SCCTS Transactions on Reconfigurable Computing, ISSN: 3049-1533 Vol. 3, No. 1, 2026 (pp. 79-89)
RESEARCH ARTICLE ECEJOURNALS.IN

Dynamic Task Migration and Partial Reconfiguration in Heterogeneous FPGA Clusters for Adaptive Computation Offloading

S. Sindhu[1]*, S.Poornimadarshini[2]

[1] Research Analyst, Centivens Institute of Innovative Research, Coimbatore, Tamil Nadu, India.
[2]Jr Researcher, National Institute of STEM Research, India.

Keywords:
Dynamic Task Migration,
Partial Reconfiguration,
Heterogeneous FPGA Clusters,
Adaptive Computation Offloading,
Runtime Reconfigurable Systems,
Edge Computing Acceleration

Author's Email:
sindhuanbuselvaneniya@gmail.com
poornimadarshini22@gmail.com

DOI: 10.31838/RCC/03.01.09

Received : 22.10.2025
Revised : 24.11.2025
Accepted : 16.01.2026

Abstract

The growing requirement of higher performance and energy efficient computing at the edge and cloud-edge has escalated the use of heterogeneous FPGA clusters. The distinct selling point of such platforms is their reconfigurability that best suits workloads that are dynamic and where latencies are an issue. This review entails an in-depth discussion of two important factors in the enablement of runtime flexibility of such systems: dynamic task migration, and partial reconfiguration (PR). Dynamic task migration enables migration of tasks across the processing units depending on requirements of the available resources, heat, or performances. At the same time, partial reconfiguration ensures that it is possible to modify, on the fly, hardware modules in the system without affecting overall system behavior. Composing these methods provides the opportunity to flexibly and in fine detail control the computation offloading approaches, making the use and allocation of resources more effective and efficient in terms of energy consumption. This paper summarizes the state of art in the area of task migrating policy, PR methodologies, runtime frameworks and design tools, as well as outlining the synergies and trade-offs. Moreover, there is a discussion on the real-world applications, such as edge AI inference, secure system reconfiguration, and adaptive multimedia processing to show some practical applications. Lastly, proved challenges in research and future work directions within the paper are the following: intelligent scheduling, PR-aware system architecture and multi-vendor FPGA ecosystem standardization. This review can be used as the basic point of reference by researchers and practitioners who need to develop adaptation and reconfiguration computing platforms with FPGA.

How to cite this article: Sindhu S, Poornimadarshini S (2026). Dynamic Task Migration and Partial Reconfiguration in Heterogeneous FPGA Clusters for Adaptive Computation Offloading. SCCTS Transactions on Reconfigurable Computing, Vol. 3, No. 1, 2026, 79-89

Introduction

Exponential rise of data-intensive and latency-sensitive applications (autonomous systems and edge-based artificial intelligence, real-time video analytics, etc.) has led to the transition to heterogeneous adaptable computing platforms. The relatively high power consumption, as well as the inability to satisfy strict criteria of deterministic performance and the need to adapt to dynamic resource requirements, of traditional CPU- and GPU-based architectures often make them insufficient to address the demands of

edge and near-edge deployments. In this regard, the programmable FPGAs have been identified as an interesting alternative because of reconfigurable hardware substrate and its parallel processing capacity and based on their energy efficient utilization as well.

A promising architectural paradigm of next generation computing infrastructures is the heterogeneous FPGA cluster that combines processing/memory resource-diverse FPGAs, or pairs FPGAs with general-purpose processors. The advantage is that these clusters permit multiple workloads in parallel, the ability to scale within and across cloud and edge deployments and the ability to adapt to changing circumstance. But to obtain full potential of its power, two advanced capabilities come into necessity: dynamic task migration and partial reconfiguration (PR).

Dynamic task migration is the feature enabling moving the computational task on other nodes or other execution units of the FPGA cluster at runtime. The systems will be able to evenly distribute workloads, work around thermal hotspots, and withstand fluctuations in resources or performance requirements using this process. Task migration, when orchestrated correctly, helps to balance the loads, counter faults and provide responsiveness in real time. Partial reconfiguration, on the other hand, enables controlled reconfiguration of certain areas of the FPGA without taking down the whole system. This will allow reprogramming hardware modules on the fly, which allows hardware multiplexing, updates of hardware functions, dynamic adaption to changing application contexts.

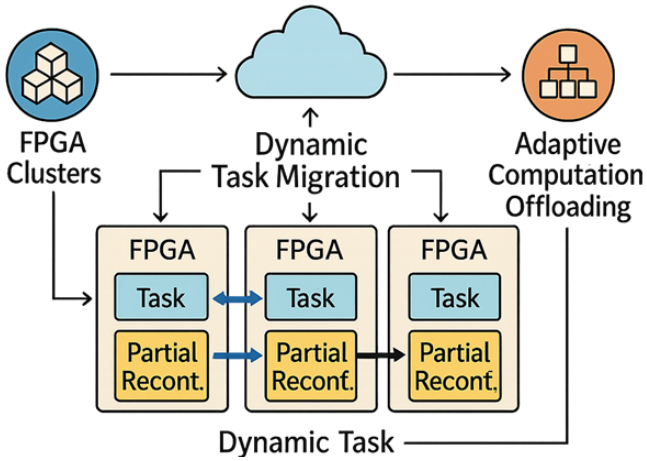

Fig. 1: System Diagram of Task Migration and Partial Reconfiguration in FPGA Clusters

