Dora: A Low-Latency Partial Reconfiguration Controller for Reconfigurable System

Guosheng Zhang, Lisong Shao, Hanqian Wu, Xiaotian Gu, and Xinyi Zhang

***Abstract*—Dora, a low-latency Field-programmable gate array (FPGA) partial reconfiguration (PR) controller, is proposed in this brief to address the latency challenge encountered by traditional solutions in highly real-time reconfigurable systems. First, based on the constructed refined cost model, key factors for minimizing latency in Host-to-FPGA reconfiguration process are analyzed. Subsequently, utilizing the established producer-consumer model, the reconfiguration mechanism (scatter gather-streaming-based hybrid transmission) and training method (adaptive overclocking of the Internal Configuration Access Port, ICAP) in Dora aim to enhance production of configuration bitstream while achieving efficient consumption. Ultimately, experiments demonstrate that Dora outperforms existing solutions in pivotal aspects: (i) It slashes FPGA resource utilization by over 60% while attaining a reconfiguration rate close to the theoretical peak of 99.6%, (ii) it reduces latency to just 11.1 ms, representing only 2.6% of the latency of the standard Xilinx solution, (iii) additionally, its open-source nature fosters broader adoption and utilization among the research community.**

***Index Terms*—FPGA, reconfigurable system, partial reconfiguration, ICAP, reconfiguration latency.**

I. Introduction

RAPID technological advances pose challenges to traditional fixed hardware, leading to the emergence of reconfigurable systems (RS) [1] capable of adapting during runtime or downtime. RS enhances efficiency in diverse domains, including software-defined satellites and software-defined radio (SDR) [2], [3], by leveraging dynamic hardware adaptation. FPGA serves as the fundamental hardware backbone for RS, facilitating swift optimization, bolstering flexibility and customization capabilities [4]. However, the reconfigurability of FPGA-based RS is constrained by timing and scale considerations, rendering it incapable of achieving dynamic application reconfiguration during runtime.

Received 15 July 2024; revised 29 August 2024; accepted 23 September 2024. Date of publication 26 September 2024; date of current version 27 December 2024. This brief was recommended by Associate Editor A. Yan. *(Corresponding author: Hanqian Wu.)*

Guosheng Zhang and Hanqian Wu are with the College of Software Engineering, Southeast University, Nanjing 210096, China (e-mail: zgs18655058726@gmail.com; hanqian@seu.edu.cn).

Xiaotian Guis is with the School of Electronic Science & Engineering, Southeast University, Nanjing 210096, China (e-mail: 220221674@seu.edu.cn).

Xinyi Zhang is with the School of Integrated Circuit, Southeast University, Nanjing 210096, China (e-mail: 220226337@seu.edu.cn).

Lisong Shao is with the the High-Performance R&D Department, Phytium Technology, Changsha 410028, China (e-mail: shaolisong@phytium.com.cn).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TCSII.2024.3468317.

Digital Object Identifier 10.1109/TCSII.2024.3468317

PR [5] is a key feature of FPGA, enabling rapid hardware function switching during runtime to improve efficiency and flexibility. It further facilitates on-the-fly programming [6], enabling the selective reconfiguration of specific FPGA areas without disrupting ongoing operations. Due to its advantages, PR is extensively utilized in applications such as SDR, network protocol processing, and security encryption. This approach enables FPGA to accomplish multiple application switches, promoting flexible reconfiguration while optimizing resource utilization [7]. Despite the above-mentioned benefits of PR, the significant latency issues in strictly time-bound scenarios (see [8] for aerospace applications) are also encountered. Rapid switching and repair are imperative in the event of system hardware faults, while high latency tends to prolonged fault handling time, potentially leading to system malfunction or posing safety threats. Reconfigurable applications are typically sizable [9], [10], spanning from tens to hundreds of megabytes. For instance, the Hardware ICAP (HWICAP) PR controller proposed in [11], suffers from a reconfiguration latency measured in seconds, which significantly exceeds the milliseconds or microseconds demanded in highly time-sensitive applications [12]. Consequently, this latency frequently emerges as a performance bottleneck in such critical scenarios.

