SSR19] Sharma et al., "Implementation of CNN on Zynq based FPGA for Real-time Object Detection"

[SS20] Sledevič and Serackis, "mNet2FPGA: A Design Flow for Mapping a Fixed-Point CNN to Zynq SoC FPGA"

[Qas+21] Qasaimeh et al., "Benchmarking vision kernels and neural network inference accelerators on embedded platforms"

each device suffers different challenges, making it difficult for applications with several optimization targets to have a single optimal device. While the GPU presents a suitable option for applications requiring high throughput, the low energy efficiency and high latency execution model prevent the GPU from satisfying the range of CNN application requirements [Lv+20; Mel+18]. On the other hand, while FPGAs are considered more energy-efficient and flexible to be customized for either throughput or latency optimization, they require designing a corresponding accelerator. That requires digital design skills and non-standardized IP interfaces, limiting the adoption of FPGAs by data scientists. Finally, while ASIC solutions might perform the best for current CNN architectures, rapid development often imposes hardware upgrades, which are considered an expensive alternative with respect to Non-Recurring Engineering (NRE) costs.

[Lv+20] Lv et al., "Research on Dynamic Reconfiguration Technology of Neural Network Accelerator Based on Zynq"

[Mel+18] Meloni et al., "NEURAghe: Exploiting CPU-FPGA Synergies for Efficient and Flexible CNN Inference Acceleration on Zynq SoCs"

In recent years, machine learning tools have garnered a lot of attention because they can learn relationships between inputs and outputs of large complex systems directly from data and generate predictions [Zhu+21; Sch+21; Zhu+21].

[Zhu+21] Zhu et al., "High-Fidelity Prediction of Megapixel Longitudinal Phase-Space Images of Electron Beams Using Encoder-Decoder Neural Networks"

[Sch+21] Scheinker et al., "An adaptive approach to machine learning for compact particle accelerators"

In this chapter, we compare the performance of a customized CNN architecture developed for the European XFEL. We adopt the Xilinx Vitis AI[1] design flow to assess the accuracy, throughput, and latency of the implementation in comparison to the state-of-the-art GPU, CPU, and MPSoC FPGA. In addition, we analyze these results and draw conclusions about the performance of the Deep-Processing-Unit (DPU) on the Versal ACAP in comparison to the Zynq UltraScale+ counterpart.

[1] https://github.com/Xilinx/Vitis-AI

Our contribution is a thorough evaluation of the various options for accelerating the inference of CNNs on hardware devices. We evaluate the Vitis AI quantization methods for accuracy-critical regression of a customized CNN architecture.

The rest of the chapter is organized as follows: in Section 5.2, we present some related work; in Section 5.3, we describe the application and the target CNN architecture; in Section 5.4, we give some details about the special purpose processors used for the hardware acceleration of the CNN; in Section 5.5, we present the experimental results; Finally, in Section 5.6, we discuss the results with respect to the overall fault detection system and conclude Part II.

5.2 RELATED WORK

Several studies have successfully implemented CNNs on hardware accelerators, with a focus on throughput, latency, or energy, as their optimization target. In [SSR19], the study evaluates Python-Productivity-for-Zynq (PYNQ) to implement 4 different CNNs on the Zynq FPGA. Results show that using the PYNQ design flow, the combination of SSD and *Inception v2*, the model was the most accurate with near real-time object detection performance. In [SS20], the study presents a framework that maps CNNs to cost-optimized SoC FPGAs. Developed in Matlab, with restrictions on the layer design, CNNs are translated into instructions to run over the predefined CNN core. The CNN core was able to provide 32 GOP/s, utilizing up to 67 % of the programmable logic, 41 % of the flip-flops, 30 % of the BRAMs and 31 %

[SSR19] Sharma et al., "Implementation of CNN on Zynq based FPGA for Real-time Object Detection"

[SS20] Sledevič and Serackis, "mNet2FPGA: A Design Flow for Mapping a Fixed-Point CNN to Zynq SoC FPGA"

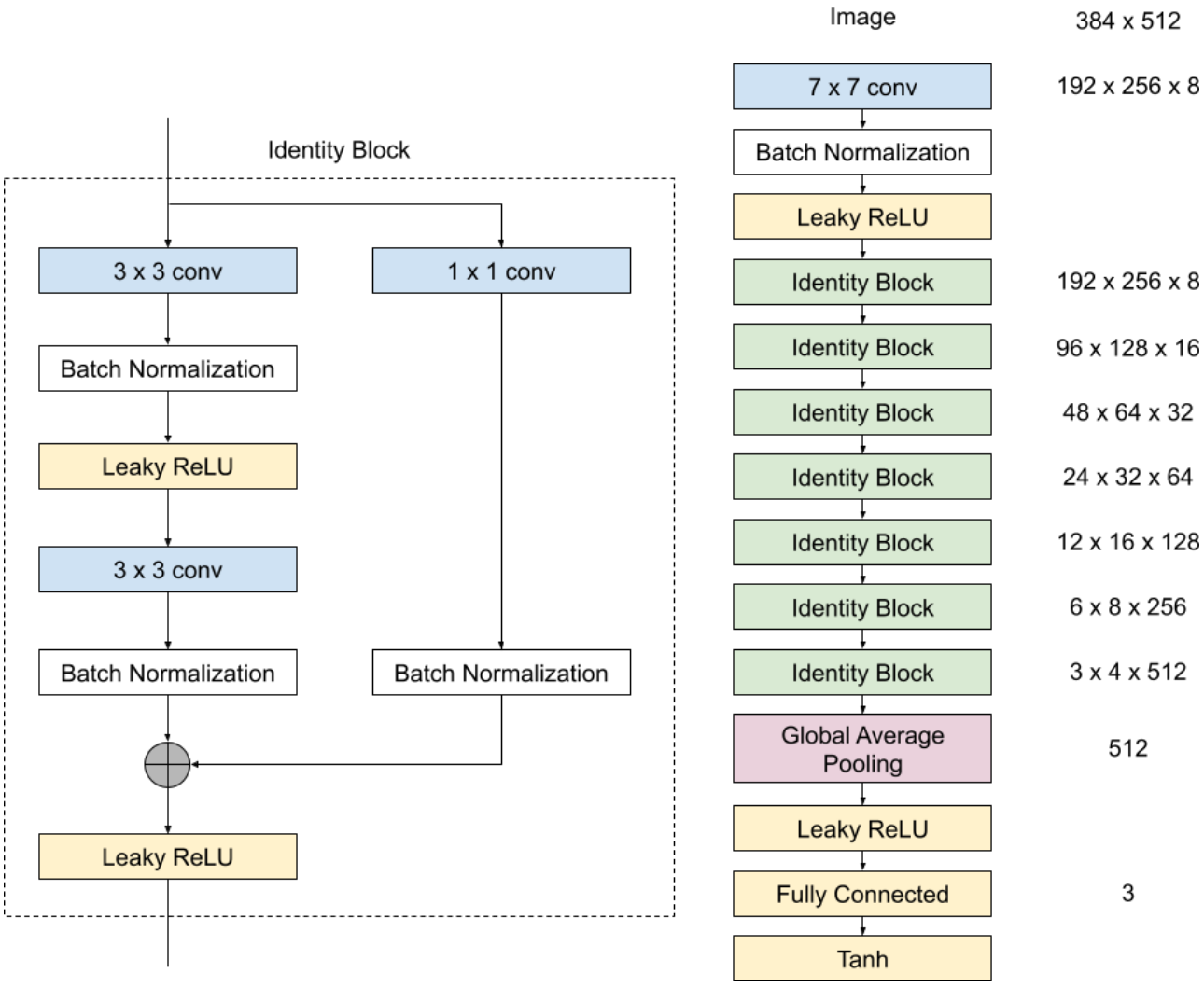


FIGURE 5.1: Target CNN Architecture.

of the DSPs. In [Qas+21], the authors present the results of benchmarking vision kernels and neural network inference on embedded CPU, GPU, and FPGA. The results show that while easy-to-parallelize vision kernels perform well on the GPU, the FPGA is considered more efficient for a complete vision pipeline, where the FPGA was between 2.1× and 2.9× faster and between 1.1× and 2.4× more energy efficient than the GPU. In [Lv+20], the authors propose a dynamic reconfigurable CNN accelerator, reconstructing the different subgraphs based on the partition of the model task graph. The results show 30.35 GOP/s was obtained, with an increased throughput of around 1.2× compared to a non-dynamic reconfigurable accelerator, while suffering an increase in latency of 1.5×. In [Mel+18], the authors presented $\text{NEURA}_{\text{ghe}}$, a CNN acceleration for the Zynq SoCs. Provided by a convolution engine and programmable softcore processor, releasing the ARM processor from its management duties, dedicating it to a more cooperative acceleration of tasks. Experimental results show an improvement in the state-of-the-art performances on well-known CNN networks. Similar studies can be found in [Guo+18; BA19].

[Qas+21] Qasaimeh et al., "Benchmarking vision kernels and neural network inference accelerators on embedded platforms"

[Guo+18] Guo et al., "Angel-Eye: A Complete Design Flow for Mapping CNN Onto Embedded FPGA"

[BA19] Bachtiar and Adiono, "Convolutional Neural Network and Maxpooling Architecture on Zynq SoC FPGA"

TABLE 5.1: Zynq UltraScale+ vs Versal ACAP DPU.

Features	DPU Series	
	Zynq UltraScale+	*Versal ACAP*
Processor Core	PL	AIE
Frequency Domain	1	2
Data Channel	HP Ports	NoC
Batch Handler	N/A	Available

5.3 APPLICATION AND TARGET CNN

The neural network used for this evaluation has been trained to predict the RF phases (the RF gun, A1, and AH1 cryomodules) from the longitudinal phase-space image at the European XFEL photoinjector. Details about the experiment can be found in [Zhu+21].

[Zhu+21] Zhu et al., "High-Fidelity Prediction of Megapixel Longitudinal Phase-Space Images of Electron Beams Using Encoder-Decoder Neural Networks"

Figure 5.1 displays the target CNN architecture. The CNN receives a single channeled input image of size 384×512 pixels and outputs a vector of length 3. The network has 7 similar identity blocks, each combining the output of a (1×1)-convolution residual block with the output of two stacked (3×3)-convolutions with Leaky ReLU activation. The total number of parameters is 13299195, of which 13293083 are trainable.

5.4 DATA PROCESSING UNITS

A Data Processing Unit (DPU) is a programmable processor optimized for data processing. There exist a variety of DPUs specialized for operations related to deep neural networks. The Xilinx IP library offers a parameterizable set of IP cores, providing an inference acceleration platform. During the compilation process, the quantized model is converted to DPU-specific instructions.

In this study, two DPU families have been used: one available for the Zynq UltraScale+ architecture and the other available for the Versal ACAP platform. Table 5.1 highlights some of the key differences between the two DPUs.

The DPU for the Versal ACAP AI series is realized by grouping the Artificial Intelligence Engines (AIE), 32 bit VLIW processors, designed specifically for machine learning and artificial intelligence workloads, while in the earlier devices, it is realized by the Programmable Logic (PL). Additionally, the Zynq UltraScale+ DPU works in one clock domain (usually 333 MHz), while the ACAP decouples the processing and data handling into two separate domains (1.3 GHz and 333 MHz, respectively). Also, while the instructions and data on the Zynq UltraScale+ pass through the DRAM controller and High-Performance (HP) ports, the Versal ACAP uses a high bandwidth Network on Chip (NoC). Finally, while not available in earlier series, the Versal ACAP DPU encapsulates batch handler cores, allowing higher throughput.

Features	DPU Architecture	
	B4096	*C32B3*
Cores	Dual (PL)	Single (96 AIE)
Batch Handler	0	3
Load/Save parallel	2/2	7/2
LUT	53540	227420
Register	105008	285434
BRAM/UltraRAM	257/0	0/332
DSP	562	779

TABLE 5.2: *B4096* vs *C32B3* DPU architectures.

5.5 EXPERIMENTAL RESULTS AND ANALYSIS

In this section, experimental results are displayed and analyzed, highlighting key points in the performance of the CNN on the Versal ACAP AI series with respect to the earlier Zynq UltraScale+ series, a high-performance CPU, and a GPU.

5.5.1 *Setup*

In this study, Vitis AI 2.0[2] has been used to implement the CNN on both FPGA boards, while the model has been developed and trained using TensorFlow 2.3 [Aba16].

[2] https://github.com/Xilinx/Vitis-AI

[Aba16] Abadi, "TensorFlow: learning functions at scale"

Specifically, the target CPU and GPU in this study were the AMD EPYC Rome 2 CPU and Nvidia GeForce RTX 3090 GPU, respectively. For the FPGA acceleration, the VCK190 board from the Versal ACAP AI series and the ZCU104 board from the UltraScale+ MPSoC series were used. The DPUs on the FPGA platforms were the *C32B3* and *B4096*, respectively. Table 5.2 highlights the key differences between the two DPUs. The *B4096* DPU is considered the most capable DPU architecture on the Zynq UltraScale+, making it a good candidate for performance comparison. The *C32B3* has 3 batch handlers, able to load 7 images and save 2 per each, allowing the *C32B3* DPU to load 21 images and save 6 images at the same time, while the *B4096* can manage only two images at a time, which is reflected in a higher resources utilization for the *C32B3*.

5.5.2 *Methodology*

Several changes have been made to the CNN architecture not only to obtain a compatible model with Vitis AI but avoid any CPU offloads. First, dimension expansion operators had to be removed as they were not supported. Instead, the input dimension has been adjusted, and data reformatted to fit directly into the DPU. Second, activations were changed to LeakyRelu in order to be supported by the quantization tool for DPU execution, strictly, with parameter α equal to 0.1. Any change to the value of α resulted in mapping the activation to the CPU and degrading the overall performance. Third, while the GlobalAveragePoolinglayer2D is supported by the Versal ACAP

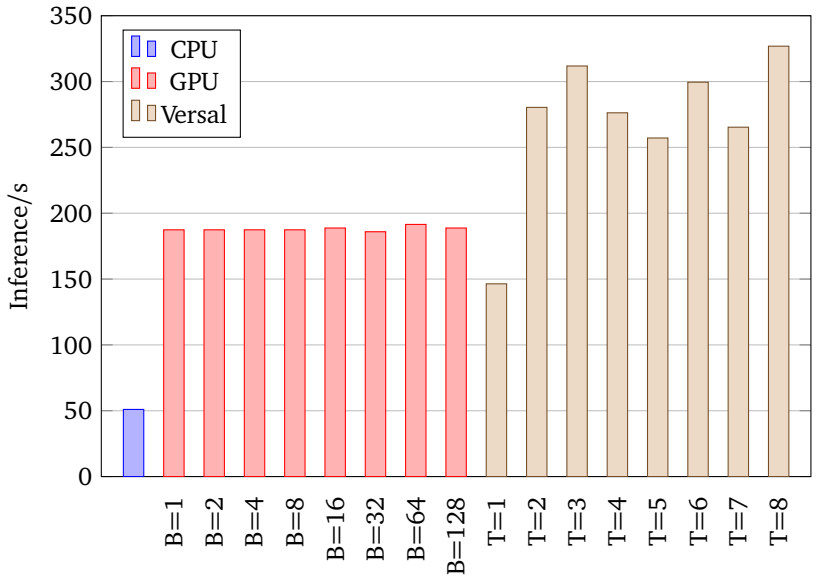


FIGURE 5.2: Versal ACAP vs. GPU vs. CPU throughput (Partially Supported Configuration). B is the batch size, and T is the number of threads.

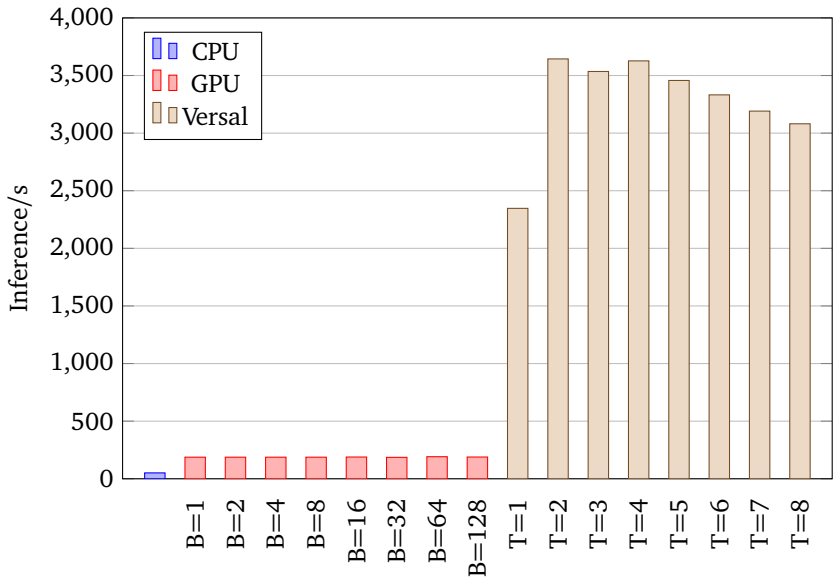


FIGURE 5.3: Versal ACAP vs. GPU vs. CPU throughput (Fully supported Configuration). B is the batch size, and T is the number of threads.

DPU, it is not supported by the Zynq UltraScale+ DPU. To accomplish the global average pooling functionality, GlobalAveragePoolinglayer2D has been replaced with a normal AveragePooling2D layer whose pool size is set to the size of the input feature maps and whose stride length is set to the width of the features. Finally, we stripped the model from the tanh-layer manually after quantization, and the output of the compiled DPU model is post-processed by each of the threads to output the desired result.

On the other hand, two applications were developed to carry the CNN execution on the DPU, single and multithreaded implementations. The single-threaded implementation was used for latency measurements, while the latter was used to collect throughput readings. For the acquisition of the performance readings in both implementations, we used the Vitis AI profiler, which allows for tracing the CNN execution over the DPU. Each reading has been done 10 times to eliminate any jitters in the results and compared to reading from the Python time module for confirmation.

Configuration	Device Latency [ms]			
	CPU	GPU	*B4096*	*C32B3*
Partial Support	51.40	55.68	53.04	24.01
Full Support	50.80	54.30	4.60	2.98

TABLE 5.4: Devices Latency.

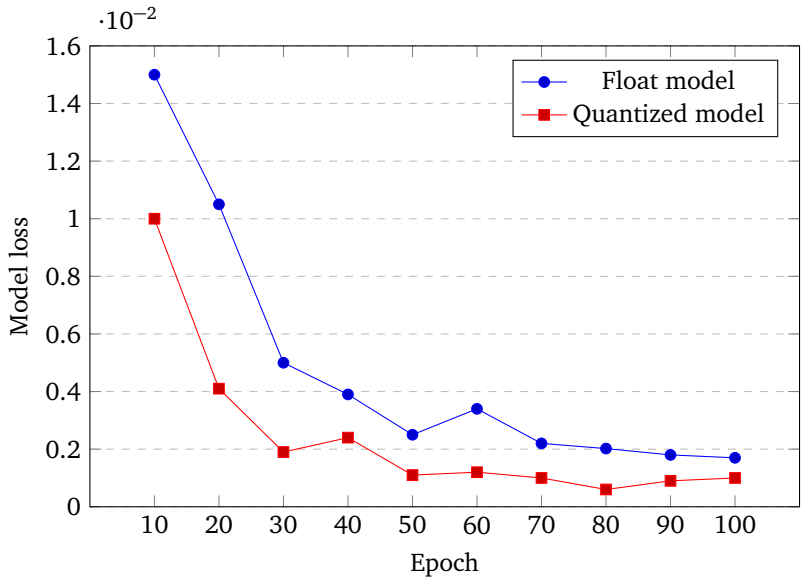


FIGURE 5.4: Training: Float and Quantized Loss ($lr = 3^{-4}$).

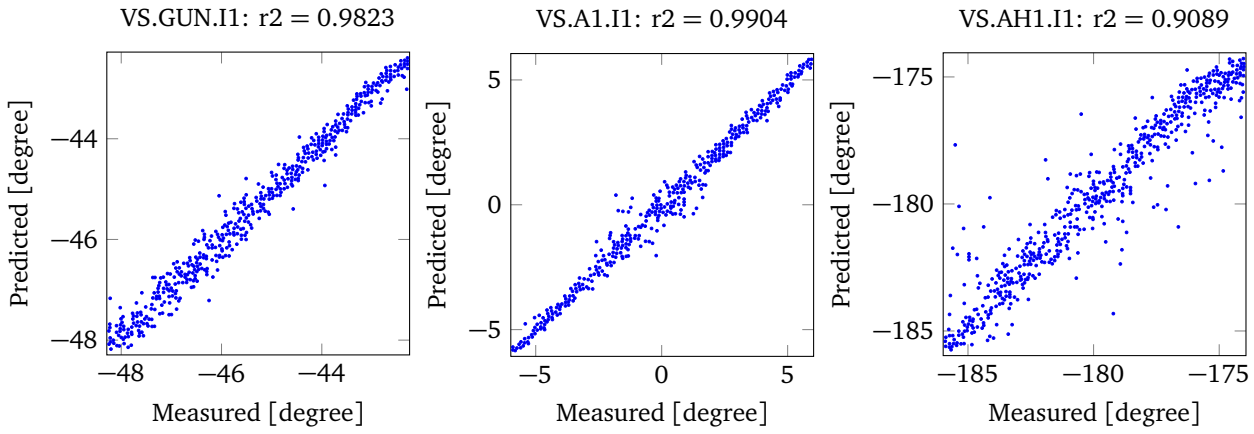
5.5.3 *Throughput*

We first evaluate the performance of the CNN in terms of throughput, i.e., the number of inferences achieved per second. In Table 5.3, the throughput of both the Versal ACAP and Zynq UltraScale+ DPUs is displayed over the deployed number of threads. Both DPUs perform more efficiently in multi-threaded execution (7 and 8 for the Zynq and Versal, respectively), and the Versal ACAP achieves about 3.9× the throughput. When compared to the CPU and GPU, the Versal achieves about 5.2× and 1.8× higher throughput, respectively. Note that the drop in multithreaded performance is due to the overhead, which has to be managed by a dual-core CPU. Figure 5.2 shows the CPU and GPU performance in comparison to the Versal device. The reason behind the lower-than-expected results for the Versal board is the existence of the unsupported activation layer in the CNN model. Such implementations suffer the fact that now the CPU integrated into the Versal board is the bottleneck of the execution, hindering the faster and more efficient DPU.

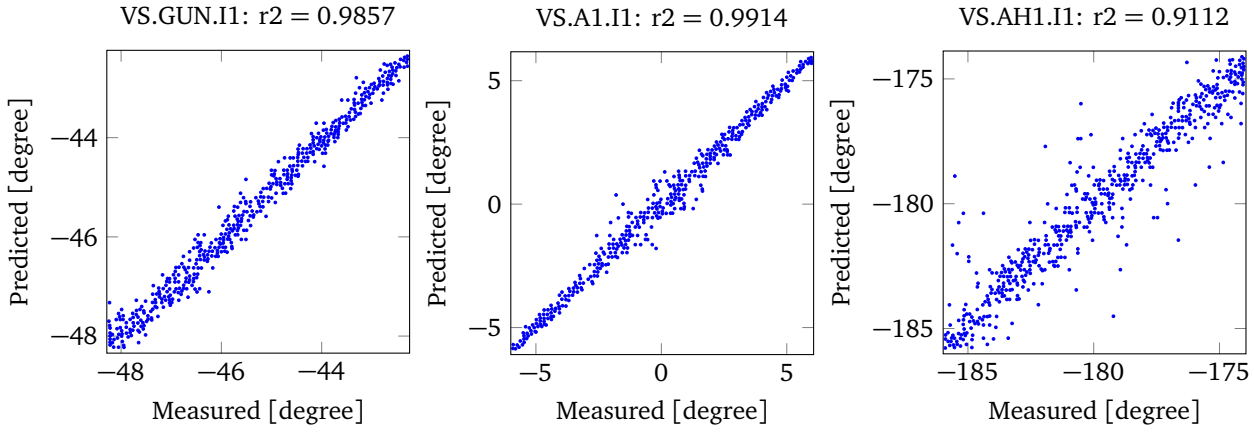
In order to expose the true capabilities of the Versal DPU, we performed new tests after eliminating the tanh-layer. Figure 5.3, shows the performance of the CPU, GPU, and Versal ACAP when the operators are fully supported by the DPU. While the performance of the CPU and GPU were almost not affected by the removal of the activation layer, the Versal ACAP DPU performance was greatly improved, as the throughput performance is now 53.35× and 17.39× and better than the CPU and GPU, respectively.

TABLE 5.3: Versal AI vs. Zynq UltraScale+ throughput.