Dynamic task migration and partial reconfiguration offers a good adaptive system of offending the computation in heterogeneous environments. Computation offloading, which entails offloading processing tasks placing a heavy load on limited edge devices to more resourced processing modules, can be made much more efficient through the addition of reconfiguration-aware and such that migration is possible. Such synergy enables optimization of performance, energy consumption, and resource consumption during run-time and specifically in edge-centric computing infrastructures where resource constraints are on power budgets and area budgets. Combining task migration and partial reconfiguration, although promising, is also very complex in terms of design. The issues like bitstream management, latency in configuration and runtime scheduling of tasks, hardware/software partitioning have to be solved using high-tech frameworks, toolchains and middlewares. In the last ten years, a variety of architectural countermeasures, scheduling algorithms and tool-aided design flows methodologies have been suggested to solve these issues with varying degrees of performance, flexibility, and overhead.

In this review paper, the intent is to conduct a general survey of the status quo of the dynamic task migration and partial reconfiguration of heterogeneous FPGA clusters allowing adaptive offloading of computation. It touches on baseline principles, types of current approaches, overviews design method(ologies), and studies their applicability in a range of possible real-life applications. Moreover, it determines the existing constraints and provides the possible directions of work in this rather dynamic area such as machine learning-based intelligent migration, PR-aware reconfiguration, and thermal or energy-aware task scheduling. This is to provide an amalgamated knowledge of the techniques and technologies that are about to define the future of adaptive FPGA-based computing systems to the researchers, engineers and system designers.

Background and Motivation

Heterogeneous FPGA Clusters

Heterogenous FPGA clusters are a type of computing systems that combines many FPGAs, typically with varied logic, memory, and inter-connect resources and couples them with general-purpose processors such as CPUs, GPUs or ARM cores. These environments

facilitate the execution of tasks in a collaborative fashion composed of heterogeneous components that give rise to each processing element to address a certain application requirement.

Architecturally, a typical heterogeneous FPGA cluster has several interconnected nodes where each node contains a reconfigurable fabric (FPGA) that can be reprogrammed to some degree while running. These sorts of FPGAs typically are used in collaboration with embedded processing subsystems, including ARM Cortex-A/R/M cores, digital signal processors (DSPs) and common memory blocks, permitting tight integration of the software and hardware components. Intra-cluster communication is further boosted by high-speed interconnects like, PCIe, AXI, and NoC (Network-on-Chip) fabrics providing the provision of scalable parallel processing.

Examples of business importance are:

- Xilinx Zynq UltraScale+ MPSoC, a mix of programmable logic and the quad-core ARM Cortex-A53 with two-core Cortex-R5 processors that offers the capability of real-time processing and dynamic hardware acceleration.
- Intel Stratix 10 SoC that combines the high-performance FPGA fabric with various ARM Cortex-A53 processor cores and provides new capabilities of partial reconfiguration, high-bandwidth memory interfaces, and transceiver features that address data-intensive tasks.

The rationale behind the implementation of such clusters can be seen in the fact that performance-per-watt optimization is increasingly demanded in the edge and near-edge computing. Homogeneous computing platforms that are conventional are not generally efficient in terms of workload which is generalized to include AI inference, image processing, signal decoding, or real-time controls. FPGAs are low-latency, fine-grained, and configurable; this aspect qualifies them to serve as an application-specific accelerator. The relative scale of systems equipotentially across several applications and dynamically adapt to run-time constraints like thermal stress, power budgets and job variability can be achieved by forming clusters of heterogeneous FPGAs. The flexibility is central to the areas of autonomous cars, smart surveillance, medical diagnostics, and industrial automation.

Computation Offloading Paradigms

Computation offloading is a technique where computationally demanding or latency-sensitive tasks are moved or relocated to stronger processing nodes or servers to offload the computationally limited devices (like edge nodes or sensors). The goal of this strategy is to achieve the greatest responsiveness of the system on a local level, use the least amount of energy of the local devices, and increase throughput by leveraging the capacities of the remote or neighboring compute nodes. Offloading policies may be categorised into a static and a dynamic paradigm.

- Static offloading would be use of pre-defined rules/heuristics to decide what will be offloaded locally and what will be done remotely, and they are usually laid down during the design time. Although simple and efficient in resources, static schemes are not flexible enough to meet changes in real-time of the system load or network conditions.
- Dynamic offloading, on the other hand, bases adaptive decisions on the context that is available at a given point of time, i.e., the current power levels, the thermal situation, the network latency, the task priority, etc. This type of solution is particularly useful in heterogeneous FPGA clusters where the mobility of workload and partially reconfiguring can be coordinated using the system telemetry.

