A Survey of FPGA Floorplanning for Dynamic Partial Reconfiguration: From Heuristic Approaches to Autonomous AI-Driven Methods

Ibrahim LIMEM, Sadok BAZINE, Abdessalem BEN ABDELALI
Laboratory of Electronics and Microelectronics-Faculty of Sciences of Monastir, University of Monastir, Tunisia

***Abstract*—Dynamic Partial Reconfiguration (DPR) has emerged as a key enabler of runtime adaptability and hardware-software co-design in modern FPGA-based heterogeneous systems. However, with the transition toward 5nm technologies and multi-die 3D-IC architectures, spatial resource management faces a "complexity wall," where traditional manual floorplanning techniques struggle to satisfy timing, utilization, and scalability constraints. This study presents a systematic literature review and proposes a comprehensive taxonomy of FPGA floorplanning and placement methodologies developed over the past two decades. The proposed classification organizes existing approaches into three generations: 1) the Heuristic Era, focused on rule-based automation and physical feasibility; 2) the Optimization Era, characterized by formal mathematical models and Mixed-Integer Linear Programming (MILP) for heterogeneous resource allocation; and 3) the Autonomous Era, which leverages AI-driven techniques, including Reinforcement Learning and intelligent scheduling, to enable predictive and shape-adaptive placement strategies. This evolution reflects a fundamental shift from static grid-based management toward elastic, self-optimizing FPGA fabrics. We further examine emerging architectural constraints, including Super Logic Region (SLR) boundaries and hierarchical nested Partial Reconfigurable Regions (PRRs). Beyond this taxonomy, the survey identifies a critical scalability–optimality trade-off, highlighting the need for hybrid frameworks that combine the formal guarantees of optimization-based methods with the real-time adaptability of AI-driven approaches. It further establishes a unifying perspective in which DPR is evolving from a logic-reconfiguration mechanism into a thermal–spatial management paradigm for mitigating heat in high-density 3D-IC systems. Finally, the analysis reveals a significant functional–physical gap in current autonomous design tools, emphasizing the need for context-aware agents capable of jointly reasoning about temporal task dependencies and spatial floorplanning constraints. This review provides a structured roadmap for the development of next-generation intelligent control frameworks for edge and cloud-scale reconfigurable computing systems.**

Keywords*—Field Programmable Gate Arrays (FPGA); Floorplanning, Dynamic Partial Reconfiguration (DPR); bitstream relocation; hardware autonomy; reinforcement learning; heterogeneous computing; 2D shape-adaptive placement

I. Introduction

Field-Programmable Gate Arrays (FPGAs) are versatile, reconfigurable hardware platforms widely used in high-performance computing, embedded systems, and adaptive applications [1], [2], [3]. Unlike dedicated hardware (ASICs), FPGAs can be reprogrammed post-fabrication, enabling rapid prototyping, flexible computing, and efficient resource utilization.

Dynamic Partial Reconfiguration (DPR) enhances FPGA flexibility by permitting selective regions of the fabric to be reconfigured at runtime without halting the static logic [4], [5]. This capability allows adaptive and resource-constrained systems to improve hardware utilization. It allows modules to be dynamically loaded or swapped at runtime. Nonetheless, DPR introduces additional design complexity, requiring careful spatial and temporal management to prevent resource fragmentation and minimize reconfiguration overhead.

Floorplanning and placement are the critical stages in the design flow that dictate the performance and efficiency of DPR-enabled systems. While floorplanning defines the coarse-grained allocation of Static Regions (SRs) and Partially Reconfigurable Regions (PRRs), placement determines the precise coordinates of logic elements to minimize wirelength and timing delays [6], [7]. In modern heterogeneous FPGAs, these tasks are complicated by the non-uniform distribution of CLBs, BRAMs, and DSP blocks, alongside the emerging physical boundaries of Super Logic Regions (SLRs) in 3D-IC architectures.

To address these challenges, configuration and relocation have gained attention, enabling pre-generated partial bitstreams to be applied to multiple compatible regions. Relocation-aware floorplanning and placement strategies are essential to reduce bitstream storage, improve fabric utilization, and enhance system flexibility. However, traditional manual and heuristic-based flows are increasingly inadequate for the "Complexity Wall" of modern high-density devices, driving the need for more autonomous and optimization-driven methodologies.

II. Background

A. Overview of FPGA Architecture

FPGAs are fundamentally composed of an array of configurable logic blocks (CLBs), digital signal processors (DSPs), block RAMs (BRAMs), and programmable interconnects [8]. Unlike application-specific integrated circuits (ASICs), FPGAs can be reprogrammed after manufacturing, making them suitable for rapid prototyping and adaptive computing and hardware process acceleration [2].

Contemporary FPGA architectures are characterized by a heterogeneous architecture that supports specialized resources, which are not limited to digital signal processing slices, high-speed transceivers, and embedded memory blocks, along with additional generalized logic functionality that serves a variety of computational tasks [1].