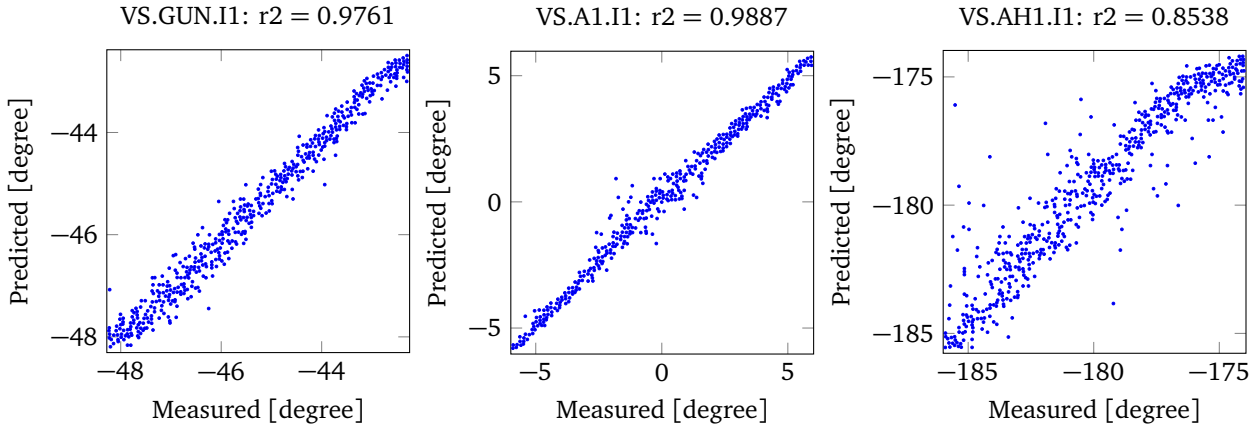
Number of threads	Device	
	B4096	*C32B3*
1	20.73	146.76
2	41.12	280.79
3	61.30	312.17
4	80.78	276.63
5	93.99	258.11
6	71.71	300.43
7	107.81	266.03
8	101.82	327.83



(a) Float Model



(b) Quantized Model



(c) Calibrated Model

FIGURE 5.5: Measured vs. predicted values.

5.5.4 *Latency*

Latency is another important evaluation metric for the application of CNNs. Table 5.4 shows the latency of the CPU, GPU, Zynq UltraScale+ DPU, and Versal ACAP DPU for both the fully and partially supported CNN layout.

The Versal ACAP DPU achieved a lower latency in both cases, while the CPU and GPU performance remained almost similar. For the original CNN configuration, the Versal ACAP DPU achieved 2.14×, 2.32×, and 2.21× better performance over the CPU, GPU, and Zynq UltraScale+ DPU, respectively. The speedups were 17.04×, 18.22×, and 1.54× for the fully supported network. These results confirm the aforementioned discussion, where CPU execution forms a bottleneck for inference, either optimized for latency or throughput.

5.5.5 *Network accuracy*

As the average squared difference between the estimated and observed values, the Mean Squared Error (MSE) is the loss function used in the training and evaluation of how accurate the CNN predictions are. Trained with 2400 images, 100 epochs with a learning rate of 3^{-4}, Figure 5.4 shows the loss values during training for both the float (32-bit) and quantized (8-bit) models. Despite the reduction in bit representation, the quantized model did outperform the original model with lower loss across all epochs.

In Table 5.5, the losses of both the original and quantized models are listed with predictions for 600 images. The quantized model yields a more accurate implementation than the float, while the calibrated model suffered the biggest loss. This is expected as the calibration is post-training and is only concerned with the distribution of activation values and requires a small unlabelled set of training data. To visualize the models' accuracy, Figure 5.5 shows the correlations between the predicted and the measured values for each of the three models. The quantized model has minimal loss across the three predicted parameters: the RF gun, A1, and AH1 phases. The R^2 score, also known as the coefficient of determination, is given by

$$R^2 = 1 - \frac{\sum_i^n (y_i - \hat{y}_i)^2}{\sum_i^n (y_i - \overline{y})^2},$$

where y_i are the measured values, $\hat{y}_i$ are the predicted values, and $\overline{y_i}$ are the mean of the measured data. The R^2 scores for the three parameters are 98.57 %, 99.14 %, and 91.12 %, respectively, for the quantized model, resulting the highest. The better performance of the quantized model could be justified by the fact that having a reduced bit representation allows the optimizer to perform better during training.

TABLE 5.5: Float vs. quantized models prediction loss.

Model	Loss $[\cdot 10^{-2}]$
Float	1.3281
Quantized	1.2534
Calibrated	2.0340

5.6 SUMMARY

In this chapter, we show how it is possible to implement a CNN on various hardware accelerators. The main device under test is the new Versal ACAP platform. We evaluated its performance in terms of throughput, latency, and accuracy of both a fully supported and partially supported model. The Versal ACAP did outperform a high-end CPU, GPU, and the Zynq UltraScale+

DPU, and despite being quantized into 8-bits, it provided even more accurate estimates than the float 32-bit model. However, the true peak performance can only be achieved once the target CNN is fully supported by the DPU used. In that case, the Versal ACAP achieved higher throughput and lower latencies than the other hardware accelerators tested.

With these results, we have a complete flow to perform hardware-accelerated anomaly detection using DPUs in FPGAs. The advantage of using a DPU with respect to a dedicated architecture, e.g., the architecture obtained by using hls4ml [Dua+18; Fas21], is that it is possible to easily exchange or retrain the network without having to recompile the entire architecture. The other advantage is that the resource utilization does not scale with the size of the network used. The cost to pay is a higher latency.

In this part, we have discussed and compared various techniques for performing anomaly detection. We have also shown specific problems that can be solved with anomaly detection. We have finally given a possible solution for the hardware acceleration of time-series anomaly detection. Future work in anomaly detection for fault detection will tackle the acceleration of time point anomaly detection.

In the next part, we will describe a set of techniques that can be combined to derive runtime monitors from existing hardware designs. This is very useful to integrate fault detection capabilities into existing designs without duplicating or triplicating the hardware necessary.

[Dua+18] Duarte et al., "Fast inference of deep neural networks in FPGAs for particle physics"

[Fas21] FastML Team, *fastmachinelearning/hls4ml*

Part III

RUNTIME MONITORING OF FIRMWARE COMPONENTS

6

Syntax-Guided Enumeration of Temporal Properties

"Any fool can know. The point is to understand."
—Albert Einstein

▸ SYNOPSIS This chapter discusses the automatic extraction of temporal properties from existing designs. It is based on the papers

> Görschwin Fey, Tara Ghasempouri, Swen Jacobs, Gianluca Martino, Jaan Raik, and Heinz Riener. "Design Understanding: From Logic to Specification". In: *IFIP/IEEE International Conference on Very Large Scale Integration, VLSI-SoC 2018, Verona, Italy, October 8-10, 2018*. IEEE, 2018, pp. 172–175. DOI: 10.1109/VLSI-SoC.2018.8644732.

and

> Gianluca Martino and Görschwin Fey. "Syntax-Guided Enumeration of Temporal Properties". In: *Forum for Specification and Design Languages (FDL)*. ed. by Tom J. Kazmierski, Reinhard von Hanxleden, and Terrence S. T. Mak. IEEE, 2019, pp. 1–8. DOI: 10.1109/FDL.2019.8876892.

This is the first chapter of Part III. The technique described in this chapter represents the first step in the flow that leads to the automatic generation of runtime monitors from existing designs. We first discuss the challenges of design understanding. Then, we present the main works done in the area of design understanding existing in the literature and describe their limitations. Finally, we give some preliminary concepts and describe our approach. The experiments performed with commonly used benchmarking design validate the general effectiveness of the approach.

6.1 INTRODUCTION

Deriving a set of properties from an existing hardware design has multiple possible applications:

1. *Reverse engineering*: When the intent of a design is not known or its Register-Transfer Level (RTL) specification got lost over the years, making sense of a gate-level description (particularly after heavy optimization) is often impossible for humans. Automated design under-

standing tools have the potential to isolate a few core properties that give an idea of the overall usage of the design.

2. *Debugging*: Finding and fixing bugs in a design is often a complicated and time-consuming task. Automatically generated properties can assist in spotting faults more quickly, e.g., when the reported properties slightly deviate from expectations.
3. *Verification*: For certain designs, equivalence checking requires considerable effort. In these cases, checking simple automatically generated properties may still be possible due to property-specific abstraction mechanisms. Property-checks can then be used as a lightweight approach to increase the confidence in the correctness of the design or to pinpoint functional differences.
4. *Runtime Monitoring*: When systems become too large for verification to be a practical approach, runtime monitoring allows us to ensure the correctness of the system by only verifying the current execution, which avoids scalability issues.

Moreover, often a designer is confronted with an unknown design, e.g., due to reusing legacy blocks, taking over from a colleague, or using third-party code. Then, reverse engineering, debugging, verification, regression testing, or understanding this unknown block are typical tasks [Ray+16; Fey+18].

[Ray+16] Ray et al., "Multilevel design understanding: from specification to logic"

[Fey+18] Fey et al., "Design Understanding: From Logic to Specification"

Design understanding is the process of reconstructing human understandable knowledge from an unknown design. This problem arises frequently in practice, e.g., when a part of a design fails or behaves unexpectedly and needs to be debugged, but the initial designer of this part left the design team. In such a situation, the documentation of the design, which is still under development, is often not available, still incomplete, or deviates from the implemented behavior. Automated design understanding tools can then be used to assist a human in gaining an understanding of the design and implementing the required changes. Overall, automated design understanding has great potential to improve fault localization, reduce time-to-market, and improve product quality. However, design understanding is also a fuzzy process and until today no commonly accepted formalism, notation, or methodology for extracting design knowledge from an unknown implementation exists.

We argue that *properties* written in a formal language—the standard formalism in formal verification—are not only useful to describe the input-output semantics of a system, but also capable of describing the interior behavior of a design in a human-understandable manner. Particular temporal logics, that describe relationships between signals of a circuit implementation over time, are attractive. We envision that a design understanding tool derives knowledge in form of temporal-logic formulæ from an existing design.

[Pnu77] Pnueli, "The Temporal Logic of Programs"

[CE81] Clarke and Emerson, "Design and Synthesis of Synchronization Skeletons Using Branching-Time Temporal Logic"

[EH86] Emerson and Halpern, ""Sometimes" and "Not Never" revisited: on branching versus linear time temporal logic"

[HBS12] Hassan et al., "Incremental, Inductive CTL Model Checking"

Specifying properties in temporal logic for defining the behavior of a digital hardware block is a well-known approach [Pnu77; CE81; EH86]. Sophisticated procedures decide whether a given property holds on a block or not [HBS12]. Once such properties are available, they precisely describe how the block works. This supports a designer in understanding the block and in reverse engineering. For regression testing known properties of old versions of the block can identify which parts of the specification changed.

This knowledge then helps in debugging.

Temporal logic is an excellent formalism for expressing such specifications of a system. Temporal logic properties specified in Linear Temporal Logic (LTL) specify guarantees over execution paths. Temporal operators describe properties of a path through the state space of a system. Computation Tree Logic (CTL) properties are powerful in specifying invariants and other types of guarantees. CTL describes properties of *computation trees* and extends Boolean logic by temporal operators and by path quantifiers **A** and **E**. Temporal operators and path quantifiers describe properties of a path through the tree. The addition of path quantifiers changes completely the semantics of the formulæ with respect to LTL, creating a temporal logic of incomparable expressive power. For example, the CTL formula **AFAG**φ cannot be expressed in LTL and, vice versa, there exist LTL formulæ, such as **FG**φ, which cannot be expressed in CTL [BK08].

[BK08] Baier and Katoen, *Principles of model checking*

However, the expressivity of temporal logic poses additional problems. There can be a gap between the formal semantics and the intended meaning of the specification as understood by a human designer. This can lead to difficulties in accurately reflecting the designer's intent and can also make it difficult for a human to understand and correctly write the specification. One common issue is the consideration of a system's behavior in communication with other systems or as a component of a larger system, which requires the specification to include assumptions about the other components. This can be challenging due to the standard interpretation of temporal logics like LTL, which assumes a worst-case scenario and can require overly specific specifications. Automated methods for generating a specification for a given system can also present challenges, as the specification may accurately reflect the system's behavior but be misinterpreted by the human designer. In order to improve design understanding and facilitate the use of formal languages in specification, it may be necessary to consider the perceptions and understanding of the human designer in addition to the formal semantics. Creating predefined templates for properties with clear meanings, using a simplified grammar, and setting limits on the size of formulas using standard measures such as length and number of subformulas can make formulas easy to understand. Additionally, more complex methods such as bounding the formulas under more complex metrics that correlate with human understanding [GP15] may also be used to ensure that formulas are easily understood. Vacuity detection [KV03] can also help address these issues.

[GP15] Ghasempouri and Pravadelli, "On the estimation of assertion interestingness"

[KV03] Kupferman and Vardi, "Vacuity detection in temporal model checking"

We propose Syntax-Guided Property Enumeration, a technique for automatically obtaining a set of short and readable temporal logic properties. The seminal idea was presented in [MRF18].

[MRF18] Martino et al., "Coverage-Guided CTL Property Enumeration for Understanding Models of Reactive Systems"

The set of properties found is complete up to a given length of the property, i.e., all temporal properties the block fulfills are found. A naïve algorithm would simply enumerate all possible formulæ according to the syntax of the temporal logic. Instead, we present an advanced algorithm including several tweaks to significantly improve the run time. Syntactic filtering removes trivial properties. The enumeration algorithm intelligently avoids obvious equivalences, e.g., due to commutativity of operators. Structural hashing and simplification eliminate further simple equivalences. Trace checking uses random simulation to efficiently falsify properties that do not

hold on the block. Finally, we apply semantic filtering based on vacuity checking [KV03] to improve the set of properties generated. The overall approach is motivated by the recent success of Syntax-Guided Synthesis (SyGuS) [Alu+15] and extends the idea to temporal logic including the proposed optimizations.

[KV03] Kupferman and Vardi, "Vacuity detection in temporal model checking"

[Alu+15] Alur et al., "Syntax-Guided Synthesis"

Our approach is unique in being able to use the full grammar of the chosen language without any restrictions. No specific simulation traces are required to find meaningful properties, only random simulation is used. By relying on enumeration we generate human-readable specifications. Our approach tightly combines concepts from syntax-guided synthesis with model checking and powerful syntactic and semantic simplification procedures.

The remaining sections of the chapter contain an overview of the main works done in the area of design understanding in Section 6.2, some preliminary concepts and the details of the proposed technique in Section 6.3 and Section 6.4, and the results of the experiments in Section 6.5. In Section 6.6, we discuss possible approaches for reducing the set of properties, and in Section 6.7, we conclude the chapter and give details about the use case in the overall framework and fault detection system.

6.2 RELATED WORK

Specification mining as an approach to infer knowledge of unknown designs has some history. Table 6.1 reports a brief comparison of selected approaches for query checking, specification mining, and our approach. Most of the approaches target automata descriptions that rather model hardware while [ABL02; Ern+07; Sho+08] directly address software. Daikon [Ern+07] only mines non-temporal Boolean expressions or software assertions, respectively. Specification mining of [ABL02] targets concurrent software and turns traced information into representative automata. The approach of [Sho+08] statically analyzes software to find automata modeling Application Programmer Interfaces (APIs); the complexity of software and consideration of API characteristics restrict the capabilities of the approach. GoldMine [HSV13] finds properties that argue over finite time windows. Similarly, [LFS10] starts from such properties but composes them into more complex automata. Inferno [DeO+09] specifically addresses interface descriptions and generalizes them to transactions. The approach in [MFF17] starts from dynamic dependency graphs to find properties. Almost all of these approaches use certain restrictions on the syntax of the specifications to be generated and start from simulation traces for (data) mining. Back-ends often rely on model checking to validate mined specifications, only [Sho+08] uses it as the core engine.

[ABL02] Ammons et al., "Mining Specifications"

[Ern+07] Ernst et al., "The Daikon system for dynamic detection of likely invariants"

[Sho+08] Shoham et al., "Static Specification Mining Using Automata-Based Abstractions"

[HSV13] Hertz et al., "Mining Hardware Assertions With Guidance From Static Analysis"

[LFS10] Li et al., "Scalable specification mining for verification and diagnosis"

[DeO+09] DeOrio et al., "Inferno: Streamlining Verification With Inferred Semantics"

[MFF17] Malburg et al., "Property Mining using Dynamic Dependency Graphs"

This brief review of related work is not complete but gives representatives for major directions.

TABLE 6.1: Approaches for specification mining.

	HW	SW	Boolean	temporal		syntactic	simulation-	formal
				finite	infinite	restrictions	guided	
Daikon [Ern+07]	-	x	x	-	-	x	x	-
Specification mining [ABL02]	-	x	x	x	x	x	x	-
API [Sho+08]	-	x	x	x	x	x	-	x
GoldMine [HSV13]	x	-	x	x	-	x	x	-
Automata-based [LFS10]	x	-	x	x	x	x	x	-
Inferno [DeO+09]	x	-	x	x	x	x	x	-
Dyn. Dependencies [MFF17]	x	-	x	x	x	x	x	-
Proposed approach	x	-	x	x	x	-	-	x

- HW/SW: Approach targets hardware or software
- Boolean: Approach generates Boolean predicates
- Finite: Approach generates invariants that argue over a fragment of a path with a fixed number of transitions
- Infinite: Approach generates temporal specifications that argue over infinite paths
- Syntactic restrictions: Syntax is restricted for the mining process
- Simulation-guided: Starts from simulation traces to find specifications
- Formal: Mining engine uses a formal approach (most approaches use model checking for post-processing only)

6.3 BACKGROUND

A *model checking procedure* checks if a given Kripke structure M models a temporal property φ and returns true iff $M, S_0 \models \varphi$.

We use the terms formula and property to distinguish between, respectively, a general temporal formula and a property of a Kripke structure M expressed in temporal logic. We write $\varphi[\gamma/\psi]$ to denote the temporal logic formula φ where all occurrences of ψ have been textually replaced by γ.

Our definition of vacuity uses the generalized definition by Kupferman and Vardi [KV03]. Suppose that M is a Kripke structure and ψ is a sub-formula of φ. If a strengthening $\varphi[\bot/\psi]$ of φ satisfies M, then ψ is irrelevant for the satisfaction of φ in M, where the substitution is assumed to respect the polarity of the sub-formula.[1] In such a case, we say φ is *vacuously-satisfied* in M and call ψ a *witness* for the vacuity of φ. Otherwise, we say φ is *nonvacuously-satisfied* in M. Testing every possible strengthening of all sub-formulæ to decide vacuity is typically impractical. Efficient vacuity detection procedures, e.g., for CTL [PS02], have been proposed that can be carried out with a small overhead.

[1] We substitute sub-formulæ in positive polarity with $\bot$ and sub-formulæ in negative polarity with $\top$.

[PS02] Purandare and Somenzi, "Vacuum Cleaning CTL Formulae"

Syntax-guided synthesis (SyGuS) [Alu+15] is a recently proposed framework for program synthesis. The SyGuS problem asks, given a context-free grammar $\mathcal{G}$ and a specification in form of a logic formula Γ over uninterpreted symbols Ψ, for finding syntactic expressions $\varphi \in \mathcal{G}$ such that $\Gamma[\varphi/\Psi]$ holds for all possible assignments to the free variables in Γ, where Γ and $\mathcal{G}$ are formulated over a fixed background theory. A SyGuS problem can have multiple solutions; a solver for a SyGuS problem is only required to provide one solution.

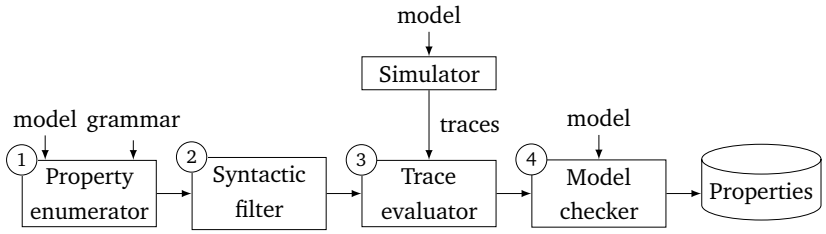


FIGURE 6.1: Overview of the property generation process.

6.4 SYNTAX-GUIDED ENUMERATION

6.4.1 *Overview*

We use a Kripke structure M to model a gate-level design. A context-free grammar $\mathcal{G}$ describes how syntactically valid formulæ are constructed for a temporal logic (or some fragment of a temporal logic).

We generalize the idea of Syntax-Guided Synthesis (SyGuS) in the context of temporal logic and model checking. For a given Kripke structure M, we generate a list $\varphi_1, \varphi_2, \ldots, \varphi_n$ of temporal logic formulæ that are satisfied by M, i.e., $M \models \varphi_i$ for $1 \leq i \leq n$, and obey to certain syntactic rules. The formulæ are generated by unwinding a context-free grammar $\mathcal{G}$ and model checked on M. Satisfied formulæ are kept and reported to a user, failing formulæ are discarded. An additional termination criterion, e.g., in form of a time limit or an upper bound on the maximum length of formulæ, is required to guarantee termination.

Given a Kripke structure M, a context-free grammar $\mathcal{G}$ and some termination criterion, the proposed method for property generation works as illustrated in Figure 6.1:

1. The process starts from the property enumerator. The enumeration requires as inputs the grammar $\mathcal{G}$ and the Kripke structure M. As a result, the property enumerator block outputs temporal formulæ until either all the search space obtained from the current grammar is explored or the termination criterion is reached.

2. Every formula generated passes through the syntactic filter. If the formula does not comply with the syntactic rules, it is discarded.

3. The remaining formulæ are checked against traces: the trace evaluator tries to find a counterexample of the formula in the database of traces; the property is discarded if the counterexample is found.

4. We finally employ the model checker to obtain only the properties that hold on M.

6.4.2 *Property enumeration*

For formula generation, we maintain a priority queue of non-concrete expressions sorted by length, i.e., the number of nodes of the DAG representing the formula. We call an expression *non-concrete* if it contains one or more non-terminal symbols and otherwise *concrete*. Initially, the queue contains only a single expression, i.e., a non-terminal symbol, which is the starting symbol of the context-free grammar. We then choose the first expression

in the priority queue, remove it from the queue, and compute the shortest path from the expression's root node to a non-terminal symbol N in the expression. We replace the non-terminal symbol N by α, for each grammar rule of form $N ::= \alpha$ in $\mathcal{G}$. If a newly generated expression does not contain non-terminal symbols, the expression is passed to the formula checking step. If the expression contains one or more non-terminal symbols, the expression is added to the priority queue, and the queue is resorted.

Example 1. $\mathcal{G} : \Psi ::= \bot \mid p \mid q \mid (\neg\Psi) \mid \mathbf{AG}(\Psi)$ is a context free grammar with start symbol Ψ. The first derived candidate formulæ look as follows:

$$\bot \mid p \mid q \mid (\neg\bot) \mid (\neg p) \mid (\neg q) \mid \mathbf{AG}(\bot) \mid \ldots$$

6.4.3 *Syntactic filtering*

Many syntactically different CTL formulæ express the same temporal relation. We employ *structural hashing* together with Boolean and temporal logic simplifications to identify and discard some functionally equivalent temporal logic formulæ during enumeration. We call this step *syntactic filtering* because only the structure of the temporal logic formulæ is analyzed.

STRUCTURAL HASHING Each temporal logic formula is represented by a rooted *directed acyclic graph* (DAG) with inner nodes and leaf nodes. Each leaf node is one element of the set $\{\top, \bot\} \cup \Sigma_O \cup \Sigma_L$, where Σ_O and Σ_L are respectively the set of output signals and the set of latch signals of the design under analysis.

Each inner node is labeled with a Boolean or temporal operator and has, depending on the operator in use, one or two children. Every tree node is associated with a number which can be used to identify a temporal logic formula and defines an order on temporal logic formulæ.

SHARING We use one multi-rooted shared formula DAG for representing all temporal logic formulæ. Common sub-expressions encountered during enumeration are shared within this graph structure. As a special case two structurally equivalent temporal logic formulæ are represented by the same node in the DAG (instead of two isomorphic nodes). Boolean and temporal logic simplification rules identify equivalent sub-expressions and improve graph sharing. An example is shown in Figure 6.2.

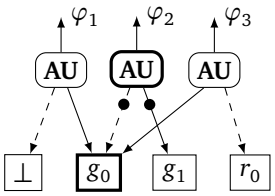


FIGURE 6.2: Shared DAG.

SIMPLIFICATIONS Filtering is achieved by detecting possible simplifications in the formulæ enumerated. We distinguish two levels of simplifications based on when they are applied: deduction and logic simplifications. Deduction simplifications are applied while a new expression is deduced from the grammar. At this stage, the expression still contains non-terminal symbols that are about to be concretized. Logic simplifications are applied to concrete expressions that have been deduced from the grammar already. In both cases, the formula that can be simplified is simply discarded. This is possible because the enumeration guarantees that the simpler version is checked before.

The simplification rules that we use are the following:

1. Double application: formulæ containing double applications of operators which can be applied multiple times, producing equivalent formulæ each time, are immediately recognized and discarded, e.g., the LTL temporal operators **G**, **F** or the CTL temporal operators **AG**, **AF**, **EG**, **EF**.

2. Idempotence and commutativity: the operands of the binary operators $\wedge$ and $\vee$ must be sorted and unequal; the operands of the binary operators **AU**, **EU**, and **U** must be unequal.

3. Boolean constant propagation: simplifications due to Boolean constants are immediately applied, e.g., the formula $\bot \wedge \Psi$ with some non-terminal Ψ is simplified to $\bot$.

4. Temporal simplifications: temporal logic simplifications that reduce the length of the number of applied operators are immediately applied. Suppose that Ψ and Ω are non-terminals. Some of the rewriting rules for CTL are the following:

   - $\mathbf{AF}(\mathbf{AG}(\mathbf{AF}(\Psi))) = \mathbf{AG}(\mathbf{AF}(\Psi))$,
   - $\mathbf{AG}(\mathbf{AF}(\mathbf{AG}(\Psi))) = \mathbf{AF}(AG(\Psi))$,
   - $\mathbf{AG}(\Psi) \wedge \mathbf{AG}(\Omega) = \mathbf{AG}(\Psi \wedge \Omega)$,
   - $\mathbf{A}(\Psi\mathbf{UA}(\Psi\mathbf{U}\Omega)) = \mathbf{A}(\Psi\mathbf{U}\Omega)$.