As part of the general cloudedge continuum, computation offloading allows a task hierarchy. Cloud servers have a high-scale compute capacity compared to edge clusters which have lower latency and have data closer to a location. Homogenous FPGA clusters in the edge are critical to help fill the gap of the latency criterion vs. performance, especially in real-time where the delay in moving the data to the cloud is not acceptable. FPGA allows offloading operations to a great extent and makes the system responsive and throughput. FPGAs should be able to have custom accelerators to infer convolutional neural networks (CNN), process cryptography, filter signals, compress data, which are often offloaded to edge devices. Moreover, the ability of partial reconfiguration enables FPGAs to change their functionality depending on the workload that arrives at them, with the logic resources previously dedicated to the infrequent or the bursty functions no longer needed.

Overall, dynamic computation offloading, with heterogeneous FPGA cluster in addition to the adaptability to the requirements at the runtime by means of dynamic task migration and reconfiguration, proves to be a firm basis of intelligent, power-aware, and high-performance edge computing systems. Such features not only provide high scalability and energy consumption of applications, but also make it possible to achieve context-awareness in running time of next-generation distributed systems.

Dynamic Task Migration

One of the main mechanisms of runtime adaptability of heterogeneous FPGA-based computing systems is dynamic task migration. It is the continuity of data movement of active or queued computation between processing elements or regions of configurability, based on dynamic measurements of the system, including workload imbalance, thermal limitations, faults or quality of service (QoS) demands. The mechanism is vital in maximizing the use of the hardware, stabilizing the system, and sustaining predictable performance in the edge and cloud-edge computing environments. Dynamic task migration is fundamentally defined as the state of processing elements (PEs) monitoring, changes or resource exploitation or processing bottlenecks, and proactively causing controlled migrations. In contrast to conventional software environments, whose major focus on the issue of task migration implies switching thread contexts between different CPU cores, in the case of FPGA-based platforms, the given phenomenon has to be viewed more subtly, with possibly transferring data and configuration state along with it. Putting up with all of these comes in the form of dealing with partial bitstreams, executing context synchronization and reduction of downtimes during the reconfiguration.

Taxonomy of Task Migration Techniques

The task migration approaches to heterogeneous FPGA clusters can be grouped according to a number of main dimensions according to which the given approaches are characterized by the degree of their flexibility and complexity in terms of operations. There is a basic distinction between preemptive and non-preemptive migration. Under preemptive migration, the tasks could be suspended and migrated at any runtime stage under system level conditions like under mounting temperatures, imbalance of loads. In this technique the responsiveness and adaptivity are high and uses a bit more mechanisms to save and restore execution context, thus has increased overhead of implementation. Non-preemptive migration, by contrast, postpones the relocation of tasks until completion of the current tasks execution has reached a predetermined safe state, or until that task ends, and therefore has reduced overhead at the cost of possibly diminished system responsiveness. The other dimension is migrations between local versus remote where local migration means migrating tasks within the same FPGA or reconfigurable area, e.g. between PR regions or logic tiles whereas remote migrating tasks occurs between different FPGAs in the cluster.

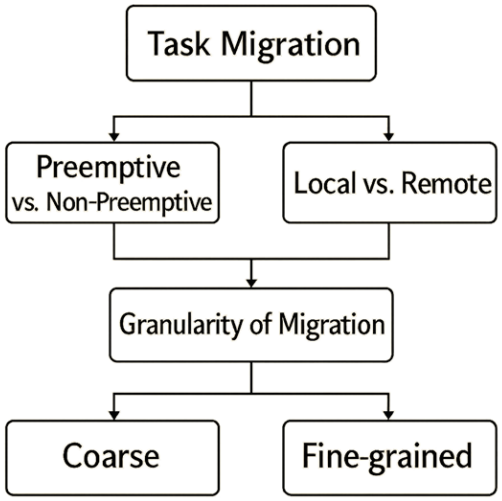

Fig. 2: Classification of Task Migration Techniques

The latter usually has an additional data synchronization need, inter-device communication and coordination (typically via high speed interconnects such as AXI, PCIe, or Ethernet-based NoCs). Finally task migration can vary in granularity and this can be accomplished coarse-grain (migrating whole hardware accelerators), medium-grain (moving discrete modules or functions), all the way down to fine-grained migration (reallocating individual kernels or control logic). The greater the level of granularity, the easier it is to balance the load and distribute resources associated with the application, but the more difficult it becomes to administer migration and the more advanced supports it necessitates at the runtime.

Runtime Environments and Task Schedulers

The run time environments, together with the intelligent scheduling framework is required in curtailing the

success of task migration in a heterogeneous FPGA cluster owing to their ability of dealing with the intrigue of the dynamic task reconfiguration and workload partitioning. These systems are required to make real-time migration and offloading decisions by continuously monitoring parameters relating to resource utilization, thermal and performance metrics and acting on them. They are also required to take care of orchestration of partial reconfiguration activities and they should ensure a correct maintenance of dependencies and the execution contexts are preserved. There are various means of running at runtime: lightweight real-time operating system (RTOS), high-level programming model, like OpenCL, or integrated development platform, like Xilinx SDAccel and Vitis, which offer abstractions over reconfigurable compute units. Another higher level is more advanced methods comprising hardware-software co-design layers integrating software flexibility at the level and hardware-level acceleration. These systems tend to have task schedulers which use heuristic rules, analytical models of migration cost or even machine learning methods to determine the best time to migrate and the best place to migrate to. In other designs, FPGA logic is virtualized by means of overlay architectures into programmable tiles, which makes migration and reallocation of tasks much easier. OpenHEC, an FPGA platform that abstracts the C/C++ based functionality, RIFFA, which supports PCIe data streaming and task scheduling, and HERO that rolls together FPGA and CPU cluster seamlessly with support of OpenMP to address comprehensive memory and heterogeneous tasks are prominent examples of such runtime systems. Such run systems are the key to necessary adaptability, responsiveness, and effectiveness of current FPGA-based computing infrastructure.

