ISBN: 978-3-95450-241-7 ISSN: 2673-5350 doi:10.18429/JACoW-IBIC2022-MOP42

KINGFISHER: A FRAMEWORK FOR FAST MACHINE LEARNING INFERENCE FOR AUTONOMOUS ACCELERATOR SYSTEMS

L. Scomparin*, E. Blomley, T. Boltz†, E. Bründermann, M. Caselle, T. Dritschler, A. Kopmann, A. Mochihashi, A.-S. Müller, A. Santamaria Garcia, P. Schreiber, J. L. Steinmann, M. Weber
Karlsruhe Institute of Technology, Karlsruhe, Germany

Abstract

Modern particle accelerator facilities allow new and exciting beam properties and operation modes. Traditional real-time control systems, albeit powerful, have bandwidth and latency constraints that limit the range of operating conditions currently made available to users. The capability of Reinforcement Learning to perform self-learning control policies by interacting with the accelerator is intriguing. The extreme dynamic conditions require fast real-time feedback throughout the whole control loop from the diagnostic, with novel and intelligent detector systems, all the way to the interaction with the accelerator components. In this contribution, the novel KINGFISHER framework based on the modern Xilinx Versal devices will be presented. Versal combines several computational engines, specifically combining powerful FPGA logic with programmable AI Engines in a single device. Furthermore, this system can be natively integrated with the fastest beam diagnostic tools already available, e.g. KAPTURE and KALYPSO.

INTRODUCTION

Ensuring stable operation of the future particle accelerator facilities will pose a challenging problem, where traditional control systems are expected to not reach the desired performance. An interesting possibility is the use of Machine Learning (ML) techniques. Reinforcement Learning (RL) [1] is a prominent approach, which recently has shown several relevant results [2–5]. The basic idea is to model the control problem as the interaction between an agent and a system. The agent obtains some observable from the system that are expected to carry information about an underlying and often hidden state and chooses an action from an action space. Then, a reward is obtained from the system giving a metric of how good the taken action was. Reinforcement Learning is a series of algorithms allowing to train such an agent by using the rewards given by the system.

One of these challenging problems is the control of micro-bunching instabilities (MBI) [6] in synchrotron light sources, where the interaction of the beam head with the wake field produced by the Synchrotron Radiation (SR) emitted from the tail leads to the development of substructures in the longitudinal phase-space. One successful attempt was already performed at SOLEIL [7], in this case using traditional control techniques based on chaos theory. The underlying idea of this feedback loop is to stabilize the system around pre-existing unstable equilibrium points.

One intriguing capability of RL is its great versatility: by changing the reward function different goals are achievable, even when they might not be characteristic of a given stable condition. For example, the emission of Coherent Synchrotron Radiation (CSR) could be enhanced or suppressed while setting an acceptable level of fluctuation in the emitted power. Moreover, the enhancement of a specific radiation frequency range could be achieved.

MODULAR ARCHITECTURE FOR FAST INFERENCE

One of the issues that can be encountered when implementing RL techniques with accelerators are latency limitations. Namely, the complete feedback loop (comprising beam diagnostic detector readout, feature extraction, observable evaluation with the agent and action taking) needs to be taken in a time frame comparable to the dynamics of the physics that needs to be controlled. In the case of the MBI, this imposes latency constraints in the order of the synchrotron periods, i.e. of a few tens of microseconds.

A fundamental requirement necessary in order to minimise deployment and testing time for different RL algorithms is a modularity oriented model. Such a system can be divided into four main parts. First, a detector readout interface (1) is needed to retrieve data with low-latency and high-throughput. This data stream from the detector is then fed into the feature extraction part (2), which produces higher-level observables that are then fed into the agent module (3) that chooses an action. The action needs then to be applied to the system, in this case the accelerator, through an action taking part (4), that can interface to a part of the machine control system, ideally with a low-latency connection. A schematic view of this system is shown in Fig. 1.

