PROCEEDINGS OF SCIENCE

FPGA-Based Data Processing using High-Level Synthesis on the Antarctic Demonstrator for the Advanced Particle-astrophysics Telescope (ADAPT)

Marion Sudvarg,[a,b,*] **Longhao Huang,**[c] **Boran Yang,**[b] **Blake Bal,**[a] **Roger Chamberlain,**[b] **Jeremy Buhler,**[b] **Leonardo Di Venere,**[d] **Davide Serini**[d] **and James Buckley**[a] **for the APT collaboration**

[a] *Dept. of Physics & McDonnell Center for Space Sciences, Washington Univ., St. Louis, MO, United States*
[b] *Dept. of Computer Science & Engineering, Washington Univ., St. Louis, MO, United States*
[c] *Dept. of Electrical & Systems Engineering, Washington Univ., St. Louis, MO, United States*
[d] *Istituto Nazionale di Fisica Nucleare (INFN), Bari, Italy*
E-mail: msudvarg@wustl.edu, h.longhao@wustl.edu, boran.y@wustl.edu, b.bal@wustl.edu, roger@wustl.edu, jbuhler@wustl.edu, leonardo.devinere@ba.infn.it, davide.serini@ba.infn.it, buckley@wustl.edu

FPGAs are widely deployed on high-energy astrophysics telescopes to read out sensor data from front-end electronics. To support continuous data streams or high trigger rates, FPGA logic may be employed to process raw sensor readout values, reducing the volume of data transmitted, processed, and stored by downstream CPU-based computational platforms. Across instruments, these FPGA-based processing pipelines often have similar semantics and share common stages. However, diverse telescope designs require unique implementations of the constituent algorithms, and the logic is often rewritten from scratch for a new instrument. Writing, simulating, and debugging firmware is difficult and time consuming. However, High-Level Synthesis (HLS) allows these algorithms to be implemented in a high-level language, enabling fast prototyping and deployment. Nonetheless, writing performant HLS code is not straightforward, and requires an understanding of how the synthesis tools convert high-level language constructs, compiler-specific pragmas, and vendor-provided template libraries to hardware circuits.

This work presents an initial HLS library of common algorithms for deployment in particle astrophysics detectors. We apply it to the Antarctic Demonstrator for the Advanced Particle-astrophysics Telescope (ADAPT), an MeV–TeV gamma-ray instrument that combines a pair tracker and Compton telescope anticipated to fly on a high-altitude balloon during the 2026–27 Antarctic season. Our library enables efficient processing of digitized waveform data from ADAPT's >2000 sensor readout channels, including per-channel photon counting and cross-channel centroiding. Our library can process hundreds of thousands of event triggers per second when deployed on embedded FPGAs flying aboard the ADAPT instrument, while providing flexibility and extensibility.

39th International Cosmic Ray Conference (ICRC2025)
15–24 July 2025
Geneva, Switzerland

*Speaker

1. Introduction

FPGAs are widely deployed on high-energy astrophysics telescopes to read out sensor data from front-end electronics. To support continuous data streams or high trigger rates, FPGA logic may be employed to process raw sensor readout values, reducing the volume of data transmitted, processed, and stored by downstream CPU-based computational platforms. Examples include the Flash ADC electronics system for VERITAS [1] and the focal plane module readout on NUSTAR [2].

Across instruments, these FPGA-based processing pipelines share common functionality. On imaging atmospheric Cherenkov telescopes like VERITAS [3] and CTA [4] and Compton telescopes like APT [5] and ADAPT [6], optical light is captured using arrays of PMTs or SiPMs. Signals from each "pixel" are read by analog waveform digitizers, such as the TARGET [7] or NECTAr0 [8] ASICs. To enable back-end scientific analysis, an FPGA may extract high-level information about the interaction, e.g., position and energy.

However, diverse telescope designs require unique implementations of the constituent algorithms, and the logic is often rewritten from scratch for a new instrument. Traditionally, this logic is specified in a hardware description language (HDL) such as VHDL or Verilog. Writing, simulating, and hardware debugging of HDL-based firmware is rarely straightforward and introduces significant overhead to the instrument development cycle.

