2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 14-18, 2024. Abu Dhabi, UAE

SwiftEagle: An Advanced Open-Source, Miniaturized FPGA UAS Platform with Dual DVS/Frame Camera for Cutting-Edge Low-Latency Autonomous Algorithms

Christian Vogt[1], Michael Jost[1] and Michele Magno[1]

***Abstract*— Low-latency sensing and decision-making processing are critical requirements for the highly dynamic control and perception applications often found in Unmanned Areal Systems (UASs). Novel sensors such as Dynamic Vision Sensors (DVSs) are enhancing the pure performance of the perception component with orders of magnitude lower latency. However, they are typically not optimally integrated with the computing hardware, which effectively reduces the potential in both latency and power consumption. In addition to the non-optimal integration, low latency processing of such high data rate generating sensors is often challenging on resource-constrained platforms. Here, Field Programmable Gate Arrays (FPGAs) platforms offer a promising set of features, including low-level access to hardware and memories, as well as high-speed interfaces. On the other hand, FPGA platforms are not as popular on UASs due to their complex programming and development environments, steep initial learning curve, and challenges in achieving optimal performance. To accelerate future development, this paper presents SwiftEagle, an open-source, cutting-edge $720\,\mathrm{g}$ FPGA based UAS based on a custom hardware design including a dual camera interface RGB/DVSs on a multi-sensors subsystem and an initial software and firmware stack designed for high precision recording of machine learning datasets in-flight with sub-micro-second time resolution and on-FPGA rendering of DVS event frames. Utilizing this developed platform, as proof-of-concept the end-to-end latency of a novel, just released DVS is shown to be below $210\,\mu\mathrm{s}$ as a worst-case scenario, enabling future cutting-edge autonomous algorithms.**

I. Introduction

The most important tasks of UAS, such as autonomous obstacle avoidance [1], or racing [2], require fast reaction times. To push these boundaries, very low latency of the sensors and decision-making processing is crucial. One promising technology to accelerate perception latency involves event-based neuromorphic sensors and processing (e.g. [3]), which process input data as soon as it becomes available from the sensors.

Given the low-latency and high data bandwidth requirement for UASs, gathering information from the environment and supplying this information as soon as it is available to the processing system is essential. This may be achieved with classical sensors via data-ready interrupts, such as intelligent Inertial Measurement Units (IMUs) detecting an event (tapping, shaking), electromagnetic sensing (UWB distance, RADAR sensors, LiDAR ...) [4], [5] or event-based DVS cameras. These revolutionary cameras offering high spatial and temporal information density in a wide range of applications [6], [7], [8], as they do not rely on waiting for the completion of whole image frames, but send changes of pixel intensities immediately. This allows for high frame rates [8] and dynamic range compared to standard frame-based cameras [9]. However, the performance of these cameras in challenging fast-moving environments is often constrained by their reliance on USB connections for data transmission and the consecutive limited data rate [10], as highlighted in [11], [7].

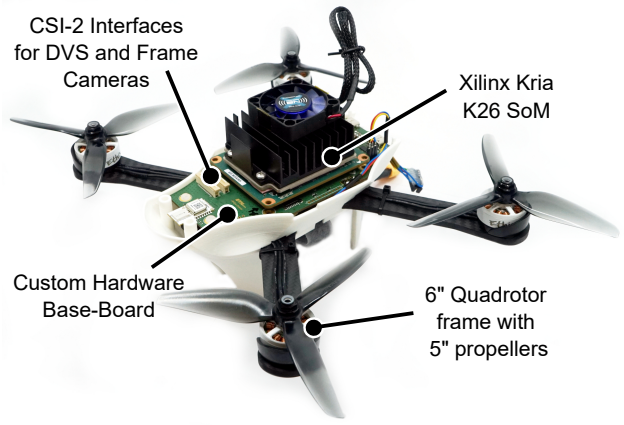

Fig. 1. Developed open-source UAS platform mounted on an 6 inch quadrotor frame.

To utilize this event-based sensing paradigm optimally, also the processing system needs to be capable of executing low-latency computations and be aware of precise timing relations between the sensor values. This precision is vital for both, dataset recording facilitating offline algorithm development, and for online real-time processing [5]. While ms-resolution timestamping can be achieved sampling sensors on a micro controller or even operating system, moving to applications targeting novel sensors like DVS, the timestamping precision becomes more important. However, existing datasets (e.g. [12]) often fail to provide high-performance synchronization of DVS, frame, and IMU data, a shortfall largely attributable to the limited capabilities of recording platforms and sensor interfaces. While various studies have leveraged DVS for low-latency applications, they frequently do not characterize the latency inherent to the camera interfaces.

This work was supported in part by the Sony Research Award Program 2022

[1] All authors are associated with Center for Project Based Learning, D-ITET, ETH Zurich, Switzerland. (Corresponding author: Christian Vogt, vogtch@ethz.ch)

Changing from the sensing and dataset collection to the on-board real-time processing, existing digital platforms for UAS, such as GPU platforms, are not yet fully equipped to fully exploit the low-latency potential of these novel sensors. Therefore, our approach to ensure faster interface and processing is the employment of FPGA-based processing elements on UAS platforms. Such FPGA based drones have been shown to speed up tasks such as sensor fusion and inference [13] on frame cameras and mmWave RADARs, or detection of landing signs [14].