On top of this inference signal path two other systems are needed: a slow-control system and a training system. Both of these are less time-critical compared to the feedback path. The training system needs to keep track of the observables, of the actions taken, and of the rewards, so that this information can be retrieved during training to update the agents's parameters. Meanwhile, the slow-control allows the operator to set different parameters and tune the system based on the needs of the facility.

KINGFISHER

The different modules and signal paths described in the previous section have different computational requirements. For instance, fast but simple operations are required when interfacing with the readout electronics, while the agent

* luca.scomparin@kit.edu
† now at SLAC, Menlo Park, California

ISBN: 978-3-95450-241-7 ISSN: 2673-5350 doi:10.18429/JACoW-IBIC2022-MOP42

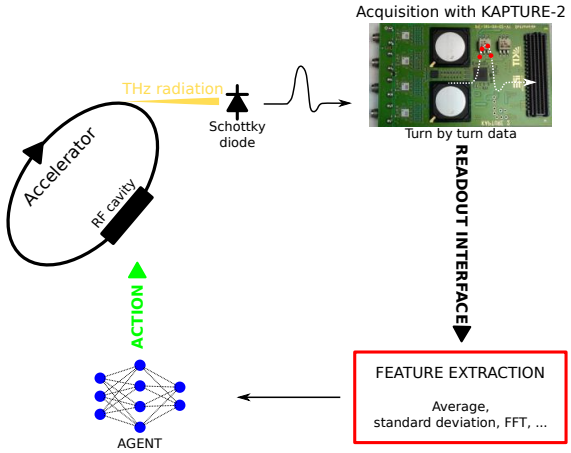

Figure 1: Schematic view of the modular RL feedback system. Changing detector, interface, or agent should minimally impact the remaining blocks of the system.

could be very computationally intensive due to the need of several floating point multiply and accumulate operations. Moreover, in order to employ the several ML frameworks already widely used and available [8, 9] in the training path, high-level operations are needed.

To fulfill all these requirements a heterogeneous platform is necessary. One possible choice is the Xilinx VCK190 evaluation board [10] based on the novel Xilinx Versal XCVC1902 Adaptive Compute Acceleration Platform (ACAP). This novel device combines an FPGA and an ARM processor in the usual System-on-Chip (SoC) configuration with a powerful AI Engine Array allowing fast floating point and integer arithmetic operations. These three components are deeply interconnected with a low-latency, high-throughput Network-on-Chip (NoC) that is also integrated with high speed transceivers, for example a 100 gigabit ethernet interface.

The modular approach described in the previous section is implemented on the VCK190 as the KINGFISHER platform (Fig. 2). The system comprises several interfaces to detector systems (KAPTURE [11], KALYPSO [12], and THERESA [13]), and to the available accelerator longitudinal feedback systems (Low Level RF (LLRF) and Bunch-By-Bunch feedback system (BBB) [14]), all implemented as FPGA blocks. The more computationally heavy operations, namely feature extraction and agent inference, are carried out by the the AI Engines, while the ARM processor has the task to run the slow-control and the training algorithms as Petalinux applications.

Agent

Reinforcement Learning allows great variability on the nature of an agent. For the current problem it was shown that a feed-forward neural network exhibits a sufficient degree of control [3]. In order to evaluate the performance of the system a simple feed-forward neural network (eight input and output neurons, four 64 neurons wide hidden layers) was developed on the AI Engine array, as described in [15]. The latency was shown to be almost four times better (4.5 µs) with respect to the same network deployed on a Zynq Ultrascale+ FPGA [15], while using 32 bit floating point arithmetics compared to 8 bit integers. Moreover, the system is fully reconfigurable from the processor allowing the online updating of weights and biases that are then used for the next computation.