To further delve into the underlying reasons for the high latency associated with PR, it is imperative to analyze its occurrence process. PR occurs within an FPGA and across devices, including Host-to-FPGA reconfiguration (typically in remote and online update scenarios). When it comes to reconfiguration within an FPGA, HWICAP serves as a standard solution, offering high flexibility but with the trade-off of increased latency [9]. This high latency is attributed to inefficiencies in the HWICAP controller, including a redundant Finite State Machine (FSM), unseparated control and data paths, as well as the utilization of the low-rate Programmed I/O (PIO) for data transfer. In cross-device scenarios, Host-to-FPGA reconfiguration typically extends techniques devised for internal FPGA reconfiguration, introducing additional complexity. A two-stage approach is used in [13], [14] to mitigate this: (i) transferring the configuration bitstream to the off-chip DDR of FPGA; (ii) reading/writing it into the FPGA's configuration memory using DMA. The employment of off-chip storage media effectively mitigates the overwhelming demand for direct Host-to-FPGA communication. However, this approach simultaneously introduces an increase in overall reconfiguration latency. Additionally, it requires high-capacity storage (e.g., DDR) on the FPGA board, posing significant challenges for cross-platform implementation and interoperability.

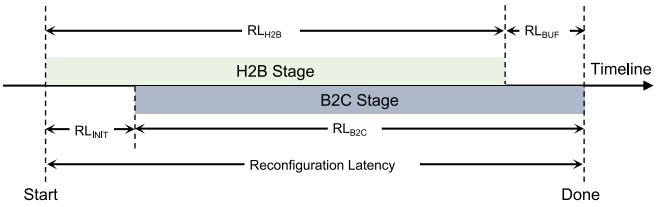

Fig. 1. Diagram of the H2C Reconfiguration Cost Model.

Building on the above discussion, a low-latency PR controller is proposed. The key contributions of this brief are summarized below, along with the availability of an open-source implementation of the controller in [15] for further research and development.

- The Host-to-FPGA reconfiguration process has been systematically analyzed, and the proposed enhanced cost model aims to identify key factors that can significantly reduce reconfiguration latency. These key factors encompass an optimized configuration buffer depth, a high-rate PCIe-DMA data path and higher ICAP frequency (Section II).
- Leveraging the producer-consumer model, Dora, a low-latency PR controller, is proposed to optimize the transmission of configuration bitstream and enhance reconfiguration frequency. By incorporating a scatter gather (SG)-streaming-based hybrid reconfiguration mechanism and an adaptive ICAP overclocking method, Dora boosts bitstream production and consumption, leading to a substantial reduction in overall reconfiguration latency (Section III).
- Utilizing Dora, a reconfigurable system for performance testing is constructed, and experimental results demonstrate that it consumes fewer FPGA resources while achieving lower latency compared to similar PR controllers (Section IV).

The remainder of this brief is structured as follows: Section II outlines a reconfiguration cost model that centers on the process from Host to FPGA. Section III introduces Dora, a low-latency PR controller. Section IV delves into the performance analysis of Dora, and Section V concludes the discussion.

II. Development of a Cost Model

To analyze the reconfiguration latency (RL) of the reconfiguration process from a Host to an FPGA, the transmission of the configuration bitstream must be divided into two stages: (i) Transmission from the Host to the FPGA's configuration buffer (H2B) using DMA, and (ii) Transmission from the FPGA's configuration buffer to its configuration memory (B2C), through the ICAP configuration port. Based on the cost model proposed by Papadimitriou et al. [16] and combined with the specific requirements of the application scenario in this brief, the following cost model is proposed, as shown in Fig. 1.

The use of a configuration buffer allows for parallel execution of H2B and B2C stages, resulting in temporal overlap. The total stage time is the sum of times taken by multiple transactions within each stage. The overall reconfiguration latency should meet Eq. (1). H2B stage latency depends on data transfer transactions and their execution time. B2C stage latency depends on ICAP configuration transactions and their execution time.

$$RL_{H2C} \le RL_{H2B} + RL_{B2C} = \sum_{i=1}^{calls_{H2B}} RL_{H2B_i} + \sum_{i=1}^{calls_{B2C}} RL_{B2C_i} \quad (1)$$

The given equation can estimate the upper bound of reconfiguration latency, but an accurate calculation requires considering the stall time caused by buffer read-write rate mismatch. Eq. (2) focuses on B2C stage and configuration initialization latency, while Eq. (3) emphasizes H2B stage and reading remaining bitstream from the buffer. The two methods yield consistent results. Analysis shows that minimizing idle periods reduces reconfiguration latency, requiring an optimized configuration buffer depth to match the read and write rates (**Key 1**).

