URAM-based asynchronous FIFO design for improved throughput and FPGA RAM usage

Martim Rosado*†‡, Pedro Tomás‡, Nuno Roma‡, and André David*†
*On behalf of the CMS Collaboration
† Experimental Physics Department, CERN, Switzerland
‡INESC-ID, Instituto Superior Técnico, Universidade de Lisboa, Portugal

***Abstract*—First-In First-Out (FIFO) memories are essential for data transfers in Field Programmable Gate Array (FPGA) designs, providing buffering and communication facilities between different components and clock domains. Traditional implementations rely on Block-RAM (BRAM) cells, but transferring large data packets can lead to excessive BRAM usage, potentially limiting resources for other parts of the system. This paper proposes an alternative Ultra-RAM (URAM)-based asynchronous FIFO design, allowing to optimise the overall FPGA RAM usage for applications that require deep and wide FIFOs. Several implementation strategies are explored to ensure maximum data throughput and support different packet sizes. Experimental results obtained on an AMD Zynq UltraScale+ FPGA running at 500 MHz demonstrate the proposed FIFO's ability to successfully transfer complete packets (as large as 1024 64-bit words) between two distinct clock domains without introducing any stalls.**

***Index Terms*—Asynchronous FIFO, Clock domain crossing, Ultra-RAM, Field-Programmable Gate Array**

I. Introduction

FIFOs memories play a crucial role in digital circuit design by allowing the buffering of in-transit data between components [1]. In particular, a key feature of asynchronous FIFOs is their ability to transfer data across different clock domains, making them widely used in numerous applications [2]–[7].

To ease the implementation of asynchronous FIFOs in FPGAs, manufacturers already provide some tools (e.g. FIFO generator IP [8] and Xilinx Parameterized Macros (XPMs) [9]) that allow designers to configure different parameters such as the FIFO depth, width, and memory type. However, such tools only target distributed RAM and BRAM cells when contemplating asynchronous FIFOs, not supporting the exploitation of URAM cells for this end. Such a limitation is particularly relevant when considering the asynchronous nature of these FIFOs, as URAM cells are single-clock-only memory blocks. Hence, despite its availability in recent AMD FPGA technologies, the use of URAMs cells is often limited to memory intensive applications that require on-chip storage for significant amounts of data [10], [11].

On the other hand, although BRAM-based FIFOs are commonly used for large data buffers, they often pose scalability limitations, as their flexibility leads to high demand for system resources that are frequently required for other purposes within the target system architecture. URAMs, on the other hand, provide a more storage-efficient solution but lack dual-clock support, making their direct use in asynchronous FIFOs particularly challenging. While some designs already make use of URAMs for different applications [12]–[15], including the design of Ethernet packet transmission [16], their employment in the design of asynchronous FIFOs is still largely unexplored and no tool exists to facilitate their usage.

Accordingly, this work proposes a tool to automate the instantiation of deep and wide asynchronous FIFOs implemented using URAM cells in AMD FPGAs, thus addressing the absence of native support of AMD's FIFO generator and XPMs. Hence, the key contributions of this work are:

- A new method for implementing asynchronous FIFOs using URAM cells (despite its single-clock limitation).
- Design considerations to optimise the operating frequency of both the write and read clock domains.
- Study of multiple FIFO topologies aimed at maximising the throughput and attaining an efficient utilisation of URAM resources.
- A hardware testing system demonstrating the performance and feasibility of the proposed approach.

Although the methodology presented herein is focused on AMD FPGAs, a similar approach could equally be adapted for Intel FPGAs featuring Embedded SRAM (eSRAM), which share characteristics with AMD's URAM.

II. Proposed URAM-based FIFO Topologies

Since URAMs are single-clock-only devices, their use in asynchronous FIFOs cannot be accomplished by the addition of logic alone. In particular, internal FIFO functionalities, including the actual storage and clock domain crossing, have to be split so the URAM can be efficiently used for most of the storage. To accomplish these requisites, the proposed topologies make use of a smaller buffer, based on either distributed RAM or BRAM cells, also configured as asynchronous FIFO, that is responsible for handling the data crossing between clock domains. Furthermore, as URAMs are more limited than other FPGA cells in what concerns its maximum operating frequency [17], preference should be given to clocking the URAMs in the slower clock domain whenever possible.