The single precision floating point arithmetic and reconfiguration capabilities are extremely intriguing: the training of the neural networks can be performed on an identical network deployed on the ARM processor, where all widespread ML Python libraries are available, and the new agent coefficients are directly uploaded to the agent network on the AI Engines, without the need of quantization and pruning steps usually necessary when deploying a model on FPGA.

Feature Extraction

Features based on the work in reference [3] were implemented. Namely, a stream of CSR power measurements acquired using the KAPTURE system with a sampling rate equal to the revolution frequency of the accelerator (in this case ≈ 2.7 MHz) were divided in batches that are 1024 samples long. The extracted observables are respectively the mean and variance of the acquired samples, the maximum bin of the Discrete Fourier Transform, the maximum bin squared absolute value normalized to the sum of the remaining bins, and its ratio of imaginary to real part.

Average and standard deviation are computed using the following equations

$$\bar{x} = \frac{1}{N}\sum x_i \tag{1}$$

$$\sigma_x^2 = \frac{1}{N}\sum x_i^2 - \left(\frac{1}{N}\sum x_i\right)^2 \tag{2}$$

this specific expression for the variance is used because it does not require knowledge of the value of the average during computation and thus can be computed on the streaming data, therefore reducing latency.

Regarding the Fourier Transform (FT) some care is necessary: the AI Engine tiles are not able to natively compute sine and cosine functions with floating point precision in an efficient manner. Moreover, there is not enough memory on the engines to store a complete lookup table of the Discrete FT (DFT) samples (32 kB available, even though the neighbouring tiles can share memory bank, allowing a maximum of 160 kB). A possible way to circumvent this issue is to employ an FFT algorithm, for example the Cooley–Tukey algorithm, but this requires to have all the samples before computing the final FFT, thus severely impacting latency. For the sake of computational optimization, the calculation is divided into chunks that are then summed up when a new FT output is needed. In the formula below the summation is divided into M chunks, each one needing a sequence of N/M samples for the update. The final FT bin is composed using the M chunks. The value of M can be chosen arbitrarily, provided it is a divisor of N.

ISBN: 978-3-95450-241-7 ISSN: 2673-5350 doi:10.18429/JACoW-IBIC2022-MOP42

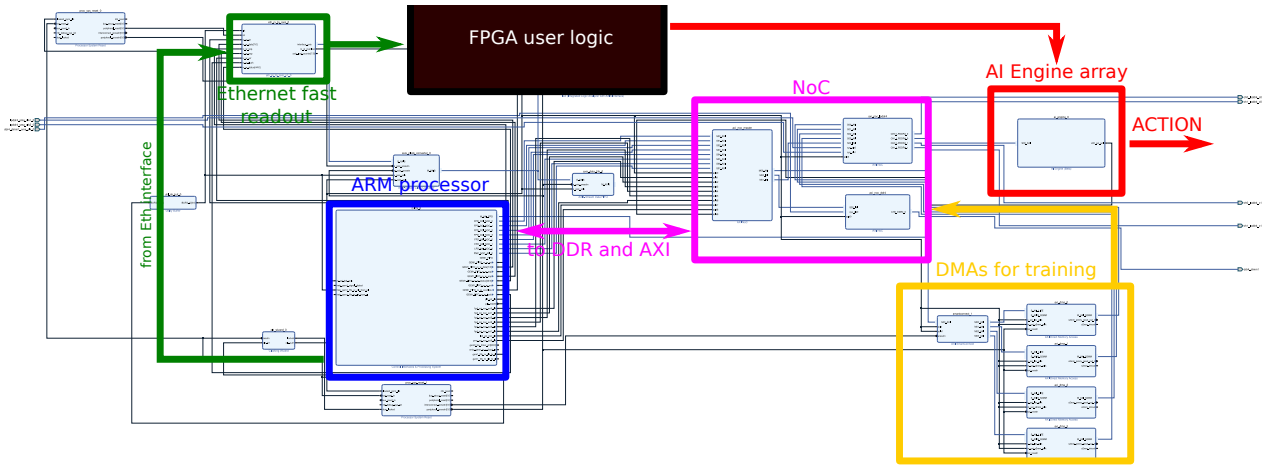

Figure 2: Schematic diagram showing the signal path inside of the KINGFISHER platform.

