Dynamic Resource Management in Reconfigurable SoC for Multi-Tenancy Support

Sohyeon Kim[1], Injun Choi[2], Minkyu Je[2], and Ji-Hoon Kim[1]
E-mail: sohyeon.kim@ewhain.net {injunchoi, mkje}@kaist.ac.kr jihoonkim@ewha.ac.kr
[1]Department of Electronic and Electrical Engineering, Ewha Womans University, Seoul, Korea
[2]School of Electrical Engineering, Korea Advanced Institute of Science and Technology (KAIST), Daejeon, Korea

2024 IEEE International Symposium on Circuits and Systems (ISCAS) | 979-8-3503-3099-1/24/$31.00 ©2024 IEEE | DOI: 10.1109/ISCAS58744.2024.10558110

***Abstract*—This study introduces a partially reconfigurable system-on-chip (SoC) platform leveraging dynamic resource management facilitated by a dynamic reconfigurable control processor (DRCP). By addressing the inherent reconfiguration time overheads of reconfigurable SoCs, the study demonstrates performance improvements through a runtime resource management strategy. The introduced management scheme effectively reduces the frequency of reconfigurations, thus lessening the associated overheads and increasing the operational efficiency of the SoC platform, which is designed to support on-chip multi-tenancy. Utilizing DRCP for dedicated resource management, the proposed SoC platform exhibited substantial reductions in reconfiguration times. When the partially reconfigurable SoC platform employed four partial regions (PRs), the reconfiguration counts decreased by 37.6%. Furthermore, upon extending the PRs to eight, there was a notable reduction in reconfiguration counts, achieving a decrease of 47.0%.**

***Index Terms*—Reconfigurable SoC, FPGA, Dynamic partial reconfiguration, Dynamic resource management, Multi-tenancy**

I. Introduction

Recently, system-on-chip (SoC) architectures incorporating embedded reconfigurable logic, known as embedded field-programmable gate array (eFPGA), have gained recognition for their energy efficiency and adaptability to varying computational requirements [1]–[5]. As depicted in Fig. 1, compared to the conventional SoC architecture with fixed hardware accelerators (**Type I**), the utilization of reconfigurable logic allows for on-the-fly customization and optimization of hardware resources. By dynamically configuring the reconfigurable fabric (**Type II**), SoCs can adapt their hardware configurations to specific workloads, enhancing overall system performance. The integration of shared reconfigurable logic with dynamic partial reconfiguration (DPR) facilitates the creation of a pooled eFPGA architecture (**Type III**), enabling efficient resource partitioning and utilization among multiple users [6], [7]. The multi-tenancy support provided by the pooled eFPGA architecture has significant implications in various domains, such as cloud computing and reconfigurable computing [8], [9].

Bitstreams for digital signal processing (DSP) kernels needed throughout the system are pre-generated and stored in bitstream memory, allowing the requested DSP kernels by CPU cores to be dynamically allocated to reconfigurable regions during runtime and deallocated once used. This process

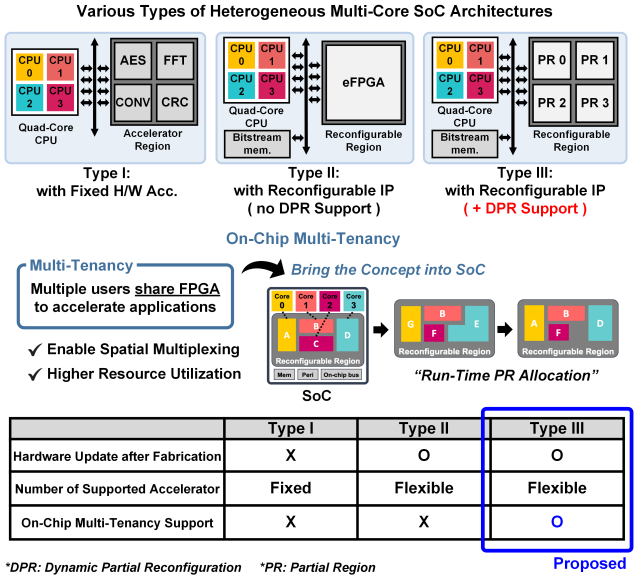


Fig. 1. Various types of heterogeneous multi-core SoC architecture and the concept of on-chip multi-tenancy

enables the support of a more significant number of kernels within a finite area. However, programming new bitstreams during runtime incurs a reconfiguration time overhead, which can negatively impact the overall system performance due to millisecond-level time delays [10]. Moreover, as the number of DSP kernels supported exceeds the number of limited partial regions, the likelihood of replacements increases, thus amplifying this overhead. Therefore, in systems supporting multi-tenancy based on DPR, it is crucial to minimize reconfiguration time overhead to maintain performance and necessitate systematic research in resource management.