FPGAs are generally more challenging to program and require specific skills and time, despite their potential to achieve higher performance [15]. Therefore, today only a few designs with FPGA based UAS exist. These designs are mostly assembled from off-the-shelf FPGA evaluation boards and separate flight controllers, interconnected by low-speed serial interfaces for transmission of set-points to the flight controller, limiting the access of the high-performance computing to low-level raw sensor data and motor actuation. To the best of our knowledge none is currently available as open-source in hardware and software [16].

To address these challenges, we have developed an open-source FPGA-based UAS platform named SwiftEagle (Fig. 1). The platform is able to provide both, precise temporal synchronization in the $10\,\mathrm{ns}$ range and high-throughput data processing. The precise temporal alignment is crucial for synchronizing high-speed event-based data with other sensor inputs, enhancing data coherence for processing. This enables researchers to exploit the full potential of sensor technologies and hardware-accelerated algorithms.

Within this work, we have furthermore meticulously characterized the end-to-end latency of our DVS interface by making use of our custom integrated precision timestamping of sensor values. Moreover, SwiftEagle incorporates a direct integration of a standard Camera Serial Interface 2 (CSI-2) camera interface, effectively eliminating the bottleneck issues related to latency and data throughput with off-the-shelf USB interfaced cameras, streamlining the data flow for maximum efficiency.

The contribution of this paper can be summarized as follows:

- **Design of the SwiftEagle, an open-source hardware and software platform leveraging FPGA technology within a UAS**, designed specifically for exceptional low-latency processing performance. This state-of-the-art platform seamlessly integrates a flight controller, offering comprehensive accessibility to all onboard sensors directly through the FPGA. This enables advanced real-time data processing and decision-making capabilities, especially in the field of low-latency autonomous navigation and detection algorithms.
- **Enabling the acquisition of highly synchronized datasets, featuring microsecond-precision in sensor timing alignment**, thanks to the integration of a DVS camera with a frame-based counterpart on the fully functional, UAS platform. This combination of sensors, directly interfaced with the FPGA, ensures the lowest possible latency and best timing alignment. This not only enhances data quality but also allows exploration of algorithms sensitive to misalignment of data timestamps.
- **Showcasing the capability for dataset acquisition featuring microsecond-precision** in sensor timing alignment. A proof-of-concept shows the platform's ability to perform rapid, on-FPGA processing of DVS data—decoding event data and rendering event frames in real-time, while also evaluating the DVS end-to-end latency.

The custom hardware and firmware of the SwiftEagle are available as open-source on `https://github.com/ETH-PBL/SwiftEagle`.

II. Related Work

This section gives an overview of UAS platforms with on-board FPGAs, as well as available software solutions to integrate FPGA accelerators in classical robotics software stacks and applications. Finally, sensor synchronization with a focus on robotics is discussed.

A. UAS FPGA Platforms

UAS platforms integrating dedicated computing accelerators are progressing, especially for high performance applications. [17] present and benchmark the ReDroSe UAS equipped with a Xilinx UltraScale+ FPGA as well as a Jetson Xavier NX evaluation board to accelerate Light Detection and Ranging (LiDAR) and Simultaneous Localization and Mapping (SLAM) on an approx. $4\,\mathrm{kg}$ UAS. Similarly, [13] show an FPGA (Avnet Ultra-96 board) equipped with a Global Navigation System (GNS), Universal Serial Bus (USB) frame camera and a mmWave Radio Detection and Ranging (RADAR) at a take-off weight of $612\,\mathrm{g}$. [14] also uses an Avnet Ultra-96 FPGA board with a Pixhawk flight controller on a custom-build quadrotor.

Our SwiftEagle platform presents a fully integrated approach of the hardware components and the flight control system at $720\,\mathrm{g}$. It fully integrates the camera interfaces with high speed connections, freeing the USB ports for future extensions.

B. FPGAs for Robotics

With respect to integrating FPGAs and System on Chip (SoC) into common robotic stacks with low effort, such as Robot Operating System (ROS), significant progress was made in recent years. For example in ROS2 [18] , [19] show tool-chains for simple integration of ROS nodes either on the Central Processing Unit (CPU) or on the logic fabric, allowing for custom hardware accelerators to be easily interfaced with widely available ROS nodes. Similarly, [20] show an FPGA interface, also incorporating communication with an external Pixhawk flight controller.

FPGA integrated hardware accelerators have been shown in robotics for common navigation algorithms like Visual Inertial Odometry (VIO) or Visual Odometry (VO), which offer multiple opportunities for acceleration. For example,

[21] accelerate deep learning VO (Monodeph2, DeepVO and DFVO) on an Xilinx Zynq UltraScale+ FPGA and Nvidia Jetson Nano. Similarly, [22] present an VO pipeline integrated with High Level Synthesis for an FPGA. Also subsets of VIO algorithms, like Harris-corner detectors [23] are prime-candidates for dedicated hardware acceleration. Also more resource-demanding algorithms are accelerate, such as SLAM [24], implemented as an hardware co-processor. Also DVS accelerators in UAS applications exist, e.g. for tracking detection [25].

In our application, we accelerate the DVS event to event-frame rendering directly on the FPGA, as well as implement precise sensor time-stamping, freeing the CPU for high level tasks.