$$X_k = \sum_{n=0}^{N-1} x_n \, e^{-i\frac{2\pi}{N}kn} = \sum_{m=0}^{M-1} \sum_{j=0}^{N/M-1} x_{\frac{N}{M}m+j} \, e^{-i\frac{2\pi}{N}k(\frac{N}{M}m+j)}$$
$$= \sum_{m=0}^{M-1} e^{-i\frac{2\pi}{M}mk} \sum_{j=0}^{N/M-1} x_{\frac{N}{M}m+j} \, e^{-i\frac{2\pi}{N}kj} \tag{3}$$

This version allows to dramatically reduce the size of the lookup table: each coefficient depends either on m or on j, never both, so instead of using N coefficients per FT bin, only $\frac{N}{M} + M$ are needed.

When computing one 1024 sample FT every 256 input samples the measured latency is in the order of a few tens of microseconds. Such latency is strongly dependent on how the computation is divided among different AI Engine tiles. Additional optimization is ongoing in order to improve latency by more cleverly managing the distribution of workload and communications between tiles.

Training Module

In order to obtain all information necessary for training the agent, an infrastructure composed of several Xilinx Direct Memory Access (DMA) IPs are employed. This part is able to monitor all data streams in the feedback path and store them in one of the two DDR memories of the VCK190. In the current design 8 GB of DDR4 memory are used to run the Petalinux OS, while the remaining 8 GB of LPDDR4 store data from the DMA.

It is then possible to write data to a file for off-line analysis and also use it directly for training employing the widespread ML libraries and then to upload the new set of parameters to the agent.

Pseudo-random Number Generator

Most of RL algorithms need a random source in order to guide the exploration of new policies [1]. A pseudo-random number generator has been developed and implemented on the FPGA in Verilog. This module is based on the Permuted Congruential Generator XOR SHift Random Rotation (PCG-XSH-RR) algorithm [16] with a 64 bit hidden state. This specific implementation works at a maximum clock frequency of 250 MHz and produces a 32 bit random unsigned integer or floating point number every two clock cycles. This algorithm was specifically chosen due to its simple implementation and good quality of random number output.

Actuator

The beamline where the diagnostic instrumentation is located is physically separated from the control system instrumentation employed to apply the feedback operation. To reduce the issues associated with this separation a Xilinx ZCU102 evaluation board [17] was used to implement a system that is controllable via Gigabit Ethernet and allows to select different action types, e.g. generate several sinusoidal analog signals with selectable amplitude and frequency, together with a white noise source (using the random number generator from the previous section) with an output sample rate of ≈ 2.7 MHz, allowing a turn-by-turn controllable beam kick at KARA in single-bunch operation.

The latency of the system was measured by sending a command from KINGFISHER via a two meter long CAT6 Ethernet cable and checking with an oscilloscope the time necessary to apply it to the analog output. The latency measured is of 2.5 µs, e.g. less than 7 full revolutions of the beam at KARA.

KAPTURE Integration

The diagnostic signal used for the current tests is the THz radiation sampled by a Schottky diode readout with the KAPTURE-2 system [11]. KAPTURE-2 allows the sampling of a fast signal on a turn-by-turn basis for all bunches in the accelerator. Compared to the previous versions of the system using the Highflex [18] board, for this integration the new Highflex-2 [15] was used. The firmware was ported to the newer Xilinx Zynq Ultrascale+ architecture and the

ISBN: 978-3-95450-241-7 ISSN: 2673-5350 doi:10.18429/JACoW-IBIC2022-MOP42

embedded Gigabit Ethernet MAC (GEM) was controlled from the FPGA in order to continuously stream data of up to 5 bunches. This interface is then connected to the GEM on the VCK190 board as shown in the picture in Fig. 3. In the future, the system will be improved further allowing the streaming of all data on fast high-data throughput optical data links.