In this paper, we present a partially reconfigurable SoC platform that can efficiently support multi-tenancy using runtime resource management with a dynamic reconfigurable control processor (DRCP). The DRCP incorporates a partial region management unit (PRMU), responsible for managing the partial region (PR) status table and PR allocation to minimize time-consuming reconfigurations utilizing a simple RISC core internally for scheduling and allocation algorithm. Additionally, a partial region transparency unit (PRTU) maintains physical allocation information, enabling CPU cores to

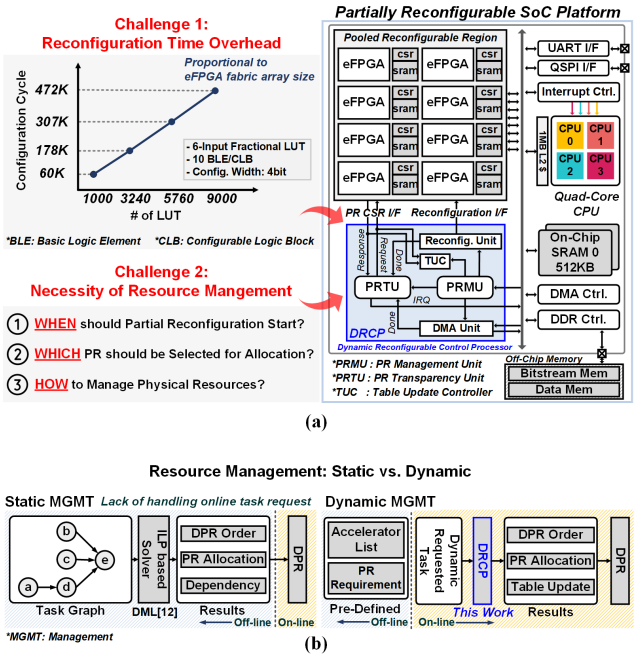


Fig. 2. Research motivation: (a) Challenges of conventional reconfigurable SoC. The proposed partially reconfigurable SoC platform with a dynamic reconfiguration control processor (DRCP) as a solution. (b) Resource management schemes, static and dynamic.

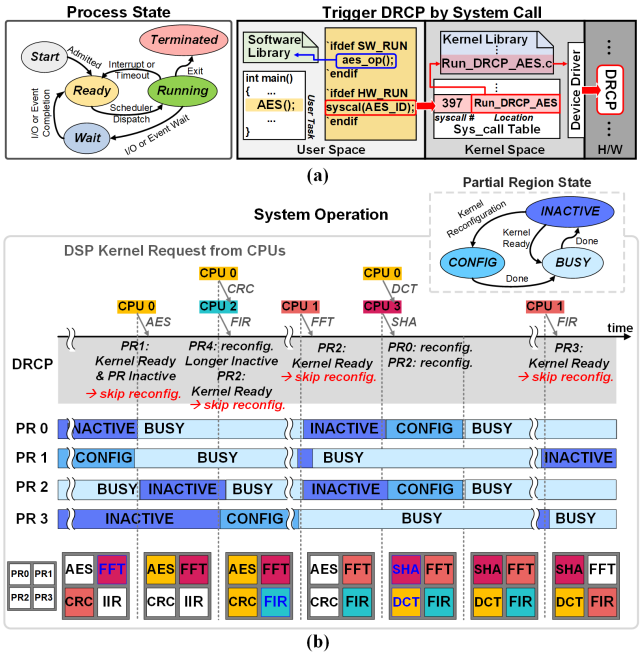


Fig. 3. Overview of the operational process for the proposed reconfigurable SoC platform: (a) The mechanism by which CPU cores request PR allocation from the DRCP for task offloading. (b) The system operation scenario depicting how the DRCP allocates and reallocates PR resources.

request and receive tasks from DSP kernels without knowing the dynamically changing physical allocation information. By efficiently managing PRs and maintaining physical allocation transparency, the DRCP enhances the performance and flexibility of the proposed partially reconfigurable SoC platform with multi-tenancy support.