As an alternative, High-Level Synthesis (HLS) tools enable these algorithms to be implemented in a high-level language, such as C or C++, which eases modifications and enables fast prototyping and deployment. Nonetheless, writing performant HLS code is not always straightforward, and requires an understanding of how the synthesis tools convert high-level language constructs, compiler-specific pragmas, and vendor-provided template libraries to hardware circuits.

To broaden its adoption, this work presents an initial HLS library — written for the AMD Xilinx Vitis HLS platform — of common algorithms for deployment in particle astrophysics detectors. These encompass per-channel preprocessing and filtering, including ① data acquisition, ② pedestal subtraction, ③ photon counting (via peak detection, waveform integration, and gain correction), and ④ zero-suppression as well as cross-channel analysis and event building stages, including ⑤ island detection and centroiding. It uses preprocessor macros and generic programming to enable easy specification of data types corresponding to physical attributes of the telescope.

We apply it to the Antarctic Demonstrator for the Advanced Particle-astrophysics Telescope (ADAPT), a prototype high-altitude balloon mission with an anticipated flight during the 2026–27 Antarctic season. ADAPT is an MeV–TeV gamma-ray instrument that combines a pair tracker and Compton telescope in a single monolithic design by using a combination of scintillating fiber trackers and CsI:Na tiles read out with wavelength shifting (WLS) fibers for imaging and edge-mounted SiPMs for calorimetry. With ADAPT, we demonstrate the several advantages of our library, which enables efficient processing of analog waveform data from ADAPT's >2000 sensor readout channels. It is *performant*, processing hundreds of thousands of event triggers per second. It is *lightweight*, enabling deployment on embedded FPGAs flying aboard the instrument. It is *flexible*, allowing easy modification to explore the trade off space between speed and resource utilization. And it is highly *extensible*, enabling straightforward swapping of algorithmic modules as simulations and lab tests reveal new processing and analysis requirements.

2. Background on High Level Synthesis

High-Level Synthesis (HLS) tools compile algorithms expressed in an imperative language (e.g., C/C++) into a digital logic design that can be deployed as FPGA firmware. HLS compilers have recently become commercially available, potentially eliminating (or at least reducing) the need to use a low-level hardware description language (HDL) such as Verilog or VHDL.

When authoring an application using HLS, the language is augmented with vendor-specific pragmas and template libraries, examples of which are shown in Section 4. These are crucial in assisting the compiler tools to achieve the desired performance in the synthesized design. For example, Sohrabizadeh et al. [9] describe a convolutional neural network (CNN) implemented in 24 lines of HLS code that runs 80× slower than a single-threaded CPU-based version. The insertion of an additional 28 pragmas permits a 7000× speedup. In [10], we demonstrated that an appropriate pragma-defined architecture permits a 4200× increase in the event rate that may be accommodated by the ADAPT high-energy telescope's readout electronics, compared to a naïve implementation. These examples illustrate the wide performance variability present in HLS implementations. Developers therefore face the challenge of annotating the code to achieve the desired performance, which may come at the expense of additional resource utilization on the FPGA. Depending on how many configurable logic blocks the FPGA has (its *area*), the efficiency of computation that can be achieved may be limited by the functional requirements of the application.

For example, a high-energy telescope data analysis pipeline has functional requirements defined by the number of sensor readout channels, as well as the processing steps for individual channels (e.g., pedestal subtraction and charge integration) and across channels (e.g., calculations of interaction intensity and location). With sufficient FPGA area, each channel may be processed in parallel. Moreover, processing steps can be pipelined, allowing multiple data items to be processed simultaneously (e.g., the data read out from a triggered event may have pedestal subtraction applied while data from the previous trigger is still being charge-integrated), increasing throughput. However, if the additional digital logic to achieve these speed optimizations cannot fit on the target FPGA, some channels might be processed in sequence, or pipelining might be disabled. In [10], we explore these tradeoffs for some of the preprocessing algorithms that will be deployed on the ADAPT telescope (detailed in Section 3). In this paper and the associated library, we present our most speed-optimized implementations of these (and more) algorithms; future releases of the library will include more options to take advantage of the speed/area tradeoff space.