The non-uniform distribution of specialized resources complicates placement and floorplanning, necessitating sophisticated optimization techniques to optimize the effectiveness of these diverse resources [9].

B. Dynamic Partial Reconfiguration

Runtime Dynamic Partial Reconfiguration (DPR) represents one of the most powerful capabilities of modern FPGAs, enabling hardware modules to be reconfigured without interrupting the operation of the rest of the system [5]. Unlike static reconfiguration, where the entire device must be reprogrammed, DPR allows designers to modify only specific reconfigurable regions (RRs) while maintaining the functionality of the static region. This property makes FPGAs ideal for adaptive and resource-efficient embedded systems, where computational tasks evolve dynamically according to application requirements.

The implementation of DPR requires careful partitioning of the FPGA fabric into static and reconfigurable areas during the design phase [10]. As shown in Fig. 1, each Reconfigurable Partition (RP) hosts one or more Reconfigurable Modules (RMs) that can be swapped in and out during runtime [11]. This modularity improves hardware utilization and system scalability, allowing multiple functions such as signal processing [12], image filtering [13], or cryptographic operations [14] to share the same physical resources.

From a design perspective, DPR introduces new floorplanning and placement challenges. The designer must ensure that reconfigurable regions are physically isolated, their interfaces are aligned and consistent, and communication between static and dynamic regions remains stable during reconfiguration.

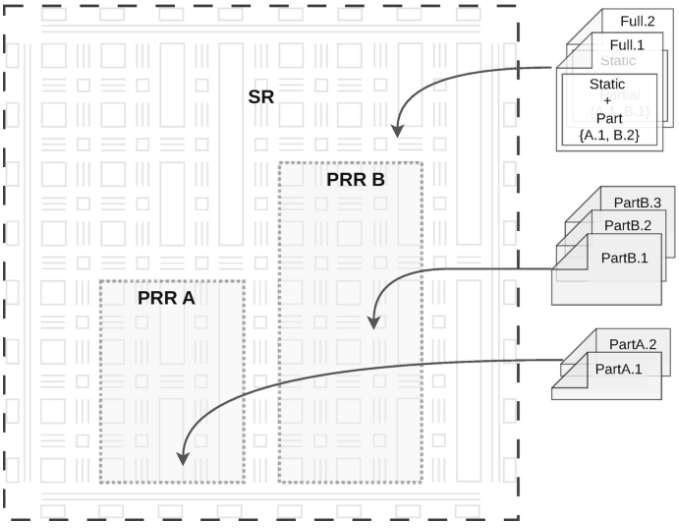

Fig. 1. Dynamic partial reconfiguration.

C. What is Floorplanning?

Floorplanning is a complex stage of the FPGA design flow. It consists of allocating specific physical regions of the FPGA's underlying architecture to each module or component that makes up the whole design [15] [16]. This critical phase is an important intermediate that connects the high-level synthesis stage with the subsequent detailed placement and routing processes.

The efficacy of floorplanning is paramount for realizing several significant design objectives, which include but are not limited to:

- Minimizing routing congestion [17].
- Reducing critical path delay [18].
- Optimizing power consumption [19].
- Support for partial and dynamic reconfiguration [20] [4].

For static floorplanning, the layout is strongly fixed and does not change during compile time, whereas partial reconfiguration [4] technology can facilitate dynamic alteration of requested portions of the FPGA fabric at run-time, therefore necessitating an even more meticulous approach to spatial and temporal planning to ensure optimal performance.

D. What is Placement?

Placement is a crucial stage in the FPGA design flow that follows floorplanning. It consists of assigning exact physical positions to logic elements such as Look-Up Tables (LUTs), flip-flops, Digital Signal Processing (DSP) blocks, and Block RAMs (BRAMs) within the physical regions previously defined during the floorplanning phase. The primary goal of placement is to arrange these components efficiently to minimize interconnect length, reduce routing congestion, and improve overall timing performance [6], [7].

An effective placement directly affects several critical design metrics, including delay, power consumption, and resource utilization. Proper positioning of interdependent components shortens signal paths, thus enhancing operating frequency and reducing dynamic power [21]. In the context of Dynamic Partial Reconfiguration (DPR), placement must additionally preserve region boundaries and ensure consistent communication interfaces between the static and reconfigurable regions of the FPGA [22], [23].

E. Interplay Between Floorplanning and Placement

Although floorplanning and placement are closely related (Fig. 2), they represent distinct stages in the design flow [6] [7] :

- Floorplanning defines the spatial organization of modules on the FPGA. It assigns specific regions where each module can be placed while satisfying design constraints.
- Placement, however, concerns the precise positioning and ordering of logical elements and instances, i.e., Look-Up Tables (LUTs), BRAM, ... into the predefined regions identified during the floorplanning process, thereby optimizing the order in terms of performance and efficiency based on the defined spatial rules.