The remainder of this paper is organized as follows. Section II describes the motivation for this study. Section III presents the hardware structure of DRCP and the overall system operation of the proposed partially reconfigurable SoC platform. Section IV discusses the FPGA implementation results of the proposed SoC platform, and finally, it concludes in Section V.

II. Motivation

This section examines the challenges inherent in conventional reconfigurable SoC platforms. It then articulates the critical need for effective resource management, thereby underpinning the motivation for this study.

A. Conventional Reconfigurable SoC Platform

Various reconfigurable SoCs leverage eFPGA for the configuration and offload of DSP kernels have been proposed, aiming to concurrently improve energy efficiency and computational performance [1]–[5]. eFPGA occupies a significant area within a chip, and as depicted in Fig. 2(a) *Challenge 1*, the reconfiguration time tends to increase in proportion to the size of the eFPGA fabric. To mitigate this overhead, techniques like DPR [6], [7] have been proposed to reduce bitstream size, yet overheads are still in the order of milliseconds [10]. By employing DPR, multiple users can pool and utilize shared FPGA resources, thus highlighting the benefits of resource utilization, which has brought multi-tenancy research into focus [8]. However, this approach has introduced new challenges, particularly the decision of when, where, and how to allocate resources, as mentioned in Fig. 2(a) *Challenge 2* [11], [12].

B. Resource Management Scheme

Research is being conducted on resource management techniques based on a dynamic partial reconfiguration platform. [12] proposed a methodology for statically scheduling workloads and performed run-time partial reconfiguration. As shown in Fig. 2(b), the static resource management approach is practical when applied to predetermined sequences of tasks. However, the software-based approach has limitations in tracking the continually changing status of each PR shared by the CPU core online. Dynamic resource management, in contrast, provides enhanced adaptability and efficiency in handling unpredictable workloads, enabling real-time allocation that optimizes resource utilization and performance [13].

To mitigate the challenges mentioned earlier, this paper proposes DRCP, which uses status tables for dynamic resource management within a partially reconfigurable SoC platform, as shown on the right side of Fig. 2(a). DRCP is designed to improve the system's overall performance by minimizing the occurrence of random DSP kernel reconfigurations requested by independent quad-core CPUs.

III. Proposed Dynamic Resource Management

In this section, we describe the hardware architecture of the DRCP that enables dynamic resource management, as well as the overall system operation.

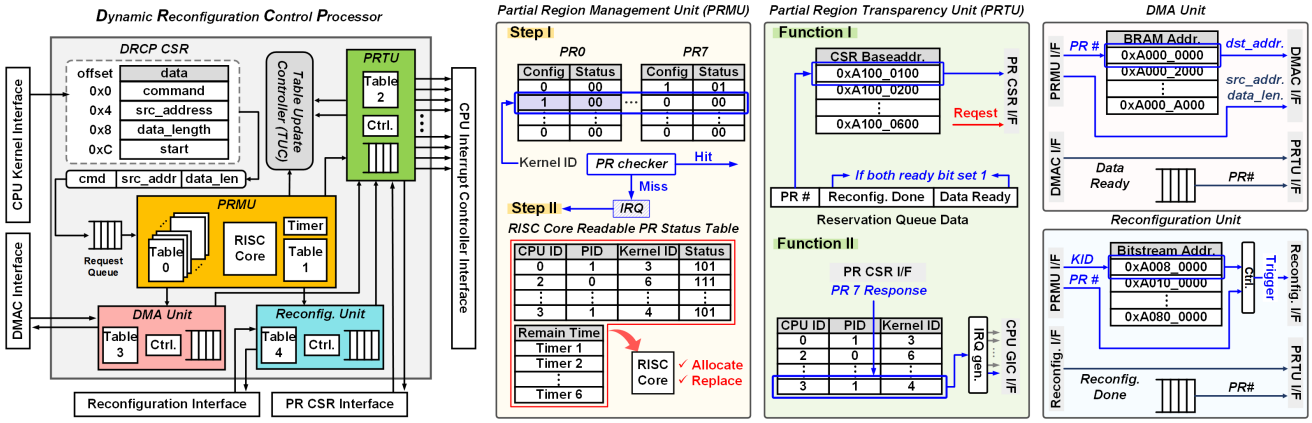


Fig. 4. Overall hardware architecture of dynamic reconfiguration control processor (DRCP) and detailing the partial region management unit (PRMU), the partial region transparency unit (PRTU), the DMA unit, and the reconfiguration unit.