3. Target Instrument: ADAPT

The Advanced Particle-astrophysics Telescope (APT) [5] is a concept space-based observatory that will provide complementary gamma-ray and cosmic-ray measurements. Its science goals include fast GRB localization — using the methods presented in [11, 12] at this year's ICRC — and prompt alert delivery to enable multi-messenger follow-up observations of astrophysical transients.

To meet the computational demands of high-rate sensor data processing pursuant to accurate GRB localization in real time, APT will fly with onboard FPGA hardware to accelerate front-end data reduction and preprocessing. The Antarctic Demonstrator for APT (ADAPT) is a prototype high-altitude balloon mission anticipated to fly during the 2026–27 season. ADAPT will demonstrate, at a smaller-scale, APT's instrument design and computational capabilities, including the requisite FPGA-based processing. Several elements of its FPGA pipeline are discussed in [10, 13].

ADAPT has an active detector area of 0.45 × 0.45 m^2 and includes 4 imaging CsI:Na calorimeter (ICC) modules (Figure 1) interleaved with 4 layers of scintillating-fiber trackers. Optical light produced by the 3×3 arrangement of CsI:Na scintillating crystal tiles in each layer are captured by perpendicular arrays of 2 mm wavelength-shifting (WLS) optical fibers running across the top and bottom surfaces of each tile for spatial resolution of Compton interactions. To improve light detection and energy estimates, a 36-SiPM multiplexed detector module is placed over each of the 6 tile edges on the adjacent ICC sides opposite the WLS fiber SiPMs.

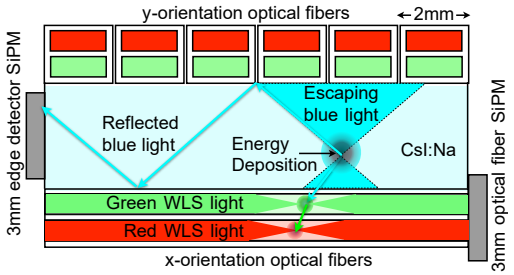

Fig. 1: Cross-section of ADAPT ICC module [5].

Each layer-axis has 225 optical fibers multiplexed into 80 readout channels captured by waveform digitizer ASICs [13]. ADAPT will demonstrate the capabilities of two ASICs: the ALPHA developed by collaborators at the University of Hawai'i [14], and the commercial HDSoC chip from Nalu Scientific [15]. The ALPHA has 16 input channels, each reading 10 ns samples into a ring buffer of 256 analog memory cells. The HDSoC has 32 channels, reading 1–4 ns samples into a 1984-sample buffer. When sufficient energy is detected in the instrument based on voltages from the edge detectors, the ASICs are simultaneously triggered, digitizing and reading out to FPGAs. Per-channel readout and processing of each axial array of WLS fibers (two per ICC layer) is handled by five ALPHAs or two HDSoCs and one FPGA. An additional FPGA per ICC layer handles aggregate analysis — e.g., island detection and centroiding across the arrays to infer interaction coordinates and energies — and network communication to the CPU.

4. Algorithms

This section describes the FPGA-based algorithms that we have developed and their inclusion in ADAPT's processing pipeline. These are implemented using the AMD Xilinx Vitis HLS suite; pragmas and libraries mentioned in this and the next section are specific to those tools.

4.1 Pipeline

A high-level overview of the architecture of ADAPT's FPGA-based data preprocessing and reduction pipeline is shown in Figure 2. Three HDSoC or five ALPHA ASICs per ICC layer-axis send data in serial to an FPGA; a communication handling and de-serialization module (not shown) written in a low-level HDL produces data words which are processed by a packet-handling module based on a finite-state machine (FSM). Although the packet formats are unique to each ASIC, the packet handler abstracts away the difference, enabling a common downstream format.