Also of note is that floorplanning can indeed be considered an essential constraint, or guideline, of placement algorithms

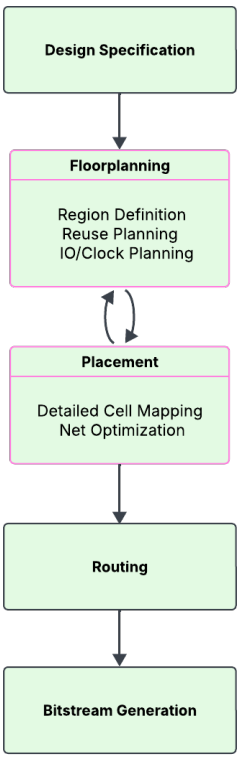

Fig. 2. Design flowchart.

used in the design process, and this first stage can play a major role in the potential impact on the placement algorithms, and ultimately on the achievement of design execution.

F. What is Relocation ?

1) Definition: Relocation is an advanced technique in Dynamic Partial Reconfiguration (DPR) that enables a single partial bitstream corresponding to a reconfigurable module to be loaded into multiple physical locations on an FPGA at runtime. This capability significantly enhances the flexibility and efficiency of reconfigurable systems by decoupling module functionality from fixed placement constraints.

2) Limitation of vendor flows: In conventional vendor-supported design flows, such as those provided by Xilinx, partial bitstreams are inherently location-dependent. Consequently, a distinct partial bitstream must be generated and stored for each potential physical location where a reconfigurable module may be deployed. This requirement introduces a major limitation in dynamically reconfigurable systems, as it results in substantial storage overhead and limits scalability, particularly in applications involving numerous tasks or reconfigurable regions.

To overcome this limitation, Fig. 3 illustrates that relocation relies on generating a single partial bitstream for an initial reference location and dynamically adapting it during system operation to target alternative physical regions on the FPGA. This approach enables the same hardware functionality to be instantiated at different locations without regenerating or storing multiple bitstream versions.

3) Benefits: As a result, relocation provides several key advantages for dynamically reconfigurable FPGA systems. From a memory efficiency perspective, it significantly reduces storage requirements by eliminating redundant location-specific bitstreams. Moreover, it enhances system flexibility by enabling dynamic placement and migration of hardware tasks across multiple Dynamically Partially Reconfigurable Regions (DPRRs). By overcoming inherent limitations of standard design tools, relocation supports more adaptive, scalable, and resource-efficient runtime reconfiguration strategies.

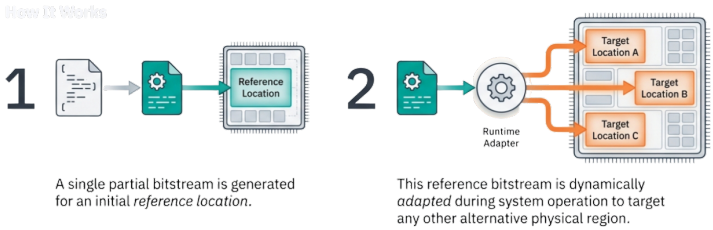

Fig. 3. Relocation technique.

G. Modern Commercial Architectural Constraints: Xilinx UltraScale+ and Intel Agilex

Advanced FPGA architectures targeting 5nm processes and multi-die integration introduce structural constraints that invalidate traditional 2D floorplanning assumptions. Modern devices, such as Xilinx UltraScale+ and Intel Agilex, require architecture-aware approaches, particularly for Dynamic Partial Reconfiguration (DPR).

1) Xilinx UltraScale+ and SLR constraints: Xilinx UltraScale+ devices, including Versal ACAPs, use Stacked Silicon Interconnect (SSI) to partition the fabric into Super Logic Regions (SLRs). Inter-SLR communication is limited by a small number of Super Long Lines (SLLs), leading to non-uniform routing costs. Consequently, PRRs are typically confined within a single SLR to avoid timing degradation, requiring SLR-aware placement strategies.

2) Intel agilex and 3D tile-based architectures: Intel Agilex employs a chiplet-based design using Embedded Multi-die Interconnect Bridge (EMIB) and 3D packaging. Resources such as transceivers and HBM are distributed across separate tiles, increasing heterogeneity and complicating PRR allocation for latency-sensitive applications.

These architectures also introduce thermal challenges due to localized heat generation in 3D stacks. Floorplanning must therefore consider thermal-aware placement and, in some cases, dynamic relocation of modules to mitigate hotspots and avoid performance throttling.

3) Implications for floorplanning: Overall, these constraints highlight the need for architecture-aware floorplanning methods that jointly consider interconnect cost, resource heterogeneity, and thermal behavior under runtime reconfiguration.

H. Floorplanning Metrics

The optimization of floorplanning and placement in Dynamic Partial Reconfiguration (DPR) systems is driven by a

set of quantitative metrics. These metrics aim to maximize spatial and temporal efficiency while ensuring compliance with system-level constraints. These metrics define the objective functions used to evaluate solution quality in reconfigurable FPGA architectures.

1) Timing and performance: Meeting timing constraints across all configurations is a primary objective. Floorplanning and placement decisions directly impact signal delay, critical path lengths, and achievable operating frequency. The goal is to minimize interconnect delay and improve spatial locality among timing-critical modules.