   Some of the rewriting rules for LTL are the following:

   - $\mathbf{F}(\mathbf{G}(\mathbf{F}(\Psi))) = \mathbf{G}(\mathbf{F}(\Psi))$,
   - $\mathbf{G}(\mathbf{F}(\mathbf{G}(\Psi))) = \mathbf{F}(\mathbf{G}(\Psi))$.

TRACE EVALUATION A counterexample is a Kripke structure $\tilde{M} \subseteq M$ which contains only the necessary states to demonstrate in a rigorous manner how $M \not\models \varphi$. Such a Kripke structure $\tilde{M}$ can be reduced to an execution trace if the formula φ is an LTL formula or a CTL formula which is also expressible in LTL [CV03].

[CV03] Clarke and Veith, "Counterexamples Revisited: Principles, Algorithms, Applications"

For this reason, it is possible to use execution traces to disprove temporal logic formulæ. We can also prove that a formula φ satisfies a Kripke structure M using traces thanks to the fact that if $M \not\models \varphi$ then $M \models \neg\varphi$.
Due to the nature of the enumeration process, the properties generated are often not expressing meaningful behaviors of the design M under consideration. For this reason, looking for counterexamples to those properties, among a set of traces is very effective in filtering out large amounts of properties.

In our approach, the traces are generated once before the start of the enumeration process and then used for all the formulæ generated. The usage of random traces is motivated by the necessity to have a fast way to produce an arbitrary number of traces. The experimental results proved that the random traces are indeed very effective in filtering out formulæ.

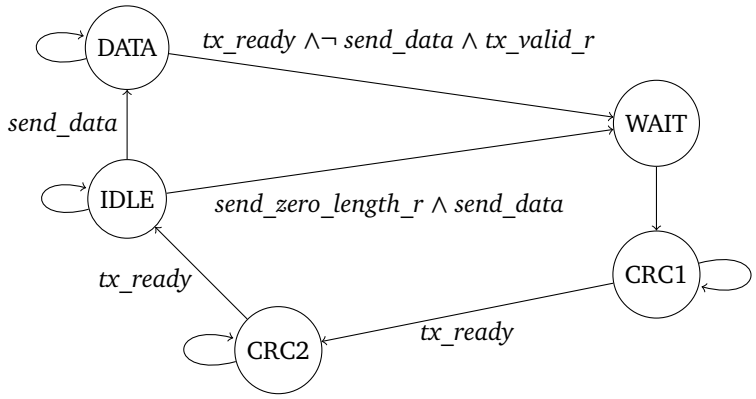


FIGURE 6.3: State transition relations of the Packet Assembler module.

Condition	Output
tx_ready ∧ tx_valid_r ∧ state=DATA	rd_next
tx_valid_d	tx_valid
(send_token ∨ send_data) ∧¬ tx_first_r	tx_first
send_token ∨ last	tx_valid_last

TABLE 6.2: Output's behavior of the Packet Assembler module.

6.5 EVALUATION

Our implementation of the syntax-guided enumeration tool supports gate-level designs and generates CTL properties, LTL properties or any restriction of the two specification languages. The choice of the model checker, used for the model checking step, is tied to the design representation used and to the properties generated. In our implementation, we support VIS [Bra+96], for both LTL and CTL properties, and IIMC [HBS12] for CTL properties. We implemented our property mining tool in C++ using the enumeration library behemoth[2] and existing model checkers as building blocks. The termination criterion of this implementation is the maximum length of the formula, also denominated cost.

[Bra+96] Brayton et al., "VIS: A System for Verification and Synthesis"

[HBS12] Hassan et al., "Incremental, Inductive CTL Model Checking"

[2] behemoth, `https://github.com/hriener/behemoth`

6.5.1 *Case study*

In the first experiment, we perform a case study to evaluate the technique's behavior when confronted with a design understanding task. The case study helps in comprehending how our tool compares to existing tools when generating properties to reconstruct knowledge about a design.

For this case study, all tests were conducted on a laptop equipped with a CPUs Intel(R) Core(R) i7 operating at a frequency of 2.9 GHz and 32 GB of RAM.

In the case study, we compare it to the GoldMine approach [HSV13]. In this experiment we use the PRISM miner of GoldMine, being the approach that gives the best results in terms of number of properties generated. We report execution time, number of properties generated, and the completeness of the set of properties generated with respect to the specification of the module under analysis.

[HSV13] Hertz et al., "Mining Hardware Assertions With Guidance From Static Analysis"

The design chosen for the case study is the Packet Assembler module, which is part of the protocol layer of the USB 2.0 core. This block assembles

TABLE 6.3: Summary of the comparison with GoldMine[HSV13].

	Properties	Time [s]	Behavior reconstructed [%]
SGPE*	75784	2520	70
GoldMine	17	192	44

* Syntax-Guided Property Enumeration

packets and places them into the output FIFO. It first assembles the header, inserting a proper PID and checksums, then adds a data field if requested. The block is implemented as a finite state machine containing 5 states. The state transition relations are described in Figure 6.3. The behavior of the output signals of the system is described in Table 6.2. This design is one of the examples distributed with GoldMine.

The execution of GoldMine on the design returns a set composed of 17 SVA properties. The properties are returned in 192 seconds excluding the time needed for model checking.

With the properties obtained, we were able to reconstruct 44% of the information contained in Figure 6.3 and in Table 6.2. The properties obtained allow recreating the full table for output behavior but do not allow recreating the state transition table.

For the syntax-guided enumeration of properties, we used the LTL back-end in order to be able to compare the set of properties obtained with the SVA properties obtained by GoldMine. The cost was set to 5 and the properties were generated in negation normal form.

The summary of this comparison is presented in Table 6.3. With the properties obtained, we were able to reconstruct 70% of the information. The execution of our tool produces 75784 properties, 24749 of which contain at least a temporal operator. The result is obtained in 2520 seconds but all properties returned have been already model checked. With the properties obtained, we can partially reconstruct both the state transition table and the output table. The only relations not present were the ones containing a condition with more signals than the enumeration can reach due to the termination criterion. Additionally, the set of properties contains also properties explaining the behavior of other registers of the module.

Based on the results, we can conclude that this technique is very effective at generating short properties. The results can be improved combining this technique with dynamic methods for the generation of longer and specific properties. The readability of the results can be improved using techniques for automatic generation of finite state machines, e.g, [UBS18].

[UBS18] Ulyantsev et al., "Exact finite-state machine identification from scenarios and temporal properties"

6.5.2 *Filtering evaluation*

The second experiment is done to evaluate the effectiveness of each filtering stage of the proposed approach, as well as to understand how a change in the grammar changes the behavior of each filtering stage. For this test, we implemented a proof-of-concept version of the tool which, at the model checking step of the process, invokes multiple instances of the model checker

TABLE 6.4: Summary of the enumeration of ACTL formulæ.

Name	Number of properties in output				Runtime		
	property generator	trace evaluator	model checker	vacuity detector	IIMC only [s]	IIMC + traces [s]	speedup
amba4	117156	70694	40769	3232	1708	1347	1.27
amba8	333696	197075	113191	7876	11705	9033	1.30
b13	215604	125188	86416	7846	5585	3821	1.46
gcd	155700	82118	53504	3537	2787	1904	1.46
rrobin	2100	890	276	17	4	2	2.00
twoQ	477476	255789	177284	13001	24466	15795	1.55
vMiim	605556	355198	218677	15837	34766	28827	1.21

TABLE 6.5: Summary of the enumeration of ACTL invariants.

Name	Number of properties in output				Run time		
	property generator	trace evaluator	model checker	vacuity detector	IIMC only [s]	IIMC + traces [s]	speedup
amba4	105924	39059	1758	1214	1651	872	1.89
amba8	302214	108664	2019	1097	11854	5962	1.99
b13	195146	88070	13289	1659	8167	5530	1.48
gcd	140850	47609	472	158	2782	1413	1.97
rrobin	1850	452	5	5	4	1	4.00
twoQ	432604	153243	10871	399	29659	12992	2.28
vMiim	548774	279215	42983	3470	35678	25423	1.40

in different threads. The maximum number of threads used for the model checking was limited to 64. All tests were conducted on a Linux server equipped with 96 CPUs Intel(R) Xeon(R) Gold 6146 operating at a frequency of 3.20 GHz and 1510 GB of RAM.

For this experiment we compared two different grammars; both are subsets of CTL:

- In the first test, the grammar is composed of the Boolean operators $\neg, \wedge, \vee$ and of the temporal operators **AX**, **AU**, **AG**, **AF**.
- In the second test, the grammar remains the same but every formula is inside an **AG** operator.

We used ACTL in this experiment because it is a widely used subset of CTL, e.g., the specification given for the two AMBA buses used as benchmarks in this experiment are written in ACTL. In both cases, the formulæ are composed in negation normal form. We used an additional filtering stage, denominated vacuity detector, which detects and discards vacuous formulæ. The tests use selected benchmarks presented in Table 6.6.

The benchmarks are taken from the examples provided in AIGER format by IIMC and are selected in order to maintain the variety in terms of complexity of the design and functionalities, but limiting the number of signals.

TABLE 6.6: Benchmark parameters.

Model	I*	O†	L‡
amba4	15	19	27
amba8	27	32	46
b13	10	10	53
gcd	17	8	45
rrobin	2	5	2
twoQ	11	11	68
vMiim	41	28	83

* Number of inputs.
† Number of outputs.
‡ Number of latches.

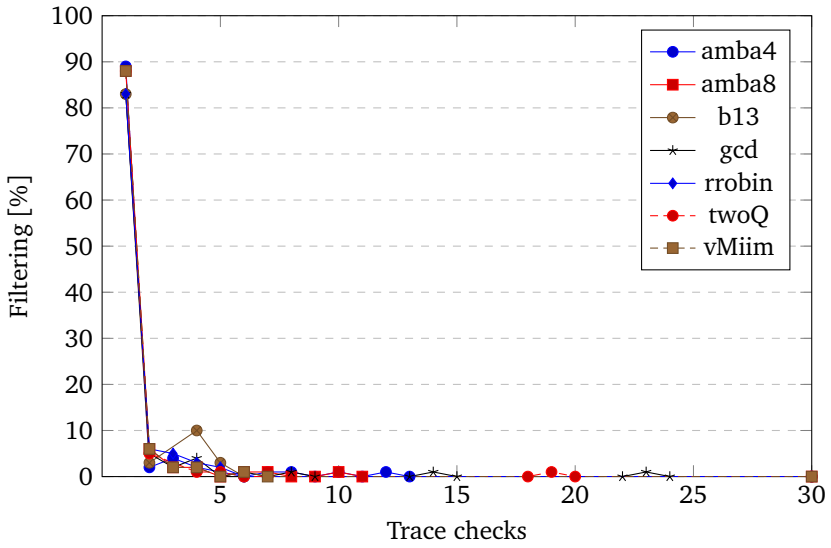


FIGURE 6.4: Percentage of filtered formulæ for each trace.

The signals considered are limited to latched inputs, latches and outputs. For both tests, the cost was set to 4.

In Table 6.4 and Table 6.5 we report the number of properties in output at each step of the process in the section "Number of properties in output" of the tables. The run times and the speedups when using the trace evaluation filter are reported in the run time section of the tables.

The analysis of the results suggests that the trace evaluation step is very good at filtering out properties: the trace evaluation is able to filter out on average 63% of the properties in output of the syntactic filter block in case of ACTL invariants; in case of ACTL the filtering is of 45% on average. Vacuity checking is very effective in case of ACTL but not as much effective in filtering out formulæ in case of ACTL invariants. The average filtering is respectively 93% and 60%.

We measured also how many times each trace was used in the trace checking step and how many times a trace was a counterexample. Figure 6.4 shows how many formulæ, in percentage, each trace filters out. We verified that on average 98% of all filtered formulæ in this step are filtered by the first 10 random traces. Interestingly, using just one trace is enough to filter out an average of 86% of all filtered formulæ.

Using this technique, with the termination criteria chosen, we were able to enumerate 5 out of 18 and 9 out of 34, respectively, of the properties manually written for the two AMBA buses designs present in the benchmarks. Using a higher cost as termination criteria, it is possible to enumerate the entire set of properties of the specification.

From this experiment, we can conclude that the effectiveness of each filtering stage heavily depends on the grammar used. Anyway, this has not a big impact on the run time. The additional filtering step, based on vacuity detection, reduces considerably the number of properties, removing uninteresting properties and improving the readability of the result.

Finally, in Figure 6.5, we show that syntax-guided property enumeration can be effectively multi-threaded.

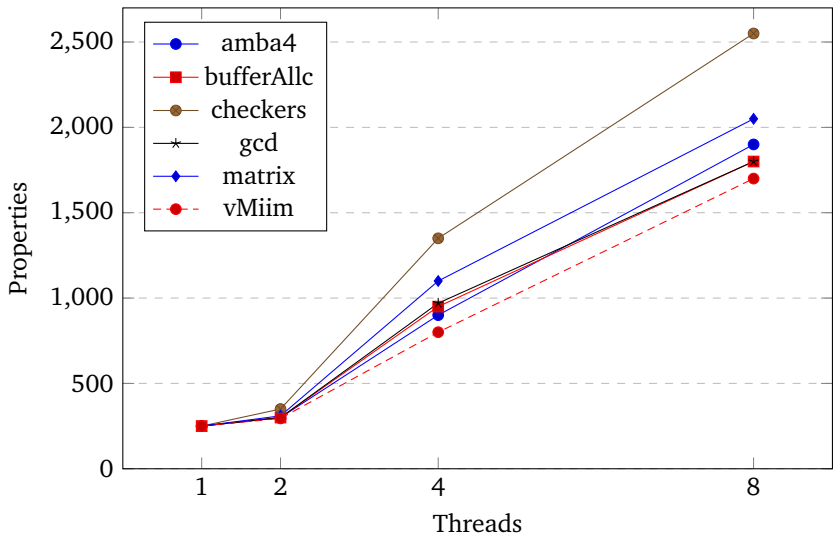


FIGURE 6.5: Multi-threaded invariant generation (time limit: 90s).

6.6 PROPERTY SET REDUCTION

Since the technique proposed produces a large set of properties, a ranking method that can estimate the quality of an assertion is necessary. Multiple ranking techniques exist in literature [FFP07; KGG99; Hos+99; HSV13; GP15]. The proposed ranking techniques use a variety of metrics to estimate the quality of the set of properties. For instance, in [HSV13], an approach has been proposed that estimates the quality based only on the number of propositions. In [GP15], assertion quality is estimated based on their frequencies and correlation during simulation.

The experimental results show that the proposed methodologies allow us to reduce the set of properties without loss of accuracy when fault and code coverage analyses are taken into account.

[FFP07] Fedeli et al., "Properties incompleteness evaluation by functional verification"

[KGG99] Katz et al., ""Have I Written Enough Properties?" - A Method of Comparison Between Specification and Implementation"

[Hos+99] Hoskote et al., "Coverage Estimation for Symbolic Model Checking"

[HSV13] Hertz et al., "Mining Hardware Assertions With Guidance From Static Analysis"

[GP15] Ghasempouri and Pravadelli, "On the estimation of assertion interestingness"

6.7 SUMMARY

In this chapter, we described a novel approach for generating a complete set of temporal properties of a hardware design. The core engine is a syntax-guided enumerator. For this, we generalized the idea of syntax-guided enumeration to temporal logic and model checking. Various technical improvements provide a drastic speed-up over the basic algorithm and filter equivalent properties from the output. Unlike previous work, this approach finds all valid properties up to a given length with respect to the underlying grammar and does neither require any templates nor predefined meaningful simulation traces. Moreover, the presented syntax-guided enumerator is not tied to a specific language to express properties of a system. The properties obtained with this method can be ranked in order to highlight the most interesting ones.

In the next chapter, we describe how the set of properties obtained can be used to generate runtime monitors to be embedded in the starting design.

7

Runtime Monitoring of c-LTL Specifications on FPGAs using HLS

"Deadlines just aren't real to me until I'm staring one in the face. "
—Rick Riordan

▸ SYNOPSIS In this chapter, we describe a framework that automatically produces synthesizable C++ code implementing runtime monitors, using c-LTL semantics, starting from an LTL property. This chapter is based on the paper

> Gianluca Martino and Görschwin Fey. "Runtime Monitoring of c-LTL Specifications on FPGAs Using HLS". in: *18th International Conference on Synthesis, Modeling, Analysis and Simulation Methods and Applications to Circuit Design, SMACD 2022, Villasimius, Italy, June 12-15, 2022*. IEEE, 2022, pp. 1–4. DOI: 10.1109/SMACD55068.2022.9816308.

The next step after creating a method for producing a set of properties from an existing design is to develop a technique for creating runtime monitors to verify the properties. In this chapter, we give some details about the interpretation of LTL in running systems and then describe the framework developed.

7.1 INTRODUCTION

Ensuring the correctness of a modern system is a difficult task due to its increasing complexity. Formal verification can ensure that the system design satisfies the requirements, solving the first issue. However, not all systems, especially large ones, can be practically verified using formal verification due to the computational complexity of correctness proofs. Additionally, systems based on SRAM memories, e.g., FPGAs, are affected by Single Event Upsets (SEUs) that can modify the bit value of a memory cell during the system's runtime.

Runtime monitoring helps in both cases by implementing hardware property checkers that ensure that the system satisfies a correctness specification during operation [LS09]. Like formal verification properties, runtime monitoring properties are expressed in a specification language, e.g., LTL. However, the semantics of the language is often slightly modified to account for the finite nature of the traces observed by monitors at runtime.

Multiple possible semantics have been introduced [BLS06; Eis+03; GV13;

[LS09] Leucker and Schallhart, "A brief account of runtime verification"

[BLS06] Bauer et al., "Monitoring of Real-Time Properties"

[Eis+03] Eisner et al., "Reasoning with Temporal Logic on Truncated Paths"

[GV13] Giacomo and Vardi, "Linear Temporal Logic and Linear Dynamic Logic on Finite Traces"

[BLS10] Bauer et al., "Comparing LTL Semantics for Runtime Verification"

[Bar+18] Bartocci et al., "A Counting Semantics for Monitoring LTL Specifications over Finite Traces"

BLS10]. However, c-LTL [Bar+18] adds the possibility of predicting the result of a property evaluation based on the previous behavior of the system. We propose the usage of the predictive result to give early warnings in case of a prediction of unsatisfaction of the correctness specification. Such an early warning can safely stop a system before an unwanted state is reached. One class of systems that benefits from such a monitoring technique is the class of fail-safe systems. In such systems, a failure can happen, but the system must continue to remain safe. The ability to predict a risky situation while maintaining the strictness guaranteed by an LTL-based monitoring technique can reduce the safety intervals without loss of safety. Another application for such a monitoring technique is in fail-secure systems. In such a system, security has to be maintained even in case of faults. For example, let us consider a system where secrets are stored: the runtime monitor can act as a watchdog and disable the system before the secret leaks to an attacker.

Our contribution is the first framework that, starting from an LTL property, automatically generates synthesizable C++ monitors based on c-LTL semantics. Using HLS [Cou+09], the monitors generated are technology-independent and take advantage of the features offered by different architectures. Moreover, HLS introduces fine-grain optimizations that reduce area consumption and increase performance. Our framework reduces the generated code by employing structural hashing and applying Boolean simplifications. The high-level resource sharing yields highly optimized C++ code, e.g., if multiple monitors share the same sub-formula, sub-monitors, and evaluation sections are shared. Finally, the framework supports the usage of HLS directives to guide the HLS synthesis process and achieve further optimizations.

[Cou+09] Coussy et al., "An Introduction to High-Level Synthesis"

The rest of the chapter is structured as follows: in Section 7.2, we discuss the related work and how they differ from our work; in Section 7.3, we define all the concepts used in the remainder of the chapter; in Section 7.4, we explain the optimizations applied to the evaluation of c-LTL properties when operating on streaming data; in Section 7.5, we describe the optimizations applied when generating multiple monitors at once; in Section 7.6, we show the results of the experiments; in Section 7.7, we conclude the chapter and give some final remarks.

7.2 RELATED WORK

Since the main use case of runtime monitors is when large systems become too difficult to verify, software systems were the first systems to have such a need. Nowadays, several monitoring frameworks and techniques exist for software [DGR04; Cin+16]. Such a push has been experienced recently also for hardware systems, particularly the ones implemented in FPGAs. In [II+20], the authors propose a toolchain for constructing monitors of weak t-LTL properties using an automata representation. The paper compares different possible implementations using Grapefruit [Rah+20] as the translation layer between the automata representation and the HDL code. In [BZ08], the authors propose a synthesis technique for properties expressed in PSL. Also in this case, the approach uses automata as building blocks for the subsequent synthesis. In [Sel+17], the authors propose a technique

[DGR04] Delgado et al., "A Taxonomy and Catalog of Runtime Software-Fault Monitoring Tools"

[Cin+16] Cinque et al., "Characterizing Direct Monitoring Techniques in Software Systems"

[II+20] II et al., "Runtime Verification on FPGAs with LTLf Specifications"

[Rah+20] Rahimi et al., "Grapefruit: An Open-Source, Full-Stack, and Customizable Automata Processing on FPGAs"

[BZ08] Boule and Zilic, "Automata-based assertion-checker synthesis of PSL properties"

[Sel+17] Selyunin et al., "Runtime Monitoring with Recovery of the SENT Communication Protocol"

for the realization of monitors that can perform runtime monitoring and recovery of the SENT communication protocol. Such monitors are expressed as Timed Regular Expressions (TRE) and Signal Temporal Logic (STL) and are then synthesized using HLS starting from SystemC code. R2U2 [MRS17] is a framework for runtime monitoring of security properties aimed at diagnosing security threats on-board Unmanned Aerial Systems (UAS). The monitors are built using a Bayesian network on top of Metric Temporal Logic (MTL) properties.

[MRS17] Moosbrugger et al., "R2U2: monitoring and diagnosis of security threats for unmanned aerial systems"

The main difference between our proposed framework and the related work is the capability of our framework of generating highly-optimized c-LTL monitors, which introduces the ability to predict the result of the evaluation.

7.3 BACKGROUND

In this section, we define the main concepts used throughout the chapter.

7.3.1 *Trace*

A trace ω is a finite or infinite sequence of states of a system. Formally, we define a set AP of atomic proposition and call ω a word over the finite alphabet $\Sigma = 2^{AP}$. If the trace ω is finite then $\omega \in \Sigma^*$, if the trace ω is infinite $\omega \in \Sigma^\omega$.

7.3.2 *5-Valued Logic*

The usage of a 5-valued logic composed of true ($\top$), presumably true ($\top_p$), inconclusive (?), presumably false ($\bot_p$), and false ($\bot$) lets us distinguish between cases in which the satisfaction or violation of a property can be determined in the trace ($\top$ and $\bot$, respectively) and the cases in which past observations give us a non-contradictive support of a future satisfaction or violation ($\top_p$ and $\bot_p$, respectively). With $\top, \top_p, ?, \bot_p, \bot \in \mathbb{B}_5$, we can fully define the 5-valued logic for the usual Boolean operators using the truth tables in Table 7.1 and Table 7.2 and the usual equivalences.

TABLE 7.1: 5-valued logic truth table for $a \vee b$.

$a \vee b$	$\bot$	$\bot_p$	?	$\top_p$	$\top$
$\bot$	$\bot$	$\bot_p$	?	$\top_p$	$\top$
$\bot_p$	$\bot_p$	$\bot_p$	?	$\top_p$	$\top$
?	?	?	?	$\top_p$	$\top$
$\top_p$	$\top_p$	$\top_p$	$\top_p$	$\top_p$	$\top$
$\top$	$\top$	$\top$	$\top$	$\top$	$\top$

TABLE 7.2: 5-valued logic truth table for $\neg a$.

a	$\neg a$
$\bot$	$\top$
$\bot_p$	$\top_p$
?	?
$\top_p$	$\bot_p$
$\top$	$\bot$

7.3.3 *LTL Semantics on Finite Traces*

For LTL, we have the 3-valued LTL semantics (3-LTL) [BLS06], the LTL on truncated paths semantics (t-LTL) [Eis+03], and the LTL counting semantics (c-LTL) [Bar+18]. In the 3-LTL semantics, a satisfaction result, i.e., true ($\top$), is obtained at timestep N if all possible infinite extensions result in the property's satisfaction. The violation result, i.e., false ($\bot$), is obtained if all possible infinite extensions result in the property's violation. The 3-LTL semantics additionally introduces the inconclusive (?) result and assigns it to all cases in which not all possible infinite extensions of the trace evaluate to either satisfaction or violation of the property. The t-LTL semantics introduces a weak and a strong view of the property evaluation: in the weak view, the satisfaction result is returned for all timesteps after the end of the truncated trace; the violation result is returned in the strong view. Finally, c-LTL uses the history of satisfaction or violation of a finite trace to predict whether the formula will be satisfied when extending the formula to an infinite one.

[BLS06] Bauer et al., "Monitoring of Real-Time Properties"

[Eis+03] Eisner et al., "Reasoning with Temporal Logic on Truncated Paths"

Such a prediction, expressed as a 5-valued logic, can be used to react early to the threat of unsafe operation.