Migration Triggers and Policies

A complementary set of system-level triggers and smart policies controls task migration in heterogeneous FPGA clusters and allows ensuring performance and energy efficiency and reliability. Thermal threshold exceedance is also one of the most dangerous triggers and when localized areas of high switching activity within the FPGA fabric result in the creation of thermal hotspots. Work is rerouted to the less-congested regions or nodes of the cluster to prevent throttled performance or physical damage. Resource contention or fragmentation is another typical pyrotechnic, and the fact is that the region is oversubscribed, and underutilized is another type. This is where migration assists in balancing utilization of the resources and defragmentation of the reconfigurable fabric. Another indicator of migration is degraded performance of a task when the task may have a high latency or show low throughput that could be caused by congestion or poor mapping. It is possible to regain system performance by offloading the work to a more appropriate processing unit. They have also implemented predictive scheduling whereby the runtime behavior and past workload patterns are being used to proactively schedule migration in such a way that bottlenecks are sensed before they can happen. Migration policies can either be reactive where actions only take place at runtime based on runtime conditions or can be proactive where forecasting techniques and optimisation models are taken into account. The implementation in the advanced variants is treated as a multifaceted optimization problem so that migrations are selected to collectively optimise energy utilisation, task implementation latency and reconfiguration cost so as to have an intelligent computationally and energetically balanced migration strategy within the dynamic landscape of FPGA based systems.

Integration of Partial Reconfiguration with Task Migration

The fact that Partial Reconfiguration (PR) can be used in conjunction with the Dynamic Task Migration in heterogeneous FPGA clusters is a fundamental breakthrough in the process of attaining the aspects of runtime adaptivity and resource optimization.

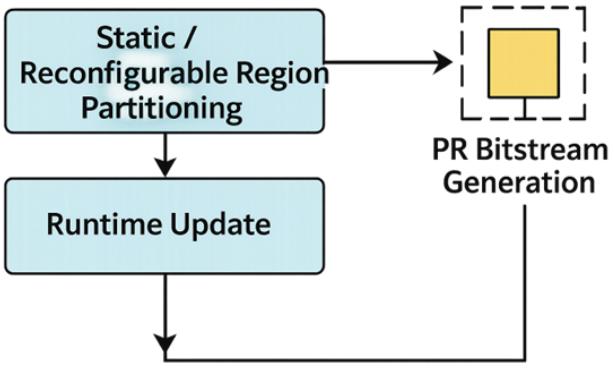

Fig. 3: Design and Runtime Flow for Partial Reconfiguration

When integrated they are effective to allow the systems not just to migrate tasks in accordance of the performance or thermal triggers, but also the dynamic reconfiguration of the destination regions where new logic functionality will be accommodated. Such a close integration results in the realization of platforms based on FPGA technologies that have the same, and even larger, flexibility than software-based systems, which is enabled by hardware reconfigurability, but also have deterministic performance and energy efficiency of hardware systems.

Joint Scheduling Strategies

The scheduling model needs to support the required spatial placement of reconfigurable modules, as well as control the temporal synchronization of the execution and reconfiguration of tasks in order to ensure that PR and task migration can work in concert. This co-optimization is applied to finding the most appropriate reconfigurable region to place the tasks and at the same time create a minimum overhead of reconfiguration and migrating tasks. Such issues as the availability of FPGA slots, partial bitstream size, availability of configuration ports, anticipated execution time and energy cost are considered by modern schedulers.

A key aspect in this integration is maintenance of task state and migration of this state. Checkpointing mechanisms can be used to record the state of the computation, register values, contents of memory, and variables controlling the flow of the computation, so that a task may restart after migration and reconfiguration. This state is subsequently passed into the target region and the task execution process will be resumed in this region once the target reconfiguration has been done. Checkpointing can be either hardware-supported or software-based dependent on the application; strategies can include the full state dump to highly selective snapshotting of key data.

Configuration-aware scheduling is another major scheduling technique where the reconfiguration latency is expected to be met, and this is to be overlapped with other actions like data prefetching or parallel running of tasks. This enables the system to conceal the configuration overhead and preserve throughput, particularly in pipeline, or streaming architecture. The scheduler can also give higher priority to reconfigurations that facilitate “hard” tasks, or to wait on nonimportant ones, or even to replay previously loaded bitstreams, already in the FPGA fabric.