In DPR systems, each configuration must independently satisfy timing constraints, requiring configuration-aware performance evaluation. Therefore, optimization aims to maximize clock frequency while ensuring timing closure across all reconfiguration scenarios.

2) Area and resource utilization: Efficient utilization of FPGA resources is essential to support both static logic and multiple reconfigurable modules. Optimization aims to minimize resource fragmentation and ensure a balanced distribution of heterogeneous resources, including configurable logic blocks (CLBs), digital signal processors (DSPs), and block RAMs (BRAMs).

Proper sizing of reconfigurable partitions reduces unused capacity while preserving flexibility for module allocation. High area efficiency improves scalability and enables the integration of additional functionalities within the same device.

3) Communication and routing cost: Communication efficiency is a key determinant of overall system performance. The optimization process seeks to reduce inter-module communication distance, minimize total wirelength, and alleviate routing congestion.

Placement strategies that consider connectivity graphs and communication intensity improve routability and lead to more predictable delay behavior. Although routing occurs in later design stages, routing-aware floorplanning increases the likelihood of stable and efficient implementations across configurations.

4) Reconfiguration overhead: A key optimization objective in DPR systems is the reduction of reconfiguration overhead. This includes minimizing partial bitstream size, reconfiguration latency, and configuration bandwidth consumption. The geometric characteristics of reconfigurable partitions such as size, shape, and alignment with architectural frames directly affect reconfiguration efficiency.

Optimizing these parameters reduces reconfiguration downtime and improves runtime throughput.

5) Power and energy: Power-aware optimization encompasses both operational energy consumption and reconfiguration-related overhead. Placement decisions that reduce interconnect length and switching activity contribute to lower dynamic power consumption.

Additionally, reducing the size and frequency of reconfiguration operations lowers the energy cost of configuration memory access. Energy-efficient optimization improves system sustainability and thermal behavior.

The floorplanning problem in FPGA systems is inherently NP-hard. As design complexity increases, finding exact solutions becomes computationally infeasible. Consequently, a wide range of heuristic and metaheuristic approaches have been developed to provide near-optimal solutions within acceptable time constraints.

III. Systematic Literature Review Protocol

To ensure the rigor, completeness, and reproducibility of this survey, a systematic literature review (SLR) was conducted following a structured and well-defined protocol. This methodology was designed to identify, filter, and analyze relevant studies on FPGA floorplanning for Dynamic Partial Reconfiguration (DPR) over the past two decades.

A. Search Strategy and Data Sources

The literature search targeted primary studies published between 2006 and 2026 to capture the evolution of FPGA floorplanning methodologies. The following major scientific databases were systematically queried: IEEE Xplore Digital Library, ACM Digital Library, SpringerLink, ScienceDirect (Elsevier) and arXiv (to include recent developments in AI-driven approaches).

Search queries were constructed using Boolean operators to combine keywords related to FPGA architecture, floorplanning, and optimization techniques. The main query used was:

"FPGA" OR "Field-Programmable Gate Array") AND ("Floorplanning" OR "Placement") AND ("Dynamic Partial Reconfiguration" OR "DPR" OR "Bitstream Relocation") AND ("Optimization" OR "Machine Learning" OR "Reinforcement Learning"

B. Inclusion and Exclusion Criteria

To ensure relevance and quality, studies were selected based on clear inclusion and exclusion criteria. Only peer-reviewed papers addressing FPGA floorplanning or placement in dynamic partial reconfiguration (DPR) contexts, and providing experimental evaluation, were included. Works focusing on ASICs, static FPGA designs, or lacking sufficient technical contribution were excluded. The selection process involved removing duplicates, screening titles and abstracts, and performing full-text analysis, resulting in a refined set of studies for detailed review.

C. Data Extraction and Taxonomy Mapping

For each selected study, key information was extracted, including methodology type, optimization objectives, computational complexity, and evaluation metrics. The studies were then categorized according to the proposed three-generation taxonomy: Heuristic, Optimization-based, and Autonomous approaches.

Each method was analyzed using a unified set of performance criteria, including:

- Timing and performance
- Area and resource utilization
- Communication and routing cost

- Reconfiguration overhead
- Power and energy consumption

This structured analysis enabled consistent comparison across methodologies and supported the identification of research trends and open challenges.

IV. Evolution of FPGA Floorplanning Methodologies: A Taxonomy and State-of-the-Art Review

The evolution of floorplanning methodologies for Dynamic Partial Reconfiguration (DPR) reflects the increasing complexity of modern FPGA architectures and the growing demand for runtime adaptability. Over the past two decades, approaches have progressed from manual and heuristic techniques to formal optimization models and, more recently, to AI-driven autonomous systems.

In this work, we propose a structured taxonomy that categorizes these methodologies into three distinct generations based on their level of algorithmic intelligence, hardware awareness, and adaptability. This classification (Table I) provides a unified framework for understanding the progression of the field and identifying emerging research trends.