A. DRCP Hardware Architecture

Fig. 4 shows the hardware architecture of DRCP. Through the CPU's device driver, DRCP receives CPU ID, process ID (PID), and kernel ID (KID) as commands. It also receives memory address and size information of the data to be processed through the internal control status register (CSR) and forwards all this information to the PRMU. The PRMU first searches the status information of each PR that can be determined by the hardware (**Step I**). If the requested kernel is already configured and the PR is inactive, it is directly allocated. Otherwise, PRMU sends an IRQ to the internal small RISC core, and it schedules the requested task by the management program, performing allocation and replacement based on the PR status (**Step II**). The resource management by the PRMU reduces the number of reconfigurations, which leads to the total execution time reduction.

Once the PR is allocated, PRMU sends the signal to DMA unit, Reconfiguration unit, and PRTU. Both the DMA unit and reconfiguration unit are similar to PRTU. They also manage tables to provide transparency of PR to CPU cores. DMA unit searches a table that manages BRAM addresses of PR, and it sends signals to the DMA controller to move the data to the physical address of the allocated BRAM. Reconfiguration Unit, when necessary, sends the PR number and pre-defined bitstream memory address for the target kernel ID (KID) in the list to the Configuration Controller to trigger PR reconfiguration.

The PRTU receives information about the PRs allocated by PRMU and stores it in the reservation queue while waiting for reconfiguration and data move to be completed. Once both are completed, PRTU sends a processing request signal to the CSR of the corresponding PR (**Function I**). When the computation is finished in the PR, it sends a response signal to PRTU, which generates an IRQ signal in the CPU interrupt controller for the requested CPU to trigger an interrupt (**Function II**). This allows the CPU to have the effect of not needing the physical allocation information of the dynamically changing allocation, as it can handle it through the status table.

B. Overall System Operation

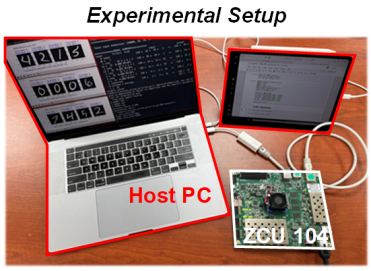
Fig. 3 demonstrates the overall process of the proposed partially reconfigurable SoC platform, including (i) how CPU cores request PR allocation to the DRCP for task offloading and (ii) how the DRCP allocates and reallocates PR resources while performing task scheduling. For example, as illustrated in Fig. 3(a), CPU cores use a system call to request AES (Advanced Encryption Standard) kernel offloading to PRs, where OS (Operation System) sends a request command to the DRCP through the device driver and triggers the DRCP. The state of the process, which sends the offloading request, changes from *Running* to *Wait* and back to *Ready* after the completion of the configured PR operation.

As shown in Fig. 3(b), when the DRCP receives the request command from CPU cores, it checks the status of PRs with PRMU. With the pre-determined set of DSP kernels, the DRCP checks whether the request kernel is already configured in any PR. If the DRCP finds the *INACTIVE* PR with the same kernel, then directly allocates, and the PR state turns *BUSY*. We can skip a time-consuming reconfiguration process (*Hit*). If the corresponding PR with the same kernel is *BUSY*, the process changes to *Wait* until it becomes *INACTIVE* to avoid reconfiguration. If there is no PR configured with the requested kernel, the DRCP identifies the PR that has remained *INACTIVE* for the most prolonged duration, leveraging temporal locality to allocate the requisitioned kernel via reconfiguration. Since the reconfiguration is time-consuming, the DRCP tries to keep the frequently used kernel in the pooled reconfigurable logic.

For efficient PR resources management, the DRCP should keep the inactive PR list for the request kernel allocation and track the status of all PRs with the PRMU. Also, the DRCP can decouple the PR allocation process from the CPU process with the PRTU, which avoids the software complexity increase for the proposed partially reconfigurable SoC platform.