Because the ALPHA has 16 channels, while the HDSoC has 32, we architect a set of parallel pipelines downstream, each handling 16 of the readout channels. This means the packet handler for the HDSoC splits the data into two paths, while that for the ALPHA outputs only one path. These pipelines perform the pedestal subtraction, photon counting, and zero-suppression steps detailed below. These pipelines are then merged into a single stream for island detection and centroiding.

Dataflow Programming enables deep task-level pipelining, allowing functional modules to execute in parallel. Data are arranged by the packet handlers to encode a SAMPLES×CHANNELS matrix of values. Furthermore, some of the downstream per-channel processing algorithms execute over each time sample sequentially; this means that one module can process time samples as soon as they have

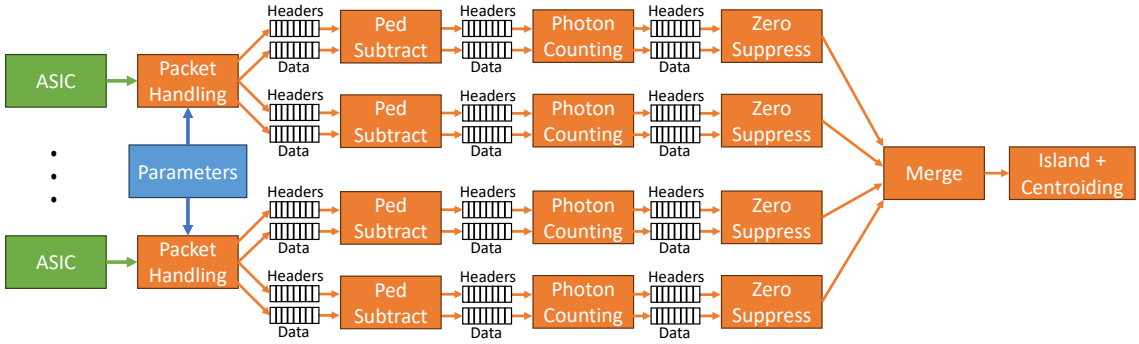

Fig. 2: ADAPT's prototype FPGA-based data processing pipeline.

been processed by the previous one, rather than delaying execution until the entire packet has been processed. By using the Vitis HLS `dataflow` pragma, we realize significant performance gains.

Each module is connected by two FIFO queues, implemented using the Vitis `hls::stream` container template, which can be instantiated for arbitrary data types. The synthesis tools automatically implement all necessary control logic for the underlying FIFO. One queue passes data between modules, while the other passes header information. The HLS tools allow the header to be defined as a C-style `struct` with corresponding attributes accessible by name in the source code.

We make use of vector data types, which Vitis HLS provides as a generic `hls::vector` type. By vectorizing across multiple input channels on an ASIC, an entire time-sample can be retrieved from or written to a data FIFO concurrently. Furthermore, charge integration can naturally be extended to sum over vectorized representations, rather than each channel individually.

4.2 Data Acquisition

Packet Handling We developed separate, swappable modules with FSMs to handle packets from either the ALPHA or HDSoC ASIC. The FSM is implemented using a variable to track the current region in the data packet, and a C-style `switch` statement that processes based on the state.

Pedestal Subtraction The capacitive charge pedestal from the ASIC's analog memory cells must be subtracted from its digitized readouts to obtain the true sampled signal values. Pedestal values are unique to each analog memory cell; these are stored in Block RAM (BRAM) on the FPGA. Both the ALPHA and HDSoC digitizers begin readout from the physical buffer relative to the current write position when a trigger is received. Thus, for each logical sample, the appropriate physical offset into the pedestal array must be computed. The header passed from the FSM packet handler includes metadata indicating the physical memory cell associated with the start of the readout. The algorithm to perform offset indexing and subtraction is similar to the one we presented in [10], and achieves excellent performance without the addition of pragmas.

4.3 Photon Counting

To infer the positions and energies of each gamma-ray interaction in the ICC, we must estimate the number of photons captured by each fiber. We provide three approaches for doing so.