C. Sensor Synchronization

In navigation algorithms, such as VIO, the synchronization between IMU and frame camera plays a significant role. In common datasets and algorithms, effort is taken to estimate and synchronize the IMU and frame-based sensor data streams with respect to each other, e.g. [26] or the VINS pipeline [27]. For example, [28] compare OKVIS [29], VINS-Mono [27] and their custom self-alignment algorithm with respect to IMU and camera frame miss-alignment. Here, a simulated delay of 30 ms already increases the relative translation error by a factor of 5 when using the Euroc [30] dataset, if not corrected.

Our proposed platform enables the collection of datasets exceeding the state of the art by sub-micro-second sensor data alignment, while reducing the overall sensor-to-recording latency by directly interfacing with high data rate sensors, such as the DVS.

III. Methodology

This section gives an overview of the open-source hardware and software/firmware developed, as well as the design choices and latency measurement setups.

A. FPGA-based drone hardware

The proposed hardware for the FPGA drone is based on a standard quad-rotor frame (Armattan Chameleon OG 6" base plate), with 2306 2345kV motors (Ethix) and a 60 A quad motor controller BLHeli32-DS1200 (Hobbywing). Optional RC remote control is provided by a commercial 2.4 GHz remote control (FrSky XM PLUS Mini). For connectivity e.g. ROS2 telemetry, a WiFi USB dongle is attached (Ralink 5390 USB).

Next to these commercial components, a custom hardware depicted in Fig. 2 is employed, designed to use a Xilinx K26 System on Module (SoM) FPGA module as main computing module. The SoM contains a single chip integrating a quad-core ARM A53 Application Processing Unit (APU), a dual-core ARM R5 Real-time Processing Unit (RPU), as well as a FPGA fabric allowing full or partial configuration by the ARM processors. Additionally, 4 Gbit DDR and 16 Gbit flash memory are available on the SoM. The custom hardware directly integrates the power supplies

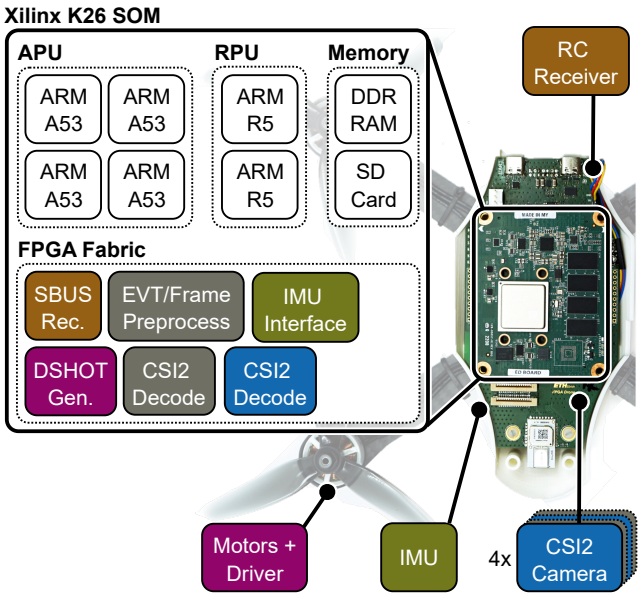

Fig. 2. Overview of the FPGA-based drone hardware depicting the external hardware, custom hardware, as well as the implemented digital modules located in the FPGA fabric.

including on-board current measurement on each power rail (Texas Instruments INA3221), from the battery voltage (4s LiPo, 1800 mA h, Tattu R-line). Furthermore, the IMU (ST Microelectronics LSM6DSV) and an extension interface for additional sensors (Crazyflie 2 shield extension [31]) are integrated in the custom hardware, as well as a USB PHY (Microchip USB3320). A DC isolated interface for the motor driver (Texas Instruments ISO7140), a level shifter for the 5 V radio control receiver signals as well as interfaces for up to four CSI-2 based cameras (2 lanes each) are available. Currently one frame-based camera (SONY IMX219) and one DVS camera (Prophesee GenX320) are attached to these ports. The overall weight of the custom electronics hardware is 64 g, the hardware including the SoM and cooler amounts to 193 g. The SwiftEagle possesses a total take-off weight of 720 g (including quadrotor frame, motors, frame-based camera, DVS and battery).

To support future low latency processing applications, all sensors and external interfaces are connected directly to the FPGA fabric, allowing custom logic to process the sensor signals or generate e.g. the motor signals before forwarding them to the APU and RPU processors. These can directly access either the FPGA fabric or the memory to interact with any pre-processed sensor data.

B. Firmware and Logic Design

The main firmware of the FPGA drone is structured into three layers, with latency and real-time requirements going from low in 1) to high in 3):

1) Petalinux: the Xilinx-supported Linux operating system based on the YOCTO distribution. It operates on the quad-core ARM Cortex A53 processor. The ROS environment (ROS2 humble) and all nodes are hosted on this platform inside a docker container.

2) CF2 firmware: the open-source drone controller used by the Crazyflie2 was ported to run on the dual-core ARM Cortex R5 processors, interfacing with the IMU and the motor controller via the FPGA fabric. The corresponding hardware drivers were adapted on the stock-Crazyflie firmware. Information with the ROS2 nodes on the APU is exchanged via the Xilinx IPI interface, e.g. controller set-points, drone orientation and state.