7.3.4 *LTL Counting Semantics*

c-LTL uses the history of satisfaction or violation of a finite trace to predict whether the formula will be satisfied when extending the formula to an infinite one. Intuitively, c-LTL first calculates the minimal number of steps needed to witness the satisfaction or the violation of the formula. The number of steps is expressed as a pair of values $(s, f) \in \mathbb{N}_+$, where $\mathbb{N}_+ = \mathbb{N}_0 \cup \{\infty, -\}$ with $\forall n \in \mathbb{N}_0, n < \infty < -$. If s assumes the values ∞ or $-$, this denotes that the trace can either be satisfied in an infinite number of steps or not at all, respectively. The value of f expresses the number of steps needed for the violation of the property. From the values (s, f), an evaluation expressed as a 5-valued logic is obtained as follows:

- if the result determines already a satisfaction or violation, either $\top$ or $\bot$, respectively, is returned;
- if there is a possibility for satisfaction or violation in finite time, a prediction based on previous results is made and, based on that, either $\top_p$ or $\bot_p$ is returned;
- if neither there is the possibility for satisfaction or violation in finite time, nor the prediction based on previous results is conclusive, the result is based on a recursive evaluation of the unfolded formula.

[Bar+18] Bartocci et al., "A Counting Semantics for Monitoring LTL Specifications over Finite Traces"

Considering the grammar defined in Section 2.3, Bartocci et al. [Bar+18] define a set of rules for the LTL counting semantics (c-LTL) over a finite trace $\omega = a_0 a_1 \dots a_n \in \Sigma^*$ over the finite alphabet $\Sigma = 2^{AP}$. Let $\mathbb{N}_+ = \mathbb{N}_0 \cup \{\infty, -\}$, where $\forall n \in \mathbb{N}_0, n < \infty < -$, and the function $d_\omega : \Phi \times \Sigma^* \times \mathbb{N}_0 \to \mathbb{N}_+ \times \mathbb{N}_+$, where $\phi \in \Phi$, the LTL counting semantics is defined as:

$$
\begin{aligned}
d_\omega(p \in AP, i) &= \begin{cases} (0,-), & \text{if } i \le |\omega| \text{ or } p \in a_i, \\ (-,0), & \text{if } i \le |\omega| \text{ or } p \notin a_i, \\ (0,0), & \text{if } i > |\omega|, \end{cases} \\
d_\omega(\neg\phi, i) &= \sim d_\omega(\phi, i), \\
d_\omega(\phi_1 \vee \phi_2, i) &= d_\omega(\phi_1, i) \sqcup d_\omega(\phi_2, i), \\
d_\omega(X\phi, i) &= d_\omega(\phi, i+1) \oplus 1, \\
d_\omega(\phi_1 U \phi_2, i) &= \begin{cases} d_\omega(\phi_2, i) \sqcup (d_\omega(\phi_1, i) \\ \quad \sqcap d_\omega(X(\phi_1 U \phi_2), i)), & \text{if } i \le |\omega|, \\ d_\omega(\phi_2, i) \sqcup (d_\omega(\phi_1, i), \\ \quad \sqcap (-, \infty)), & \text{if } i > |\omega|, \end{cases} \\
d_\omega(F\phi, i) &= \begin{cases} d_\omega(\phi, i) \sqcup d_\omega(XF\phi, i), & \text{if } i \le |\omega|, \\ d_\omega(\phi, i) \sqcup (-, \infty), & \text{if } i > |\omega|, \end{cases}
\end{aligned}
$$

where the operators $\sim$, $\oplus$, $\sqcap$, and $\sqcup$ are defined as follows:

$$
\begin{aligned}
\sim(s,f) &= (f,s),\\
a \in \mathbb{N}_0 \oplus b \in \mathbb{N}_0 &= a+b,\\
s \in \mathbb{N}_+ \oplus f \in \mathbb{N}_+ &= \max(s,f),\\
(s,f) \oplus 1 &= (s \oplus 1, f \oplus 1),\\
(s,f) \sqcup (s',f') &= (\min(s,s'), \max(f,f')),\\
(s,f) \sqcap (s',f') &= (\max(s,s'), \min(f,f')),
\end{aligned}
$$

with $(s,f) \in \mathbb{N}_+ \times \mathbb{N}_+$.

The predictive evaluation function $e_\omega : \Phi \times \Sigma^* \times \mathbb{N}_0 \to \mathbb{B}_5$ derives a qualitative result expressed in the 5-valued logic $\mathbb{B}_5$. In all cases in which the evaluation cannot be fully determined in the trace, the decision is made based on the previous behavior of the system as observed in the trace, calculating the prediction predicate $\text{pred}_\omega(\phi,i)$, or by unfolding the formula and checking the evaluation of the sub-formulæ, calculating the value of the auxiliary function $r_\omega(\phi,i)$. Given $m_\omega = \max_{0 \le j < i}\{s' | d_\omega(\phi,j) = (s',-)\}$ and $d_\omega(\phi,i) = (s,f) \in \mathbb{N}_+ \times \mathbb{N}_+$, the prediction predicate $\text{pred}_\omega(\phi,i)$ is defined as:

$$
\text{pred}_\omega(\phi,i) = \begin{cases}
\top, & \text{if} \quad \exists j < i . d_\omega(\phi,j) = (s',-) \\
 & \quad \text{and } s \le s', \\
?, & \text{if} \quad \nexists j < i . d_\omega(\phi,j) = (s',-), \\
\bot, & \text{if} \quad \exists j < i . d_\omega(\phi,j) = (s',-) \\
 & \quad \text{and } s > m_\omega.
\end{cases}
$$

The auxiliary function $r_\omega : \Phi \times \Sigma^* \times \mathbb{N}_0 \to \mathbb{B}_5$ is defined as follows:

$$
\begin{aligned}
r_\omega(p \in AP, i) &= ?,\\
r_\omega(\neg\phi, i) &= \neg e_\omega(\phi,i),\\
r_\omega(\phi_1 \vee \phi_2, i) &= e_\omega(\phi_1,i) \vee e_\omega(\phi_2,i),\\
r_\omega(X\phi, i) &= e_\omega(\phi, i+1),\\
r_\omega(\phi_1 U \phi_2, i) &= \begin{cases}
e_\omega(\phi_2,i) \vee (e_\omega(\phi_2,i) & \\
\quad \wedge e_\omega(X(\phi_1 U \phi_2), i)), & \text{if } i \le |\omega|, \\
e_\omega(\phi_2,i), & \text{if } i > |\omega|,
\end{cases}\\
r_\omega(F\phi, i) &= \begin{cases}
e_\omega(\phi,i) \vee e_\omega(XF\phi,i), & \text{if } i \le |\omega|, \\
e_\omega(\phi,i), & \text{if } i > |\omega|.
\end{cases}
\end{aligned}
$$

Putting it all together, we can finally define the function $e_\omega(\phi,i)$ as an equivalence between all the possible results of the function $d_\omega(\phi,i)$ and the results of the function $e_\omega(\phi,i)$. With $\bot < ? < \top$, we obtain the following definition:

$d_\omega(\phi,i)$	$e_\omega(\phi,i)$	
$(a,-)$	$\top$	
(b_1,b_2)	$\top_p$	if $\text{pred}_\omega(\phi,n) > \text{pred}_\omega(\neg\phi,n)$
	$r_\omega(\phi,i)$	if $\text{pred}_\omega(\phi,n) = \text{pred}_\omega(\neg\phi,n)$
	$\bot_p$	if $\text{pred}_\omega(\phi,n) < \text{pred}_\omega(\neg\phi,n)$
(b_1,∞)	$\top_p$	if $\text{pred}_\omega(\phi,n) = \top$
	$r_\omega(\phi,i)$	if $\text{pred}_\omega(\phi,n) = ?$
	$\bot_p$	if $\text{pred}_\omega(\phi,n) = \bot$
(∞,b_1)	$\neg e_\omega(\neg\phi,i)$	
(∞,∞)	$r_\omega(\phi,i)$	
$(-,a)$	$\bot$	

7.4 C-LTL ON STREAMING DATA

The evaluation of c-LTL in an FPGA with streaming data requires all the evaluations to be done in advance, such that at each time step, the value of the prediction predicate and of all the values in the unfolded formula are available. Considering n and $n+1$, respectively, the current and the next time step in the execution trace, a c-LTL runtime monitor is composed of the following blocks:

- a calculation block, where the value of the functions $d_\omega(\phi,n)$ and $d_\omega(\phi,n+1)$ are calculated;
- a sub-monitor, where the value of the auxiliary function $r_\omega(\phi,i)$ is calculated;
- two prediction memories, for the calculation of $\text{pred}_\omega(\phi,n)$ and $\text{pred}_\omega(\phi,n+1)$.

Each sub-monitor is a complete runtime monitor of a sub-formula of the original LTL formula so that the value of $d_\omega(\phi,n)$ and $d_\omega(\phi,n+1)$ can be calculated for the corresponding LTL sub-formula in the execution trace.

Taking advantage of the fact that we operate on streaming data and calculate the result always for the time steps n and $n+1$, we optimize the calculations for some of the operators of the c-LTL semantics. For the first time step after the end of the trace $n+1$, the previous definition stands. For the time step n, we modify the expressions so that we are able to reuse the results already calculated for $n+1$:

$$
\begin{aligned}
d_\omega(p \in AP, n) &= \begin{cases} (0,-), & \text{if } i \le |\omega| \text{ and } p \in a_i, \\ (-,0), & \text{if } i \le |\omega| \text{ and } p \notin a_i, \end{cases} \\
d_\omega(\neg\phi, n) &= \sim d_\omega(\phi, n), \\
d_\omega(\phi_1 \vee \phi_2, n) &= d_\omega(\phi_1, n) \sqcup d_\omega(\phi_2, n), \\
d_\omega(X\phi, n) &= d_\omega(\phi, n+1) \oplus 1, \\
d_\omega(\phi_1 U \phi_2, n) &= d_\omega(\phi_2, n) \sqcup (d_\omega(\phi_1, n) \\
&\qquad \sqcap d_\omega(\phi_1 U \phi_2, n+1)), \\
d_\omega(F\phi, n) &= d_\omega(\phi, n) \sqcup d_\omega(F\phi, n+1),
\end{aligned}
$$

where $d_\omega(F\phi, n+1)$ and $d_\omega(\phi_1 U \phi_2, n+1)$ have already been calculated.

Also, we rewrite the auxiliary functions $r_\omega(\phi, n)$ and $r_\omega(\phi, n+1)$ for time step n:

$$\begin{aligned}
r_\omega(p \in AP, n) &= ?, \\
r_\omega(\neg\phi, n) &= \neg e_\omega(\phi, n), \\
r_\omega(\phi_1 \vee \phi_2, n) &= e_\omega(\phi_1, n) \vee e_\omega(\phi_2, n), \\
r_\omega(X\phi, n) &= e_\omega(\phi, n+1), \\
r_\omega(\phi_1 U \phi_2, n) &= e_\omega(\phi_2, n) \vee (e_\omega(\phi_2, n) \\
&\qquad \wedge e_\omega(\phi_1 U \phi_2, n+1)), \\
r_\omega(F\phi, n) &= e_\omega(\phi, n) \vee e_\omega(F\phi, n+1),
\end{aligned}$$

where $e_\omega(F\phi, n+1)$ and $e_\omega(\phi_1 U \phi_2, n+1)$ are the result of the current monitor at the next time step. Those results are already known since the results for $n+1$ do not require any previous results.

The monitors have two prediction memories so that the case in which $d_\omega(\phi, n) = (b_1, b_2)$ can be managed. The two prediction memories store the value $\max_s$, maximum of all values of s for $d_\omega(\phi, n) = (s, -)$ or $\max_f$, maximum of all values of f for $d_\omega(\phi, n) = (-, f)$. The memories are initialized with a negative value such that the prediction predicate $\text{pred}_\omega(\phi, i)$ can be redefined as follows:

$$\text{pred}_\omega(\phi, i) = \begin{cases} \top, & \text{if } \max_s \geq 0 \text{ and } s \leq \max_s, \\ ?, & \text{if } \max_s < 0, \\ \bot, & \text{if } \max_s \geq 0 \text{ and } s > \max_s. \end{cases}$$

This reformulation allows us to avoid storing all previous values.

7.5 C-LTL RUNTIME MONITORING FRAMEWORK

The c-LTL runtime monitoring framework generates synthesizable monitors as C++ code. The C++ code obtained is then compiled to either VHDL or Verilog using an HLS compiler.

Creating a monitor for a single property requires the creation of multiple runtime monitors for the calculation of $r_\omega(\phi, n)$, as explained in Section 7.4. This is done by unfolding the formula, removing one operator at a time, starting from the most external one.

Example 2. Unfolding of $G(req \rightarrow F(ack))$.

$$\begin{aligned}
G(\neg req \vee F(ack)) &\rightarrow \text{monitor}_0 \\
\neg req \vee F(ack) &\rightarrow \text{monitor}_1 \\
\neg req &\rightarrow \text{monitor}_{2l} \\
req &\rightarrow \text{monitor}_{3l} \\
F(ack) &\rightarrow \text{monitor}_{2r} \\
ack &\rightarrow \text{monitor}_{3r}
\end{aligned}$$

Reformulating all Boolean operators to either $\neg$ or $\vee$, we avoid generating additional monitors for all Boolean operators by taking advantage of the fact that $r_\omega(\neg\phi, i) = \neg e_\omega(\phi, i)$ and $r_\omega(\phi_1 \vee \phi_2, i) = e_\omega(\phi_1, i) \vee e_\omega(\phi_2, i)$. We calculate the result for the sub-formulæ of a Boolean operator and then apply the Boolean operators for as many unfolding steps as necessary. The next example shows that the effect of this on the previous property is the elimination of monitor$_1$ and monitor$_{2l}$.

Example 3. Unfolding of $G(req \rightarrow F(ack))$.

$$\begin{aligned} G(\neg req \vee F(ack)) &\rightarrow \text{monitor}_0 \\ req &\rightarrow \text{monitor}_{3l} \\ F(ack) &\rightarrow \text{monitor}_{2r} \\ ack &\rightarrow \text{monitor}_{3r} \end{aligned}$$

In case of multiple properties to be monitored, the sub-monitors can be shared among multiple monitors. This is achieved by structural hashing of the LTL property. An example is shown in Figure 7.1.

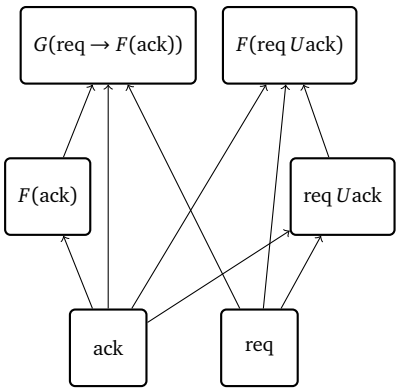


FIGURE 7.1: Sub-monitors sharing graph for the properties G(req → F(ack)) and F(req U ack).

The calculation of $d_\omega(\phi, n)$ is performed in a separate function so that the same sharing mechanism based on structural hashing of the LTL property can be applied. This is useful since the sub-monitors of a monitor have already calculated the partial values of $d_\omega(\phi, n)$.

Finally, our framework places in-code directives in functions that can be inlined to achieve better resource sharing. The HLS compiler uses the directives to decide how to implement each function and, on a more fine-grain scale, each code section.

7.6 EXPERIMENTS

In the experiments, we evaluated the monitors' resource consumption and performance when varying the parameters used for generating the LTL properties.

7.6.1 *Experimental Setting*

In all the experiments, we use the tool *randltl*, part of *spot* [DL+16], to generate the set of properties used in the experiments. Using this tool, we obtain a set of unique properties. Additionally, we can control the number of properties generated, the length, and the number of atomic propositions.

We use Xilinx Vitis HLS [Xil19] for the high-level synthesis of the C++ runtime monitors to VHDL and the Virtex-UltraScale VCU110 Evaluation Platform as a target device. The numbers used in the following tables are the estimated area and performance reported by Xilinx Vitis HLS.

In the experiment, we evaluate the optimization techniques implemented by running the high-level synthesis using all combinations of the following parameters:

- number of properties (P): 1, 10, 85,
- length of properties: 3, 4, 5, 6,
- number of atomic propositions (AP): 3, 10.

[DL+16] Duret-Lutz et al., "Spot 2.0 — a framework for LTL and ω-automata manipulation"

[Xil19] Xilinx, *Introduction to FPGA Design with Vivado High-Level Synthesis*

TABLE 7.3: Estimated number of flip-flops for varying numbers of properties (P), atomic propositions (AP), and length of the properties.

P	AP	Length 3			Length 4			Length 5			Length 6		
		unopt	opt	diff	unopt	opt	diff	unopt	opt	diff	unopt	opt	diff
1	3	208	17	−92%	52	24	−54%	43	24	−44%	71	31	−56%
10	3	284	62	−78%	540	72	−87%	721	72	−90%	437	63	−86%
	10	263	92	−65%	497	102	−79%	1054	88	−92%	1179	121	−90%
85	3	2193	384	−82%	4298	426	−90%	5327	493	−91%	6238	439	−93%
	10	3164	474	−85%	4413	490	−89%	7671	490	−94%	8450	576	−93%
Avg. diff				−82%			−75%			−76%			−79%

TABLE 7.4: Estimated number of look-up tables for varying numbers of properties (P), atomic propositions (AP), and length of the properties.

P	AP	Length 3			Length 4			Length 5			Length 6		
		unopt	opt	diff	unopt	opt	diff	unopt	opt	diff	unopt	opt	diff
1	3	3614	472	−87%	950	566	−40%	769	518	−33%	1179	667	−43%
10	3	3923	869	−78%	9522	2047	−79%	8048	1336	−83%	6248	1013	−84%
	10	4260	676	−84%	9115	1267	−86%	17331	1234	−93%	16484	1623	−90%
85	3	18335	7879	−57%	39567	10491	−73%	58331	11994	−79%	92702	11015	−88%
	10	26425	6971	−74%	52796	7234	−86%	95373	8629	−91%	103507	10582	−90%
Avg. diff				−78%			−68%			−69%			−73%

TABLE 7.5: Estimated maximum frequency [MHz] for varying numbers of properties (P), atomic propositions (AP), and length of the properties.

P	AP	Length 3			Length 4			Length 5			Length 6		
		unopt	opt	diff	unopt	opt	diff	unopt	opt	diff	unopt	opt	diff
1	3	73	120	65%	150	144	−4%	181	172	−5%	132	128	−4%
10	3	45	121	171%	54	74	36%	52	120	130%	51	150	196%
	10	66	257	290%	41	119	194%	35	80	131%	42	109	161%
85	3	64	257	298%	54	114	110%	44	111	153%	44	109	148%
	10	64	118	83%	51	219	331%	44	209	377%	40	110	176%
Avg. diff				162%			110%			130%			112%

We limit the number of properties to 85, corresponding to the maximum number of properties that *randltl* can generate with length 3 using 3 atomic propositions.

7.6.2 *Experimental Results*

The results are shown in Table 7.3, Table 7.4, and Table 7.5. In each table, we report the results obtained with the optimizations described in Section 7.5 disabled (*unopt*) and with all the optimizations enabled (*opt*). We also report the relative difference in percent between the *unopt* columns and the *opt* columns (*diff*). Finally, we report the average relative difference in percent

for each property length (*Avg. diff*). We omitted the numbers for 1 property using 10 atomic propositions for all lengths. The numbers obtained for a single property mainly depend on the structure, so they are the same for 3 and 10 atomic propositions.

Table 7.3 shows the number of flip-flops. Enabling all the optimizations, the minimum flip-flop utilization decrease is 44 % and the maximum is 94 %, with an average saving of 78 %. A larger number of properties requires more flip-flops. While generating 10 monitors of length 6 using 3 atomic predicates in the unoptimized case requires approximately 6.2 times more flip-flops than for a single monitor. With the optimizations enabled, this ratio drops to approximately 2. When considering 10 monitors with respect to 100 monitors, the ratio drops from 14.3 to 7, approximately.

In Table 7.4, we show the number of look-up tables (LUTs). The LUT utilization decreases on average 72% by enabling our optimizations. For look-up tables, generating 10 monitors of length 6 using 3 atomic predicates requires approximately 5.3 times more look-up tables than for a single monitor. With the optimizations enabled, this ratio drops to approximately 1.5.

Table 7.5 shows the maximum frequencies. We see a frequency increase of up to 377% after the optimizations are applied. On average, the frequency increase is 130%. The monitors generated have maximum frequencies ranging from 74 MHz to 257 MHz with an average frequency of 145 MHz. Differently from flip-flops and look-up tables, the variation of the maximum frequency is not related to the number of properties generated.

Another noticeable result is in Table 7.3 and Table 7.4 for the single property experiment. The property of length 3 has a higher resource utilization than the properties with lengths 4, 5, and 6 before the optimizations. However, after optimizing, this monitor ends up having the lowest resource consumption. This shows the effectiveness of the optimizations applied by the framework.

7.7 SUMMARY

In this chapter, we presented a new framework for the generation of synthesizable c-LTL monitors. Starting from LTL properties, the framework generates synthesizable C++ code that is later used for HLS synthesis. The code generated is optimized thanks to the usage of structural hashing, simplifications tailored to c-LTL, and HLS directives.

The experimental results show that our optimizations effectively reduce the area utilization and improve the maximum frequency of the monitors. The tests performed also highlight the effectiveness of the optimizations with respect to scalability. A larger number of properties does not require as much resource increase as in the case without our optimizations. The gain in frequency is instead stable even for a large number of monitors.

The code generated from this framework can be embedded in existing designs. The runtime monitors obtained can check the design during runtime for violation of the specification used as input. If the specification has been obtained by the tool in Chapter 6, the runtime monitor obtained will check

for all the properties of the system that can be expressed by properties of up to the length specified during the enumeration.

In the next chapter, we conclude Part III by describing a technique for generating the smallest circuit possible that implements a specified function.

8

Revisiting Explicit Enumeration for Exact Synthesis

"In trying to count our many blessings the difficulty is not to find things to count, but to find time to enumerate them all."
—Aiden Wilson Tozer

▸ SYNOPSIS In this chapter, we describe a parallelizable explicit enumeration algorithm that finds the smallest circuits of a given Boolean function in a single run of the algorithm. This chapter is based on the paper

> Gianluca Martino, Heinz Riener, and Görschwin Fey. "Revisiting Explicit Enumeration for Exact Synthesis". In: *23rd Euromicro Conference on Digital System Design, DSD 2020, Kranj, Slovenia, August 26-28, 2020*. IEEE, 2020, pp. 29–34. DOI: 10.1109/DSD51259.2020.00016.

In this chapter, we show our solution to the problem of exact synthesis, i.e., the problem of finding the smallest circuit implementing a given Boolean function. Addressing this problem is useful for implementing faster, more efficient, and smaller circuits. In our case, this is important for implementing runtime monitors that have the smallest footprint possible.

8.1 INTRODUCTION

A synthesizer produces one or more possible implementations starting from a correctness specification. This means answering the question: "What is the implementation I of a specification S?". Given a background theory defining all symbols to be used, e.g., gate types, and their respective interpretations, e.g., the functions implemented by each gate, all possible combinations of the symbols compose the search space. A synthesizer selects from the search space the solutions that satisfy the specification S. The problem of finding the smallest circuit, i.e., optimal with respect to the number of gates needed, is known as exact synthesis. In logic synthesis, exact synthesis is used for various optimization steps, e.g., peephole optimization and circuit rewriting [Rie+19; Ama+17; BB04; MCB06], or for composing libraries of minimal circuits to be used in those optimization steps.

As described in [Ern09], algorithms for exact synthesis can be organized into three different categories:

1. Algorithms based on functional decomposition [Kar+61; RK62; SD68; Law64], where the target function is repeatedly decomposed until a predefined set of building blocks can represent all sub-functions.

[Rie+19] Riener et al., "On-the-fly and DAG -aware: Rewriting Boolean Networks with Exact Synthesis"

[Ama+17] Amarù et al., "Enabling exact delay synthesis"

[BB04] Bjesse and Borälv, "DAG-aware circuit compression for formal verification"

[MCB06] Mishchenko et al., "DAG -aware AIG Rewriting: a Fresh Look at Combinational Logic Synthesis"

[Ern09] Ernst, "Optimal combinational multi-level logic synthesis"

[Kar+61] Karp et al., "A computer program for the synthesis of combinational switching circuits"

[RK62] Roth and Karp, "Minimization Over Boolean Graphs"

[SD68] Schneider and Dietmeyer, "An Algorithm for Synthesis of Multiple-Output Combinational Logic"

[Law64] Lawler, "An Approach to Multilevel Boolean Minimization"

[Hel63] Hellerman, "A Catalog of Three-Variable Or-Invert and And-Invert Logical Circuits"

[Smi65] Smith, "Minimal three-variable NOR and NAND logic circuits"

[DG98] Drechsler and Günther, "Exact circuit synthesis"

[MI72] Muroga and Ibaraki, "Design of optimal switching networks by integer programming"

[Bau+72] Baugh et al., "Optimal Networks of NOR-OR Gates for Functions of Three Variables"

[ML76] Muroga and Lai, "Minimization of Logic Networks Under a Generalized Cost Function"

[KKY09] Kojevnikov et al., "Finding Efficient Circuits Using SAT-Solvers"

[Haa+18] Haaswijk et al., "SAT Based Exact Synthesis Using DAG Topology Families"

[FHS17] Fiser et al., "SAT -Based Generation of Optimum Function Implementations with XOR Gates"

[Dav69] Davidson, "An Algorithm for NAND Decomposition Under Network Constraints"

[Cul+79] Culliney et al., "Results of the Synthesis of Optimal Networks of AND and OR Gates for Four-Variable Switching Functions"

[Ern09] Ernst, "Optimal combinational multi-level logic synthesis"

[HJS09] Hamadi et al., "ManySAT: a Parallel SAT Solver"

[BSS15] Balyo et al., "HordeSat: A Massively Parallel Portfolio SAT Solver"

[HW12] Hamadi and Wintersteiger, "Seven Challenges in Parallel SAT Solving"

[BM10] Brayton and Mishchenko, "ABC: An Academic Industrial-Strength Verification Tool"

2. Algorithms based on explicit enumeration [Hel63; Smi65; DG98] or implicit enumeration [MI72; Bau+72; ML76; KKY09; Haa+18; FHS17], where all possible solutions are enumerated using circuits of growing sizes. The difference between explicit and implicit enumeration lies in whether all the possible circuits are explicitly represented or whether a single structure represents the space of all possible circuits.