Figure 1 presents three alternative FIFO topologies aiming to maximise the asynchronous FIFO's operating frequencies in several situations considering the clock domain (fast/slow) present at which FIFO port (read/write) and the maximum packet size (in FIFO word entries) being processed by the target application. The proposed topologies are classified as:

1) Slow Read - Topology optimised for applications transferring data from a fast clock domain to a slow clock domain. Hence, the data is first fed to the asynchronous buffer and then stored in the URAM cells.
2) Fast Read - Topology optimised for the opposite case, when data is being transferred from a slow clock domain into a fast clock domain. The data is stored in an URAM cell and then transferred through the asynchronous buffer, based on distributed RAM or BRAM cells.
3) Large Packet - Topology required for handling packets of large dimensions, such that the implementation of the asynchronous buffer is no longer efficient using distributed or block RAM. Hence, a URAM cell is also used to store data after the crossing of clock domains.

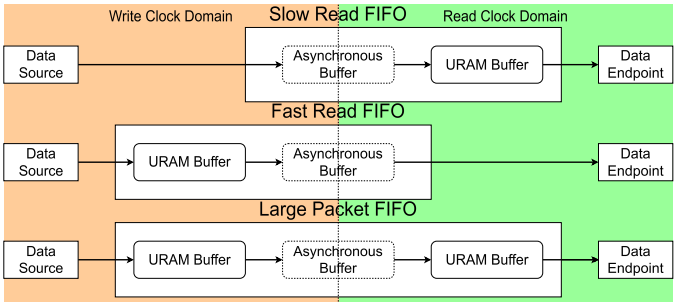

Fig. 1: Proposed FIFO topologies.

Furthermore, to ensure that a complete packet can be accommodated in the intermediary asynchronous buffer, the maximum packet size passing through the FIFO must be taken into consideration when defining the memory type that implements such an asynchronous buffer (either distributed RAM or BRAM cells). This ensures that a complete packet can be read without stalls, allowing the best possible throughput per packet to be achieved. In fact, should only a fraction of a packet be stored in the asynchronous buffer, the frequency difference of the read and write clock domains would lead to a limit situation in which the asynchronous buffer is empty (when the fast clock domain is the read clock) or full (when the fast clock domain is the write clock). Hence, stalls would be introduced in the processing, as not every clock cycle in the fast domain will correspond to a successful word transfer.

The particular cases where the FIFO's read port interfaces a pipeline or delay chain separating it from the data endpoint further benefit from guaranteeing that a complete packet can be stored in the asynchronous buffer when beginning the reading process from the FIFO. Otherwise, for every word to be read, the empty flag of the FIFO needs to be checked to avoid the reading of invalid data. This would be aggravated by the fact that this flag-check would take several clock cycles due to the pipeline latency, penalising the system's throughput. With a complete packet available in the same clock domain, its reading can be done in full, maximising the system's throughput. The same reasoning can be applied to the writing process when in a slow read configuration.

Under another perspective, while small packet sizes motivate the implementation of the asynchronous buffer with distributed RAM cells (as it would be inefficient to use a BRAM with a low occupancy), as the packet size increases, so does the resource usage to implement a large distributed RAM asynchronous buffer, justifying the use of BRAMs to minimise the resource usage. However, if the packet size increases such that it cannot be accommodated efficiently in BRAM cells, an URAM would become the natural choice for the asynchronous buffer implementation. This case is contemplated in the last FIFO proposed in fig. 1 (large packet FIFO), which requires three buffers. In this case, the previous considerations regarding the size of the asynchronous buffer do not apply as it needs not to store a complete packet, and it is implemented with distributed RAM to attain the solution with the least resources and most flexibility in the FPGA's routing due to the abundant Look-Up Table (LUT) distribution.

III. Developed Automated Tool

The developed tool[1] automates the instantiation of deep and wide asynchronous FIFOs using URAM cells in AMD FPGAs, covering the existing gap in native support of AMD's FIFO generator and XPMs. The tool receives the following pre-synthesis parameters as input:

- FIFO's topology (slow-read, fast-read, or large-packet);
- Maximum packet size to be supported;
- Memory type of the intermediate asynchronous buffer;
- Dimension of the intermediate asynchronous buffer.

The produced VHDL template can then be used in a similar way as an XPM, i.e. by instantiation in the HDL project file.

For the sake of illustration, Figure 2 details the implementation of a large-packet topology, highlighting the use of the available synchronous and asynchronous FIFO XPMs.