$$\begin{aligned} RL_{H2C} &= RL_{INIT} + RL_{B2C} + RL_{STALL} \\ &= RL_{INIT} + \sum_{i=1}^{calls_{B2C}} RL_{B2C_i} \\ &\quad + \sum_{i=1}^{calls_{STALL}} RL_{STALL_i} \end{aligned} \quad (2)$$

$$\begin{aligned} RL_{H2C} &= RL_{H2B} + RL_{BUF} + RL_{STALL} \\ &= \sum_{i=1}^{calls_{H2B}} RL_{H2B_i} + \sum_{i=1}^{calls_{BUF}} RL_{BUF_i} \\ &\quad + \sum_{i=1}^{calls_{STALL}} RL_{STALL_i} \end{aligned} \quad (3)$$

In the analysis, the time for each stage relies on its reconfiguration rate (RR). For H2B, it's governed by the write buffer's frequency and bit width; for B2C, by the ICAP's read buffer frequency and bit width. The actual rate (Eq. (4)) won't exceed the minimum of H2B and B2C theoretical rate (Eq. (5)). The H2B stage uses PCIe-DMA, and B2C's bit width is fixed. Thus, a high-rate PCIe-DMA path (**Key 2**) and higher ICAP frequency (**Key 3**) are crucial for lower reconfiguration latency.

$$RR = Bitstream_{size}/RL \quad (4)$$

$$\begin{aligned} RR &\le min\{RR_{H2B}, RR_{B2C}\} \\ &= min\{freq_{H2B} \times width_{H2B}, freq_{B2C} \times width_{B2C}\} \end{aligned} \quad (5)$$

In summary, we present a cost model for Host-to-FPGA PR, analyzing latency and rates, and identify three key factors for achieving low-latency reconfiguration.

III. Proposed PR Controller

A. Overview of the Architecture

Dora focuses on the Host-to-FPGA PR process, modeled as a producer-consumer system, with the buffer's bitstream as the production-consumption object (depicted in Fig. 2). It includes three steps: (i) **Host Initiation**, where the Clock Training module determines the optimal ICAP frequency; (ii) **Bitstream Transmission**, where the Host reads the bitstream, creates task

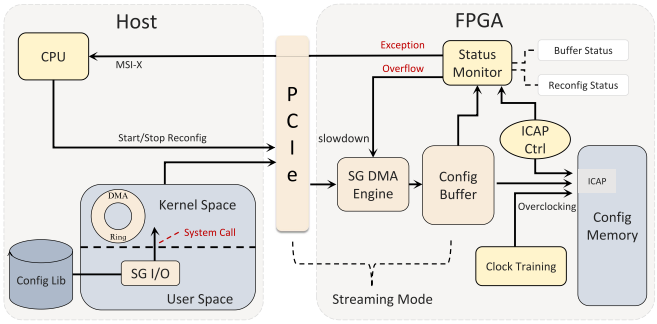

Fig. 2. Dora Low-Latency Reconfiguration Flow.

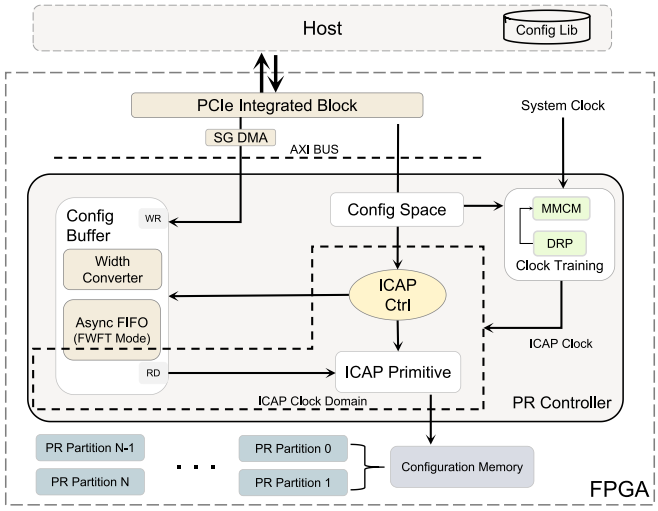

Fig. 3. Block diagram focusing on PR Controller.

descriptors, issues a Start Reconfig command to initiate SG DMA, and transmits the bitstream over PCIe to the Config Buffer in the FPGA; and (iii) **ICAP Configuration**, where the ICAP Ctrl module writes the bitstream from the Config Buffer into the Config Memory. The Status Monitor oversees the Buffer and Reconfiguration process, regulating DMA flow or halting transmission when the buffer is overflowing, and generating MSI-X interrupts to alert the host CPU of reconfiguration exceptions.