3. Hybrid approaches where both techniques are combined [Dav69; Cul+79; Ern09].

Currently, SAT-based implicit enumeration algorithms represent the state-of-the-art in exact synthesis. However, SAT-based algorithms have problems that are inherently hard to solve: the runtime is not predictable and depends on the query, and they are challenging to parallelize. Despite some parallel SAT solvers exist [HJS09; BSS15], many challenges still remain [HW12].

In this chapter, we describe an enumeration algorithm that, using constraints based on the topology of the circuit or on the properties of the gates used, quickly scans through the search space to find a minimal circuit implementing the target function. The algorithm uses a set of non-isomorphic directed acyclic graphs to define all possible circuit structures. For each node of each graph, the algorithm generates a set of possible symbols to be assigned to each node of the circuit based on the number of children. The enumeration is guided by the set of possible symbols combined with the set of constraints given by the concrete gates, symmetries of the circuit, and its functional behavior.

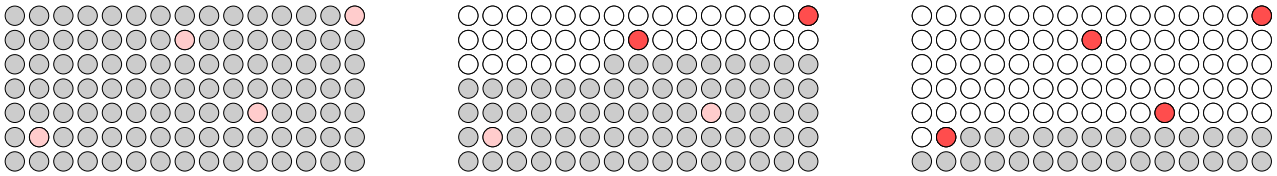
The main contribution is the proposal of an explicit enumeration algorithm that can be parallelized effectively. Searching for multiple solutions using this algorithm requires a single run of the algorithm. This means that the search has a runtime that is equal to the runtime of finding only the last circuit in the search order. For this reason, our approach is suitable for the creation of libraries of minimal-size circuits.

To evaluate the approach, we implemented the framework in a prototype header-only library. We compared the approach to the one available in ABC [BM10] for the generation of minimal circuits using and-inverter graphs (AIGs) implementing all possible 3-input functions.

The chapter is organized as follows: the concepts and the conventions used are introduced in Section 8.2; in Section 8.3 and Section 8.4, we describe the algorithm and the optimizations; in Section 8.5, all the details about the experiments are given; in Section 8.6, we conclude the chapter and Part III.

8.2 BACKGROUND

Directed acyclic graphs (DAGs) are a class of graphs that can be used for the representation of the structure of Boolean circuits. The representation of the Boolean circuit using a DAG $G = (V, E)$ requires a labeling function $f : V \to \mathbb{B}$ that maps the vertices of the DAG to Boolean operators. For short, we will use $f(G)$ to indicate that the labeling function is applied to every vertex of the graph G. As a result, a Boolean circuit is obtained. A Boolean



(a) (b) (c)

FIGURE 8.1: Representation of the circuits' search space (in red the circuits implementing the target function, in gray the circuits not yet explored, in white the circuits explored): (a) initial state of the search space, (b) 36 circuits explored, 2 target circuits found, and (c) 77 circuits explored, all target circuits found.

circuit B implements a Boolean function Γ if the truth tables T_B and T_Γ are the same, and we indicate this with $B \models \Gamma$.

The algorithm presented in Section 8.4 takes advantage of isomorphism in DAGs to reduce the number of circuits to be considered. Two graphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$ are isomorphic if they have the same number of vertices and all vertices are connected in the same way to each other. More formally, G_1 and G_2 are isomorphic if there exists a bijective function $\Phi : V_1 \to V_2$ that maps adjacent vertices in G_1 to adjacent vertices in G_2 and nonadjacent vertices in G_1 to nonadjacent vertices in G_2.

Despite more efficient algorithms exist [Bab16], canonicalization is often used as a method for detecting isomorphic graphs in a set of graphs. Graph canonicalization is a transformation that produces the same canonical representative from each isomorphic graph. To reduce a set of graphs to a set of non-isomorphic graphs, we store a canonical representative in a hashmap for each computed canonical form and remove each graph that has a representative that is already present in the hashmap. Since isomorphic DAGs represent the same circuit structure, we can reduce the set of considered graphs by selecting only non-isomorphic graphs.

[Bab16] Babai, "Graph Isomorphism in Quasipolynomial Time"

Generation of sets of non-isomorphic DAGs can be done on-the-fly, or storing the results, using dedicated libraries such as nauty [MP14] or using the implementation contained in percy [Soe+19].

[MP14] McKay and Piperno, "Practical graph isomorphism, II"

[Soe+19] Soeken et al., *The EPFL logic synthesis libraries*

8.3 EXACT SYNTHESIS USING ENUMERATION

Solving the exact synthesis problem using an enumeration algorithm consists in representing all possible circuits, exploring the search space using circuits composed of an increasing number of gates until the correct circuit has been found. Formally, solving the exact synthesis problem for the function Γ means finding the smallest graph G such that $\exists f(G) \models \Gamma$.

In the case of implicit enumeration, the problem is encoded as Conjunctive Normal Form (CNF) formulae. Boolean variables encode the structure of the circuit, the functionality of each gate, the target function, and a set of restrictions to guarantee that the circuit is minimal. With this encoding, the problem becomes very hard to solve because the number of variables required grows very rapidly with the number of inputs and the number of gates of the circuit.

In the case of explicit enumeration, the circuits are explicitly represented and simulated until a difference with the truth table of the target function is

found. Otherwise, the solution is reported to the user. The search space is explored linearly so that all the solutions can be found before the last one is found. One of the main advantages of explicit enumeration is that in case multiple target functions need to be found, e.g., for constructing a library, the execution time for finding all circuits is equal to the largest execution time. This is in contrast to the implicit enumeration, which encodes each problem in a single SAT instance. Figure 8.1 shows a representation of the search space containing all possible circuits.

However, the obvious problem of the explicit enumeration formulation is that the search space rapidly grows when increasing the number of gates. The problem can be mitigated by reducing the number of circuits composing the search space, e.g., by removing equivalent circuits.

Equivalent circuits to be removed are the ones containing structures that can be simplified using the following equivalence rules of Boolean algebra:

- idempotence: $x \wedge x = x \vee x = x$,
- absorpion: $x \wedge (x \vee y) = x \vee (x \wedge y) = x$,
- distributivity: $(x \vee y) \wedge (x \vee z) = x \vee (y \wedge z)$.

We can also use the remaining equivalence rules of Boolean algebra to avoid considering more than once the same circuit even if it has the same size:

- commutativity: $x \wedge y = y \wedge x$,
- associativity: $x \vee (y \vee z) = (x \vee y) \vee z$.

Each rule prunes large portions of the search space, reducing the search problem.

Additionally, the search space is reduced by learning equivalent circuits during the execution of the algorithm. This can be done by saving the size of the first circuit that implements a certain function, i.e., the minimal size. All the circuits encountered afterward having the same truth table are duplicated, and all circuits containing the same subcircuit are not minimal.

8.4 ENUMERATION ALGORITHM

In the following subsections, we describe:

- the data structures for representing the DAG, the symbols assigned to each node of the DAG during the enumeration, and the concrete representation for deciding whether the circuit implements the target function;
- how the algorithm uses the data structures to enumerate all possible solutions;
- the techniques for reducing the search space.

8.4.1 *Data Structures*

The first data structure used by the enumeration algorithm contains the set of non-isomorphic DAGs. A DAG represents the structure of a set of circuits and specifies the connections between the nodes. Each DAG is represented as an array of gates where each node contains the index of its children. We present an example in Figure 8.2. The index starts from 1 because 0 is used to represent the inputs.

Moreover, we define an ordered set of symbols composed of all the inputs and all the gate types that can be assigned to each node. Each node is associated with a set of possible assignments, subset of the set of symbols. The algorithm selects each of the possible assignments for each node to generate each candidate circuit.

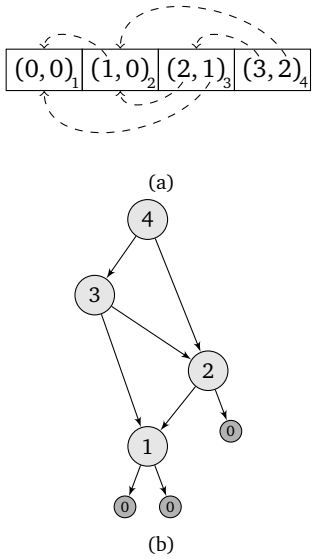


FIGURE 8.2: (a) Array of gates and (b) corresponding structure.

```
Require: F: set of target functions, D: set of DAGs
foreach d ∈ D do
    p ← generate_sets_of_possible_assignments(d);
    c ← initial_current_assignment(p);
    do
        if duplicate(c, p, d) then
            skip_circuits(c, p);
        end
        if c implements f ∈ F then
            add c to solutions;
        end
        if all solutions found then
            return solutions;
        end
    while not next_current_assignment(c, p);
end
```

Algorithm 1: Exact synthesis algorithm using explicit enumeration.

8.4.2 *Enumeration*

The algorithm requires a set of non-isomorphic DAGs to be passed to the enumerator. The set of non-isomorphic DAGs can be generated efficiently or precomputed and stored. Precomputed sets of non-isomorphic DAGs are available online[1].

[1] http://users.cecs.anu.edu.au/~bdm/data/graphs.html

For each DAG, the enumeration starts by generating the set of possible assignments for each node of the graph. This is done by checking the number of children nodes present for each node and selecting a corresponding symbol, e.g., a symbol with no children nodes can be associated only with input symbols while a node with 2 children nodes can be associated with an $\wedge$ (AND) symbol. The algorithm uses the first element of the possible assignments set for each node as the starting element for the enumeration. An example is presented in Figure 8.3. The symbols a, b, and c represent the inputs of the circuit.

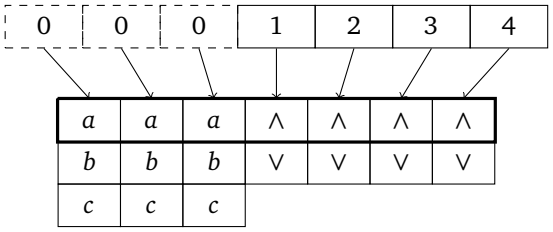


FIGURE 8.3: Nodes of the graph of Figure 8.1(b) with the relative possible assignment arrays. The highlighted element of each array represents the current assignment for each node.

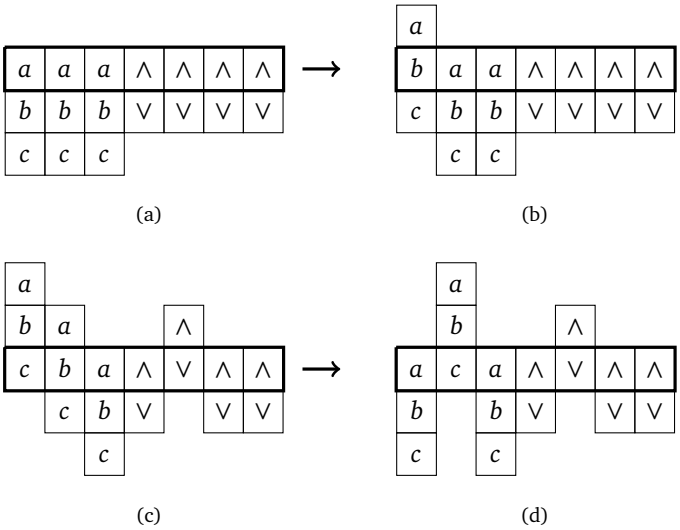


FIGURE 8.4: Various statuses of the current assignment *c*. (a) and (c) are the starting states. (b) and (d) are the next states after the call to *next_current_assignment(c, p)*.

The enumerator produces all possible assignments by generating all possible combinations. Pseudocode is given in Algorithm 1. The algorithm requires as input a set of target functions F for which the algorithm needs to find an implementation and a set of non-isomorphic DAGs D, which are used for the enumeration. The output of the algorithm is the set of minimal circuits. The *if-condition* in Line 5, applies the rules defined in Section 8.4.3 to skip entire subsections of the search space. The function *skip_circuits(c, p)*, in Line 6, sets the current assignment c to the next value which contains a non-duplicate circuit. In Line 8, the current circuit, represented by the current assignment c, is checked with respect to the target function f. This is done by comparing the truth table of the circuit with the truth table of the function. In Line 14, the next assignment is generated starting from the current assignment c and the set of possible assignments p. The algorithm uses the current assignment c as a counter where each digit has a different maximum value depending on the number of possible assignments p. Figure 8.4 presents an example. The initial state, represented in Figure 8.4(a), transitions to the next state, represented in Figure 8.4(b), after the first call of *next_current_assignment(c, p)*. In Figure 8.4(c), the current assignment c transitions to the next state represented in Figure 8.4(d). If all possible solutions have been explored, i.e., the counter has reached the maximum value, the function *next_current_assignment(c, p)* returns *false* so that the next DAG is selected.

8.4.3 *Search Space Pruning*

Pruning the search space is essential for the practicality of our algorithm. The proposed search space pruning techniques use the structure of the DAG and the properties of the gate types to filter out entire classes of circuits.

▸ SINGLE AND MULTI GATE PRUNING

Each gate type defined can have some attributes which produce some equivalences. This allows us to skip some of the candidate circuits without the risk of losing a possible solution. For idempotency and commutativity of the $\wedge$ gate, represented in Figure 8.5, six out of the nine possible circuits can be discarded.

Increasing the size of the circuit and considering multiple gates at once, the filtering becomes more effective, as shown in Figure 8.6. Other filtering opportunities are given, for example, by skipping the circuits containing twice the same gate with the same inputs.

▸ SEARCH SPACE PRUNING IN PRACTICE

The algorithm checks for commutativity by fixing a canonical symbol order at the inputs of the circuit and for idempotence by searching for equal symbols at the inputs of the circuit for each gate. Search space pruning is done by skipping multiple circuits using the current assignment c as a counter. In case one or more gates in a particular configuration produce a duplicate circuit, most of the circuits containing the same configuration can be skipped by increasing the maximum possible digit, which does not cause any relevant circuit to be skipped. The example in Figure 8.7 shows how by increasing the most significant digit between the two children nodes when the duplicate is detected, it is possible to skip three circuits out of the nine in total. In some cases, it is possible to achieve a more effective search space reduction by temporarily removing some elements from the sets of possible assignments. In the example in Figure 8.8, we can see that by removing one element from each set of possible assignments at the input nodes, only non-duplicate circuits remain. Both Figure 8.7(c) and Figure 8.8(c) represent the set of circuits in Figure 8.5(b).

8.4.4 *Parallelization*

One of the advantages of using an explicit enumeration algorithm is that it is straightforward to parallelize the algorithm. Two possible approaches are a) to use a single-producer/multiple-consumers approach or b) to fully parallelize the enumeration by having a shared set of current assignments c. In the first approach, the single producer explores the search space and places the candidate circuits in a FIFO queue. Multiple consumers compute the truth tables and compare them with the target functions. In the second approach, all the threads increase the current assignment c and check the candidate circuit. The modification of the current assignment c is a critical section and needs to be protected by a mutex.

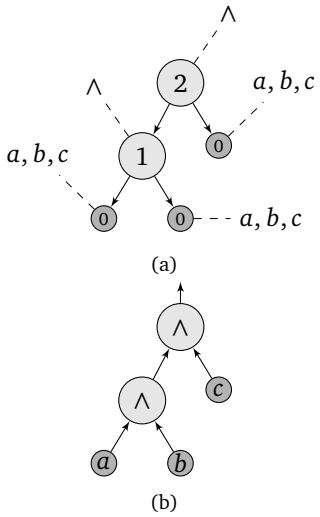
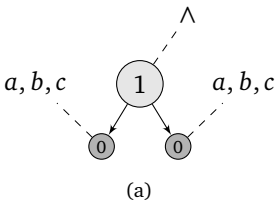
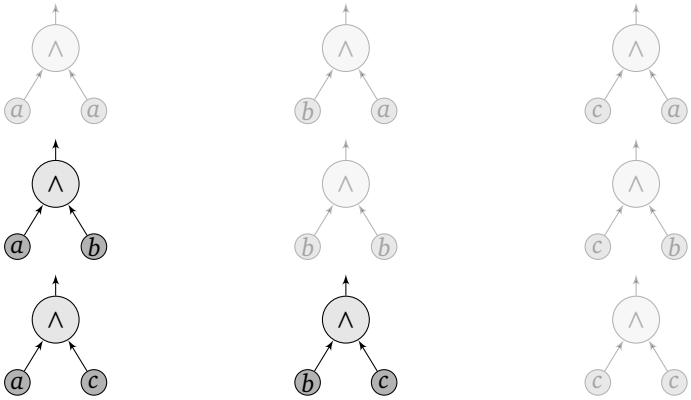


FIGURE 8.6: (a) DAG with possible assignments and (b) possible AIG concretization if the node $\wedge$ has the attributes *multinode idempotent* and *multinode commutative*.

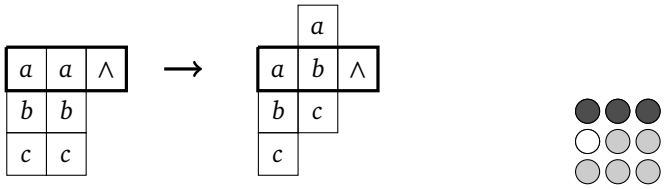


(a)



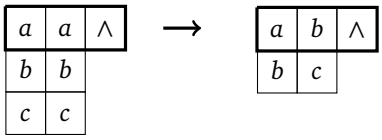
(b)

FIGURE 8.5: (a) Representation of the DAG with the associated possible assignments and (b) possible circuits.



(a) (b) (c)

FIGURE 8.7: Example of search space pruning: (a) duplicate detected; (b) status after increment; (c) representation of the search space. The black dots represent skipped circuits.



(a) (b) (c)

FIGURE 8.8: Example of search space pruning by temporarily removing possible assignments: (a) duplicate detected; (b) status after removing the possible assignment; (c) representation of the search space. The black dots represent skipped circuits.

8.5 EXPERIMENTAL RESULTS

In the following tests, the task is to generate the minimal implementation of all possible 3-input functions. For testing the algorithm, we compared it to the implementation present in ABC [BM10] using the command *allexact*. We used the flag *-a* to force ABC to use only ∧ gates. The runtimes reported are measured in ABC from the start of the execution of the command to the production of the result. The system used for testing is equipped with an Intel(R) Core(TM) i7-8750H processor running at 2.20GHz.

For both ABC and the explicit enumeration algorithm, in some instances, the runtime is smaller than the resolution of the clock. In those cases, the time is measured over 50 consecutive runs and then divided the total time by 50.

[BM10] Brayton and Mishchenko, "ABC: An Academic Industrial-Strength Verification Tool"

#gates	*explicit enumeration*		*ABC*	
	#funcs	avg. runtime [s]	#funcs	avg. runtime [s]
1	24	<0.001	0	<0.001
2	72	0.001	64	0.001
3	30	0.004	56	0.002
4	104	0.213	104	0.007
5	8	1.674	14	0.024
6	18	28.189	18	0.143

TABLE 8.1: Runtime comparison between ABC and the presented enumerator.

In the first test, we compared the average runtime for each possible number of nodes, which composes a solution. For verifying the results, for each function, we performed a combinational equivalence checking of the two solutions using the command *dcec* in ABC. The results are shown in Table 8.1. The first result was that the enumeration algorithm found a smaller solution for 32 functions. Those 32 functions can be represented with 1 or 2 gates, but ABC reports that are needed either 3 or 5 gates, hence the difference in the respective lines of the table. The reason is that ABC returns a solution that always contains all inputs. This is because ABC assumes the function to be in its minimal support representation. The average runtimes for single circuits are in favor of ABC due to the limited number of filters for search space pruning implemented.

In the second test, we tried to obtain minimal circuits implementing all possible functions with up to 4 gates. For our implementation, we just needed to give a list of target functions and a maximum circuit size. Since ABC does not give the same possibility, we summed up all execution times from the previous execution for all functions implemented with up to 4 gates. In this case, we obtain 230 circuits from the explicit enumeration algorithm in 0.75 s, while ABC returns 224 circuits in 0.94 s. As explained in Section 8.3, this is due to the fact that in our algorithm, we can find the solution by exploring the search space only once.

In the third test, we tested how the parallelization of the algorithm affects the runtimes. For the implementation, we decided to use the second approach presented in Section 8.4.4. In Table 8.2, we show how the average runtime is modified using up to 4 threads and the speedup for each test. At each run, we averaged the runtimes for each circuit size. The test shows that the improvement in runtime is linear to the number of cores used.

TABLE 8.2: Enumerator runtimes using different number of threads.

	1 thread	*2 threads*		*3 threads*		*4 threads*	
#gates	avg. runtime [s]	avg. runtime [s]	speedup	avg. runtime [s]	speedup	avg. runtime [s]	speedup
1	<0.001	<0.001	0.969	<0.001	1.048	<0.001	0.981
2	0.001	<0.001	1.261	<0.001	1.521	<0.001	1.733
3	0.004	0.002	1.618	0.001	1.994	0.001	2.125
4	0.213	0.107	1.976	0.077	2.752	0.072	2.943
5	1.674	0.923	1.890	0.702	2.485	0.623	2.798
6	28.189	14.039	2.007	8.726	3.230	7.089	3.976

8.6 SUMMARY

Although the SAT-based approach for implicit enumeration can take advantage of highly optimized SAT solvers, the explicit enumeration-based algorithm can be used as an easy-to-use tool for the enumeration of entire sets at once of minimal circuits using a set of given gates. Thanks to the simplicity of the circuit representation, it is possible to detect equivalences between sequences of internal gates in each circuit. This gives the possibility to reduce the search space such that an unoptimized implementation of the technique, containing a reduced set of simplification rules, can compete with state-of-the-art tools for small instances. The simplicity of parallelization gives many opportunities for future experiments using a machine containing a large number of cores.

This technique can be used to reduce the size of the circuits to be synthesized. Minimizing the size of synthesized circuits is useful for any application, but it is particularly important for our application in runtime monitoring. Having the smallest footprint possible reduces the chances of a malfunctioning runtime monitor due to Single Event Upsets (SEUs).

In this part, we described the overall idea for the automatic generation of runtime monitors from an existing design. As explained in Chapter 1, runtime monitors can be integrated into our general fault detection system to detect faults at the firmware level. The faults detected by the overall flow described are hardware-related. In order to detect programming errors, we need to use the list of properties obtained by the technique described in Chapter 6 to perform regression testing. Alternatively, check whether the properties obtained describe the specification written during the design phase using techniques that translate natural language to formal specification [Har12; ZH19; KH19].

With this chapter, we also conclude the technical section of this thesis. In the next chapter, we will give an overview of the entire thesis, a description of the achievements and a brief description of the ideas for future work.

[Har12] Harris, "Extracting design information from natural language specifications"

[ZH19] Zhao and Harris, "Automatic Assertion Generation from Natural Language Specifications Using Subtree Analysis"

[KH19] Keszöcze and Harris, "Chatbot-based assertion generation from natural language specifications"

Part IV

EPILOGUE

9

Conclusion

"An expert is a person who has made all the mistakes that can be made in a very narrow field."
—Niels Bohr

In this thesis, we presented a collection of results that aim at improving the fault detection capabilities of current techniques in particle accelerators and, in general, in modern distributed systems. Modern distributed systems have reached a level of integration and complexity that creates a fault space too large to be completely explored during testing. Furthermore, techniques like model checking, when feasible, can only catch design issues and not malfunctions of components. Modern particle accelerators are extreme examples of complex distributed systems. If we consider the European XFEL, the performance requirements are very challenging to reach, e.g., RF amplitude stability needs to be $<0.01\,\%$ and synchronization needs to be in the order of femtoseconds across the 3.4 km. Moreover, availability and flexibility need to be maintained while guaranteeing safe operation, e.g., normal and special modes of operations need to have short setup and tuning times. Adding to the previous points the push toward autonomous accelerators and the constantly changing requirements of such machines, a fault detection system that can detect a wide set of problems, even if never observed before, is of paramount importance.

This thesis is the first step toward the realization of a fault detection system for the European XFEL. In Chapter 1, we introduced the basic architecture of the new fault detection system, the components, and how the architecture can be integrated into the existing system. The architecture described is capable of detecting faults in the digital part of the LLRF system by using a mix of runtime monitoring and anomaly detection. With the combined usage of both techniques, we can monitor the system having the guarantees of a semi-formal approach like runtime monitoring while maintaining the flexibility of a data-driven approach like anomaly detection.