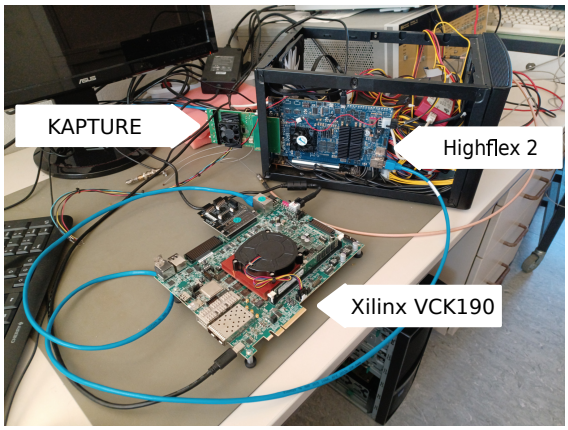

Figure 3: Picture of the VCK190 Evaluation Board connected to the Highflex-2 board where KAPTURE is mounted.

BEAM TESTS

The above KAPTURE and KINGFISHER system has been tested during several beam times at the KIT synchrotron light source KARA storage ring facility at KIT using the 1.3 GeV short-bunch operation mode. All systems were separately tested, although the closed loop tests need to wait for upgrades to the LLRF system in order to apply the required feedback signal. Nonetheless it was possible to perform stable measurements of the THz signal with feature extraction. Small longitudinal kicks where applied by connecting the actuator analog output to the longitudinal BBB feedback system while disabling the internal filters so the signal is directly applied to the beam. This configuration was shown to be working by inducing longitudinal oscillations at specific frequencies, as shown in Fig. 4, where the spectrum of a Beam Position Monitor (BPM) read with the BBB system clearly shows a peak at the excitation frequency. Unfortunately this kind of action is not strong enough in the current longitudinal BBB setup to sufficiently influence the micro-bunching instability.

CONCLUSION

The novel KINGFISHER platform was introduced as a general low-latency inference system for Reinforcement Learning for intelligent control of particle accelerators. The main blocks were discussed, as well as their architecture. First results during beamtime have shown that all blocks work correctly, leaving the addition of a low-latency feedback input to the accelerator control system as the last step before the final testing of the complete feedback loop can be carried out.

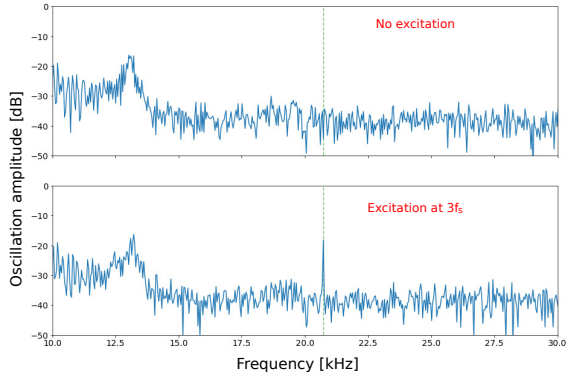

Figure 4: Sinusoidal excitation at three times the synchrotron frequency: top plot shows the BPM spectrum when the excitation is off, bottom plot when the excitation is on. The vertical dashed green line shows the frequency of the applied excitation.

ACKNOWLEDGEMENTS

We would like to thank Y.-L. Mathis and his team for beamtime and support at the KARA IR1 beamline.

The work was in part supported by BMBF ErUM-Pro FKZ 05K19VKC (TiMo) and in part by the Innovationspool Project ACCLAIM.

REFERENCES

[1] R. S. Sutton and A. G. Barto, *Reinforcement Learning, second edition*. Cambridge, MA, USA: MIT Press, 2018.