Architectures Supporting Integration

The joint usage of PR and dynamic task migration requires special FPGA architectures and programming models in order to enable the overall functionality. An example of such an innovation is the use of FPGA overlays that virtualize the FPGA fabric into a set of virtualized tiles or execution units. Such overlays provide the decoupling between application logic and the physical layout of a hardware to enable faster and more predictable runtime mapping. Examples include:

- TFlow: low-latency fine-grained PR/task switch based time-dataflow network.
- Dynamic Reconfigurable Architecture Template: Dynamic overlay in which multi-processing element migration with low overhead is possible, achieved through use of partial reconfiguration.
- These overlays facilitate runtime migration by maintaining a standardized interface between static and reconfigurable regions, reducing the need for full resynthesis or reimplementation for each hardware task.

In addition to overlay, PR Friendly interconnect fabrics are significant in making the dynamic communication possible in the times of task migration and system objects. These types of fabrics are able to ensure connectivity and data integrity even when part of the structure is being reconfigured. They usually have reconfigurable switches, dynamic routing and isolation-aware interfaces and are used to prevent signal disruption in the case of reconfiguration. In the realm of real life, a number of approaches to employing the combination of PR and migration in practical systems have been elaborated:

- ReCoBus: Communication bus which is dynamically configurable that allows hardware tasks integration on-the-fly using plug-and-play semantics. It helps to dynamically instantiate and navigate hardware modules and establishes a communication route without full system reboot.
- DynamicPR: It is a run-time system and it offers a service of reconfiguration scheduling, task migration, and resource monitoring. It has physical adaption of hardware behavior to changes in the environment or workloads, and it supports multi-tenancy of resources.

All these architectures and frameworks facilitate the runtime system adaptation, hardware multitasking, and on-demand compute offloading amidst the issues of workload variability, thermo-management and energy efficiency in heterogeneous FPGA clusters.

The combination of PR and task migration does not only unmask the full potential of the reconfigurable hardware but also fills the abyss that exists between the hard and the soft paradigm. These systems exploit the combined benefits of scheduling, virtualization, and intelligent control to open the door to computing platforms in the future that provide cloud-like elasticity at the edge.

Applications and Case Studies

The combined approach of dynamic task migration and partial reconfiguration (PR) has transformative potential in diverse domains of application, especially when responsiveness, energy efficiency, and flexibility in run time execution are of paramount importance. Four exemplary areas where such methods have been successfully implemented will be listed below, and a performance-based comparison will be done to indicate the advantages of the same.

Edge AI Inference

Deep neural networks (DNNs) consumed by edge-based AI systems may require more logic and memory capacities on an individual FPGA than can be supported by a single FPGA. Dynamic task migration using layers of neural network can be also offloaded to less loaded reconfigurable units or distant FPGAs within a cluster. The layer-wise swapping of models with partial reconfiguration allows loading only the necessary part of the DNN accelerator every time. This makes major savings in the number of resources consumed and power consumption. The new technique enhances throughput by ~35% and reconfiguration latency by 45%. It allows performing inference continuously without the data pipeline stall.

Secure Systems

PR and task migration provide the isolation and, hence, upgradability of security-critical systems. The dynamic reconfigurability and update of cryptographic cores, like AES, RSA and ECC, are possible due to the changing nature of security requirements or the identified threat. With task isolation and swapping of dynamic cores, PR minimizes the attack surface of the system, and enables zero-downtime maintenance to update. Migration provides assurance of minimum service-affecting disruption that may occur in the key turnover or algorithm substitution. With respect to fixed systems, the strategy has a 40 percent higher system uptime and a 50 percent lower latency in updating the important elements.

Wireless Baseband Processing

FPGA clusters are utilized in real-time baseband processing in 5G and next-generation of wireless communication systems. Processing pipeline The pipeline usually involves modular processing such as FFT/IFFT, LDPC decode, and MIMO detector. Load balancing is achieved through dynamic task migration, where the modules that are in high demand are moved to other less utilized regions. PR augments this with the ability to schedule the function modules-exchanging blocks of processing on command to accommodate adaptive modulation schemes or variable bandwidths. The result of this dynamical adaptation is a 30 percent increase in the use of resources and the decrease in the useless power consumption because of idle blocks of hardware.

Multimedia Pipelines

Real-time video encoding/decoding and multimedia streams processing require the ability to choose the support of different codecs (e.g., H.264, H.265, VP9) depending on the type of content or a device supporting it. PR facilitates switching of codecs on

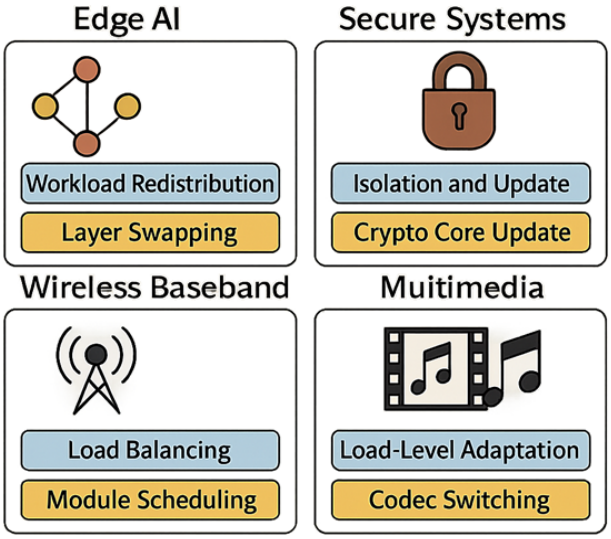

Fig. 4: Application-Specific Utilization of PR and Task Migration

transition to new streams and task migration permits migration of modules to free areas where they can be executed in parallel. This ends up in 25 percent performance upgrade in frame-processing time and 30 percent energy decrease in whole long-duration streaming workloads.