While early work focused on the physical legality of bitstream alignment (Era 1), middle-period research introduced formal mathematical models to handle heterogeneous resources (Era 2). Currently, the field is transitioning toward autonomous, AI-augmented systems that prioritize predictive management and elastic resource allocation (Era 3).

This taxonomy highlights a clear shift from static, constraint-based design toward dynamic, intelligent, and adaptive floorplanning strategies.

The following subsections describe each generation in detail, highlighting their key methodologies, innovations, and limitations.

A. Era 1: Heuristic and Rule-Based Foundations

This era is primarily characterized by constraint-driven design and limited automation, focusing on physical feasibility rather than optimality.

1) Hierarchical PBlock methodology: Lysaght et al. [22] introduced structured, constraint-driven partitioning using *PlanAhead*. PRRs were created within fixed physical blocks (PBlocks), allowing early estimation of resource usage, timing analysis, and routing isolation. Static regions were implemented first, followed by PRRs containing Reconfigurable Modules (RMs) independently.

2) Automation of resource alignment: OpenPR [24] and GOAHEAD [23] automated AREA GROUP constraints, bus macro placement, and routing alignment. Clock and frame alignment ensured PRR relocatability and predictable timing, simplifying subsequent integration with partial bitstreams.

3) Slot-based and tile abstractions: ReCoBus-Builder [25] introduced fine-grained resource slots enabling modular relocation, while CoPR [26] adopted tile-based abstractions of CLBs, BRAMs, and DSPs to create precise rectangular PRR layouts, facilitating efficient routing and minimal fragmentation.

4) Low-level logic exploration: RapidSmith 2 [27] exposed BEL-level FPGA structures to CAD tools, enabling placement, packing, and routing optimizations while retaining full compatibility with vendor bitstream generation.

This era focused on legal PRR creation, interface consistency, and minimal human intervention, but largely relied on heuristics, fixed templates, and rule-based strategies without formal guarantees on optimality.

This lack of optimality and inability to handle increasing resource heterogeneity necessitated the shift toward the formal mathematical models characterizing Era 2

B. Era 2: Formal Optimization and Analytical Modeling

This era introduces formal optimization techniques, enabling systematic handling of heterogeneous resources and improving solution quality. As FPGA fabrics became heterogeneous, floorplanning evolved toward mathematical modeling, Mixed Integer Linear Programming (MILP), and nonlinear optimization to achieve predictable and high-quality layouts.

1) Nonlinear and integer programming: PRFloor [28] combined heuristic and NLP optimization to reduce inter-module communication cost while respecting heterogeneous resources. DUPRFloor [29] applied MILP formulations for L-shaped PRRs, ensuring no overlaps and optimized interconnect usage. Deak et al. [30] used Columnar Kernel Tessellation and NLIP to align rectangular PRRs to FPGA frames. FLORA [31] treated FPGA resources as weighted grids to minimize wirelength and routing congestion, considering both logic and memory utilization.

2) Model-driven task mapping: Ben Abdelali et al. [32] developed a two-level L1/L2 scheduling framework. L1 analyzed functional dependencies and reconfiguration transitions between tasks, while L2 scheduled bitstream loading and task execution, optimizing runtime overhead and reducing reconfiguration latency.

3) Metaheuristic and joint optimization: Sadeghi et al. [33] and Ding et al. [34] used Genetic Algorithms and Simulated Annealing (Partitioned Sequence Triple, P-ST) to jointly optimize PRR placement, communication, and heterogeneous resource usage, improving layout quality for complex multi-module workloads.

4) High-level synthesis integration: HiPR [35] linked HLS with PR workflows, enabling designers to define partial functions in C/C++ while automatically generating legal PRR placements. HiPlanner produced optimized floorplans in under a second, minimizing wirelength and ensuring PRR alignment with frames.

5) Heterogeneous resource awareness: The PR FP tooll [36] employs a scarcity-prioritized iterative refinement approach using simulated annealing to handle irregular fabric structures, balancing logic, DSP, and memory usage across multiple PRRs.

Era 2 represents the transition from heuristic approaches to formal, resource-aware optimization, providing precise PRR layouts, efficient inter-module communication, and systematic management of heterogeneous FPGA fabrics, forming the foundation for runtime-adaptive reconfiguration.

TABLE I. TAXONOMY OF FPGA FLOORPLANNING METHODOLOGIES ACROSS GENERATIONS (2006–2026)

Generation	Timeline	Core Methodology	Relocation Support	Primary Focus
Era 1	2006–2015	Heuristic / Rule-based	1D Columnar / Manual	Physical Legality
Era 2	2016–2022	Formal Optimization	Block-Level / MILP	Timing & Congestion
Era 3	2022–2026	AI-Augmented / RL	**2D Shape-Adaptive**	**Autonomous Autonomy**

C. Era 3: Autonomous and AI-Augmented Systems

Era 3 marks a shift toward self-optimizing FPGA systems, where AI techniques are used to address the growing complexity of advanced nodes (e.g., 5nm) and heterogeneous 3D-IC architectures. In contrast to earlier static or design-time approaches, this era integrates intelligence across both runtime adaptation and design-time optimization, structured around three complementary pillars.