Signal Integration is a common method: the total area under a defined portion of the captured waveform is calculated, then scaled by the average area under a single-photoelectron (PE) impulse response to estimate photon count. Our algorithm in [10] loops over the pedestal-subtracted sample vectors, and if the loop index is within the defined bounds, it adds the vector to the current tracked integral value. We enable multiple integrals, capturing different parts of the waveform in parallel.

This is achieved by writing a standard C++ `for` loop over the integrals, then decorating the loop with `#pragma HLS unroll factor=NUM_INTEGRALS`. By "unrolling" the loop, the synthesis tools create four copies of its underlying digital logic circuit. Additional techniques, like copying data into temporary variables and performing conditional updates using the C++ ternary operator, allows the synthesis tools to pipeline execution at the operation level, allowing a new sample to be processed every cycle. An alternative method based on a prefix-sum computation, detailed in [10], is also available.

Threshold-Based Analysis The energy deposited by a Compton scatter is low enough that typically <20 optical photons are captured by each WLS fiber SiPM. Figure 3 shows a simulated digitized waveform, with peaks corresponding to PE impulses. Due to input noise and the variability in peak height, photon counts inferred from signal integration are inaccurate given the low SNR. We therefore implement an alternative method that identifies photons by counting peaks in the waveform. It uses the three thresholds illustrated in Figure 3. Any sample above the "single-photon threshold" but below the "dual-photon threshold" is assumed to be the peak of a single-PE impulse response. The higher "dual-photon threshold" is reached when two photons arrive at roughly the same time. And when two adjacent samples are above the "split" threshold, but below the others, it's assumed that the peak of a single-PE impulse response straddles the two.

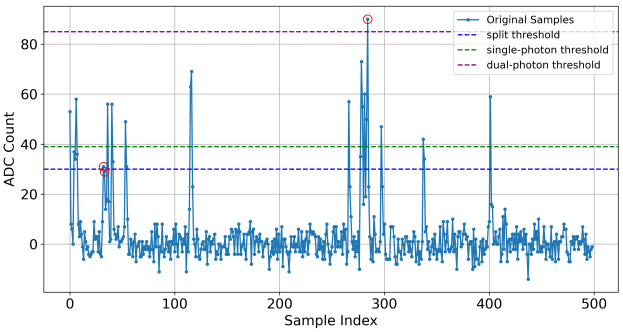

Fig. 3: Illustration of threshold-based photon counting.

Hybrid Approach Although the threshold-based approach to photon counting is robust to noise in signal-free regions of the waveform, it does not scale well to interactions at higher energy regimes, e.g., from pair-production or cosmic-ray showers where more than two optical photons are likely to pile up. Moreover, it requires careful tuning of the three threshold values. We have therefore devised a third approach which combines aspects of both. It uses a single threshold to identify a likely PE, then integrates within a configurable window around any sample exceeding the threshold.

The implemented algorithm uses a two-pass approach to avoid double counting. It maintains a bitmask implemented as a matrix of size SAMPLES×CHANNELS, where each element has the same number of bits (16) as a sample value. It scans all samples in sample/channel order. For those identified to exceed the threshold, it sets all bits in the corresponding entry of the bitmask matrix — and entries for some configurable number of samples ahead and behind it — to 1. In a second pass, it integrates over the bitwise `AND` of the sampled values and the bitmask. This masking technique permits fully vectorized addition in the integration step.

4.4 Island Detection and Centroiding

Sufficiently low photon are assumed to be caused by various noise sources; even for the threshold-based photon counting, thermally-generated electrons in the SiPMs may produce PE "dark counts." A **zero suppression** module, described in [10], after photon counting therefore sets values below a configurable threshold to zero. The final 16-channel photon count vectors are then concatenated across each ICC layer-axis in a **merge** module, implemented as a simple loop.

Table 1: FPGA speed and area statistics for synthesized algorithms.