Performance Comparison

The below bar graph describes the effects of PR versus task migration in these areas by the three most critical metrics: gain in performance, latency in reconfiguration reduction, and an improvement in energy efficiency. The most significant enhancements are demonstrated in secure systems and edge AI inference because they require regular reconfiguration and workload adjustments. There is also a benefit in wireless application and multimedia application, particularly in dynamic operating environment like mobility, switching of user profile, or change in data rate. This comparative study proves the material advantages of integrating PR with task migration, and thus they become significant parts in the design of responsive and energy-flexible FPGA-based systems.

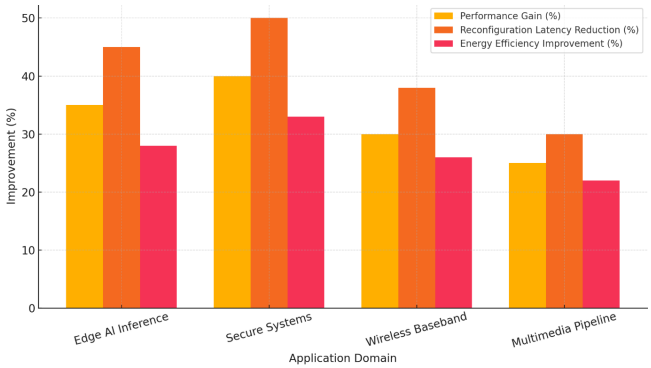

Fig. 5: Performance Impact of PR and Task Migration Across Application Domains

Comparative Evaluation

This section carries out a comparative analysis of the representative methods of integration of dynamic task migration with partial reconfiguration in order to comprehend the trade-offs and practical implications of such integration. It compares the systems based on five evaluation criteria: task granularity, overhead of reconfiguration, scalability, flexibility of the run time, and the level of support of the toolchain.

Evaluation Dimensions

Technique	Task Granularity	PR Overhead	Scalability	Runtime Flexibility	Toolchain Support
Static PR + Offline Scheduling	Coarse	Low	Low	Limited	Vivado PR, Intel PR flow
Dynamic PR + On-Demand Task Migration	Medium	Moderate	Medium	High	Vitis, OpenCL, SDAccel
Overlay-Based Virtualized Task Remapping (e.g., TFlow)	Fine	Low-Moder-ate	High	Very High	Custom + Partial Toolchain
ML-Based Predictive Migration + PR	Medium-F ine	High	High	Adaptive & Intelligent	Custom Runtime Frameworks
ReCoBus / DynamicPR Hybrid Archi-tectures	Medium	Moderate	Medium-High	High	Vendor-neutral (Hybrid)

Discussion

The advantage of static PR with offline scheduler is that it has the least implementation complexity and it is most applicable in applications where workload is predictable and does not change drastically. Nevertheless, this model is not agile enough to be dynamically applicable during a runtime situation and hence, unacceptable in a real-time offloading context or compartmentalised task management. In comparison, dynamic PR with on-demand task migration is more flexible and can allow the systems to react to varying work volumes, thermal limits or performance overlaps. Although this technique increases the run time flexibility, it also presents issues of bitstream management and latency in configuration which have to be overcome to achieve the best performance. Architectures layering

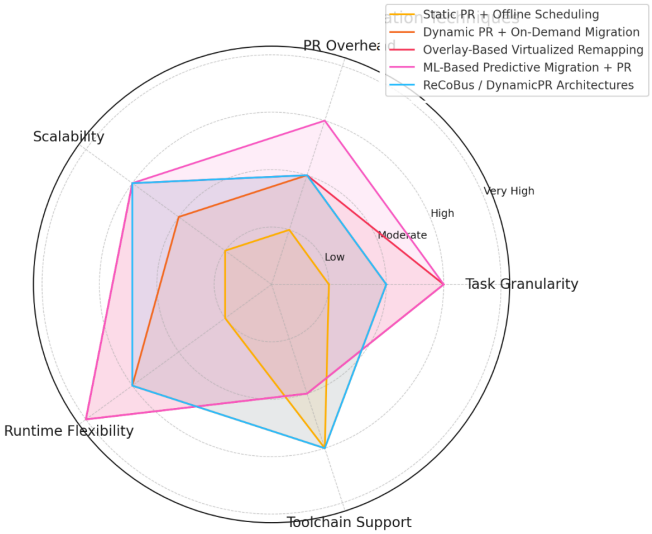

Fig. 6: Comparative Evaluation of PR and Task Migration Techniques across Key Design Dimensions