3) FPGA Logic: the nodes integrated in the FPGA fabric contain operations with the highest latency and timing sensitivity, such as generating the DSHOT protocol for the motor drivers, pre-processing the SBUS signal of the RC remote and providing immediate motor shutdown when the safety-switch on the remote is pressed. Furthermore, the logic integrates time-stamping for data-alignment between the different sensors (IMU, frame cameras and DVS cameras), as well as various commercial logic nodes, such as the MIPI-CSI2 decoder core (Xilinx).

C. 10 nano-second aligned dataset recording

In order to record a high quality dataset with the presented platform, the data streams of the individual sensors are time-stamped within the FPGA logic and thus can be aligned within the precision of the current system clock (100 MHz), allowing 10 ns timestamp resolution. Figure 3 presents the general overview of the extendable time stamping logic at the example of the IMU, DVS and frame camera, with the data streams to the real-time processing not shown. For new sensors, the designed HDL modules could be individually added to the monitored sensor input. The lightweight time stamping logic allows precise recording of both camera inputs simultaneously, even at full speed. The direct CSI-2 data processing and storage via the FPGA fabric is designed to handle the maximum DVS data rate of two DVS sensors (2x1.5 Gbit s^{-1} [32]), removing the bottleneck of previous USB implementations [10].

Each of the sensors offers a different kind of data stream, with the IMU raising regular interrupts when new samples are ready (set to 1 kHz), to the DVS camera with its own time stamps sending events at irregular intervals and finally the frame camera, with 50 frames per second.

At each regular data ready interrupt of the IMU the current system time counter is stored and read together with the IMU sample data by the firmware of the RPU and stored in the DDR memory bank. In contrast, the frame camera timestamps are generated when the transmission start of a new camera frame is detected and inserted before the frame image into the data stream, which is directly written to the DDR memory. Similarly, the time stamps for the DVS camera are also directly inserted into the event stream, however, as the event data does not consist of frames, the system timestamps are inserted regularly (every 4096 bytes) into the data stream. By comparing DVS and fabric rate at time frames where the event rate drops from high values to zero, the latency of the internal FPGA data processing can be estimated.

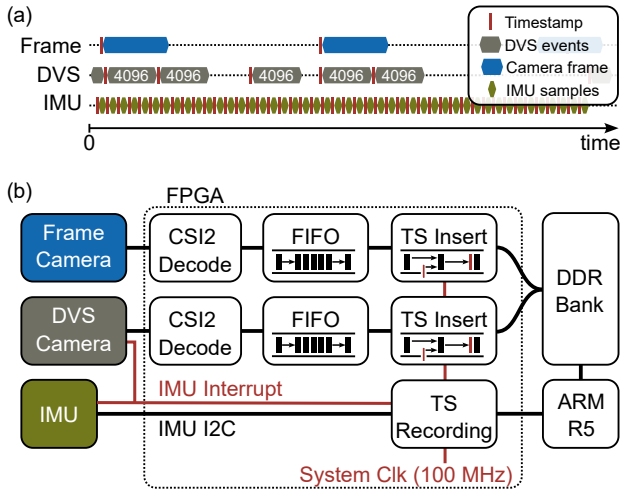

Fig. 3. Example timestamp insertion for different data arrival, e.g. frame camera, IMU and DVS data (a). Implemented time stamping functionality based on the FPGA system clock of 100 MHz. For the frame and DVS camera, the time stamps are directly inserted in the data flow, for the IMU the timestamps are created at each sensor interrupt and then collected by the RPU. (b)

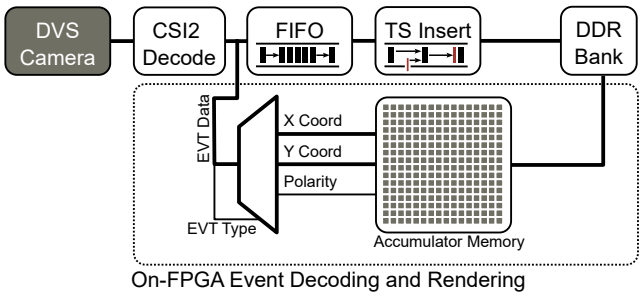

Fig. 4. Parallel On-FPGA event decoding and aggregation into frames with transfer of variable integration-time frames to the DDR memory bank for access by the APU or other logic.

D. Event data decoding and processing

The event data from the DVS is encoded in the EVT2.1 format [33] and received from the camera via a CSI-2 interface with 1 data lane. Figure 4 depicts the high-level processing done on the received CSI-2 data.

First, the CSI-2 format is decoded by a commercial logic core (Xilinx). Afterwards, the decoded data is processed by a custom event decoder and event frame aggregator. The main advantage of this design is the inherently low latency by design. The complete decoding and accumulator logic is designed to operate at the system clock and process all data received by the CSI-2 decoder within two system clock cycles (50 MHz or 20 ns). The resulting data rate (1.6 Gbit s^{-1} for the EVT2.1 format) exceeds the DVS camera interface (1.5 Gbit s^{-1}), ensuring no event data loss.

Apart from the raw input event processing, the accumulator memory can be adapted on the fly for different accumulation times, providing the resulting event frames to the DDR memory for further processing by other logic blocks or the APU.