In Part II, we described the work done in the field of anomaly detection. With our tests we have shown that the usage of anomaly detection has several advantages:

1. early identification of issues: anomaly detection can identify unusual patterns or deviations in data that may indicate a problem or issue. This allows for early intervention, which can potentially prevent or mitigate the impact of the problem;
2. automation: anomaly detection can be automated, which allows for continuous monitoring of data and identification of issues in real time;

3. handling of large and complex data: anomaly detection can be applied to large and complex data sets, making it well-suited for use in big data environments;
4. fewer false alarms: anomaly detection can be more precise than traditional rule-based systems in identifying actual problems, reducing the number of false alarms.

Moreover, we have shown that even in complex systems, very simple and fast algorithms for time point anomaly detection are sufficient for detecting a wide variety of faults. Finally, in Part II, we have tested two algorithms for time-series anomaly detection, their use for the detection of more complex issues, and a possible solution for the hardware implementation of CNN-based algorithms.

In Part III, we described a set of techniques that can be combined for creating runtime monitors from existing HDL designs. Runtime monitoring has several advantages, including:

1. proactive maintenance: it allows identify potential issues and take action before they lead to system failure, enabling proactive maintenance and reducing downtime;
2. debugging: it helps the developer identify the exact point of error in the code;
3. real-time monitoring: monitoring the system in real-time can help you identify and diagnose issues as they happen, instead of waiting for an end-user to report a problem.

9.1 ACHIEVEMENTS

The results presented in this dissertation contribute to the development of methods for enhancing the reliability and availability of digital systems governing the Low-Level Radio Frequency (LLRF). We addressed a series of complex problems, each contributing collectively to the achievement of specific goals. The literature offers a wide variety of algorithms for anomaly detection. Tackling the first two problems (Problems 1 and 2) of selecting an appropriate time point anomaly detection algorithm and an appropriate time series anomaly detection algorithm for fault detection in a specific complex distributed system was crucial. Having tackled this problem, we reached our first goal (Goal 1) of algorithm selection and adaptation, leveraging advanced machine learning and statistical methods. Addressing the challenge of selecting the right implementation platform for hardware acceleration of a convolutional neural network (CNN) (Problem 3), significantly contributed to our second goal of latency optimization (Goal 2) and, simultaneously, enhanced the opportunities for quickly correlating multiple events (Goal 5), enabling efficient processing of complex event correlations and enhancing the system's analytical ability.

Further, we generated a readable set of assertions for an existing system, addressing Problem 4, and created a runtime monitor capable of predicting future specification violations, addressing Problem 5. This automatic specification generation enable us to streamline the runtime monitor deployment, thus contributing to Goals 3 and 4 of automated runtime monitor generation

and real-time parameter monitoring. This ensured continuous monitoring of critical parameters.

In conclusion, by systematically addressing these intricate problems, this dissertation has not only advanced the field of LLRF system management but also laid a solid foundation for future research and development. The methodologies and solutions developed here pave the way for more resilient, efficient, and intelligent digital systems, marking a significant step forward in our increasingly digital world. In conclusion, this dissertation represents a comprehensive effort to solve critical problems in LLRF system management.

9.2 FUTURE WORK

Future work requires the development of a fully autonomous self-healing system. The goal of the self-healing process is to initiate a procedure that can bring the system state back to a healthy state. The first practical approach is to create a database of faults and their solutions. However, due to the complexity and evolving nature of the system, not all faults can be contained in a database. Instead, the healing system needs to have a list of standard resolution methods that can be applied starting from the fault root cause location. The principle is to try healing the system starting with the measures having the least impact. An example of such a list of resolution methods for a PCI Express (PCIe) bus is the following:

1. reset of the hardware transceiver;
2. reset of IP cores controlling the transceiver;
3. reset of failing board;
4. partial reconfiguration of the board;
5. operator intervention request.

The integration of a self-healing system combined with techniques for predictive maintenance is essential for creating the necessary conditions for a fully autonomous particle accelerator.

Bibliography

[Axi] *AMBA AXI and ACE – Protocol Specification*. Standard. Arm Limited, June 2003 (cit. on p. 25).

[Aba16] Martín Abadi. "TensorFlow: learning functions at scale". In: *Proceedings of the 21st ACM SIGPLAN International Conference on Functional Programming, ICFP 2016, Nara, Japan, September 18-22, 2016*. Ed. by Jacques Garrigue, Gabriele Keller, and Eijiro Sumii. ACM, 2016, p. 1. DOI: 10.1145/2951913.2976746 (cit. on p. 63).

[Abe05] Shigeo Abe. *Support Vector Machines for Pattern Classification*. Advances in Pattern Recognition. Springer, 2005. ISBN: 978-1-85233-929-6. DOI: 10.1007/1-84628-219-5 (cit. on p. 37).

[Abe+06] Rafael Abela, Arthur Aghababyan, Massimo Altarelli, Carlo Altucci, Gayane A. Amatuni, Philip Anfinrud, Patrick Audebert, Valeri Ayvazyan, Nicoleta Baboi, et al. *XFEL: The European X-Ray Free-Electron Laser - Technical Design Report*. Hamburg: DESY, 2006, pp. 1–646. ISBN: 978-3-935702-17-1. DOI: 10.3204/DESY_06-097 (cit. on p. 6).

[Ahm+19] Sagheer Ahmad, Sridhar Subramanian, Vamsi Boppana, Shankar Lakka, Fu-Hing Ho, Tomai Knopp, Juanjo Noguera, Gaurav Singh, and Ralph Wittig. "Xilinx First 7nm Device: Versal AI Core (VC1902)". In: *2019 IEEE Hot Chips 31 Symposium (HCS), Cupertino, CA, USA, August 18-20, 2019*. IEEE, 2019, pp. 1–28. DOI: 10.1109/HOTCHIPS.2019.8875639 (cit. on p. 59).

[Al-+22] Ahmad Al-Zoubi, Gianluca Martino, Fin Hendrik Bahnsen, Jun Zhu, Holger Schlarb, and Görschwin Fey. "CNN Implementation and Analysis on Xilinx Versal ACAP at European XFEL". In: *35th IEEE International System-on-Chip Conference, SOCC 2022, Belfast, United Kingdom, September 5-8, 2022*. Ed. by Sakir Sezer, Thomas Büchner, Jürgen Becker, Andrew Marshall, Fahad Siddiqui, Tanja Harbaum, and Kieran McLaughlin. IEEE, 2022, pp. 1–6. DOI: 10.1109/SOCC56010.2022.9908101 (cit. on pp. 15, 59, 128, 129).

[Alu+15] Rajeev Alur, Rastislav Bodík, Eric Dallal, Dana Fisman, Pranav Garg, Garvit Juniwal, Hadas Kress-Gazit, P. Madhusudan, Milo M. K. Martin, Mukund Raghothaman, Shamwaditya Saha, Sanjit A. Seshia, Rishabh Singh, Armando Solar-Lezama, Emina Torlak, and Abhishek Udupa. "Syntax-Guided Synthesis". In: *International Symposium on Dependable Software Engineering: Theories, Tools, and Applications (SETTA)*. 2015, pp. 1–25. DOI: 10.3233/978-1-61499-495-4-1 (cit. on pp. 16, 74, 75).

[Ama+17] Luca Gaetano Amarù, Mathias Soeken, Patrick Vuillod, Jiong Luo, Alan Mishchenko, Pierre-Emmanuel Gaillardon, Janet Olson, Robert K. Brayton, and Giovanni De Micheli. "Enabling exact delay synthesis". In: *2017 IEEE/ACM International Conference on Computer-Aided Design, ICCAD 2017, Irvine, CA, USA, November 13-16, 2017*. Ed. by Sri Parameswaran. IEEE, 2017, pp. 352–359. DOI: 10.1109/ICCAD.2017.8203799 (cit. on p. 97).

[ABL02] Glenn Ammons, Rastislav Bodík, and James R. Larus. "Mining Specifications". In: *ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages (POPL)*. 2002, pp. 4–16. DOI: 10.1145/503272.503275 (cit. on pp. 74, 75).

[AP02] Fabrizio Angiulli and Clara Pizzuti. "Fast Outlier Detection in High Dimensional Spaces". In: *Principles of Data Mining and Knowledge Discovery, 6th European Conference, PKDD 2002, Helsinki, Finland, August 19-23, 2002, Proceedings*. Ed. by Tapio Elomaa, Heikki Mannila, and Hannu Toivonen. Vol. 2431. Lecture Notes in Computer Science. Springer, 2002, pp. 15–26. DOI: 10.1007/3-540-45681-3_2 (cit. on p. 35).

[Avi+04] Algirdas Avizienis, Jean-Claude Laprie, Brian Randell, and Carl E. Landwehr. "Basic Concepts and Taxonomy of Dependable and Secure Computing". In: *IEEE Trans. Dependable Secur. Comput.* 1.1 (2004), pp. 11–33. DOI: 10.1109/TDSC.2004.2 (cit. on p. 4).

[Bab16] László Babai. "Graph Isomorphism in Quasipolynomial Time". In: *ACM Symposium on Theory of Computing*. ACM, 2016, pp. 684–697. ISBN: 9781450341325. DOI: 10.1145/2897518.2897542 (cit. on p. 99).

[BA19] Y.A. Bachtiar and T. Adiono. "Convolutional Neural Network and Maxpooling Architecture on Zynq SoC FPGA". In: *2019 International Symposium on Electronics and Smart Devices (ISESD)*. 2019, pp. 1–5. DOI: 10.1109/ISESD.2019.8909510 (cit. on p. 61).

[BK08] Christel Baier and Joost-Pieter Katoen. *Principles of model checking*. MIT Press, 2008, pp. 330–331. ISBN: 978-0-262-02649-9 (cit. on p. 73).

[BSS15] Tomás Balyo, Peter Sanders, and Carsten Sinz. "HordeSat: A Massively Parallel Portfolio SAT Solver". In: *18th International Conference on Theory and Applications of Satisfiability Testing - SAT 2015, Austin, TX, USA*. Ed. by Marijn Heule and Sean A. Weaver. Vol. 9340. Lecture Notes in Computer Science. Springer, 2015, pp. 156–172. DOI: 10.1007/978-3-319-24318-4_12 (cit. on p. 98).

[Bar+18] Ezio Bartocci, Roderick Bloem, Dejan Nickovic, and Franz Roeck. “A Counting Semantics for Monitoring LTL Specifications over Finite Traces”. In: *International Conference on Computer Aided Verification (CAV)*. 2018, pp. 547–564. DOI: 10.1007/978-3-319-96145-3_29 (cit. on pp. 13, 16, 86–88).

[BGS08] Arslan Basharat, Alexei Gritai, and Mubarak Shah. “Learning object motion patterns for anomaly detection and improved object detection”. In: *IEEE Conference on Computer Vision and Pattern Recognition*. 2008, pp. 1–8. DOI: 10.1109/CVPR.2008.4587510 (cit. on pp. 11, 32).

[BLS06] Andreas Bauer, Martin Leucker, and Christian Schallhart. “Monitoring of Real-Time Properties”. In: *Foundations of Software Technology and Theoretical Computer Science (FSTTCS)*. 2006, pp. 260–272. DOI: 10.1007/11944836_25 (cit. on pp. 85, 87).

[BLS10] Andreas Bauer, Martin Leucker, and Christian Schallhart. “Comparing LTL Semantics for Runtime Verification”. In: *Journal of Logic and Computation* 20.3 (2010), pp. 651–674. DOI: 10.1093/logcom/exn075 (cit. on p. 86).

[Bau+72] Charles R. Baugh, C. S. Chandersekaran, Richard S. Swee, and Saburo Muroga. “Optimal Networks of NOR-OR Gates for Functions of Three Variables”. In: *IEEE Trans. Computers* 21.2 (1972), pp. 153–160. DOI: 10.1109/TC.1972.5008920 (cit. on p. 98).

[Bel+21] Andrea Bellandi, Łukasz Butkowski, Burak Dursun, Annika Eichler, Çagıl Gümüş, Michael Kuntzsch, Ayla Nawaz, Sven Pfeiffer, Holger Schlarb, Christian Schmidt, Klaus Zenker, and Julien Branlard. “Online Detuning Computation and Quench Detection for Superconducting Resonators”. In: *IEEE Transactions on Nuclear Science* 68.4 (2021), pp. 385–393. DOI: 10.1109/TNS.2021.3067598 (cit. on p. 54).

[Ben+09] Jacob Benesty, Jingdong Chen, Yiteng Huang, and Israel Cohen. “Pearson Correlation Coefficient”. In: *Noise Reduction in Speech Processing*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009, pp. 1–4. ISBN: 978-3-642-00296-0. DOI: 10.1007/978-3-642-00296-0_5 (cit. on p. 40).

[BB04] Per Bjesse and Arne Borälv. “DAG-aware circuit compression for formal verification”. In: *2004 International Conference on Computer-Aided Design, ICCAD 2004, San Jose, CA, USA, November 7-11, 2004*. IEEE Computer Society / ACM, 2004, pp. 42–49. DOI: 10.1109/ICCAD.2004.1382541 (cit. on p. 97).

[Ble+15] Joaquim Blesa, Pedro Jimenez, Damiano Rotondo, Fatiha Nejjari, and Vicenç Puig. “An Interval NLPV Parity Equations Approach for Fault Detection and Isolation of a Wind Farm”. In: *IEEE Trans. Ind. Electron.* 62.6 (2015), pp. 3794–3805. DOI: 10.1109/TIE.2014.2386293 (cit. on p. 5).

[BH02] Richard J. Bolton and David J. Hand. “Statistical Fraud Detection: A Review”. In: *Statistical Science* 17.3 (2002), pp. 235–249. DOI: 10.1214/ss/1042727940 (cit. on pp. 11, 32).

[BZ08] Marc Boule and Zeljko Zilic. “Automata-based assertion-checker synthesis of PSL properties”. In: *ACM Transactions on Design Automation of Electronic Systems* 13.1 (2008), 4:1–4:21. DOI: 10.1145/1297666.1297670 (cit. on p. 86).

[Bra+12] Julien Branlard, Gohar Ayvazyan, Valeri Ayvazyan, MK Grecki, M Hoffmann, T Jezynski, IM Kudla, T Lamb, F Ludwig, U Mavric, et al. “The European XFEL LLRF system”. In: *IPAC* 12 (2012), pp. 55–57 (cit. on pp. 7, 32).

[Bra+96] Robert K. Brayton, Gary D. Hachtel, Alberto L. Sangiovanni - Vincentelli, Fabio Somenzi, Adnan Aziz, Szu - Tsung Cheng, Stephen A. Edwards, Sunil P. Khatri, Yuji Kukimoto, Abelardo Pardo, Shaz Qadeer, Rajeev K. Ranjan, Shaker Sarwary, Thomas R. Shiple, Gitanjali Swamy, and Tiziano Villa. “VIS: A System for Verification and Synthesis”. In: *International Conference on Computer Aided Verification (CAV)*. 1996, pp. 428–432. DOI: 10.1007/3-540-61474-5_95 (cit. on p. 79).

[BM10] Robert K. Brayton and Alan Mishchenko. “ABC: An Academic Industrial-Strength Verification Tool”. In: *22nd International Conference on Computer Aided Verification, CAV 2010, Edinburgh, UK, July 15-19*. Ed. by Tayssir Touili, Byron Cook, and Paul B. Jackson. Vol. 6174. Lecture Notes in Computer Science. Springer, 2010, pp. 24–40. DOI: 10.1007/978-3-642-14295-6_5 (cit. on pp. 17, 98, 104, 129).

[Bre+00] Markus M. Breunig, Hans-Peter Kriegel, Raymond T. Ng, and Jörg Sander. “LOF: Identifying Density-Based Local Outliers”. In: *ACM SIGMOD International Conference on Management of Data*. 2000, pp. 93–104. ISBN: 1581132174. DOI: 10.1145/342009.335388 (cit. on p. 35).

[Bro20] Jason Brownlee. *A Gentle Introduction to K-Fold Cross-Validation*. 2020. URL: https://machinelearningmastery.com/k-fold-cross-validation/ (visited on 09/30/2020) (cit. on p. 43).

[Buo19] Caitlyn Buongiorno. “The Future of Particle Accelerators May Be Autonomous”. In: *Symmetry Magazine* (2019). URL: https://www.symmetrymagazine.org/article/the-future-of-particle-accelerators-may-be-autonomous (visited on 09/30/2022) (cit. on p. 7).

[But+15] Lukasz Butkowski, Tomasz Kozak, Bin Yang, Paweł Prędki, and Radoslaw Rybaniec. "FPGA firmware framework for MTCA.4 AMC modules". In: *15th International Conference on Accelerator and Large Experimental Physics Control Systems (ICALEPCS 2015), Melbourne, Australia*. 2015, pp. 17–23. DOI: 10.18429/JACoW-ICALEPCS2015-WEPGF074 (cit. on p. 24).

[CT04] Rama Calaga and Rogelio Tomás. "Statistical Analysis of RHIC Beam Position Monitors Performance". In: *Physical Review Special Topics-Accelerators and Beams* 7.4 (2004), p. 042801. DOI: 10.1103/PhysRevSTAB.7.042801 (cit. on p. 33).

[Cao+20] Keyan Cao, Yefan Liu, Gongjie Meng, and Qimeng Sun. "An Overview on Edge Computing Research". In: *IEEE Access* 8 (2020), pp. 85714–85728. DOI: 10.1109/ACCESS.2020.2991734 (cit. on p. 10).

[Cha+09] Mark S. Champion, Lance D. Cooley, Camille M. Ginsburg, Dmitri A. Sergatskov, Rongli L. Geng, Hitoshi Hayano, Yoshihisa Iwashita, and Yujiro Tajima. "Quench-Limited SRF Cavities: Failure at the Heat-Affected Zone". In: *IEEE Transactions on Applied Superconductivity* 19.3 (2009), pp. 1384–1386. DOI: 10.1109/TASC.2009.2019204 (cit. on p. 54).

[CP00] Chi-Ho Chan and G.K.H. Pang. "Fabric defect detection by Fourier analysis". In: *IEEE Transactions on Industry Applications* 36.5 (2000), pp. 1267–1276. DOI: 10.1109/28.871274 (cit. on p. 5).

[CBK09] Varun Chandola, Arindam Banerjee, and Vipin Kumar. "Anomaly detection: A survey". In: *ACM Comput. Surv.* 41.3 (2009), 15:1–15:58. DOI: 10.1145/1541880.1541882 (cit. on pp. 11, 32).

[Cha+10] Yin-Wen Chang, Cho-Jui Hsieh, Kai-Wei Chang, Michael Ringgaard, and Chih-Jen Lin. "Training and Testing Low-degree Polynomial Data Mappings via Linear SVM". In: *Journal of Machine Learning Research* 11 (2010), pp. 1471–1490. DOI: 10.5555/1756006.1859899 (cit. on p. 37).

[Cin+16] Marcello Cinque, Domenico Cotroneo, Raffaele Della Corte, and Antonio Pecchia. "Characterizing Direct Monitoring Techniques in Software Systems". In: *IEEE Transactions on Reliability* 65.4 (2016), pp. 1665–1681. DOI: 10.1109/TR.2016.2570564 (cit. on p. 86).

[CE81] Edmund M. Clarke and E. Allen Emerson. "Design and Synthesis of Synchronization Skeletons Using Branching-Time Temporal Logic". In: *Logics of Programs*. 1981, pp. 52–71. DOI: 10.1007/BFb0025774 (cit. on pp. 26, 72).

[CGP01] Edmund M. Clarke, Orna Grumberg, and Doron A. Peled. *Model checking, 1st Edition*. MIT Press, 2001. ISBN: 978-0-262-03270-4 (cit. on p. 27).

[CV03] Edmund M. Clarke and Helmut Veith. "Counterexamples Revisited: Principles, Algorithms, Applications". In: *Verification: Theory and Practice, Essays Dedicated to Zohar Manna on the Occasion of His 64th Birthday*. 2003, pp. 208–224. DOI: 10.1007/978-3-540-39910-0_9 (cit. on p. 78).

[Cou+09] Philippe Coussy, Daniel D. Gajski, Michael Meredith, and Andrés Takach. "An Introduction to High-Level Synthesis". In: *IEEE Design & Test of Computers* 26.4 (2009), pp. 8–17. DOI: 10.1109/MDT.2009.69 (cit. on p. 86).

[Cul+79] Jay Niel Culliney, Ming Huei Young, Tomoyasu Nakagawa, and Saburo Muroga. "Results of the Synthesis of Optimal Networks of AND and OR Gates for Four-Variable Switching Functions". In: *IEEE Transactions on Computers* 28.1 (1979), pp. 76–85. DOI: 10.1109/TC.1979.1675229 (cit. on p. 98).

[Dai+20] Zhiyong Dai, Jianjun Yi, Yajun Zhang, Bo Zhou, and Liang He. "Fast and accurate cable detection using CNN". In: *Appl. Intell.* 50.12 (2020), pp. 4688–4707. DOI: 10.1007/s10489-020-01746-9 (cit. on pp. 12, 59).

[DF96] Dipankar Dasgupta and Stephanie Forrest. "Novelty detection in time series data using ideas from immunology". In: *International conference on intelligent systems*. 1996, pp. 82–87 (cit. on pp. 11, 14, 32).

[DN00] Dipankar Dasgupta and Fernando Nino. "A comparison of negative and positive selection algorithms in novel pattern detection". In: *IEEE International Conference on Systems, Man, and Cybernetics*. Vol. 1. 2000, pp. 125–130. DOI: 10.1109/ICSMC.2000.884976 (cit. on pp. 11, 14, 32).

[Dav69] Edward S. Davidson. "An Algorithm for NAND Decomposition Under Network Constraints". In: *IEEE Transactions on Computers* 18.12 (1969), pp. 1098–1109. DOI: 10.1109/T-C.1969.222593 (cit. on p. 98).

[DeO+09] Andrew DeOrio, Adam Bauserman, Valeria Bertacco, and Beth Isaksen. "Inferno: Streamlining Verification With Inferred Semantics". In: *Transactions on Computer Aided Design of Circuits and Systems (TCAD)* 28.5 (2009), pp. 728–741. DOI: 10.1109/TCAD.2009.2013995 (cit. on pp. 13, 74, 75).

[DGR04] Nelly Delgado, Ann Q. Gates, and Steve Roach. "A Taxonomy and Catalog of Runtime Software-Fault Monitoring Tools". In: *IEEE Transactions on Software Engineering* 30.12 (2004), pp. 859–872. DOI: 10.1109/TSE.2004.91 (cit. on p. 86).

[DPW20] Angus Dempster, François Petitjean, and Geoffrey I. Webb. “ROCKET: exceptionally fast and accurate time series classification using random convolutional kernels”. In: *Data Min. Knowl. Discov.* 34.5 (2020), pp. 1454–1495. DOI: 10.1007/s10618-020-00701-z (cit. on pp. 55, 56).

[Din+14] Xuemei Ding, Yuhua Li, Ammar Belatreche, and Liam P. Maguire. “An experimental evaluation of novelty detection methods”. In: *Neurocomputing* 135 (2014), pp. 313–327. ISSN: 0925-2312. DOI: 10.1016/j.neucom.2013.12.002 (cit. on pp. 14, 33).

[Dom+20] Rémi Domingues, Pietro Michiardi, Jérémie Barlet, and Maurizio Filippone. “A Comparative Evaluation of Novelty Detection Algorithms for Discrete Sequences”. In: *Artificial Intelligence Review* 53.5 (2020), pp. 3787–3812. ISSN: 1573-7462. DOI: 10.1007/s10462-019-09779-4 (cit. on pp. 14, 33).

[DG98] Rolf Drechsler and Wolfgang Günther. “Exact circuit synthesis”. In: *International Workshop on Logic Synthesis (IWLS)*. 1998 (cit. on p. 98).

[Dru03] Doron Drusinsky. “Monitoring Temporal Rules Combined with Time Series”. In: *Computer Aided Verification, 15th International Conference, CAV 2003, Boulder, CO, USA, July 8-12, 2003, Proceedings*. Ed. by Warren A. Hunt Jr. and Fabio Somenzi. Vol. 2725. Lecture Notes in Computer Science. Springer, 2003, pp. 114–117. DOI: 10.1007/978-3-540-45069-6_11 (cit. on p. 13).

[DG17] Dheeru Dua and Casey Graff. *UCI Machine Learning Repository*. 2017. URL: https://archive.ics.uci.edu (cit. on pp. 45, 52).

[Dua+18] Javier Duarte et al. “Fast inference of deep neural networks in FPGAs for particle physics”. In: *JINST* 13.07 (2018), P07027. DOI: 10.1088/1748-0221/13/07/P07027. arXiv: 1804.06913 [physics.ins-det] (cit. on p. 68).

[DL+16] Alexandre Duret-Lutz, Alexandre Lewkowicz, Amaury Fauchille, Thibaud Michaud, Etienne Renault, and Laurent Xu. “Spot 2.0 — a framework for LTL and ω-automata manipulation”. In: *International Symposium on Automated Technology for Verification and Analysis (ATVA)*. Vol. 9938. Lecture Notes in Computer Science. Springer, 2016, pp. 122–129. DOI: 10.1007/978-3-319-46520-3_8 (cit. on p. 92).

[EBT23] Annika Eichler, Julien Branlard, and Jan H. K. Timm. “Anomaly detection at the European X-ray Free Electron Laser using a parity-space-based method”. In: *Phys. Rev. Accel. Beams* 26 (1 Jan. 2023), p. 012801. DOI: 10.1103/PhysRevAccelBeams.26.012801 (cit. on pp. 9, 15, 56).