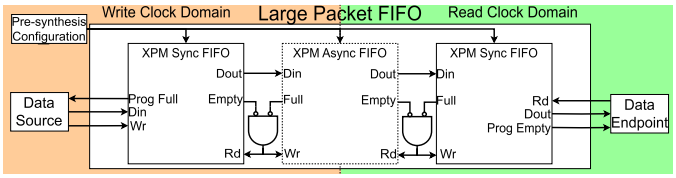

Fig. 2: Implementation details of the large-packet FIFO topology.

In the produced configuration, the programmable empty- and full-flags are tuned and exposed to the user according to the considered packet size. This has the advantage of allowing the simplification of the read/write external logic, as these signals convey the information that the FIFO has a full packet ready to be read or enough space to accommodate a new one.

IV. Performance Evaluation

The proposed FIFO topologies were extensively evaluated regarding the clock frequency, maximum throughput per packet (i.e., if a complete packet can be read without stalls), resource and power consumption. For such purpose,

[1]Tool available at: https://gitlab.cern.ch/prosado/uram-async-fifo

the system was tested on an AMD Zynq UltraScale+ ZCU106 development board, featuring a xczu7ev-ffvc1156-2-e device. To cover all discussed FIFO topologies, six FIFOs are required to test all topologies and packet sizes. In particular, three FIFOs were tested assuming a fast read configuration with: (i) a distributed RAM asynchronous buffer; (ii) a BRAM asynchronous buffer; and (iii) a large packet FIFO. The same is applied to three FIFOs in a slow read configuration.

This test system, described in fig. 3, connects the FIFOs under test (FIFO UT) to a write controller (Wr controller) and a Scoreboard. The write controller reads pre-defined packets from external BRAM memories and sends the data packets to the FIFOs. The scoreboard has access to a similar external BRAM memory in a different clock domain, loaded with the same pre-defined packets, and compares the data read from the FIFOs to the packets stored in the BRAM.

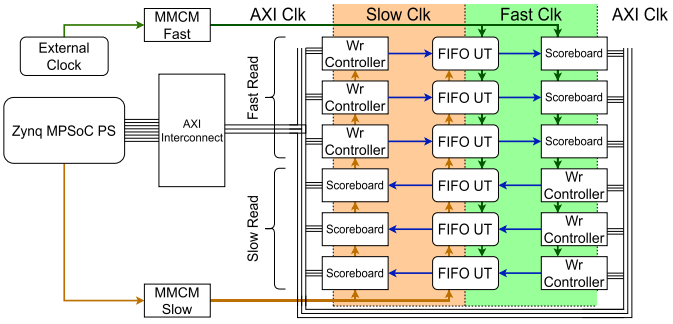

Fig. 3: Testing system architecture.

The write controller and scoreboard use an AXI protocol [18] connection to the software, which is responsible for loading the test BRAMs with the defined packets to transmit via the FIFOs under test, trigger the testing procedure, and then read the results from the available status registers.

Special care was taken to ensure the generation of unrelated clock signals, allowing to stimulate the FIFOs in a truly asynchronous environment. Hence, two distinct Mixed-Mode Clock Managers (MMCMs) were driven with two clock signals originating from different sources.

A similar test system was implemented using the traditional asynchronous FIFO solutions based on BRAMs (implemented through the use of AMD FIFO generator IP) to compare clock frequencies and resource consumption. However, only two FIFOs were implemented. This is to guarantee that the resource-wise comparisons made for the proposed FIFO topologies are being done with BRAM-only asynchronous FIFO implementations with an equivalent capacity. The proposed FIFO topologies (apart from the large packet FIFO) can be compared with the same BRAM solution, as long as the storage capacity of all solutions is approximately the same and equivalent to one URAM. A larger BRAM FIFO configured to have an equivalent storage capacity of two URAMs can be used to compare with the proposed large packet topology.

A maximum frequency of 500 MHz for both clock domains was successfully achieved by all the proposed FIFO topologies. It should be noted that the URAM connected to the slow clock domain port of the FIFO limits the large packet FIFO topology from closing timing with higher clock frequencies. The other topologies successfully closed the timing with a 600 MHz fast clock and a 500 MHz slow clock.