E. Latency measurement setup

In order to characterize the latency of the complete processing system consisting of DVS camera and decoding of the data stream inside the FPGA, a waveform generator (Keysight EDU33212A) is used to synchronously strobe a red Light Emitting Diode (LED) (Wurth 150060RS75003) at approx. 6 V as well as generate a trigger output. The trigger output is wired to an FPGA input, which is configured to time-stamp the occurrence of each trigger event. the LED flash is repeated every 100 ms with a duration of 5 ms.

Further comparing the processing inside the FPGA fabric, the regular timestamps inserted every 4096 bytes are utilized and compared to the data extracted from the DVS raw data stream. The time delay between two timestamps indicates the transfer rate inside the fabric, as well as a lower bound on the event rate. The event rate bound can be calculated by considering 8 byte per event in the data stream, therefore, between two timestamps at least 4096/8 events were transmitted. The lower bound exists, as the EVT2.1 data format allows compression of more events into the 8 bytes, in case of high event rates at the DVS sensor (up to 32 events) [33].

IV. Experimental Results

The following section lists the experimental results, starting with the evaluation of the relative sensor clock/sampling jitter on a dataset recorded by the SwiftEagle, the measured end-to-end latency of the employed DVS and finally an overview of the FPGA resources utilized by the presented design.

Due to the centralized time stamping in the FPGA, the jitter of each sensor is measured. Fig. 5 shows the time elapsed (dt) between two consecutive timestamps for the three sensors observed currently. The resulting jitter can be extracted looking at the deviation around the mean value of dt. The IMU timestamps jitter below 0.1 ms over the measurement duration with dt around 1 ms that corresponds to the expected sampling rate (1 kHz). Similarly, the frame camera shows sub-1 µs jitter between the frames and an absolute deviation from the expected frame-per-second, with an inter-frame spacing of 21 ms. Due to the dynamic movement of the UAS in the example Fig. 5, the timestamps of the DVS are closely spaced, with less than 1 ms between every 4096 bytes event data. This can also be used as simple metric to estimate the event rate of the DVS.

Most perception and navigation algorithms rely on precise knowledge of the clock or time-base drift between individual sensors (section II). Based on the timestamps extracted from the measurements, the drift is evaluated. Whereas the DVS keeps over 5 s within approx. 40 µs, the IMU drifts more than 170 ms with respect to the expected value based on the IMU settings. The developed onboard time-stamping allows also live access to this time-base drift, enabling future algorithms to adapt to changing clock rates (e.g. due to temperature changes).

Looking into the signal processing and acquisition latency, Fig. 6 presents the results of the DVS processing delay experiment. The overall generated events per micro-second

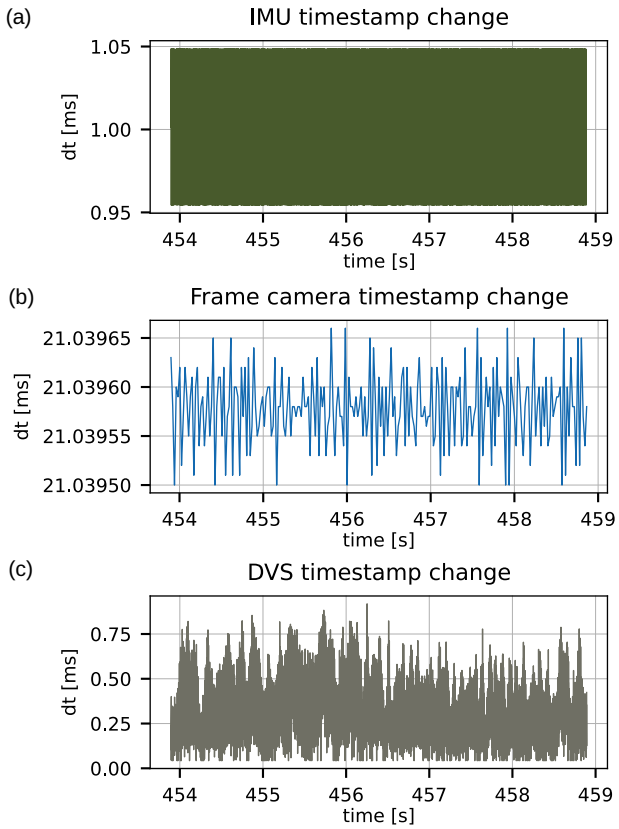

Fig. 5. Time elapsed (dt) between two consecutive timestamps of the three main sensors: IMU (a), frame camera (b) and DVS (c), as recorded by the FPGA while the drone is flying.

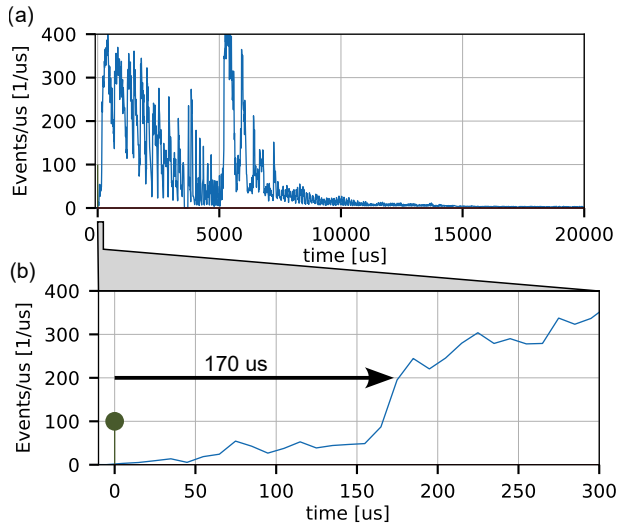

Fig. 6. Onset of LED-flash until event rate rising, collected events per μ-second (a) and detail view with an onset of events at approx. 170 µs (b).