Module	II	Rate	BRAM	%	DSP	%	FF	%	LUT	%
Pedestal Subtraction	261	383k	14	1.5	0	0	558	0.1	785	0.4
Photon Counting: Integrals	271	369k	0	0	0	0	3,656	0.9	8,282	4.1
Photon Counting: Thresholds	262	381k	0	0	0	0	1,399	0.3	4,302	2.1
Photon Counting: Hybrid	1,545	64k	44	4.9	0	0	3,671	0.9	4,813	2.4
Area to PE Conversion	21	4,761k	0	0	0	0	1,249	0.3	8,192	4.0
Zero Suppression	23	4,347k	0	0	0	0	2,115	0.5	9,263	4.5
Merge	9	11,111k	0	0	0	0	520	0.1	116	0
Island Detection & Centroiding	115	869k	0	0	30	3.6	48,424	11.9	37,043	18.2

To infer interaction coordinates, an **island detection and centroiding** module makes a single scan over this array, using registers to track whether it is currently in an "island" (a sequence of adjacent non-zero channels), and whether it has just entered or left an island. While in an island, registers maintain the island's beginning channel index, width, and total intensity, while a multiply-accumulate circuit tracks the intensity-weighted position. When leaving an island, a single divide computes the weighted mean to get the interaction position, and the centroid data is pushed into a FIFO to be transmitted via Ethernet to the processor.

5. Performance Evaluation

We implement the algorithms described in the previous section, using the dataflow-based architecture in Figure 2. Each 16-channel parallel pipeline is synthesized into a single RTL block, allowing easy duplication depending on the number of input channels. ADAPT uses 5 pipelines for each ICC layer-axis, and 10 for each axial array of its scintillating-fiber trackers mentioned in Section 3. The merge and island detection/centroiding modules are synthesized into their own RTL block, with a preprocessor macro (`#define NUM_PIPELINES`) configuring automatic synthesis to handle the appropriate number of input streams and channels in the resulting merged array.

We synthesize these designs using Vitis HLS version 2024.1, targeting the AMD Xilinx Kintex 7-series FPGA (ADAPT will fly with a Kintex 7 XC7K325T). Reported speed and area numbers are listed in Table 1 for `NUM_PIPELINES` = 2. Column "II" is the Initiation Interval, representing the number of cycles of execution that must elapse between when the module begins processing a triggered readout event, to when a subsequent data packet may enter its pipeline. Using a conservative 100 MHz clock speed, we can infer the supported throughput (event rate). Unless using the hybrid photon counting module, the reported values indicate that our pipeline supports a trigger rate of over 3×10^5. And even with the hybrid approach's two-pass scan, it can support 6.4×10^4 events per second. Note that the packet handling FSM is *not* reported here, as it presents a fundamental bottleneck: ADAPT's digitizer ASICs read out over a ~300MHz serial line. For the HDSoC, a single triggered event yields ~200kb of data, permitting a trigger rate of only ~1500/sec.

The number of BRAM blocks, DSP slices, flip-flops (FFs), and look-up tables (LUTs) used, and their percent utilization on ADAPT's FPGA, are also reported; even with `NUM_PIPELINES` = 5, these remain within the area provided by the FPGA. Not reported is the additional resource utilization incurred by the dataflow pipeline architecture. Most significantly, FIFOs between modules occupy an additional 122 BRAM blocks, 13.7% of those available.

6. Conclusions and Future Work

This paper presents a library of FPGA-based data preprocessing algorithms for high-energy telescopes. These are implemented using high-level synthesis (HLS) to ease adoption and prototyping. When implemented for the ADAPT gamma-ray/cosmic-ray suborbital telescope, these permit high (up to $>3\times10^5$) event rates. As future work for the library, we will consider additional optimizations in both speed (especially for the hybrid photon counting module) and area (especially for island detection and centroiding). For the future proposed APT instrument, we will address the bottleneck at the front end imposed by serial readouts from the digitizer ASICs.

References

[1] J.H. Buckley, P. Dowkontt, K. Kosack and P. Rebillot, *The VERITAS flash ADC electronics system*, in *Proc. of 28th Int'l Cosmic Ray Conference*, pp. 2827–2830, 2003.

[2] F.A. Harrison, S. Boggs, F. Christensen, W. Craig, C. Hailey, D. Stern et al., *The Nuclear Spectroscopic Telescope Array (NuSTAR)*, in *Space Telescopes and Instrumentation 2010: Ultraviolet to Gamma Ray*, vol. 7732, pp. 189–196, SPIE, 2010, DOI.