IV. Experimental Results

Fig. 5 shows the experimental setup for the proposed system. The proposed system implemented quad-core CPUs in the PS part and 8 PRs and DRCP in the PL part of the



Xilinx ZCU 104	Resources			
	LUTs	FFs	DSPs	BRAM
Single PR	4200	8400	56	21
DRCP	3450	1084	0	2

Fig. 5. Dynamic resource management system experimental setup

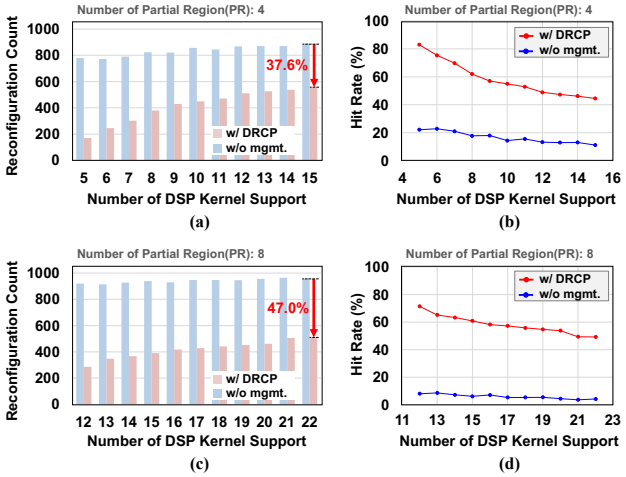


Fig. 6. Reconfiguration count and hit rate: (a) When # of PR = 4 (b) When # of PR = 4 (c) When # of PR = 8 (d) When # of PR = 8, respectively.

Xilinx ZCU104 evaluation board, operating at a frequency of 200MHz. Each single PR occupies 4200 LUTs, 8400 FFs, and 56 DSPs. The resources for a single PR are configured considering the size of the target DSP kernels.

To validate the effectiveness of the DRCP in managing resources, we configured partial regions on a ZCU104 board. We measured the number of reconfigurations and hit rate under varying conditions, as shown in Fig. 6. Specifically, this study examined the effects of increasing the number of partial regions to four and eight, as well as augmenting the quantity of DSP kernels available in bitstreams.

For comparison, the baseline scenario allocates the kernel solely to a predetermined PR for an independent CPU core request (e.g., CPU0: PR0, CPU1: PR1, CPU2:PR2, CPU3:PR3). DRCP searches the status table for all PR regions and allocates the requested DSP kernel by the CPU to the optimal PR. Upon analyzing 1000 randomly generated kernel requests, we found that as the number of supported DSP kernels grows, the hit rate generally declines while the reconfiguration count tends to increase in four PR scenarios. Nevertheless, the implementation of DRCP still managed to reduce the reconfiguration count by 37.6% compared to the baseline scenario without resource management, even with 15 DSP kernels. Despite the tendency for the hit rate to decrease and the reconfiguration count to rise with an increased number of supported DSP kernels, the reconfiguration count was still reduced by 47.0% when the number of PRs was expanded to eight and up to 22 kernels were supported compared to the baseline. These results show that the impact of DRCP becomes more pronounced as the number of partial regions increases.

TABLE I
COMPARISON WITH RELATED WORKS

	CICC'23 [1]	JSSC '22 [3]	MICRO '22 [12]	**This work**
Platform	ASIC	ASIC	ZCU106	**ZCU104**
Multi-Core	Quad	Dual	Quad	**Quad**
# of LUTs	6720	8760	10000	**4200**
# of DSP Slices	-	80	40	**56**
Multi-tenancy	X	X	X	**O**
Resource mgmt.	X	X	Static	**Dynamic**

Table I shows the comparison with state-of-the-art works. This work distinguishes itself from conventional reconfigurable SoCs by introducing a dedicated hardware mechanism for dynamic resource management, which significantly augments multi-tenancy capabilities at the on-chip level. This strategic enhancement facilitates more efficient utilization of computational resources, offering a compelling advantage for scenarios demanding robust multi-user environments.

V. CONCLUSION

This paper presents a partially reconfigurable SoC platform with DRCP to manage resources for multi-tenancy support efficiently. The proposed architecture enables a shared reconfigurable region, allowing independently operating CPUs to request and utilize hardware acceleration resources as needed at the on-chip level. The DRCP effectively minimizes reconfiguration overhead through PR status table based resource allocation, improving performance in multi-tenant environments. Experimental results highlight the DRCP's effectiveness in reducing reconfiguration counts by up to 47.0% compared to baseline scenarios, confirming the benefits of our approach in diverse workload conditions.