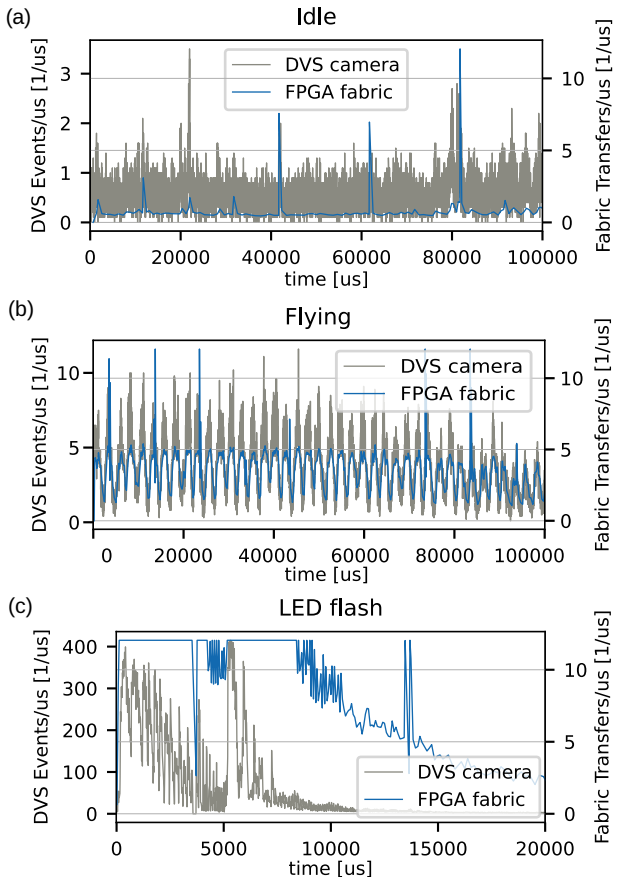

Fig. 7. DVS event rate compared with data rate in FPGA fabric at various event densities: idle (a), UAS flying (b) and LED flash (c). No data is lost, but partially buffered by the MIPI CSI-2 input decoder in the fabric.

are depicted in Fig. 6a, with peaks of up to $400\,\mathrm{Event}\,\mu\mathrm{s}^{-1}$. The first part up to $5\,\mathrm{ms}$ consists mostly of positive events, the second part of events with negative polarity. This fits well with the rising and falling edge of the light pulse intensity. The DVS latency is evaluated in Fig. 6b, with the LED-pulse trigger marked as gray dot. The onset of the light pulse in the event data is defined as the rising edge in event rate, yielding a latency of approx. $170\,\mu\mathrm{s}$ for the DVS camera. This corresponds well with reported latency range of similar DVS models [34].

To further characterize the on-board DVS processing in terms of latency, the data rate inside the fabric and recorded by the DVS sensor is compared in Fig. 7. Raw event data is recorded to DDR memory during this experiment, limiting the recording to approx. $5\,\mathrm{s}$. The low event counts in the idle case (Fig. 7a) are approximately $1\,\mathrm{Event}\,\mu\mathrm{s}^{-1}$, with the FPGA fabric transfer rate mirroring this rate. Noticeably, peaks of up to $10\,\mathrm{Transfer}\,\mu\mathrm{s}^{-1}$ can be observed, depending on the buffer fill state. Increasing the event rate in Fig. 7b, the DVS generates up to $12\,\mathrm{Event}\,\mu\mathrm{s}^{-1}$, the transfer rate in the fabric yields a maximum lower bound on transferred data between 1 and $12\,\mathrm{Transfer}\,\mu\mathrm{s}^{-1}$. In the extreme case of the LED flash, this saturation of the fabric transfer rate is clearly visible, compared to the max. $400\,\mathrm{Event}\,\mu\mathrm{s}^{-1}$ generated by

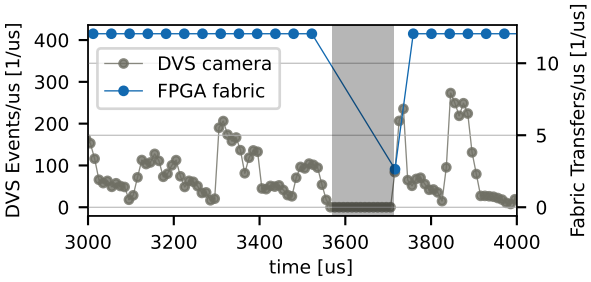

Fig. 8. Drop of fabric data transfers in region with no DVS event generation, marked in gray.

the DVS. However, no data is lost, the saturation originates from the internal MIPI CSI-2 decoder.