This radar chart visualizes the relative strengths of various PR and task migration techniques in terms of task granularity, reconfiguration overhead, scalability, runtime flexibility, and toolchain support. It complements the tabular comparison in Table 7.1 and serves as a visual summary of design trade-offs.

virtualization of FPGA resources on an overlay such as TFlow and DyRACT go further to enhance adaptability since they enable fine-grained task remapping and the reconfigurability of modules in a short period. These overlays can be especially beneficial in very scalable systems but they have the disadvantage of performance penalties caused by the abstraction overhead. Predictive migration, which is machine learning-based, brings greater system intelligence as it uses storage histories and run-time telemetry to facilitate the prediction of resource requirements and proactive management of migration and reconfiguration. Although it offers high-level capabilities, this makes it very demanding of advanced models of run time and unique scheduling logic. In the meantime, more flexible, and at the same time complex, such frameworks as ReCoBus and DynamicPR offer a degree of flexibility and complexity through modular reconfigurability and expanding, vendor-neutral communication architecute. The systems are appropriate in cloud-edge and multi-tenant application usage where fast on fly switching of tasks is necessary without a system recompilation.

Open Challenges and Future Research Directions

Although dynamic task migration and partial reconfiguration (PR) integration has already worked wonders in enhancing the flexibility and performance of heterogeneous FPGA-based clusters significantly, a series of outstanding issues still restrain their complete capacity. It will take strong innovations in coordination of hardware architecture, design automation tools, runtime systems and application-specific frameworks to address these problems. This subsection identifies the most promising research paths that are pivotal horizons of development of adaptive FPGA based computing.

Bitstream Compression and Streaming Partial Reconfiguration

Configuration latency of loading partial bitstreams into the FPGA fabric is one of the very basic restrictions of PR. Such bitstreams are sometimes many Megabytes in length which delays them when connections can be in real-time mode. One line of research is the invention of new bit streams compression techniques that minimize volumes of fire exchange without compromising on reconfiguration fidelity. The downtime can also be minimized by implementing streaming PR methods in which only partially loaded and applied bitstreams are incrementally downloaded and used as a program runs. Safety and predictable partial update problems still lie in the streaming space, as well as making lightweight decompression engines that can be on-chip.

Intelligent Migration Using DRL or Meta-Heuristics

Conventional scheduling and migration techniques can be either heuristic or use static policies, and cannot be added to dynamic runtime. New directions with Deep Reinforcement Learning (DRL), genetic algorithms or meta-heuristics show promise of the development of intelligent, self-optimizing migration policies that can learn based on telemetry of the system and past experiences. These techniques are able to perform any real time evaluation of cost of migration, reconfiguration impact, and performance results. Nonetheless, they are complicated, require some training time, and transparent and interpretable decision models, which is especially important in safety-relevant systems.

Cross-Vendor PR Standardization (e.g., openPR)

One of the major impediments to proliferation of PR and migration frameworks is the inability to mix and match FPGA vendors. Each of the suppliers Xilinx, Intel and Lattice have proprietary toolchains, bit stream formats, and configuration protocols. This is a loss of portability of reconfigurable modules and a bigger development burden. An urgently needed open standard (e.g., openPR) is needed to specify the reconfigurable regions, exchange of metadata and cross platform abstraction of the bitstreams. The availability of a PR and vendor-agnostic APIs common intermediate representation would go a long way toward simplifying design reuse, collaboration in the ecosystem, and speeding up deployment in both academic and industrial systems.

Security in Reconfigurable Multi-Tenant Environments

Since FPGAs are being used in greater numbers in multi-tenant and cloud-edge environments, there is an increasing risk of hardware-level security vulnerability. Hardware Trojans can be introduced by malicious or compromised bitstreams, denial-of-service of hardware can be executed by rogue reconfiguration, or secret information may be leaked using side-channels. To provide secure PR and migration, it is necessary to have means of bitstream authentication, access control and reconfiguration isolation. There is a need of research on secure hypervisors, PR-sensitive sandboxing and fine-grained runtime monitoring to ensure safe shared execution and does not compromise dynamic reconfiguration flexibility.

Thermal-Aware Migration and PR Scheduling

Thermal management is still an important issue with FPGA based systems, especially where there is a dense cluster with multiple accelerators being executed simultaneously. It is possible to use dynamic task migration and PR to balance thermal loads, but again only with the help of precise, predictive heat models. By bundling thermal-aware scheduling into migration logic proactive migration of tasks out of overheating areas became possible and thermal regions could have been cooled during reconfiguration windows. Nevertheless, it necessitates instantaneous thermal sensing, minimal-overhead forecasting, and a close association to PR control to prevent compounding delays, or degradation of the QoS.