1) Reinforcement Learning (RL) for runtime adaptation (Online): Reinforcement Learning is increasingly employed for online decision-making in dynamic environments. In this setting, the FPGA fabric is modeled as an environment in which an agent learns optimal placement or reconfiguration policies through interaction. Such approaches are particularly effective for thermal- and power-aware management, enabling real-time relocation of modules based on sensor feedback. For instance, Yeonjin & Kigarura [37] explore RL for autonomous edge intelligence, while Muralidharan [38] integrates embedded RL logic to trigger DPR events without disrupting ongoing static operations.

2) Evolutionary and metaheuristic search for DSE (offline): Evolutionary and metaheuristic methods have matured into powerful tools for autonomous Design Space Exploration (DSE). Building on earlier Genetic Algorithm approaches, these techniques address the NP-hard nature of floorplanning by exploring large solution spaces and converging toward near-optimal layouts[39]. Unlike RL-based methods, they are typically applied offline during compilation to generate optimized, shape-adaptive floorplans for complex and heterogeneous workloads such as CNN pipelines or signal processing chains.

3) Neural networks and generative AI (Hybrid): Recent advances incorporate neural models, including Graph Neural Networks (GNNs) and Large Language Models (LLMs), to further automate and accelerate the design process. GNNs are used for predictive analysis, such as estimating routing congestion or timing violations prior to finalizing a floorplan. In parallel, LLMs enable generative hardware design flows, translating high-level descriptions into HDL code with embedded placement constraints. For example, frameworks like ResBench [40] assess the quality of LLM-generated designs, while Boudjadar et al. [41] leverage AI-driven DSE for dynamic HW/SW partitioning between Processor Systems (PS) and Programmable Logic (PL) based on runtime requirements.

4) Architectural enablers: Hierarchical and nested PRRs: These algorithmic advances are supported by evolving FPGA architectures that enhance reconfiguration flexibility. In particular, hierarchical and nested Partial Reconfigurable Regions (PRRs) [42] replace traditional flat layouts, enabling finer-grained and independent reconfiguration within subregions (see Fig. 4). This structural elasticity reduces bitstream size, improves scalability, and reinforces the overall autonomy of the system.

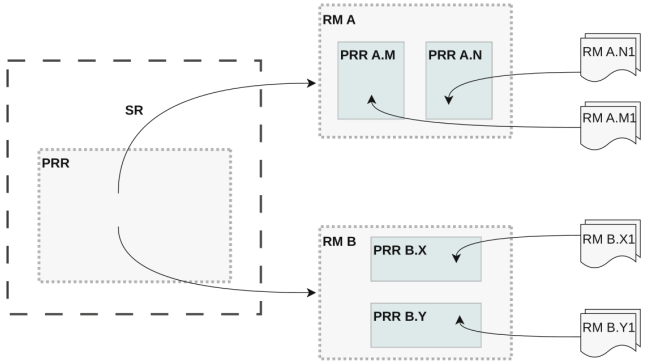

Fig. 4. Hierarchical nested PRR flow exhibiting parent-child reconfiguration layers [42].

The systematic evolution across the three eras is summarized in Table II. A closer examination of the *Major Innovation* and *Primary Metrics* columns highlights a clear progression in the maturity of the field. Era 1 established the practical feasibility of FPGA floorplanning through rule-based and one-dimensional bitstream relocation techniques. Era 2 advanced the state of the art by introducing formal optimization models to address resource heterogeneity and enforce design constraints. Most notably, Era 3 reflects a shift toward spatio-temporal autonomy, where the focus moves from static objectives such as timing closure to dynamic, adaptive metrics driven by runtime considerations, including thermal and power management enabled by AI-based methods.

Overall, the evolution of floorplanning methodologies demonstrates a progressive transition from manual and heuristic approaches to formal optimization and, ultimately, to AI-driven autonomous systems. While each generation addresses specific limitations of its predecessors, emerging challenges such as scalability, heterogeneity, and runtime adaptability continue to drive innovation in this field.

V. CRITICAL ANALYSIS AND RESEARCH GAPS

Despite significant progress across the three generations of FPGA floorplanning methodologies, several fundamental limitations remain unresolved. While heuristic approaches ensure feasibility, optimization-based methods improve solution quality, and AI-driven techniques enhance adaptability, none of these paradigms fully address the combined challenges of scalability, heterogeneity, and runtime dynamism.

This section provides a critical analysis of these limitations and identifies key research gaps that must be addressed to enable next-generation reconfigurable systems.