The maximum latency introduced by the CSI-2 to the FPGA fabric bus is depicted in Fig. 8 and is collected synchronously with the DVS data set presented in Fig. 6. The data transfer rate dropping to approx. $2.5\,\mathrm{Transfer}\,\mu\mathrm{s}^{-1}$ after a longer period with no DVS event generation. This indicates a maximum latency after FPGA processing of approx. $210\,\mu\mathrm{s}$.

The overall utilization of the FPGA logic design is listed in Tab. I, with the majority of resources being utilized for the DVS frame rendering proof-of-concept.

TABLE I
FPGA UTILIZATION OF BASIC DESIGN

Resource	Used	Available	Utilization [%]
LUT as Logic	48035	117120	41.01
LUT as Memory	32836	57600	57.01
FF	45922	234240	19.60
BRAM	110	144	76.39
DSP	168	1248	13.46

V. CONCLUSION

This paper presents SwiftEagle, a $720\,\mathrm{g}$ open source FPGA based UAS platform, including custom hardware as well as an initial software and firmware stack to operate the UAS. A sub-micro second timestamping and alignment of sensor data is integrated in the FPGA fabric next to precise dataset recording capability of IMU, frame camera and DVS data in flight. In addition to the raw sensor values the clock drifts as well as clock jitter ($< 1\,\mu\mathrm{s}$) of the sensors are evaluated on a dataset collected with the SwiftEagle. Furthermore, the latency of the implemented FPGA hardware design is measured to be below $210\,\mu\mathrm{s}$ from a light strobe at the DVS until data is available in the FPGA fabric under different conditions. Therefore, the presented platform enables future low latency algorithm development for resource constrained mobile platforms.

REFERENCES

[1] D. Falanga, K. Kleber, and D. Scaramuzza, "Dynamic obstacle avoidance for quadrotors with event cameras," *Science Robotics*, vol. 5, no. 40, p. eaaz9712, 2020.

[2] E. Kaufmann, L. Bauersfeld, A. Loquercio, M. Müller, V. Koltun, and D. Scaramuzza, "Champion-level drone racing using deep reinforcement learning," *Nature*, vol. 620, no. 7976, pp. 982–987, 2023.

[3] T. Schoepe, E. Janotte, M. B. Milde, O. J. N. Bertrand, M. Egelhaaf, and E. Chicca, "Finding the gap: neuromorphic motion-vision in dense environments," *Nature Communications*, vol. 15, no. 1, p. 817, 2024.

[4] J. Michalczyk, R. Jung, and S. Weiss, "Tightly-coupled ekf-based radar-inertial odometry," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 12 336–12 343.

[5] P. Li, K. Cai, M. R. U. Saputra, Z. Dai, and C. X. Lu, "Odombeyondvision: An indoor multi-modal multi-platform odometry dataset beyond the visible spectrum," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 3845–3850.

[6] W. Guan, P. Chen, Y. Xie, and P. Lu, "Pl-evio: Robust monocular event-based visual inertial odometry with point and line features," *IEEE Transactions on Automation Science and Engineering*, 2023.

[7] A. Di Mauro, M. Scherer, J. F. Mas, B. Bougenot, M. Magno, and L. Benini, "Flydvs: An event-driven wireless ultra-low power visual sensor node," in *2021 Design, Automation & Test in Europe Conference & Exhibition (DATE)*. IEEE, 2021, pp. 1851–1854.

[8] R. Tapia, J. P. Rodríguez-Gómez, J. A. Sanchez-Diaz, F. J. Gañán, I. G. Rodríguez, J. Luna-Santamaria, J. Martínez-De Dios, and A. Ollero, "A comparison between framed-based and event-based cameras for flapping-wing robot perception," in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2023, pp. 3025–3032.

[9] C. Li, L. Longinotti, F. Corradi, and T. Delbruck, "A 132 by 104 10μm-pixel 250μw 1kefps dynamic vision sensor with pixel-parallel noise and spatial redundancy suppression," in *2019 Symposium on VLSI Circuits*. IEEE, 2019, pp. C216–C217.

[10] J. Moosmann, J. Mandula, P. Mayer, L. Benini, and M. Magno, "Quantitative Evaluation of a Multi-Modal Camera Setup for Fusing Event Data with RGB Images," in *2023 IEEE SENSORS*. Vienna, Austria: IEEE, Oct. 2023, pp. 1–4.

[11] S. Bian, L. Schulthess, G. Rutishauser, A. Di Mauro, L. Benini, and M. Magno, "Colibriuav: An ultra-fast, energy-efficient neuromorphic edge processing uav-platform with event-based and frame-based cameras," in *2023 9th International Workshop on Advances in Sensors and Interfaces (IWASI)*. IEEE, 2023, pp. 287–292.

[12] P. De Tournemire, D. Nitti, E. Perot, D. Migliore, and A. Sironi, "A large scale event-based detection dataset for automotive," *arXiv preprint arXiv:2001.08499*, 2020.

[13] N. H. Malle, F. F. Nyboe, and E. S. M. Ebeid, "Onboard powerline perception system for UAVs using mmWave radar and FPGA-accelerated vision," *IEEE Access*, vol. 10, pp. 113 543–113 559, 2022.

[14] B. B. Kovari and E. Ebeid, "MPDrone: FPGA-based platform for intelligent real-time autonomous drone operations," in *2021 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*. IEEE, 2021, pp. 71–76.