The main controller, as shown in Fig. 3, uses custom-designed modules for Dora's portability across different development boards. It has two primary paths: (i) **Control Path**: The ICAP Ctrl and Clock Training modules are controlled through PIO within the Config Space, with the latter module including the Mixed-mode Clock Manager (MMCM) for generating clocks and the Dynamic Reconfiguration Port (DRP) for tuning clock parameters; (ii) **Data Path**: The bitstream travels through the Config Buffer, which is composed of a Width Converter for adjusting the data width between AXI and ICAP and an Async FIFO for buffering data. From there, it is written into the Configuration Memory via ICAP, which contains bitstreams for multiple PR Partitions.

B. SG-Streaming-Based Hybrid Reconfiguration Mechanism

To bolster the bitstream production rate, we introduce a reconfiguration mechanism that integrates SG and streaming optimizations. Leveraging the PCIe-DMA [17] data transmission path, this method optimizes the transmission both in software and hardware, significantly diminishing system calls and PCIe interrupts. As a result, it establishes high-rate PCIe-DMA path. The key points of this mechanism, accompanied by their corresponding key factors, are described below.

- *SG-based bitstream transmission (Key 2):* Specifically, for data transfers from user space to kernel space, SG I/O is employed, consolidating multiple discrete data transfers to reduce the number of system calls. For data transfers from kernel space to devices, SG DMA is introduced, unloading the CPU's burden while logically connecting physically dispersed DMA descriptors to support DMA engine continuous pre-fetch of multiple descriptors, thereby reducing PCIe interrupt occurrence.
- *Streaming-based bitstream transmission (Key 2):* A two-stage reconfiguration process is inefficient as it requires storing the configuration bitstream in DDR. Instead, using a streaming transmission mechanism and a configuration buffer eliminates the need for DDR storage. This buffer's non-addressing nature avoids transmission interruptions caused by 4K address boundaries during burst transfers [18], enabling the actual transmission rate to closely approach the theoretical rate.
- *Configuration buffer depth adjustment (Key 1):* Incorrect buffer depth settings can frequently cause interrupts, increasing reconfiguration latency. Adjusting buffer depth based on actual read and write rate minimizes interrupts. The calculation method for buffer depth [19] is:

$$depth = len - len \times \frac{rd_{freq}}{wr_{freq}} \times \frac{rd_{duty}}{wr_{duty}} \tag{6}$$

  where len denotes the maximum length of burst transmission.
- *FSM-free design (Key 2):* Existing solutions like HWICAP use a FSM to control reconfiguration, incurring time overhead from software configuration registers. Dora adopts an FSM-free design where hardware automatically senses the buffer status and performs the configuration.

C. Adaptive ICAP Overclocking Training Method

In order to enhance bitstream consumption rate in reconfiguration process, we develop a targeted solution: the adaptive ICAP overclocking training method. This method dynamically boosts the ICAP frequency to enhance the consumption rate, while also incorporating sophisticated timing optimizations to mitigate the risks of metastability even timing violations. The key points of this method, along with their corresponding key factors, are outlined below.

- *ICAP overclocking (Key 3):* Boosting consumption rate hinges on ICAP's frequency. On Xilinx 7-series FPGAs, a 32-bit bus enables raising ICAP frequency to 550 MHz [20], accelerated by advanced ports and logic optimizations. However, Xilinx's static clock placement [21] is a challenge. DRP allows runtime parameter adjustments without full reconfiguration [22]. The ICAP clock, from an MMCM, can be tuned by DRP to optimize and boost frequency gradually, overclocking the