TABLE II. CHRONOLOGICAL EVOLUTION OF FPGA FLOORPLANNING METHODOLOGIES: FROM HEURISTIC RULES TO AI-AUGMENTED AUTONOMY

Year	Reference	Method / Complexity	Execution Context & Cost	Constraint Guarantees	Major Innovation	Primary Optimization Metrics
ERA 1: HEURISTIC AND RULE-BASED FOUNDATIONS						
2006	Lysaght et al. [22]	Heuristic	High (Offline)	Heuristic constraints	Hierarchical PBlocks (Manual)	Timing Isolation & Resource Est.
2008	ReCoBus-Builder [25]	Heuristic	Low (Offline)	Heuristic constraints	Slot-Based Abstraction 1D	Reconfiguration Overhead
2011	OpenPR [24]	Heuristic	Low (Offline)	Heuristic constraints	Automated AREA GROUP	Timing Closure
2012	GOAHEAD [23]	Heuristic	Low (Offline)	Heuristic constraints	Clock/Frame Alignment 1D	Manual constraint reduction
2015	RapidSmith 2 [27]	Heuristic	Low (Offline)	Heuristic constraints	BEL-level Exploration	Resource Density
ERA 2: FORMAL OPTIMIZATION AND ANALYTICAL MODELING						
2016	PRFloor [28]	NP-hard (NLP)	High (Offline)	Formal guarantees	Hybrid Heuristic/NLP	Communication Cost & Wirelength
2017	Ben Abdelali et al. [32]	Heuristic	Moderate (Offline)	Heuristic constraints	Two-Level L1/L2 Model	Reconfiguration Latency
2019	FLORA [31]	Heuristic	Low (Offline)	No formal guarantees	Weighted Resource Grid	Routing Congestion
2020	DUPRFloor [29]	NP-hard (MILP)	High (Offline)	Formal guarantees	L-shaped PRR Formulation	Overlap Prevention
2022	Ding et al. [34]	NP-hard (SA)	High (Offline)	No formal guarantees	P-ST Joint Optimization	Placement Success Rate
ERA 3: AUTONOMOUS AND AI-AUGMENTED SYSTEMS						
2022	Fahmy et al. [42]	Heuristic	Low (Runtime)	Heuristic constraints	Hierarchical/Nested PRRs	Fine-grained Adaptability
2025	Gu et al. [43]	Heuristic	Low (Runtime)	No formal guarantees	Big.Little Spatio-temporal Slots	Resource Utilization Efficiency
2025	Boudjadar et al. [41]	Heuristic (DSE)	Moderate (Offline)	No formal guarantees	HW/SW Co-Design (CNN)	Throughput & Scaling
2026	Yeonjin & Kigarura [37]	Polynomial	Near Real-time	No formal guarantees	RL-based Thermal Agents	Power & Thermal Management
2026	Muralidharan [38]	Polynomial	Real-time	No formal guarantees	Embedded AI Control Logic	Non-intrusive DPR triggering
2026	Li et al. [44]	Heuristic	Near Real-time	Heuristic constraints	FEditor (2D Shape-Adaptive)	Resource & Shape Flexibility

A. Scalability vs. Optimality Trade-off

A fundamental limitation across existing methodologies is the trade-off between scalability and optimality. Formal optimization techniques, such as MILP and nonlinear programming (Era 2), provide high-quality and constraint-compliant solutions. However, their computational complexity grows exponentially with design size, making them impractical for large-scale FPGA systems.

In contrast, AI-driven approaches (Era 3), particularly reinforcement learning, can explore large design spaces efficiently and produce solutions in near real-time. However, these methods lack guarantees of feasibility and constraint satisfaction, particularly with respect to strict FPGA architectural rules.

This dichotomy highlights the need for hybrid approaches that combine fast exploration capabilities of AI with the rigor of formal verification and constraint enforcement.

B. Fragmentation in Heterogeneous 3D-ICs

The increasing adoption of heterogeneous 2.5D and 3D FPGA architectures introduces new spatial and thermal challenges that are insufficiently addressed in current literature. Most existing approaches assume a 2D fabric and fail to account for Super Logic Region (SLR) boundaries, interposer constraints, and non-uniform communication costs.

Moreover, thermal effects are becoming a critical factor in high-density FPGA systems. Current floorplanning methods largely ignore thermal-aware placement and dynamic heat distribution, which can significantly impact system reliability and performance.

Future research must integrate spatial, thermal, and communication-aware optimization models to address these emerging challenges.

C. Model-Driven Adaptability

While model-driven approaches (Era 2) effectively capture task dependencies and reconfiguration sequences, most AI-driven methods (Era 3) optimize spatial metrics such as area and wirelength in isolation. This results in a disconnect between functional behavior and physical implementation.

In practical systems, reconfigurable modules exhibit temporal dependencies, where certain tasks are frequently co-located or executed sequentially. Ignoring these relationships leads to suboptimal runtime performance and increased reconfiguration overhead.

Future approaches should integrate task-level dependency models into AI-based floorplanning to enable more context-aware and application-driven optimization.

D. Lack of Standardized Benchmarks and Evaluation Frameworks

A major limitation in the field is the absence of standardized benchmarks and evaluation methodologies. Existing works often rely on custom datasets, proprietary designs, or tool-specific configurations, making it difficult to compare approaches objectively.