ACKNOWLEDGMENT

This work was partly supported by Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No. 2021-0-00724, RISC-V based Secure CPU Architecture Design for Embedded System Malware Detection and Response, 50%) and Institute of Information & communications Technology Planning & Evaluation (IITP) grant funded by the Korea government (MSIT) (No. 2021-0-00853, Developing Software Platform for Programming of PIM, 50%)

REFERENCES

[1] T.-J. Chang, A. Li, F. Gao, T. Ta, G. Tziantzioulis, Y. Ou, M. Wang, J. Tu, K. Xu, P. J. Jackson, A. Ning, G. Chirkov, M. Orenes-Vera, S. Agwa, X. Yan, E. Tang, J. Balkind, C. Batten, and D. Wentzlaff, "Cifer: A 12nm, 16mm2, 22-core soc with a 1541 lut6/mm2 1.92 mops/lut, fully synthesizable, cachecoherent, embedded fpga," in *2023 IEEE Custom Integrated Circuits Conference (CICC)*, 2023, pp. 1–2.

[2] F. Gao, T.-J. Chang, A. Li, M. Orenes-Vera, D. Giri, P. J. Jackson, A. Ning, G. Tziantzioulis, J. Zuckerman, J. Tu, K. Xu, G. Chirkov, G. Tombesi, J. Balkind, M. Martonosi, L. Carloni, and D. Wentzlaff, "Decades: A 67mm2, 1.46tops, 55 giga cache-coherent 64-bit risc-v instructions per second, heterogeneous manycore soc with 109 tiles including accelerators, intelligent storage, and efpga in 12nm finfet," in *2023 IEEE Custom Integrated Circuits Conference (CICC)*, 2023, pp. 1–2.

[3] S. K. Lee, P. N. Whatmough, M. Donato, G. G. Ko, D. Brooks, and G.-Y. Wei, "Smiv: A 16-nm 25-mm² soc for iot with arm cortex-a53, efpga, and coherent accelerators," *IEEE Journal of Solid-State Circuits*, vol. 57, no. 2, pp. 639–650, 2022.

[4] P. D. Schiavone, D. Rossi, A. Di Mauro, F. K. Gürkaynak, T. Saxe, M. Wang, K. C. Yap, and L. Benini, "Arnold: An efpga-augmented risc-v soc for flexible and low-power iot end nodes," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 29, no. 4, pp. 677–690, 2021.

[5] F. Renzini, C. Mucci, D. Rossi, E. F. Scarselli, and R. Canegallo, "A fully programmable efpga-augmented soc for smart power applications," *IEEE Transactions on Circuits and Systems I: Regular Papers*, vol. 67, no. 2, pp. 489–501, 2020.

[6] D. Koch, "Partial reconfiguration on fpgas: architectures, tools and applications," *Springer Science & Business Media*, vol. 153, 2012.

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

[8] G. Dessouky, A.-R. Sadeghi, and S. Zeitouni, "Sok: Secure fpga multi-tenancy in the cloud: Challenges and opportunities," in *2021 IEEE European Symposium on Security and Privacy (EuroSP)*, 2021, pp. 487–506.

[9] J. M. Mbongue, A. M.-I. Shuping, P. Bhowmik, and C. Bobda, "Architecture support for fpga multi-tenancy in the cloud," in *2020 IEEE 31st International Conference on Application-specific Systems, Architectures and Processors (ASAP)*, 2020, pp. 125–132.

[10] S. Liu, R. N. Pittman, A. Forin, and J.-L. Gaudiot, "Minimizing the runtime partial reconfiguration overheads in reconfigurable systems," *The Journal of Supercomputing*, vol. 61, no. 2, pp. 894–911, 2012.

[11] Y. Xu, O. Muller, P.-H. Horrein, and F. Pétrot, "Hcm: An abstraction layer for seamless programming of dpr fpga," in *22nd International Conference on Field Programmable Logic and Applications (FPL)*, 2012, pp. 583–586.

[12] A. Dhar, E. Richter, M. Yu, W. Zuo, X. Wang, N. S. Kim, and D. Chen, "Dml: Dynamic partial reconfiguration with scalable task scheduling for multi-applications on fpgas," *IEEE Transactions on Computers*, vol. 71, no. 10, pp. 2577–2591, 2022.

[13] A. K. Singh, P. Dziurzanski, H. R. Mendis, and L. S. Indrusiak, "A survey and comparative study of hard and soft real-time dynamic resource allocation strategies for multi-/many-core systems," *ACM Computing Surveys (CSUR)*, vol. 50, no. 2, pp. 1–40, 2017.