TABLE I
RECONFIGURATION RATE AND LATENCY OF DIFFERENT PR CONTROLLERS

Design	Platform	Config ICAP Freq (MHz)	Partial Bitstream	Rate (MBps) ICAP	Rate (MBps) Overall	Latency (ms)
Xilinx HWICAP [11]	VCU709	100	15MB	N/A	36.9	425.3
Monopoli et al. with ZyCap [9]	ZCU106	200	15MB	N/A	772	19.2
Monopoli et al. with DFX [9]	ZCU106	200	13.5MB	N/A	760	17.7
Ram et al. [23]	ZYNQ	100	3.9KB	N/A	45.29	0.087
Ma et al. [14]	VCU118	100	15MB	N/A	214	70
Proposed	VCU709	360	15MB	1440	1420.7	11.1
Proposed	VCU709	100	3.9KB	400	398.4	0.01

TABLE II
RESOURCE UTILIZATION OF DIFFERENT PR CONTROLLERS

	LUTs	FFs	MUXes	BRAMs
HWICAP [11]	594	1399	1	1.5
HBICAP [24]	995	1994	0	1.5
Proposed	226	347	0	1

configuration port to its stability limits, maximizing ICAP frequency and minimizing latency.

- *Timing optimization (Key 3):* Gradually increasing the ICAP clock frequency significantly increases the probability of circuits metastability, leading to timing violations related to the setup time of the ICAP clock. To mitigate this issue, we optimize the combinational logic delay on the critical path through pipelining, register retiming and logic replication, and further reduce the occurrence of timing violations through placement and routing optimizations. These efforts enable the overall system to achieve a working frequency of 360 MHz while meeting the timing closure requirements.

IV. PERFORMANCE

A performance experimental platform, incorporating a reconfigurable system integrated with Dora, has been established. This platform is founded upon the Xilinx VC709 board and an Intel Xeon E5 server. This platform evaluates Dora's performance using resource utilization, reconfiguration rate, and latency for real-time metrics.

The resource utilization comparison of Dora with HWICAP and High Bandwidth ICAP (HBICAP), obtained after post-implementation on the VC709 board using Vivado DS and detailed in Table II, reveals that Dora significantly reduces logic resource utilization (LUTs and FFs) by 62-75% and 77-83%, respectively, and decreases storage requirements (BRAMs) by 33%. This makes Dora a preferable option for resource-constrained applications.

As depicted in Fig. 4, the utilization of the SG mode for optimization, based on the traditional AXI Memory Mapped transmission, resulted in an earlier increase in data transfer rate to above 5600 MBps. Furthermore, by incorporating the streaming transmission into the SG foundation, the overall PCIe-DMA transfer rate is significantly boosted to approximately 6700 MBps. This hybrid optimization approach led

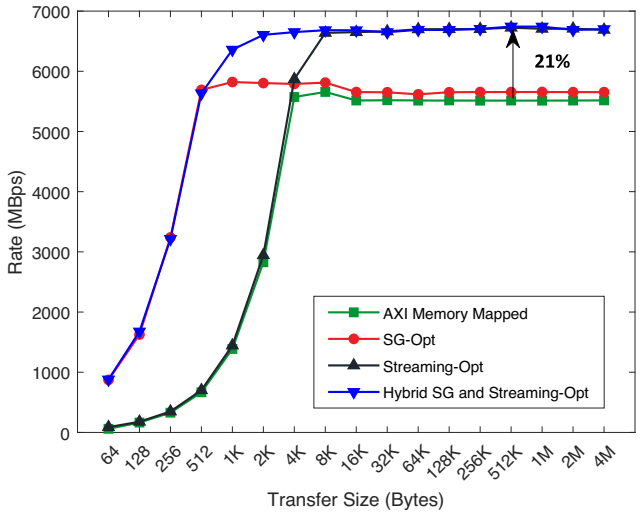

Fig. 4. PCIe-DMA data transfer rate optimized by hybrid SG and Streaming under different transfer sizes.

to a remarkable enhancement in transmission performance, achieving a 21% improvement.

Fig. 5 indicates that the reconfiguration rate (H2C) closely approaches 99.6% of the theoretical maximum (B2C) as the ICAP frequency increases. Fig. 6 further shows that latency scales linearly with the size of the configuration bitstream, and proper buffer depth adjustment can mitigate the performance impact of large bitstream, thus avoiding unnecessary system interruptions.