[Eic+21] Annika Eichler, Florian Burkart, Jan Kaiser, Willi Kuropka, Oliver Stein, Erik Bründermann, Andrea Santamaria Garcia, and Chenran Xu. “First Steps Toward an Autonomous Accelerator, a Common Project Between DESY and KIT”. In: *12th International Particle Accelerator Conference (IPAC 2021), Campinas, Brazil*. 2021. DOI: 10.18429/JACoW-IPAC2021-TUPAB298 (cit. on p. 7).

[Eis+03] Cindy Eisner, Dana Fisman, John Havlicek, Yoad Lustig, Anthony McIsaac, and David Van Campenhout. “Reasoning with Temporal Logic on Truncated Paths”. In: *International Conference on Computer Aided Verification (CAV)*. 2003, pp. 27–39. DOI: 10.1007/978-3-540-45069-6_3 (cit. on pp. 85, 87).

[EH86] E. Allen Emerson and Joseph Y. Halpern. “"Sometimes" and "Not Never" revisited: on branching versus linear time temporal logic”. In: *J. ACM* 33.1 (1986), pp. 151–178. DOI: 10.1145/4904.4999 (cit. on p. 72).

[Emm+16] Andrew Emmott, Shubhomoy Das, Thomas Dietterich, Alan Fern, and Weng-Keen Wong. “A Meta-Analysis of the Anomaly Detection Problem”. In: *arXiv e-prints*, arXiv:1503.01158 (2016), pp. 1–35. DOI: 10.48550/arXiv.1503.01158. arXiv: 1503.01158 [cs.AI] (cit. on pp. 11, 32).

[Ern09] Elizabeth Ann Ernst. “Optimal combinational multi-level logic synthesis”. PhD thesis. University of Michigan, 2009 (cit. on pp. 17, 97, 98).

[Ern+07] Michael D. Ernst, Jeff H. Perkins, Philip J. Guo, Stephen McCamant, Carlos Pacheco, Matthew S. Tschantz, and Chen Xiao. “The Daikon system for dynamic detection of likely invariants”. In: *Science of Computer Programming* 69.1 (2007). Special issue on Experimental Software and Toolkits, pp. 35–45. ISSN: 0167-6423. DOI: 10.1016/j.scico.2007.01.015 (cit. on pp. 74, 75).

[Fas21] FastML Team. *fastmachinelearning/hls4ml*. 2021. DOI: 10.5281/zenodo.1201549 (cit. on p. 68).

[Faw+20] Hassan Ismail Fawaz, Benjamin Lucas, Germain Forestier, Charlotte Pelletier, Daniel F. Schmidt, Jonathan Weber, Geoffrey I. Webb, Lhassane Idoumghar, Pierre-Alain Muller, and François Petitjean. “InceptionTime: Finding AlexNet for time series classification”. In: *Data Min. Knowl. Discov.* 34.6 (2020), pp. 1936–1962. DOI: 10.1007/s10618-020-00710-y (cit. on p. 55).

[FFP07] Andrea Fedeli, Franco Fummi, and Graziano Pravadelli. “Properties incompleteness evaluation by functional verification”. In: *IEEE Transactions on Computers* 56.4 (2007), pp. 528–544. DOI: 10.1109/TC.2007.1012 (cit. on p. 83).

[Fey+18] Görschwin Fey, Tara Ghasempouri, Swen Jacobs, Gianluca Martino, Jaan Raik, and Heinz Riener. "Design Understanding: From Logic to Specification". In: *IFIP/IEEE International Conference on Very Large Scale Integration, VLSI-SoC 2018, Verona, Italy, October 8-10, 2018*. IEEE, 2018, pp. 172–175. DOI: 10.1109/VLSI-SoC.2018.8644732 (cit. on pp. 16, 71, 72, 127, 128).

[FHS17] Petr Fiser, Ivo Hálecek, and Jan Schmidt. "SAT -Based Generation of Optimum Function Implementations with XOR Gates". In: *Euromicro Conference on Digital System Design (DSD)*. IEEE. 2017, pp. 163–170. DOI: 10.1109/DSD.2017.74 (cit. on pp. 17, 98).

[Fol+20] Elena Fol, Rogelio Tomás, J. Coello de Portugal, and Giuliano Franchetti. "Detection of Faulty Beam Position Monitors using Unsupervised Learning". In: *Phys. Rev. Accel. Beams* 23 (10 Oct. 2020), p. 102805. DOI: 10.1103/PhysRevAccelBeams.23.102805 (cit. on p. 33).

[FHS97] Stephanie Forrest, Steven A. Hofmeyr, and Anil Somayaji. "Computer Immunology". In: *Commun. ACM* 40.10 (1997), pp. 88–96. DOI: 10.1145/262793.262811 (cit. on p. 10).

[FS15] Adrian Francalanza and Aldrin Seychell. "Synthesising correct concurrent runtime monitors". In: *Formal Methods Syst. Des.* 46.3 (2015), pp. 226–261. DOI: 10.1007/s10703-014-0217-9 (cit. on p. 13).

[GT+09] Pedro Garcia-Teodoro, Jesus Diaz-Verdejo, Gabriel Maciá-Fernández, and Enrique Vázquez. "Anomaly-based network intrusion detection: Techniques, systems and challenges". In: *Computers & Security* 28.1 (2009), pp. 18–28. DOI: 10.1016/j.cose.2008.08.003 (cit. on pp. 11, 32).

[Geb+13] Johann Gebhardt, Markus Goldstein, Faisal Shafait, and Andreas Dengel. "Document Authentication Using Printing Technique Features and Unsupervised Anomaly Detection". In: *International Conference on Document Analysis and Recognition*. 2013, pp. 479–483. DOI: 10.1109/ICDAR.2013.102 (cit. on pp. 11, 32).

[GP15] T. Ghasempouri and G. Pravadelli. "On the estimation of assertion interestingness". In: *IFIP/IEEE International Conference on Very Large Scale Integration (VLSI-SoC)*. 2015, pp. 325–330. DOI: 10.1109/VLSI-SoC.2015.7314438 (cit. on pp. 73, 83).

[Gho+07] Debanjan Ghosh, Raj Sharman, H. Raghav Rao, and Shambhu J. Upadhyaya. "Self-healing systems - survey and synthesis". In: *Decis. Support Syst.* 42.4 (2007), pp. 2164–2185. DOI: 10.1016/j.dss.2006.06.011 (cit. on p. 5).

[GV13] Giuseppe De Giacomo and Moshe Y. Vardi. "Linear Temporal Logic and Linear Dynamic Logic on Finite Traces". In: *International Joint Conference on Artificial Intelligence (IJCAI)*. IJCAI/AAAI, 2013, pp. 854–860 (cit. on p. 85).

[GK95] Dina Q. Goldin and Paris C. Kanellakis. "On Similarity Queries for Time-Series Data: Constraint Specification and Implementation". In: *International Conference on Principles and Practice of Constraint Programming*. 1995, pp. 137–153. DOI: 10.1007/3-540-60299-2_9 (cit. on p. 45).

[GD12] Markus Goldstein and Andreas Dengel. "Histogram-based outlier score (HBOS): A fast unsupervised anomaly detection algorithm". In: *KI-2012: Poster and Demo Track*. 2012, pp. 59–63 (cit. on p. 36).

[GU16] Markus Goldstein and Seiichi Uchida. "A comparative evaluation of unsupervised anomaly detection algorithms for multivariate data". In: *PLoS One* 11.4 (2016). ISSN: 1932-6203. DOI: 10.1371/journal.pone.0152173 (cit. on pp. 14, 33, 34).

[Grü+21] Arne Grünhagen, Julien Branlard, Annika Eichler, Gianluca Martino, Görschwin Fey, and Marina Tropmann-Frick. "Fault Analysis of the Beam Acceleration Control System at the European XFEL using Data Mining". In: *30th IEEE Asian Test Symposium, ATS 2021, Matsuyama, Ehime, Japan, November 22-25, 2021*. IEEE, 2021, pp. 61–66. DOI: 10.1109/ATS52891.2021.00023 (cit. on pp. 31, 128, 129).

[Guo+18] Kaiyuan Guo, Lingzhi Sui, Jiantao Qiu, Jincheng Yu, Junbin Wang, Song Yao, Song Han, Yu Wang, and Huazhong Yang. "Angel-Eye: A Complete Design Flow for Mapping CNN Onto Embedded FPGA". In: *IEEE Trans. Comput. Aided Des. Integr. Circuits Syst.* 37.1 (2018), pp. 35–47. DOI: 10.1109/TCAD.2017.2705069 (cit. on p. 61).

[Gup+14] Manish Gupta, Jing Gao, Charu C. Aggarwal, and Jiawei Han. "Outlier Detection for Temporal Data: A Survey". In: *IEEE Transactions on Knowledge and Data Engineering* 26.9 (2014), pp. 2250–2267. DOI: 10.1109/TKDE.2013.184 (cit. on p. 33).

[Haa+18] Winston Haaswijk, Alan Mishchenko, Mathias Soeken, and Giovanni De Micheli. "SAT Based Exact Synthesis Using DAG Topology Families". In: *Design Automation Conference (DAC)*. ACM, 2018. ISBN: 9781450357005. DOI: 10.1145/3195970.3196111 (cit. on p. 98).

[Hai+17] Guo Haixiang, Li Yijing, Jennifer Shang, Gu Mingyun, Huang Yuanyue, and Gong Bing. "Learning from class-imbalanced data: Review of methods and applications". In: *Expert Systems with Applications* 73 (2017), pp. 220–239. ISSN: 0957-4174. DOI: 10.1016/j.eswa.2016.12.035 (cit. on p. 46).

[HJS09] Youssef Hamadi, Saïd Jabbour, and Lakhdar Sais. “ManySAT: a Parallel SAT Solver”. In: *J. Satisf. Boolean Model. Comput.* 6.4 (2009), pp. 245–262. DOI: 10.3233/SAT190070 (cit. on p. 98).

[HW12] Youssef Hamadi and Christoph M. Wintersteiger. “Seven Challenges in Parallel SAT Solving”. In: *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence, July 22-26, 2012, Toronto, Ontario, Canada*. AAAI Press, 2012, pp. 2120–2125. DOI: 10.1609/AAAI.V26I1.8438 (cit. on p. 98).

[HT01] David J. Hand and Robert J. Till. “A Simple Generalisation of the Area Under the ROC Curve for Multiple Class Classification Problems”. In: *Machine Learning* 45.2 (Nov. 2001), pp. 171–186. ISSN: 1573-0565. DOI: 10/bm6gn4 (cit. on p. 46).

[Har12] Ian G. Harris. “Extracting design information from natural language specifications”. In: *The 49th Annual Design Automation Conference 2012, DAC ’12, San Francisco, CA, USA, June 3-7, 2012*. Ed. by Patrick Groeneveld, Donatella Sciuto, and Soha Hassoun. ACM, 2012, pp. 1256–1257. DOI: 10.1145/2228360.2228591 (cit. on p. 106).

[HBS12] Zyad Hassan, Aaron R. Bradley, and Fabio Somenzi. “Incremental, Inductive CTL Model Checking”. In: *International Conference on Computer Aided Verification (CAV)*. Ed. by P. Madhusudan and Sanjit A. Seshia. Springer, 2012, pp. 532–547. DOI: 10.1007/978-3-642-31424-7_38 (cit. on pp. 72, 79).

[HTF09] Trevor Hastie, Robert Tibshirani, and Jerome H. Friedman. *The Elements of Statistical Learning: Data Mining, Inference, and Prediction, 2nd Edition*. Springer Series in Statistics. Springer, 2009. ISBN: 9780387848570. DOI: 10.1007/978-0-387-84858-7 (cit. on pp. 12, 56).

[HR04] Klaus Havelund and Grigore Rosu. “An Overview of the Runtime Verification Tool Java PathExplorer”. In: *Formal Methods Syst. Des.* 24.2 (2004), pp. 189–215. DOI: 10.1023/B:FORM.0000017721.39909.4b (cit. on p. 13).

[HXD02] Zengyou He, Xiaofei Xu, and Shengchun Deng. “Squeezer: An Efficient Algorithm for Clustering Categorical Data”. In: *Journal of Computer Science and Technology* 17.5 (Sept. 2002), pp. 611–624. ISSN: 1860-4749. DOI: 10/dckj5n (cit. on p. 36).

[HXD03] Zengyou He, Xiaofei Xu, and Shengchun Deng. “Discovering cluster-based local outliers”. In: *Pattern Recognit. Lett.* 24.9-10 (2003), pp. 1641–1650. DOI: 10.1016/S0167-8655(03)00003-5 (cit. on p. 36).

[Hel63] Leo Hellerman. “A Catalog of Three-Variable Or-Invert and And-Invert Logical Circuits”. In: *IEEE Trans. Electron. Comput.* 12.3 (1963), pp. 198–223. DOI: 10.1109/PGEC.1963.263531 (cit. on p. 98).

[HSV13] Samuel Hertz, David Sheridan, and Shobha Vasudevan. “Mining Hardware Assertions With Guidance From Static Analysis”. In: *Transactions on Computer Aided Design of Circuits and Systems (TCAD)* 32.6 (2013), pp. 952–965. DOI: 10.1109/TCAD.2013.2241176 (cit. on pp. 74, 75, 79, 80, 83).

[HA04] Victoria Hodge and Jim Austin. “A Survey of Outlier Detection Methodologies”. In: *Artificial Intelligence Review* 22.2 (2004), pp. 85–126. ISSN: 1573-7462. DOI: 10/fsrmtg (cit. on pp. 11, 32).

[Hos+99] Yatin Vasant Hoskote, Timothy Kam, Pei-Hsin Ho, and Xudong Zhao. “Coverage Estimation for Symbolic Model Checking”. In: *Design Automation Conference (DAC)*. 1999, pp. 300–305. DOI: 10.1145/309847.309936 (cit. on p. 83).

[II+20] Tommy Tracy II, Lucas M. Tabajara, Moshe Y. Vardi, and Kevin Skadron. “Runtime Verification on FPGAs with LTLf Specifications”. In: *2020 Formal Methods in Computer Aided Design, FMCAD 2020, Haifa, Israel, September 21-24, 2020*. IEEE, 2020, pp. 36–46. DOI: 10.34727/2020/isbn.978-3-85448-042-6_10 (cit. on pp. 13, 86).

[JCDLT13] László A. Jeni, Jeffrey F. Cohn, and Fernando De La Torre. “Facing Imbalanced Data-Recommendations for the Use of Performance Metrics”. In: *Humaine Association Conference on Affective Computing and Intelligent Interaction*. 2013, pp. 245–251. DOI: 10.1109/ACII.2013.47 (cit. on p. 46).

[Kar+61] Richard M. Karp, F. E. McFarlin, J. Paul Roth, and J. R. Wilts. “A computer program for the synthesis of combinational switching circuits”. In: *2nd Annual Symposium on Switching Circuit Theory and Logical Design, Detroit, Michigan, USA, October 17-20, 1961*. IEEE Computer Society, 1961, pp. 182–194. DOI: 10.1109/FOCS.1961.1 (cit. on pp. 17, 97).

[KGG99] Sagi Katz, Orna Grumberg, and Danny Geist. ““Have I Written Enough Properties?” - A Method of Comparison Between Specification and Implementation”. In: *Correct Hardware Design and Verification Methods (CHARME)*. 1999, pp. 280–297. DOI: 10.1007/3-540-48153-2_21 (cit. on p. 83).

[KH19] Oliver Keszöcze and Ian G. Harris. “Chatbot-based assertion generation from natural language specifications”. In: *2019 Forum for Specification and Design Languages, FDL 2019, Southampton, United Kingdom, September 2-4, 2019*. Ed. by Tom J. Kazmierski, Reinhard von Hanxleden, and Terrence S. T. Mak. IEEE, 2019, pp. 1–6. DOI: 10.1109/FDL.2019.8876925 (cit. on p. 106).

[Kim86] Kwang-Je Kim. “An analysis of self-amplified spontaneous emission”. In: *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment* 250.1 (1986), pp. 396–403. ISSN: 0168-9002. DOI: 10.1016/0168-9002(86)90916-2 (cit. on p. 20).

[Kim+04] Moonzoo Kim, Mahesh Viswanathan, Sampath Kannan, Insup Lee, and Oleg Sokolsky. “Java-MaC: A Run-Time Assurance Approach for Java Programs”. In: *Formal Methods Syst. Des.* 24.2 (2004), pp. 129–155. DOI: 10.1023/B:FORM.0000017719.43755.7c (cit. on p. 13).

[Kin+02] Stephen P. King, Dennis King, K. Astley, Lionel Tarassenko, Paul Hayton, and Simukai W. Utete. “The Use of Novelty Detection Techniques for Monitoring High-Integrity Plant”. In: *International Conference on Control Applications*. Vol. 1. 2002, pp. 221–226. DOI: 10.1109/CCA.2002.1040189 (cit. on pp. 11, 14, 32).

[KKY09] Arist Kojevnikov, Alexander S. Kulikov, and Grigory Yaroslavtsev. “Finding Efficient Circuits Using SAT-Solvers”. In: *Theory and Applications of Satisfiability Testing - SAT 2009, 12th International Conference, SAT 2009, Swansea, UK, June 30 - July 3, 2009. Proceedings*. Ed. by Oliver Kullmann. Vol. 5584. Lecture Notes in Computer Science. Springer, 2009, pp. 32–44. DOI: 10.1007/978-3-642-02777-2_5 (cit. on p. 98).

[KSZ08] Hans-Peter Kriegel, Matthias Schubert, and Arthur Zimek. “Angle-Based Outlier Detection in High-Dimensional Data”. In: *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2008, pp. 444–452. ISBN: 9781605581934. DOI: 10.1145/1401890.1401946 (cit. on p. 36).

[KV03] Orna Kupferman and Moshe Y. Vardi. “Vacuity detection in temporal model checking”. In: *International Journal on Software Tools for Technology Transfer (STTT)* 4.2 (2003), pp. 224–233. DOI: 10.1007/s100090100062 (cit. on pp. 73–75).

[Lan+23] Leandro Lanzieri, Gianluca Martino, Görschwin Fey, Holger Schlarb, Thomas C. Schmidt, and Matthias Wählisch. *A Review of Techniques for Ageing Detection and Monitoring on Embedded Systems*. 2023. DOI: 10.48550/ARXIV.2301.06804 (cit. on pp. 128, 129).

[Law64] Eugene L. Lawler. “An Approach to Multilevel Boolean Minimization”. In: *J. ACM* 11.3 (1964), pp. 283–295. DOI: 10.1145/321229.321232 (cit. on p. 97).

[LA90] Peter Alan Lee and Thomas Anderson. “System Structure and Dependability”. In: *Fault Tolerance: Principles and Practice*. Vienna: Springer Vienna, 1990, pp. 11–49. ISBN: 978-3-7091-8990-0. DOI: 10.1007/978-3-7091-8990-0_2 (cit. on p. 19).

[LS09] Martin Leucker and Christian Schallhart. “A brief account of runtime verification”. In: *The Journal of Logic and Algebraic Programming* 78.5 (2009), pp. 293–303. DOI: 10.1016/j.jlap.2008.08.004 (cit. on pp. 16, 85).

[LFS10] Wenchao Li, Alessandro Forin, and Sanjit A. Seshia. “Scalable specification mining for verification and diagnosis”. In: *Design Automation Conference (DAC)*. 2010, pp. 755–760. DOI: 10.1145/1837274.1837466 (cit. on pp. 74, 75).

[Lin+05] Jessica Lin, Eamonn J. Keogh, Ada Wai-Chee Fu, and Helga Van Herle. “Approximations to magic: finding unusual medical time series”. In: *Symposium on Computer-Based Medical Systems*. 2005, pp. 329–334. DOI: 10.1109/CBMS.2005.34 (cit. on pp. 11, 32).

[LTZ08] Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou. “Isolation Forest”. In: *Proceedings of the 8th IEEE International Conference on Data Mining (ICDM 2008), December 15-19, 2008, Pisa, Italy*. IEEE Computer Society, 2008, pp. 413–422. DOI: 10.1109/ICDM.2008.17 (cit. on p. 35).

[Llo82] Stuart P. Lloyd. “Least squares quantization in PCM”. In: *IEEE Trans. Inf. Theory* 28.2 (1982), pp. 129–136. DOI: 10.1109/TIT.1982.1056489 (cit. on p. 34).

[Luq+19] Amalia Luque, Alejandro Carrasco, Alejandro Martín, and Ana de las Heras. “The impact of class imbalance in classification performance metrics based on the binary confusion matrix”. In: *Pattern Recognition* 91 (2019), pp. 216–231. ISSN: 0031-3203. DOI: 10.1016/j.patcog.2019.02.023 (cit. on p. 46).

[Lv+20] Hao Lv, Shengbing Zhang, Xiaojian Liu, Shuo Liu, Yongqiang Liu, Wei Han, and Shaowei Xu. “Research on Dynamic Reconfiguration Technology of Neural Network Accelerator Based on Zynq”. In: *Journal of Physics: Conference Series* 1650.3 (Oct. 2020), p. 032093. DOI: 10.1088/1742-6596/1650/3/032093 (cit. on pp. 15, 60, 61).

[MFF17] Jan Malburg, Tino Flenker, and Goerschwin Fey. “Property Mining using Dynamic Dependency Graphs”. In: *ASP Design Automation Conference (ASPDAC)*. 2017, pp. 244–250. DOI: 10.1109/ASPDAC.2017.7858327 (cit. on pp. 74, 75).

[MS03a] Markos Markou and Sameer Singh. “Novelty detection: a review–part 1: statistical approaches”. In: *Signal Processing* 83.12 (2003), pp. 2481–2497. ISSN: 0165-1684. DOI: 10.1016/j.sigpro.2003.07.018 (cit. on p. 33).

[MS03b] Markos Markou and Sameer Singh. “Novelty detection: a review–part 2:: neural network based approaches”. In: *Signal Processing* 83.12 (2003), pp. 2499–2521. ISSN: 0165-1684. DOI: 10.1016/j.sigpro.2003.07.019 (cit. on p. 33).

[Mar+22] Gianluca Martino, Andrea Bellandi, Julien Branlard, Annika Eichler, Holger Schlarb, Görschwin Fey, Lawrence Doolittle, Sebastian Aderhold, Andrew Benwell, Daniel Gonnella, Sonya Hoobler, Janice Nelson, Ryan Douglas Porter, Alessandro Ratti, and Lisa Zacarias. “Anomaly Detection Based Quench Detection System for CW Operation of SRF Cavities”. In: *Proc. 31st International Linear Accelerator Conference (LINAC’22)* (Liverpool, UK). International Linear Accelerator Conference 31. JACoW Publishing, Geneva, Switzerland, Sept. 2022, THPOPA15, pp. 775–777. ISBN: 978-3-95450-215-8. DOI: 10.18429/JACoW-LINAC2022-THPOPA15 (cit. on pp. 15, 53, 128, 129).

[MF19] Gianluca Martino and Görschwin Fey. “Syntax-Guided Enumeration of Temporal Properties”. In: *Forum for Specification and Design Languages (FDL)*. Ed. by Tom J. Kazmierski, Reinhard von Hanxleden, and Terrence S. T. Mak. IEEE, 2019, pp. 1–8. DOI: 10.1109/FDL.2019.8876892 (cit. on pp. 71, 127, 129).

[MF22] Gianluca Martino and Görschwin Fey. “Runtime Monitoring of c-LTL Specifications on FPGAs Using HLS”. In: *18th International Conference on Synthesis, Modeling, Analysis and Simulation Methods and Applications to Circuit Design, SMACD 2022, Villasimius, Italy, June 12-15, 2022*. IEEE, 2022, pp. 1–4. DOI: 10.1109/SMACD55068.2022.9816308 (cit. on pp. 16, 85, 128, 129).

[Mar+21] Gianluca Martino, Arne Grünhagen, Julien Branlard, Annika Eichler, Görschwin Fey, and Holger Schlarb. “Comparative Evaluation of Semi-Supervised Anomaly Detection Algorithms on High-Integrity Digital Systems”. In: *24th Euromicro Conference on Digital System Design, DSD 2021, Palermo, Italy, September 1-3, 2021*. Ed. by Francesco Leporati, Salvatore Vitabile, and Amund Skavhaug. IEEE, 2021, pp. 123–130. DOI: 10.1109/DSD53832.2021.00028 (cit. on pp. 14, 31, 127, 129).

[MRF18] Gianluca Martino, Heinz Riener, and Görschwin Fey. “Coverage-Guided CTL Property Enumeration for Understanding Models of Reactive Systems”. In: *International Workshop on Logic Synthesis (IWLS)*. 2018 (cit. on pp. 14, 16, 73, 127).

[MRF19] Gianluca Martino, Heinz Riener, and Görschwin Fey. “Complete Specification Mining”. In: *6th Workshop on Design Automation for Understanding Hardware Designs (DUHDE), Florence, Italy*. 2019 (cit. on p. 127).

[MRF20] Gianluca Martino, Heinz Riener, and Görschwin Fey. “Revisiting Explicit Enumeration for Exact Synthesis”. In: *23rd Euromicro Conference on Digital System Design, DSD 2020, Kranj, Slovenia, August 26-28, 2020*. IEEE, 2020, pp. 29–34. DOI: 10.1109/DSD51259.2020.00016 (cit. on pp. 16, 17, 97, 127, 129).