[2] W. Wang *et al.*, "Accelerated Deep Reinforcement Learning for Fast Feedback of Beam Dynamics at KARA," *IEEE Trans. Nucl. Sci.*, vol. 68, pp. 1794–1800, 2021. doi:10.1109/TNS.2021.3084515

[3] T. Boltz, "Micro-bunching control at electron storage rings with reinforcement learning," Ph.D. dissertation, Karlsruher Institut für Technologie (KIT), Germany, 2021. doi:10.5445/IR/1000140271

[4] J. S. John *et al.*, "Real-time artificial intelligence for accelerator control: A study at the fermilab booster," *Phys. Rev. Accel. Beams*, vol. 24, p. 104 601, 2021. doi:10.1103/PhysRevAccelBeams.24.104601

[5] F. H. O'Shea, N. Bruchon, and G. Gaio, "Policy gradient methods for free-electron laser and terahertz source optimization and stabilization at the fermi free-electron laser at elettra," *Phys. Rev. Accel. Beams*, vol. 23, p. 122 802, 2020. doi:10.1103/PhysRevAccelBeams.23.122802

[6] M. Brosi *et al.*, "Systematic studies of the microbunching instability at very low bunch charges," *Phys. Rev. Accel. Beams*, vol. 22, p. 020 701, 2019. doi:10.1103/PhysRevAccelBeams.22.020701

[7] C. Evain *et al.*, "Stable coherent terahertz synchrotron radiation from controlled relativistic electron bunches," *Nat. Phys.*, vol. 15, pp. 635–639, 2019. doi:10.1038/s41567-019-0488-6

ISBN: 978-3-95450-241-7 ISSN: 2673-5350 doi:10.18429/JACoW-IBIC2022-MOP42

[8] A. Paszke *et al.*, "Pytorch: An imperative style, high-performance deep learning library," in *Proc. Advances in Neural Information Processing Systems*, vol. 32, 2019, pp. 8024–8035.

[9] Martín Abadi *et al.*, *TensorFlow: Large-scale machine learning on heterogeneous systems*, Software available from tensorflow.org, 2015. https://www.tensorflow.org

[10] *Xilinx VCK190 Evaluation Board*. https://www.xilinx.com/products/boards-and-kits/vck190.html

[11] M. Caselle, "KAPTURE-2. a picosecond sampling system for individual THz pulses with high repetition rate," *J. Instrum.*, vol. 12, p. C01040, 2017. doi:10.1088/1748-0221/12/01/c01040

[12] L. Rota *et al.*, "KALYPSO: Linear array detector for high-repetition rate and real-time beam diagnostics," *Nucl. Instrum. Methods Phys. Res., Sect. A*, vol. 936, pp. 10–13, 2019. doi:10.1016/j.nima.2018.10.093

[13] O. Manzhura *et al.*, "Terahertz Sampling Rates with Photonic Time-Stretch for Electron Beam Diagnostics," in *Proc. IPAC'22*, Bangkok, Thailand, 2022, pp. 263–266. doi:10.18429/JACoW-IPAC2022-MOPOPT017

[14] Dimtel, Inc., https://www.dimtel.com

[15] W. Wang, "Towards intelligent data acquisition systems with embedded deep learning on MPSoC," en, Ph.D. dissertation, Karlsruher Institut für Technologie (KIT), Germany, 2021. doi:10.5445/IR/1000133898

[16] M. E. O'Neill, "PCG: A family of simple fast space-efficient statistically good algorithms for random number generation," Harvey Mudd College, Tech. Rep. HMC-CS-2014-0905, 2014.

[17] *Xilinx ZCU102 Evaluation Board*. https://www.xilinx.com/products/boards-and-kits/ek-u1-zcu102-g.html

[18] M. Caselle *et al.*, "A high-speed DAQ framework for future high-level trigger and event building clusters," *J. Instrum.*, vol. 12, p. C03015, 2017. doi:10.1088/1748-0221/12/03/c03015