[15] Z. Wan, A. Lele, B. Yu, S. Liu, Y. Wang, V. J. Reddi, C. Hao, and A. Raychowdhury, "Robotic computing on fpgas: Current progress, research challenges, and opportunities," in *2022 IEEE 4th International Conference on Artificial Intelligence Circuits and Systems (AICAS)*. IEEE, 2022, pp. 291–295.

[16] Z. Wan, B. Yu, T. Y. Li, J. Tang, Y. Zhu, Y. Wang, A. Raychowdhury, and S. Liu, "A survey of fpga-based robotic computing," *IEEE Circuits and Systems Magazine*, vol. 21, no. 2, pp. 48–74, 2021.

[17] S. Rahn, P. Gehricke, C.-L. Petermöller, E. Neumann, P. Schlinge, L. Rabius, H. Termühlen, C. Sieh, M. Tassemeier, T. Wiemann, and M. Porrmann, "ReDroSe — reconfigurable drone setup for resource-efficient SLAM," in *DroneSE and RAPIDO: System Engineering for constrained embedded systems*. ACM, 2023, pp. 20–30.

[18] V. Mayoral-Vilches, S. M. Neuman, B. Plancher, and V. J. Reddi, "RobotCore: An open architecture for hardware acceleration in ROS 2," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 9692–9699.

[19] C. Lienen, S. H. Middeke, and M. Platzner, "fpgaDDS: An intra-FPGA data distribution service for ROS 2 robotics applications."

[20] F. F. Nyboe, N. H. Malle, and E. Ebeid, "MPSoC4drones: An open framework for ROS2, PX4, and FPGA integration," in *2022 International Conference on Unmanned Aircraft Systems (ICUAS)*. IEEE, 2022, pp. 1246–1255.

[21] S. Chen and K. Mai, "Towards specialized hardware for learning-based visual odometry on the edge," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 10 603–10 610.

[22] R. Ye, K. Iordanou, G. Riley, and M. Luján, "Exploring sparse visual odometry acceleration on FPGA with high-level synthesis," *IEEE Access*, pp. 1–1, 2023.

[23] P. Gu, Z. Meng, and P. Zhou, "Real-time visual inertial odometry with a resource-efficient harris corner detection accelerator on FPGA platform," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 10 542–10 548.

[24] Y. Tan, H. Deng, M. Sun, M. Zhou, Y. Chen, L. Chen, C. Wang, and F. An, "A reconfigurable coprocessor for simultaneous localization and mapping algorithms in FPGA," *IEEE Trans. Circuits Syst. II*, vol. 70, no. 1, pp. 286–290, 2023.

[25] S. N. Aspragkathos, E. Ntouros, G. C. Karras, B. Linares-Barranco, T. Serrano-Gotarredona, and K. J. Kyriakopoulos, "An event-based tracking control framework for multirotor aerial vehicles using a dynamic vision sensor and neuromorphic hardware," in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2023, pp. 6349–6355.

[26] J. Liu, Z. Hu, and W. Gao, "Online temporal calibration of camera and IMU using nonlinear optimization," in *2018 24th International Conference on Pattern Recognition (ICPR)*. IEEE, 2018, pp. 1761–1766.

[27] T. Qin, P. Li, and S. Shen, "VINS-mono: A robust and versatile monocular visual-inertial state estimator," *IEEE Transactions on Robotics*, vol. 34, no. 4, pp. 1004–1020, 2018.

[28] Y. Ling, L. Bao, Z. Jie, F. Zhu, Z. Li, S. Tang, Y. Liu, W. Liu, and T. Zhang, "Modeling varying camera-IMU time offset in optimization-based visual-inertial odometry," in *Computer Vision – ECCV 2018*, V. Ferrari, M. Hebert, C. Sminchisescu, and Y. Weiss, Eds. Springer International Publishing, 2018, vol. 11213, pp. 491–507, series Title: Lecture Notes in Computer Science.

[29] S. Leutenegger, S. Lynen, M. Bosse, R. Siegwart, and P. Furgale, "Keyframe-based visual–inertial odometry using nonlinear optimization," *The International Journal of Robotics Research*, vol. 34, no. 3, pp. 314–334, 2015.

[30] M. Burri, J. Nikolic, P. Gohl, T. Schneider, J. Rehder, S. Omari, M. W. Achtelik, and R. Siegwart, "The euroc micro aerial vehicle datasets," *The International Journal of Robotics Research*, 2016. [Online]. Available: http://ijr.sagepub.com/content/early/2016/01/21/0278364915620033.abstract

[31] "Expansion decks of the crazyflie 2.x," https://www.bitcraze.io/documentation/system/platform/cf2-expansiondecks/, accessed: 2024-02-15.

[32] "Genx320 datasheet," this confidental preliminary datasheet has been provided by Prophesee.

[33] "Prophesee evt 2.1 format," https://docs.prophesee.ai/stable/data/encoding_formats/evt21.html, accessed: 2024-02-15.

[34] G. Gallego, T. Delbruck, G. Orchard, C. Bartolozzi, B. Taba, A. Censi, S. Leutenegger, A. J. Davison, J. Conradt, K. Daniilidis, and D. Scaramuzza, "Event-based vision: A survey," vol. 44, no. 1, pp. 154–180.