The reconfiguration rate and latency of Dora are compared with existing works in Table I. Under the same configuration, Dora outperforms traditional solutions, achieving a rate of 1420.74 MBps and a latency of 11.1 ms at 360 MHz, which is 2.6% of HWICAP's latency. At 100 MHz ICAP frequency and with a partial bitstream size of 3.9 KB, Dora shows an 8.7 times faster reconfiguration rate and an 88.5% reduction in latency compared to the controller proposed by Ram et al. [23]. Such performance meets the requirements of real-time systems [12].

V. CONCLUSION

This brief proposes Dora, an FPGA PR controller designed for low-latency, addressing the challenge of rapid reconfiguration. Leveraging the constructed refined PR cost model, Dora boasts a SG-streaming-based hybrid reconfiguration mechanism and an adaptive ICAP overclocking training method,

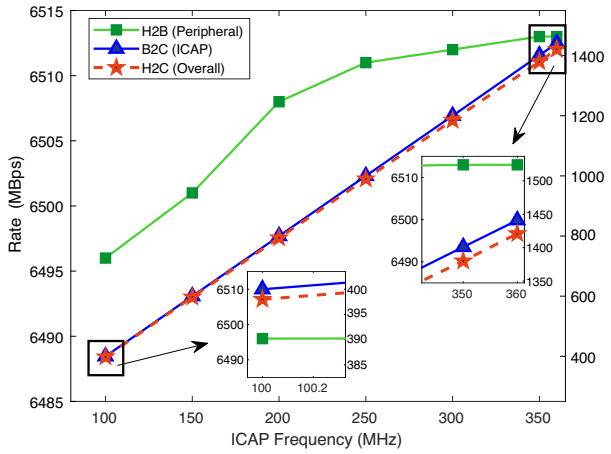

Fig. 5. Reconfiguration rates at different reconfiguration phases and ICAP frequencies.

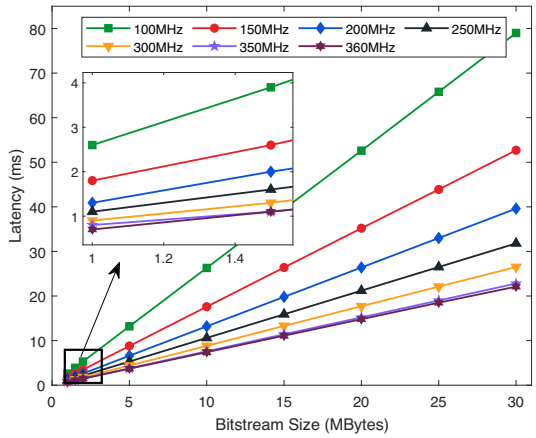

Fig. 6. Reconfiguration latency at different bitstream sizes and ICAP frequencies.

optimizing resource utilization and achieving a high reconfiguration rate with minimal latency. Its performance surpasses existing solutions, positioning Dora as an exceptional choice across various scenarios and domains.

REFERENCES

[1] P. Babu and E. Parthasarathy, "Reconfigurable FPGA architectures: A survey and applications," *J. Inst. Eng. (India), Ser. B*, vol. 102, pp. 143–156, Feb. 2021.

[2] W. Jiang, "Software defined satellite networks: A survey," *Digit. Commun. Netw.*, vol. 9, no. 6, pp. 1243–1264, 2023.

[3] A. Ramírez-Pérez, R. Aldana-López, O. Longoria-Gandara, J. Valencia-Velasco, L. Pizano-Escalante, and R. Parra-Michel, "Modular arithmetic CPM for SDR platforms," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 69, no. 4, pp. 2111–2115, Apr. 2022.

[4] B. Xu, H. Geng, L. Jiang, S. Zou, K. Chen, and Z. Liu, "FPGA implementation of memristor emulators using fractional order calculus: A high-precision reconfigurable approach," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 71, no. 4, pp. 1615–1627, Apr. 2024.

[5] K. Vipin and S. A. Fahmy, "FPGA dynamic and partial reconfiguration: A survey of architectures, methods, and applications," *ACM Comput. Surv. (CSUR)*, vol. 51, no. 4, pp. 1–39, 2018.

[6] N. Aimaier, Y. Blaquière, N. Constantin, and G. E. R. Cowan, "An FPGA-based on-the-fly reconfigurable low-power SHEPWM inverter with a compact SiP implementation," *IEEE Trans. Power Electron.*, vol. 39, no. 5, pp. 5942–5953, May 2024.