[3] T. Weekes, H. Badran, S. Biller, I. Bond, S. Bradbury, J. Buckley et al., *VERITAS: the very energetic radiation imaging telescope array system*, *Astroparticle Physics* **17** (2002) 221.

[4] CTA Consortium, M. Actis, G. Agnetta, F. Aharonian, A. Akhperjanian, J. Aleksić et al., *Design concepts for the Cherenkov Telescope Array CTA: an advanced facility for ground-based high-energy gamma-ray astronomy*, *Experimental Astronomy* **32** (2011) 193.

[5] J. Buckley et al., *The Advanced Particle-astrophysics Telescope (APT) Project Status*, in *Proc. of 37th Int'l Cosmic Ray Conference*, vol. 395, pp. 655:1–655:9, July, 2021, DOI.

[6] W. Chen, J. Buckley et al., *Simulation of the instrument performance of the Antarctic Demonstrator for the Advanced Particle-astrophysics Telescope in the presence of the MeV background*, in *Proc. of 38th Int'l Cosmic Ray Conference*, vol. 444, pp. 841:1–841:9, July, 2023, DOI.

[7] K. Bechtol, S. Funk, A. Okumura, L. Ruckman, A. Simons, H. Tajima et al., *TARGET: A multi-channel digitizer chip for very-high-energy gamma-ray telescopes*, *Astroparticle Physics* **36** (2012) 156.

[8] E. Delagnes et al., *NECTAr0, a new high speed digitizer ASIC for the Cherenkov Telescope Array*, in *IEEE Nuclear Science Symposium Conference Record*, pp. 1457–1462, IEEE, 2011, DOI.

[9] A. Sohrabizadeh, C.H. Yu, M. Gao and J. Cong, *AutoDSE: Enabling software programmers to design efficient FPGA accelerators*, *ACM Transactions on Design Automation of Electronic Systems* **27** (2022) 32:1.

[10] M. Sudvarg, C. Zhao, Y. Htet, M. Konst, T. Lang, N. Song et al., *HLS taking flight: Toward using high-level synthesis techniques in a space-borne instrument*, in *Proc. of 21st Int'l Conference on Computing Frontiers*, pp. 115–125, ACM, May, 2024, DOI.

[11] Y. Htet et al., *Performance Modeling and Improvements on the GRB Source Localization Streaming Pipeline Aboard the Antarctic Demonstrator for the Advanced Particle-Astrophysics Telescope (ADAPT)*, in *Proc. 39th Int'l Cosmic Ray Conf.*, vol. 501, pp. 679:1–679:9, July, 2025.

[12] J. Buhler and M. Sudvarg, *Real-time Likelihood Map Generation to Localize Short-duration Gamma-ray Transients*, in *Proc. 39th Int'l Cosmic Ray Conf.*, vol. 501, pp. 587:1–587:9, July, 2025.

[13] M. Sudvarg et al., *Front-End Computational Modeling and Design for the Antarctic Demonstrator for the Advanced Particle-astrophysics Telescope*, in *Proc. of 38th Int'l Cosmic Ray Conference*, vol. 444, pp. 764:1–764:9, July, 2023, DOI.

[14] M. Kuwahara and G.S. Varner, *Design and Development of Advanced Low Power Hybrid Acquisition (ALPHA) ASIC for Antarctic Demonstrator for the Advanced Particle-astrophysics Telescope (ADAPT)*, in *Proc. of Nucl. Science Symp., Medical Imaging Conf. and Int'l Symp. on Room-Temp. Semicond. Detectors*, 2023.

[15] L. Macchiarulo, I. Mostafanezhad, G. Varner, K. Lauritzen, K. Hoe, G. Uehara et al., *Measurement results for the high density digitizer system on chip (HDSoC): A waveform digitizer for high density detectors*, in *Proc. of Nuclear Science Symposium and Medical Imaging Conference*, IEEE, 2022.