Taking the above into consideration, table I presents the clock frequencies of the Fast and Slow clock pairs used in the hardware tests, where the values were chosen such that several assumptions and metrics were tested and evaluated, namely: maximum clock frequency, the frequencies employed in the case study described in section IV-E, and that a pair of clock frequencies with a significant difference and its reverse are evaluated. To maximise the frequency distance, the minimum frequency able to be generated by the MMCMs [17] was paired with the maximum frequency attained by the FIFOs. The testing of the reversed frequency pair, i.e., testing the FIFOs with a slower clock frequency on the fast clock port, has the goal of validating the FIFOs' functionality in a non-optimal situation.

TABLE I: Validated frequency pairs (MHz) in the hardware test system.

Fast Clock	500	380	500	6.3
Slow Clock	500	320	6.3	500

A. Throughput Per Packet

Appropriate maximum packet size parameters were chosen for each topology to evaluate the attained maximum throughput per packet of the proposed infrastructure. They correspond to 1024 64-bit words for the large packet topology and half the capacity of the asynchronous buffer for the other cases, aiming to select packet sizes adapted to the respective chosen RAM technology. Likewise, a packet of 8 64-bit words is considered for the FIFOs with a distributed RAM asynchronous buffer, and 128 64-bit words for a BRAM-based asynchronous buffer.

To test the throughput per packet in each FIFO, runs of 10'000 fixed size packets were transmitted through. A packet is successfully extracted from the FIFO when fully available in the read clock domain or written when enough space was freed on the write clock domain. The FIFO's programmable empty and full flags signalled these conditions to the scoreboard or wr controller. The Scoreboard not only checked for errors in the transmitted data but also verified the maximum number of consecutive words that were successfully read per processed packet. All FIFOs transmitted all packets without errors. Furthermore, the packets were read out consecutively and in full, i.e., without stalls between packet words, thus maximising the achievable throughput per packet of each FIFO.

B. Resource Usage

Table II presents a resource comparison between a BRAM-only solution and the proposed URAM FIFOs topologies using one URAM. These results correspond to an implementation of the test system on the ZUC106 development board's ZU7EV System on Chip (SoC) device with both fast and slow distinct clock domains defined as 500 MHz.

Despite an increased overhead in LUTs and Flip-Flops (FFs) in the proposed URAMs solutions, the relative occupancy of

these cells in the target device is ≈0.1% for almost all cases. This increase is due to the need for implementing the necessary write and read control circuits for more than one buffer by the XPMs. In contrast, the BRAM-only solution only needs to implement such control logic for a single buffer. Furthermore, LUTs are also used to implement the distributed RAM present in the solutions that implement the asynchronous buffer with this memory type. The RAM usage presents different results as 7.5 BRAMs are replaced by one URAM for the FIFO's storage. Moreover, the relative occupancy of the RAM cells is halved, with 1% of URAM usage corresponding to 2.4% of BRAM usage in the target device.

TABLE II: Resource usage for fast read (FR) and slow read (SR) FIFOs.

Design	LUT	FF	BRAM	%	URAM	%
BRAM equivalent	78	266	7.5	2.4	-	-
URAM dist. RAM FR	185	349	-	-	1	1.0
URAM dist. RAM SR	184	350	-	-	1	1.0
URAM BRAM FR	193	277	1	0.3	1	1.0
URAM BRAM SR	194	278	1	0.3	1	1.0

The results presented in table III, relative to the large packet FIFOs topology and the respective BRAM-equivalent counterpart, yield similar conclusions with an increase in LUT and FF consumption, although it is still in the order of 0.1% in the context of the used SoC device. Regarding RAM utilisation, 2 URAMs can replace 14.5 BRAMs, also corresponding to half of the relative device occupancy.

TABLE III: Resource usage for large packet FIFOs.

Design	LUT	FF	BRAM	%	URAM	%
BRAM equivalent	122	282	14.5	4.6	-	-
URAM fast read	267	485	-	-	2	2.1
URAM slow read	266	485	-	-	2	2.1

The resource usage results align with this contribution's proposed objective, with BRAM usage being reduced in exchange with URAM cells, allowing the former to be reallocated for applications requiring smaller and more flexible memory cells.

C. Power Estimation

The power estimation presented in table IV has been extracted from Vivado 2023.2 for a target frequency of 500 MHz for both clocks. A reduction in the power dissipation can be observed in all FIFO topologies.

TABLE IV: Vivado power estimation.