[MP14] Brendan D. McKay and Adolfo Piperno. “Practical graph isomorphism, II”. In: *Journal of Symbolic Computation* 60 (2014), pp. 94–112. ISSN: 0747-7171. DOI: 10.1016/J.JSC.2013.09.003 (cit. on p. 99).

[Mel+18] Paolo Meloni, Alessandro Capotondi, Gianfranco Deriu, Michele Brian, Francesco Conti, Davide Rossi, Luigi Raffo, and Luca Benini. “NEURAghe: Exploiting CPU-FPGA Synergies for Efficient and Flexible CNN Inference Acceleration on Zynq SoCs”. In: *ACM Trans. Reconfigurable Technol. Syst.* 11.3 (2018), 18:1–18:24. DOI: 10.1145/3284357 (cit. on pp. 15, 60, 61).

[Mic+02] Sam Michiels, Lieven Desmet, Nico Janssens, Tom Mahieu, and Pierre Verbaeten. “DistriNet: Self-adapting concurrency: the DMonA architecture”. In: *Proceedings of the First Workshop on Self-Healing Systems, WOSS 2002, Charleston, South Carolina, USA, November 18-19, 2002*. Ed. by David Garlan, Jeff Kramer, and Alexander L. Wolf. ACM, 2002, pp. 43–48. DOI: 10.1145/582128.582137 (cit. on p. 9).

[Mic] *Micro Telecommunications Computing Architecture – Short Form Specification*. Standard. PICMG, Sept. 2006 (cit. on p. 8).

[Mid+21] Matthew Middlehurst, James Large, Michael Flynn, Jason Lines, Aaron Bostrom, and Anthony J. Bagnall. “HIVE-COTE 2.0: a new meta ensemble for time series classification”. In: *Mach. Learn.* 110.11 (2021), pp. 3211–3243. DOI: 10.1007/s10994-021-06057-9 (cit. on p. 56).

[MCB06] Alan Mishchenko, Satrajit Chatterjee, and Robert Brayton. “DAG -aware AIG Rewriting: a Fresh Look at Combinational Logic Synthesis”. In: *Design Automation Conference (DAC)*. IEEE, 2006, pp. 532–535. ISBN: 1-59593-381-6. DOI: 10.1145/1146909.1147048 (cit. on p. 97).

[MRS17] Patrick Moosbrugger, Kristin Y. Rozier, and Johann Schumann. “R2U2: monitoring and diagnosis of security threats for unmanned aerial systems”. In: *Formal Methods in System Design* 51.1 (2017), pp. 31–61. DOI: 10.1007/s10703-017-0275-x (cit. on pp. 13, 87).

[Mou22] Nicolas Mounet, ed. *European Strategy for Particle Physics - Accelerator R&D Roadmap*. 2022 (cit. on p. 7).

[MI72] Saburo Muroga and Toshihide Ibaraki. “Design of optimal switching networks by integer programming”. In: *IEEE Transactions on Computers* 100.6 (1972), pp. 573–582. DOI: 10.1109/TC.1972.5009010 (cit. on p. 98).

[ML76] Saburo Muroga and Hung Chi Lai. “Minimization of Logic Networks Under a Generalized Cost Function”. In: *IEEE Trans. Computers* 25.9 (1976), pp. 893–907. DOI: 10.1109/TC.1976.1674714 (cit. on p. 98).

[Naw+18] Ayla Nawaz, Sven Pfeiffer, Gerwald Lichtenberg, and Philipp Rostalski. “Anomaly Detection for the European XFEL using a Nonlinear Parity Space Method”. In: *IFAC-PapersOnLine* 51.24 (2018). 10th IFAC Symposium on Fault Detection, Supervision and Safety for Technical Processes SAFEPROCESS 2018, pp. 1379–1386. ISSN: 2405-8963. DOI: 10.1016/j.ifacol.2018.09.554 (cit. on pp. 5, 15, 56).

[Pag+05] Carlo Pagani, Rocco Paparella, Angelo Bosotti, Paolo Pierini, Paolo Michelato, and Nicola Panzeri. *The fast piezo-blade tuner for SCRF resonators*. Tech. rep. 2005 (cit. on p. 22).

[Pea01] Karl Pearson. “On lines and planes of closest fit to systems of points in space”. In: *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science* 2.11 (1901), pp. 559–572. DOI: 10.1080/14786440109462720 (cit. on p. 38).

[Ped+11] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake VanderPlas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Edouard Duchesnay. “Scikit-learn: Machine Learning in Python”. In: *J. Mach. Learn. Res.* 12 (2011), pp. 2825–2830. DOI: 10.5555/1953048.2078195 (cit. on pp. 36, 43, 44).

[Phu+10] Clifton Phua, Vincent C. S. Lee, Kate Smith- Miles, and Ross W. Gayler. “A Comprehensive Survey of Data Mining-based Fraud Detection Research”. In: *CoRR* abs/1009.6119 (2010). arXiv: 1009.6119 (cit. on pp. 11, 32).

[Pim+14] Marco A.F. Pimentel, David A. Clifton, Lei Clifton, and Lionel Tarassenko. “A review of novelty detection”. In: *Signal Processing* 99 (2014), pp. 215–249. ISSN: 0165-1684. DOI: 10.1016/j.sigpro.2013.12.026 (cit. on p. 33).

[PMF21] Swantje Plambeck, Gianluca Martino, and Görschwin Fey. “Metrics for the Evaluation of Approximate Sequential Streaming Circuits”. In: *24th Euromicro Conference on Digital System Design, DSD 2021, Palermo, Italy, September 1-3, 2021*. Ed. by Francesco Leporati, Salvatore Vitabile, and Amund Skavhaug. IEEE, 2021, pp. 208–211. DOI: 10.1109/DSD53832.2021.00041 (cit. on pp. 127, 129).

[Pnu77] Amir Pnueli. “The Temporal Logic of Programs”. In: *Symposium on Foundations of Computer Science (FOCS)*. 1977, pp. 46–57. DOI: 10.1109/SFCS.1977.32 (cit. on pp. 25, 72).

[Por00] Leonid Portnoy. “Intrusion detection with unlabeled data using clustering”. Bachelor’s Thesis. Columbia University, 2000 (cit. on pp. 11, 32).

[PS02] Mitra Purandare and Fabio Somenzi. “Vacuum Cleaning CTL Formulae”. In: *International Conference on Computer Aided Verification (CAV)*. 2002, pp. 485–499. DOI: 10.1007/3-540-45657-0_39 (cit. on p. 75).

[Qas+21] Murad Qasaimeh, Kristof Denolf, Alireza Khodamoradi, Michaela Blott, Jack Lo, Lisa Halder, Kees A. Vissers, Joseph Zambreno, and Phillip H. Jones. “Benchmarking vision kernels and neural network inference accelerators on embedded platforms”. In: *J. Syst. Archit.* 113 (2021), p. 101896. DOI: 10.1016/j.sysarc.2020.101896 (cit. on pp. 59, 61).

[Rah+20] Reza Rahimi, Elaheh Sadredini, Mircea Stan, and Kevin Skadron. “Grapefruit: An Open-Source, Full-Stack, and Customizable Automata Processing on FPGAs”. In: *28th IEEE Annual International Symposium on Field-Programmable Custom Computing Machines, FCCM 2020, Fayetteville, AR, USA, May 3-6, 2020*. IEEE, 2020, pp. 138–147. DOI: 10.1109/FCCM48280.2020.00027 (cit. on p. 86).

[Ray+16] Sandip Ray, Ian G. Harris, Görschwin Fey, and Mathias Soeken. “Multilevel design understanding: from specification to logic”. In: *International Conference on Computer-Aided Design (ICCAD)*. 2016, 133:1–133:6. DOI: 10.1145/2966986.2980093 (cit. on p. 72).

[Rey15] Douglas A. Reynolds. “Gaussian Mixture Models”. In: *Encyclopedia of Biometrics, Second Edition*. Ed. by Stan Z. Li and Anil K. Jain. Springer US, 2015, pp. 827–832. DOI: 10.1007/978-1-4899-7488-4_196 (cit. on p. 35).

[Rie+19] Heinz Riener, Winston Haaswijk, Alan Mishchenko, Giovanni De Micheli, and Mathias Soeken. “On-the-fly and DAG -aware: Rewriting Boolean Networks with Exact Synthesis”. In: *Design, Automation and Test in Europe Conference (DATE)*. IEEE. 2019, pp. 1649–1654. DOI: 10.23919/DATE.2019.8715185 (cit. on p. 97).

[Rom+21] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. *High-Resolution Image Synthesis with Latent Diffusion Models*. 2021. arXiv: 2112.10752 [cs.CV] (cit. on pp. 19, 25).

[RK62] J. Paul Roth and Richard M. Karp. “Minimization Over Boolean Graphs”. In: *IBM J. Res. Dev.* 6.2 (1962), pp. 227–238. DOI: 10.1147/RD.62.0227 (cit. on pp. 17, 97).

[RD99] Peter J. Rousseeuw and Katrien van Driessen. “A Fast Algorithm for the Minimum Covariance Determinant Estimator”. In: *Technometrics* 41.3 (1999), pp. 212–223. DOI: 10.1080/00401706.1999.10485670 (cit. on p. 37).

[Sch+21] Alexander Scheinker, Frederick Cropp, Sergio Paiagua, and Daniele Filippetto. “An adaptive approach to machine learning for compact particle accelerators”. In: *Scientific Reports* 11.1 (Sept. 2021), p. 19187. ISSN: 2045-2322. DOI: 10.1038/s41598-021-98785-0 (cit. on pp. 12, 59, 60).

[Sch98] Thomas Schilcher. “Vector Sum Control of Pulsed Accelerating Fields in Lorentz Forces Detuned Superconducting Cavities”. Universität Hamburg, Diss., 1998. PhD thesis. Universität Hamburg, 1998, p. 137 (cit. on p. 21).

[Sch+08] Christian Schmidt, Gerwald Lichtenberg, W. Koprek, W. Jalmuzna, Herbert Werner, and Stefan Simrock. “Parameter estimation and tuning of a multivariable RF controller with FPGA technique for the Free Electron Laser FLASH”. In: *American Control Conference, ACC 2008, Seattle, WA, USA, 11-13 June 2008*. IEEE, 2008, pp. 2516–2521. DOI: 10.1109/ACC.2008.4586869 (cit. on p. 23).

[Sch06] Peter Schmüser. “Basic principles of RF superconductivity and superconducting cavities”. In: (2006). DOI: 10.5170/CERN-2006-002.183 (cit. on p. 53).

[SD68] Peter R. Schneider and Donald L. Dietmeyer. “An Algorithm for Synthesis of Multiple-Output Combinational Logic”. In: *IEEE Trans. Computers* 17.2 (1968), pp. 117–128. DOI: 10.1109/TC.1968.227399 (cit. on p. 97).

[Sch+99] Bernhard Schölkopf, Robert C. Williamson, Alexander J. Smola, John Shawe-Taylor, and John C. Platt. “Support Vector Method for Novelty Detection”. In: *Advances in Neural Information Processing Systems 12, [NIPS Conference, Denver, Colorado, USA, November 29 - December 4, 1999]*. Ed. by Sara A. Solla, Todd K. Leen, and Klaus-Robert Müller. The MIT Press, 1999, pp. 582–588. URL: http://papers.nips.cc/paper/1723-support-vector-method-for-novelty-detection (cit. on p. 37).

[Sel+17] Konstantin Selyunin, Stefan Jaksic, Thang Nguyen, Christian Reidl, Udo Hafner, Ezio Bartocci, Dejan Nickovic, and Radu Grosu. “Runtime Monitoring with Recovery of the SENT Communication Protocol”. In: *International Conference on Computer Aided Verification (CAV)*. Vol. 10426. Lecture Notes in Computer Science. Springer, 2017, pp. 336–355. DOI: 10.1007/978-3-319-63387-9_17 (cit. on pp. 13, 86).

[SSR19] Aman Sharma, Vijander Singh, and Asha Rani. “Implementation of CNN on Zynq based FPGA for Real-time Object Detection”. In: *10th International Conference on Computing, Communication and Networking Technologies, ICCCNT 2019, Kanpur, India, July 6-8, 2019*. IEEE, 2019, pp. 1–7. DOI: 10.1109/ICCCNT45670.2019.8944792 (cit. on pp. 59, 60).

[Sho+08] Sharon Shoham, Eran Yahav, Stephen J. Fink, and Marco Pistoia. “Static Specification Mining Using Automata-Based Abstractions”. In: *IEEE Transactions on Software Engineering* 34.5 (2008), pp. 651–666. DOI: 10.1109/TSE.2008.63 (cit. on pp. 74, 75).

[Shy+03] Mei-Ling Shyu, Shu-Ching Chen, Kanoksri Sarinnapakorn, and LiWu Chang. “A Novel Anomaly Detection Scheme Based on Principal Component Classifier”. In: *IEEE Foundations and New Directions of Data Mining Workshop*. 2003, pp. 172–179 (cit. on p. 38).

[SS20] Tomyslav Sledevič and Artūras Serackis. “mNet2FPGA: A Design Flow for Mapping a Fixed-Point CNN to Zynq SoC FPGA”. In: *Electronics* 9.11 (2020). ISSN: 2079-9292. DOI: 10.3390/electronics9111823 (cit. on pp. 59, 60).

[Smi65] Robert Allan Smith. “Minimal three-variable NOR and NAND logic circuits”. In: *IEEE Transactions on Electronic Computers* 14.1 (1965), pp. 79–81. DOI: 10.1109/PGEC.1965.264064 (cit. on p. 98).

[Soe+19] Mathias Soeken, Heinz Riener, Winston Haaswijk, Eleonora Testa, Bruno Schmitt, Giulia Meuli, Fereshte Mozafari, and Giovanni De Micheli. *The EPFL logic synthesis libraries*. arXiv:1805.05121v2. Nov. 2019. DOI: 10.48550/arXiv.1805.05121 (cit. on p. 99).

[Sto11] J Stohr. “Linac Coherent Light Source II (LCLS-II) Conceptual Design Report”. In: (Nov. 2011). DOI: 10.2172/1029479 (cit. on p. 6).

[TS17] A.S. Tanenbaum and M. van Steen. *Distributed Systems*. CreateSpace Independent Publishing Platform, 2017. ISBN: 9781543057386 (cit. on p. 6).

[UBS18] Vladimir Ulyantsev, Igor Buzhinsky, and Anatoly Shalyto. “Exact finite-state machine identification from scenarios and temporal properties”. In: *International Journal on Software Tools for Technology Transfer* 20.1 (2018), pp. 35–55. ISSN: 1433-2787. DOI: 10.1007/s10009-016-0442-1 (cit. on p. 80).

[Vas+10] Shobha Vasudevan, David Sheridan, Sanjay J. Patel, David Tcheng, William Tuohy, and Daniel R. Johnson. “GoldMine: Automatic assertion generation using data mining and static analysis”. In: *Design, Automation and Test in Europe Conference (DATE)*. 2010, pp. 626–629. DOI: 10.1109/DATE.2010.5457129 (cit. on p. 13).

[Xil19] Xilinx. *Introduction to FPGA Design with Vivado High-Level Synthesis*. English. Version Version 1.1. Jan. 22, 2019. 92 pp. (cit. on p. 92).

[YD03] Dit-Yan Yeung and Yuxin Ding. "Host-based intrusion detection using dynamic and static behavioral models". In: *Pattern Recognition* 36.1 (2003), pp. 229–243. ISSN: 0031-3203. DOI: 10.1016/S0031-3203(02)00026-2 (cit. on pp. 11, 32).

[Zha+17] Bendong Zhao, Huanzhang Lu, Shangfeng Chen, Junliang Liu, and Dongya Wu. "Convolutional neural networks for time series classification". In: *Journal of Systems Engineering and Electronics* 28.1 (2017), pp. 162–169. DOI: 10.21629/JSEE.2017.01.18 (cit. on pp. 12, 55, 56).

[ZH19] Junchen Zhao and Ian G. Harris. "Automatic Assertion Generation from Natural Language Specifications Using Subtree Analysis". In: *Design, Automation & Test in Europe Conference & Exhibition, DATE 2019, Florence, Italy, March 25-29, 2019*. Ed. by Jürgen Teich and Franco Fummi. IEEE, 2019, pp. 598–601. DOI: 10.23919/DATE.2019.8714857 (cit. on p. 106).

[ZNL19] Yue Zhao, Zain Nasrullah, and Zheng Li. "PyOD: A Python Toolbox for Scalable Outlier Detection". In: *Journal of Machine Learning Research* 20.96 (2019), pp. 1–7 (cit. on p. 44).

[Zhu+21] Jun Zhu, Ye Chen, Frank Brinker, Winfried Decking, Sergey Tomin, and Holger Schlarb. "High-Fidelity Prediction of Megapixel Longitudinal Phase-Space Images of Electron Beams Using Encoder-Decoder Neural Networks". In: *Physical Review Applied* 16 (2 Aug. 2021), pp. 1–10. DOI: 10.1103/PhysRevApplied.16.024005 (cit. on pp. 12, 59, 60, 62).

APPENDIX

A

Own Publications

Görschwin Fey, Tara Ghasempouri, Swen Jacobs, Gianluca Martino, Jaan Raik, and Heinz Riener. "Design Understanding: From Logic to Specification". In: *IFIP/IEEE International Conference on Very Large Scale Integration, VLSI-SoC 2018, Verona, Italy, October 8-10, 2018*. IEEE, 2018, pp. 172–175. DOI: 10.1109/VLSI-SoC.2018.8644732.

Gianluca Martino, Heinz Riener, and Görschwin Fey. "Coverage-Guided CTL Property Enumeration for Understanding Models of Reactive Systems". In: *International Workshop on Logic Synthesis (IWLS)*. 2018.

Gianluca Martino, Heinz Riener, and Görschwin Fey. "Complete Specification Mining". In: *6th Workshop on Design Automation for Understanding Hardware Designs (DUHDE), Florence, Italy*. 2019.

Gianluca Martino and Görschwin Fey. "Syntax-Guided Enumeration of Temporal Properties". In: *Forum for Specification and Design Languages (FDL)*. ed. by Tom J. Kazmierski, Reinhard von Hanxleden, and Terrence S. T. Mak. IEEE, 2019, pp. 1–8. DOI: 10.1109/FDL.2019.8876892.

Gianluca Martino, Heinz Riener, and Görschwin Fey. "Revisiting Explicit Enumeration for Exact Synthesis". In: *23rd Euromicro Conference on Digital System Design, DSD 2020, Kranj, Slovenia, August 26-28, 2020*. IEEE, 2020, pp. 29–34. DOI: 10.1109/DSD51259.2020.00016.

Swantje Plambeck, Gianluca Martino, and Görschwin Fey. "Metrics for the Evaluation of Approximate Sequential Streaming Circuits". In: *24th Euromicro Conference on Digital System Design, DSD 2021, Palermo, Italy, September 1-3, 2021*. Ed. by Francesco Leporati, Salvatore Vitabile, and Amund Skavhaug. IEEE, 2021, pp. 208–211. DOI: 10.1109/DSD53832.2021.00041.

Gianluca Martino, Arne Grünhagen, Julien Branlard, Annika Eichler, Görschwin Fey, and Holger Schlarb. "Comparative Evaluation of Semi-Supervised Anomaly Detection Algorithms on High-Integrity Digital Systems". In: *24th Euromicro Conference on Digital System Design, DSD 2021, Palermo, Italy, September 1-3, 2021*. Ed. by Francesco Leporati,

Salvatore Vitabile, and Amund Skavhaug. IEEE, 2021, pp. 123–130. DOI: 10.1109/DSD53832.2021.00028.

Arne Grünhagen, Julien Branlard, Annika Eichler, Gianluca Martino, Görschwin Fey, and Marina Tropmann-Frick. "Fault Analysis of the Beam Acceleration Control System at the European XFEL using Data Mining". In: *30th IEEE Asian Test Symposium, ATS 2021, Matsuyama, Ehime, Japan, November 22-25, 2021*. IEEE, 2021, pp. 61–66. DOI: 10.1109/ATS52891.2021.00023.

Gianluca Martino and Görschwin Fey. "Runtime Monitoring of c-LTL Specifications on FPGAs Using HLS". in: *18th International Conference on Synthesis, Modeling, Analysis and Simulation Methods and Applications to Circuit Design, SMACD 2022, Villasimius, Italy, June 12-15, 2022*. IEEE, 2022, pp. 1–4. DOI: 10.1109/SMACD55068.2022.9816308.

Gianluca Martino, Andrea Bellandi, Julien Branlard, Annika Eichler, Holger Schlarb, Görschwin Fey, Lawrence Doolittle, Sebastian Aderhold, Andrew Benwell, Daniel Gonnella, Sonya Hoobler, Janice Nelson, Ryan Douglas Porter, Alessandro Ratti, and Lisa Zacarias. "Anomaly Detection Based Quench Detection System for CW Operation of SRF Cavities". In: *Proc. 31st International Linear Accelerator Conference (LINAC'22)* (Liverpool, UK). International Linear Accelerator Conference 31. JACoW Publishing, Geneva, Switzerland, Sept. 2022, THPOPA15, pp. 775–777. ISBN: 978-3-95450-215-8. DOI: 10.18429/JACoW-LINAC2022-THPOPA15.

Ahmad Al-Zoubi, Gianluca Martino, Fin Hendrik Bahnsen, Jun Zhu, Holger Schlarb, and Görschwin Fey. "CNN Implementation and Analysis on Xilinx Versal ACAP at European XFEL". in: *35th IEEE International System-on-Chip Conference, SOCC 2022, Belfast, United Kingdom, September 5-8, 2022*. Ed. by Sakir Sezer, Thomas Büchner, Jürgen Becker, Andrew Marshall, Fahad Siddiqui, Tanja Harbaum, and Kieran McLaughlin. IEEE, 2022, pp. 1–6. DOI: 10.1109/SOCC56010.2022.9908101.

Leandro Lanzieri, Gianluca Martino, Görschwin Fey, Holger Schlarb, Thomas C. Schmidt, and Matthias Wählisch. *A Review of Techniques for Ageing Detection and Monitoring on Embedded Systems*. 2023. DOI: 10.48550/ARXIV.2301.06804.

Individual Contributions

Throughout the course of my Ph.D., I collaborated with multiple people in different forms. In the following, I describe my contributions to the projects that resulted in the publications listed above. For [Fey+18], I wrote the first version of the property enumeration tool using the generic enumeration library behemoth[1] and performed the evaluation of the code using a set of standard benchmarks. I participated in the writing of the introduction and wrote Section 2 (Syntax-Guided Property Enumeration). This work was the

[Fey+18] Fey et al., "Design Understanding: From Logic to Specification"

[1] behemoth, https://github.com/hriener/behemoth

result of the special session about design understanding of VLSI-SoC 2018. For [MF19], I enhanced the code of the property enumeration tool and tested it against Goldmine. I also designed the case study to illustrate the real-world capabilities of the tool. Afterward, I wrote the entire text explaining the technique and the experiments performed. For [MRF20], I used the same enumeration technique, previously used to enumerate properties, and used it to enumerate circuits. I extended the code base with circuit-specific optimizations. I then tested the implementation and compared it to ABC [BM10]. For [PMF21], I contributed to the discussions about the idea and advised the first author. As an advisor to the first author, I helped in refining and focusing the research question, ensuring a more targeted and impactful study. Additionally, I meticulously reviewed and provided detailed feedback on the manuscript, suggesting enhancements in the structure, clarity, and coherence of the text. For [Mar+21], I reviewed the literature, selected the algorithms, and performed the comparative evaluation. Finally, I wrote the text presenting the algorithms and the experiments performed. For [Grü+21], I advised the first author and offered suggestions on how to improve the text. For [MF22], I wrote the code of the entire framework. I then designed and ran the experiments. Finally, I wrote the text describing the framework and discussing the results of the experiments. For [Mar+22], I wrote the code of both the fault display and the quench detection. The fault display was deployed as part of the operation toolset for LCLS-II. The quench detection code was used to perform tests using the commissioning data from LCLS-II. Lastly, I wrote the text detailing both contributions. For [Al-+22], I acquired and helped set up the Versal board used for the evaluation. I then advised the first author and conducted thorough reviews of the draft, offering constructive feedback aimed at enhancing its clarity, coherence, and overall quality. My suggestions included recommendations for more precise language and better structuring of arguments. For [Lan+23], I advised the first author, contributed to the text, and offered suggestions on how to improve the text.

[MF19] Martino and Fey, “Syntax-Guided Enumeration of Temporal Properties”

[MRF20] Martino et al., “Revisiting Explicit Enumeration for Exact Synthesis”

[BM10] Brayton and Mishchenko, “ABC: An Academic Industrial-Strength Verification Tool”

[PMF21] Plambeck et al., “Metrics for the Evaluation of Approximate Sequential Streaming Circuits”

[Mar+21] Martino et al., “Comparative Evaluation of Semi-Supervised Anomaly Detection Algorithms on High-Integrity Digital Systems”

[Grü+21] Grünhagen et al., “Fault Analysis of the Beam Acceleration Control System at the European XFEL using Data Mining”

[MF22] Martino and Fey, “Runtime Monitoring of c-LTL Specifications on FPGAs Using HLS”

[Mar+22] Martino et al., “Anomaly Detection Based Quench Detection System for CW Operation of SRF Cavities”

[Al-+22] Al-Zoubi et al., “CNN Implementation and Analysis on Xilinx Versal ACAP at European XFEL”

[Lan+23] Lanzieri et al., *A Review of Techniques for Ageing Detection and Monitoring on Embedded Systems*