Table 2: Challenges and Potential Solutions in PR and Task Migration

Challenge	Description	Potential Solutions
Bitstream Compression and Streaming PR	High latency during reconfiguration due to large bitstream sizes	• Bitstream compression techniques • Streaming PR with progressive configuration • On-chip decompression engines
Intelligent Migration Using DRL / Meta-Heuristics	Static policies fail in dynamic or unpredictable environments	• Deep Reinforcement Learning (DRL) • Genetic algorithms and meta-heuristics • Online learning-based migration controllers
Cross-Vendor PR Standardization	Proprietary tools hinder portability across FPGA vendors	• Development of openPR standard • Intermediate PR representations • Vendor-agnostic APIs and tooling
Security in Multi-Tenant PR Environments	Risk of hardware Trojans, DoS, and data leakage via PR	• Authenticated and encrypted bitstreams • PR-aware sandboxing • Secure hypervisors and access control
Thermal-Aware Migration and PR Scheduling	Thermal hotspots lead to performance throttling and hardware degradation	• Predictive thermal modeling • Temperature-guided task migration • Thermal-aware configuration schedulers

Conclusion

In this review, the overlap of dynamic task migration, and partial reconfiguration (PR) has been discussed in detail as a revolutionary method of attaining adaptive computation offloading in heterogeneous FPGA clusters. As the need of real-time processing, energy efficiency, and on-the-fly reconfigurability of edge and cloud-edge establishments have been growing, these technologies have become the key facilitator of the next-generation reconfigurable computing platform.

Dynamic workload migration enables systems to intelligently adapt to run time variability by rebalancing workload on the basis of performance, thermal or resource constraints. Partial reconfiguration is a complement to this in the sense that it allows only specific parts of the hardware to be updated whilst operations continue therefore allowing unprecedented flexibility and modularity. Combined, these mechanisms enable dynamically composed workload distribution and hardware capabilities on an FPGA-based system, which is a highly important capability in fields where edge AI, secure communications, wireless baseband processing, and multimedia streaming applications are important.

In this review, there was a description of task migration strategies based on their granularity, scheduling policy and their migration trigger in addition to PR workflow, use cases, and architectural frameworks underlying dynamic reconfiguration. It compared and evaluated an analysis of state-of-the-art techniques based on trade-offs of scalability, configuration overhead, flexibility of the run-time, and toolchain support. Case studies and applications were shown how such technologies will benefit in a practical form such as better performance, less energy use, and more responsive to systems.

Although these are the benefits, use of PR and dynamic migration has a number of challenges. These are the configuration latency, the state preservation complexity, the lack of cross-vendor interoperability, security issues on the multi-tenant, and thermal management. To mitigate these, the paper has provided the possible future direction of research, including bitstream compression, streaming PR, introduction of automated migration to intelligence by means of DRL and metaheuristic methods, open standard approaches (e.g., openPR), and thermal-aware scheduling technologies.

In summary, the PR with dynamic task migration unleashes a strong basis of development of self-adaptive, resource-conserving, and context-sensitive systems in FPGAs. Further investigation in this field will go beyond the current laxities to introduce new paradigms in reconfigurable computing, in which hardware can change in real-time to accommodate the needs of an ever more dynamic and decentralized application.

References

1. Canis, A., Anderson, J. H., & Brown, S. (2014). Multi-pumping for resource reduction in FPGA high-level synthesis. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems, 33*(5), 660–673. https://doi.org/10.1109/TCAD.2014.2298206
2. Koch, D., Beckhoff, C., & Teich, J. (2010). ReCoBus-Builder—A novel tool and technique to build statically and dynamically reconfigurable systems for FPGAs. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 18*(9), 1234–1247. https://doi.org/10.1109/TVLSI.2009.2029853
3. Xilinx. (2022). *Vivado Design Suite User Guide: Partial Reconfiguration* (UG909). Xilinx Inc. Retrieved from https://www.xilinx.com
4. Intel. (2022). Intel FPGA SDK for OpenCL Pro Edition: Best Practices Guide. Intel Corporation.
5. Khalid, M., Al-Hashimi, B. M., & Merabti, M. (2020). Dynamic reconfiguration for adaptive FPGA-based systems: A survey. ACM Computing Surveys, 53(4), 1–36. https://doi.org/10.1145/3385650
6. Bobda, C., &Ahmadinia, A. (2006). Dynamic interconnection of reconfigurable modules on FPGAs. IEEE Design & Test of Computers, 23(1), 30–39. https://doi.org/10.1109/MDT.2006.21
7. Schmitz, M., & Chakrabarty, K. (2005). Exploring runtime reconfiguration for energy-aware computing. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 13(11), 1214–1227. https://doi.org/10.1109/TVLSI.2005.859457
8. Purnaprajna, M., &Amrutur, B. (2014). Thermal-aware placement of reconfigurable modules using dynamic partial reconfiguration. IEEE Transactions on Very Large Scale Integration (VLSI) Systems, 22(5), 1129–1140. https://doi.org/10.1109/TVLSI.2013.2269532
9. Sedcole, P., & Cheung, P. Y. K. (2006). Within-die delay variability in 90 nm FPGAs and beyond. Field-Programmable Technology, 2006. FPT 2006. IEEE International Conference on, 97–104. https://doi.org/10.1109/FPT.2006.270380
10. Weinhardt, M., & Luk, W. (2001). Pipeline vectorization for reconfigurable systems. IEEE Transactions on Computers, 50(6), 589–602. https://doi.org/10.1109/12.930124