Design	Power (mW)	URAM/BRAM (%)
BRAM equivalent	61	-
BRAM equivalent Large	72	-
URAM dist. RAM FR	32	52
URAM dist. RAM SR	31	51
URAM BRAM FR	30	49
URAM BRAM SR	29	48
URAM Large FR	52	72
URAM Large SR	53	74

D. Limitations and Future Work

The tests performed on the proposed FIFO topologies used fixed-size packets. This allowed the passing of a "packet ready" signal across clock domains using the programmable full and empty flags, thus simplifying the test system hardware. A configuration restriction is applied to such flags, according to the relevant XPM documentation [9], limiting the packet length to a minimum of 7 words. Similarly, variable-size packets can also be used with the proposed FIFOs but require additional hardware to convey the "packet ready", as the FIFO programmable flags can only convey information regarding a single fixed packet size.

An extension of this work could be to transmit packets with variable size through the asynchronous FIFOs, also circumventing the programmable flag limitations. It should be noted that this additional hardware is required for both URAM-based and traditional BRAM-only solutions.

E. Case Study

The proposed large packet topology FIFO was implemented in the FPGA design for the back-end Data Acquisition and control (DAQ) system of the High-Granularity Calorimeter (HGCAL) [19], under development for the Phase-2 upgrade of the Compact Muon Solenoid (CMS) [20], [21] experiment at European Organisation for Nuclear Research (CERN)'s Large Hadron Collider (LHC) [22], [23]. This design implements 54 asynchronous FIFOs, requiring a depth of 8k 64-bit words written in a 320 MHz clock domain and read in an asynchronous clock domain of up to 380 MHz. The use of the proposed FIFO reduced in $\approx 30\%$ the use of BRAMs in the target FPGA (VU13P) at the cost of an URAM increase of only $\approx 9\%$, freeing up the BRAM resources for other uses.

V. Conclusion

URAM memories offer a viable trade-off between larger storage and flexibility loss when compared with BRAMs. The inability of URAMs to act as dual-clock cells limits their native use as asynchronous FIFOs. For FIFOs handling large packets of wide data words, this poses a limitation as the use of BRAMs leads to an inefficient FPGA resource use.

This work proposed an automated tool to facilitate the exploitation of URAMs cells for the implementation of asynchronous FIFOs, thus releasing the highly flexible BRAM cells for less memory-intensive uses without throughput sacrifice. The chosen memory type for the asynchronous buffer implementation takes into account the maximum handled packet size, which, combined with the use of programmable flags, optimises the FIFO's read/write procedures. This, in turn, removes any intra-packet stalls, improving the throughput.

The proposed FIFOs were tested in a ZCU106 AMD development board working with different clock pairs up to 500 MHz and different packet sizes of up to 1024 64-bit words.

Acknowledgements

Work supported by national funds through Fundação para a Ciência e a Tecnologia (FCT) under projects 2022.06780.PTDC (DOI: 10.54499/2022.06780.PTDC), UIDB/50021/2020 (DOI: 10.54499/UIDB/50021/2020), grant PT/BD/154723/2022 (DOI: 10.54499/PRT/BD/154723/2022), and as part of a doctoral studentship at CERN.

REFERENCES

[1] L. Lin, "Optimization of on-chip memory on fpga device," Master's thesis, TU Delft, August 2024, available at https://resolver.tudelft.nl/uuid:964a1b68-8b50-4668-af66-a0463c376e58.

[2] C. E. Cummings, "Simulation and synthesis techniques for asynchronous fifo design," in *SNUG 2002 (Synopsys Users Group Conference, San Jose, CA, 2002) User Papers*, vol. 281. Citeseer, 2002.

[3] R. W. Apperson, Z. Yu, M. J. Meeuwsen, T. Mohsenin, and B. M. Baas, "A scalable dual-clock fifo for data transfers between arbitrary and haltable clock domains," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 15, no. 10, pp. 1125–1134, 2007.

[4] H. Han and K. S. Stevens, "Clocked and asynchronous fifo characterization and comparison," in *2009 17th IFIP International Conference on Very Large Scale Integration (VLSI-SoC)*, 2009, pp. 101–108.

[5] H. Chandu, "Advancements in asynchronous fifo design: A review of recent innovations and trends," *INTERNATIONAL JOURNAL OF RESEARCH AND ANALYTICAL REVIEWS*, vol. 10, pp. 573–578, 06 2023.

[6] Q. Wang, "A study of advances in asynchronous fifo design," *Applied and Computational Engineering*, vol. 121, pp. 14–23, 01 2025.