[7] P. Darbani, N. Rohbani, H. Beitollahi, and P. Lotfi-Kamran, "RASHT: A partially reconfigurable architecture for efficient implementation of CNNs," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 30, no. 7, pp. 860–868, Jul. 2022.

[8] N. Bai, X. Xiao, Y. Xu, Y. Wang, L. Wang, and X. Zhou, "Soft-error-aware SRAM with multinode upset tolerance for aerospace applications," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 32, no. 1, pp. 128–136, Jan. 2023.

[9] M. Monopoli, L. Zulberti, G. Todaro, P. Nannipieri, and L. Fanucci, "Exploiting FPGA dynamic partial reconfiguration for a soft GPU-based system-on-chip," in *Proc. 18th Conf. Ph. D Res. Microelectron. Electron. (PRIME)*, 2023, pp. 181–184.

[10] A. R. Bucknall and S. A. Fahmy, "ZyPR: End-to-end build tool and runtime manager for partial reconfiguration of FPGA SoCs at the edge," *ACM Trans. Reconfig. Technol. Syst.*, vol. 16, no. 3, pp. 1–33, 2023.

[11] *AXI HWICAP V3.0: LogiCORE IP Product Guide*, Xilinx, San Jose, CA, USA, 2020.

[12] A. K. Rana and A. Ravi Teja, "A fault-tolerant power converter with multi-switch fault diagnosis and repair capability for 4-phase 8/6 SRM drives," *IEEE Trans. Transp. Electrif.*, vol. 8, no. 3, pp. 3896–3906, Sep. 2022.

[13] R. S. Ram and M. L. C. Prabhaker, "Intelligent optimization approaches for a secured dynamic partial reconfigurable architecture-based health monitoring system," *J. Circuits, Syst. Comput.*, vol. 32, no. 3, 2023, Art. no. 2350047.

[14] L. Ma, C.-W. Sham, J. Sun, and R. V. Tenorio, "A real-time flexible telecommunication decoding architecture using FPGA partial reconfiguration," *IEEE Trans. Circuits Syst. II, Exp. Briefs*, vol. 67, no. 10, pp. 2149–2153, Oct. 2020.

[15] "Dora." Accessed: May 31, 2024. [Online]. Available: https://github.com/amarisex/dora

[16] K. Papadimitriou, A. Dollas, and S. Hauck, "Performance of partial reconfiguration in FPGA systems: A survey and a cost model," *ACM Trans. Reconfig. Technol. Syst. (TRETS)*, vol. 4, no. 4, pp. 1–24, 2011.

[17] M. Ouyang, F. Gao, Y. Wang, S. Zhang, P. Li, and J. Ren, "Computer vision-aided reconfigurable intelligent surface-based beam tracking: Prototyping and experimental results," *IEEE Trans. Wireless Commun.*, vol. 22, no. 12, pp. 8681–8693, Dec. 2023.

[18] W. Wang, C. He, and J. Shi, "A secure SoC architecture design with dual DMA controllers," *AIP Adv.*, vol. 14, no. 2, 2024, Art. no. 25125.

[19] S. Abdel-Hafeez and A. Gordon-Ross, "Reconfigurable fifo memory circuit for synchronous and asynchronous communication," *Int. J. Circuit Theory Appl.*, vol. 49, no. 4, pp. 938–952, 2021.

[20] K. Vipin and S. A. Fahmy, "A high speed open source controller for FPGA partial reconfiguration," in *Proc. Int. Conf. Field-Program. Technol.*, 2012, pp. 61–66.

[21] A. Boutros and V. Betz, "FPGA architecture: Principles and progression," *IEEE Circuits Syst. Mag.*, vol. 21, no. 2, pp. 4–29, May 2021.

[22] T. Addabbo, A. Fort, R. Moretti, F. Spinelli, and V. Vignoli, "Multi-phase frequency measurement exploiting FPGA mixed-mode clock management for QCM-D technology," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, 2023, pp. 1–5.

[23] R. S. Ram, M. L. C. Prabhaker, K. Suresh, K. Subramaniam, and M. Venkatesan, "Dynamic partial reconfiguration enchanced with security system for reduced area and low power consumption," *Microprocess. Microsyst.*, vol. 76, Jul. 2020, Art. no. 103088.

[24] *AXI HBICAP V1. 0: LogiCORE IP Product Guide*, Xilinx, San Jose, CA, USA, 2019.