[7] F. Huemer and A. Steininger, "Timing domain crossing using muller pipelines," in *2020 26th IEEE International Symposium on Asynchronous Circuits and Systems (ASYNC)*, 2020, pp. 44–53.

[8] AMD, "AMD FPGA FIFO Generator IP," FIFO generator IP user guide. [Online]. Available: https://docs.amd.com/v/u/en-US/pg057-fifo-generator

[9] ——, "Xilinx Parameterized Macros," XPM user guide from UltraScale Architecture libraries guide. [Online]. Available: https://docs.amd.com/r/en-US/ug974-vivado-ultrascale-libraries/Xilinx-Parameterized-Macros

[10] C. Jia, C. Li, Y. Li, X. Hu, and J. Li, "Facl: A flexible and high-performance acl engine on fpga-based smartnic," in *2022 IFIP Networking Conference (IFIP Networking)*, 2022, pp. 1–9.

[11] B. Gadat, L. Barthe, B. Matuz, and C. Poulliat, "Ldpc codes with low error floors and efficient encoders," in *ICC 2023 - IEEE International Conference on Communications*, 2023, pp. 6658–6663.

[12] A. Forencich, A. C. Snoeren, G. Porter, and G. Papen, "Corundum: An open-source 100-gbps nic," in *2020 IEEE 28th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*, 2020, pp. 38–46.

[13] R. Kundel, L. Nobach, J. Blendin, W. Maas, A. Zimber, H.-J. Kolbe, G. Schyguda, V. Gurevich, R. Hark, B. Koldehofe, and R. Steinmetz, "Openbng: Central office network functions on programmable data plane hardware," *International Journal of Network Management*, vol. 31, no. 1, p. e2134, 2021, e2134 nem.2134. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/nem.2134

[14] J. Meng, N. Gebara, H.-C. Ng, P. Costa, and W. Luk, "Investigating the feasibility of fpga-based network switches," in *2019 IEEE 30th International Conference on Application-specific Systems, Architectures and Processors (ASAP)*, vol. 2160-052X, 2019, pp. 218–226.

[15] E. Boutillon and C. Marchand, "Buffers optimization for multi-core decoders," in *2023 IEEE Wireless Communications and Networking Conference (WCNC)*, 2023, pp. 1–6.

[16] T. Floros, "High throughput data interfacing," Master's thesis, TU Delft, February 2021, available at https://resolver.tudelft.nl/uuid:6bd0f3e6-8aa4-42ac-a2e7-c8cf3dc925d4.

[17] AMD, *Virtex UltraScale+ FPGA Data Sheet: DC and AC Switching Characteristics (DS923)*. [Online]. Available: https://docs.amd.com/r/en-US/ds923-virtex-ultrascale-plus

[18] ARM, "AMBA® AXI™ and ACE™ Protocol Specification," AXI4 protocol specifications from ARM. [Online]. Available: https://developer.arm.com/documentation/ihi0022/e/AMBA-AXI3-and-AXI4-Protocol-Specification

[19] CMS Collaboration, "The Phase-2 Upgrade of the CMS Endcap Calorimeter," CERN, Geneva, Tech. Rep., Nov 2017. [Online]. Available: https://cds.cern.ch/record/2293646

[20] ——, "The CMS experiment at the CERN LHC," *Journal of Instrumentation 3*, (2008) S08004, DOI:10.1088/1748-0221/3/08/S08004.

[21] ——, "Development of the CMS detector for the CERN LHC Run 3," *Journal of Instrumentation 19*, (2024) P05064, doi:10.1088/1748-0221/19/05/P05064.

[22] S. Mallios, P. Dauncey, A. David, and P. Vichoudis, "Firmware architecture of the back end DAQ system for the CMS high granularity endcap calorimeter detector," *Journal of Instrumentation*, vol. 17, no. 04, p. C04007, apr 2022. [Online]. Available: https://doi.org/10.1088/1748-0221/17/04/c04007

[23] M. Rosado, A. Şarkışla, S. Mallios, B. Akgün, A. David, N. Roma, P. Tomás, P. Vichoudis, and on behalf of the CMS collaboration, "Back-end daq system prototype testing and integration on a full detector test system for the cms hgcal detector," *Journal of Instrumentation*, vol. 20, no. 03, p. C03028, mar 2025. [Online]. Available: https://dx.doi.org/10.1088/1748-0221/20/03/C03028