This lack of reproducibility hinders fair evaluation and slows progress in the field. Future research should focus on developing open benchmarks and unified evaluation metrics to enable consistent comparison across methodologies.

E. Limited Integration with Industrial CAD Flows

Despite significant academic advances, many proposed methodologies are not fully compatible with industrial FPGA design tools such as Vivado or Quartus. This gap limits their practical adoption and real-world impact.

Bridging the gap between research prototypes and industrial toolchains remains a key challenge for the field.

F. Future Research Directions

Based on the identified gaps, several promising research directions emerge:

- Hybrid AI-Optimization Frameworks: Combining reinforcement learning with formal constraint solvers to ensure both scalability and correctness.
- Thermal-Aware and 3D Floorplanning: Incorporating temperature modeling and SLR-aware placement strategies.
- Application-Aware Optimization: Integrating task dependencies and runtime behavior into floorplanning decisions.
- Standardized Benchmarks: Developing open datasets and evaluation frameworks for fair comparison.
- Toolchain Integration: Ensuring compatibility with industrial FPGA design environments.

These directions are expected to drive the next generation of intelligent and autonomous FPGA systems.

In summary, while significant progress has been achieved, FPGA floorplanning for DPR remains an open and evolving research area. Addressing these challenges will require interdisciplinary approaches that combine optimization theory, machine learning, and hardware-aware design methodologies.

VI. Conclusion

This study has presented a structured taxonomy of the evolution of FPGA floorplanning for Dynamic Partial Reconfiguration (DPR), organized into three distinct eras. The analysis highlights a clear progression from constraint-driven heuristic methods (Era 1), to formal optimization models addressing architectural heterogeneity (Era 2), and finally to AI-driven autonomous systems enabling runtime adaptation (Era 3).

Despite these advances, no single approach fully addresses the combined challenges of scalability, heterogeneity, and dynamic runtime behavior. This survey therefore emphasizes the need for more flexible and context-aware floorplanning strategies that bridge the gap between design-time optimization and runtime adaptability.

Beyond this classification, the study identifies three key research directions:

- Scalability–Optimality Trade-off: A fundamental gap exists between the formal rigor of Era 2 approaches and the exploration efficiency of Era 3 methods. Future solutions are expected to adopt hybrid frameworks that combine formal constraint handling with learning-based decision-making.
- Thermal–Spatial Management in 3D Architectures: With the emergence of advanced nodes and multi-die FPGA systems, floorplanning increasingly serves as a mechanism for runtime thermal regulation. This requires tighter integration between placement decisions and thermal-aware reconfiguration strategies.
- Temporal Awareness in Autonomous Systems: Current approaches often decouple spatial optimization from task-level temporal dependencies. Future work should focus on context-aware methods that jointly consider placement decisions and execution ordering constraints.

In conclusion, the future of reconfigurable computing depends on the integration of intelligent decision-making within FPGA design flows. Progress in standardized benchmarks and improved CAD tool interoperability will be essential to enable the next generation of adaptive and autonomous reconfigurable systems.

References

[1] A. Boutros and V. Betz, *Field-Programmable Gate Array Architecture*. Singapore: Springer Nature Singapore, 2022, pp. 1–47.

[2] V. Pangracious, Z. Marrakchi, and H. Mehrez, *Field Programmable Gate Arrays: An Overview*. Cham: Springer International Publishing, 2015, pp. 43–71.

[3] K. Vipin and S. A. Fahmy, "Fpga dynamic and partial reconfiguration: A survey of architectures, methods, and applications," *ACM Comput. Surv.*, vol. 51, no. 4, 2018.

[4] J.-P. Deschamps, G. D. Sutter, and E. Cantó, *Partial Reconfiguration on Xilinx FPGAs*. Dordrecht: Springer Netherlands, 2012, pp. 435–459.

[5] W. Chouchene, "Vers une reconfiguration dynamique partielle parallèle par prise en compte de la régularité des architectures fpga-xilinx," 2017.

[6] A. Sadeghi, M. Zolfy Lighvan, and P. Prinetto, "Automatic and simultaneous floorplanning and placement in field-programmable gate arrays with dynamic partial reconfiguration based on genetic algorithm," *Canadian Journal of Electrical and Computer Engineering*, vol. 43, no. 4, pp. 224–234, 2020.

[7] J. Cong, M. Romesis, and J. R. Shinnerl, "Fast floorplanning by look-ahead enabled recursive bipartitioning," 2005. [Online]. Available: https://doi.org/10.1145/1120725.1120838

[8] A. Badhoutiya, Z. Jaffer, H. M. Hussein, A. Juyal, M. Mittal, and R. Anand, "Field programmable gate array: An extensive review, recent trends, challenges and applications," in *2024 11th International Conference on Computing for Sustainable Global Development (INDIACom)*, 2024, pp. 1084–1090.

[9] L. Singhal and E. Bozorgzadeh, "Heterogeneous floorplanner for fpga," in *15th Annual IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM 2007)*, 2007, pp. 311–312.

[10] 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.

[11] "Supported/Unsupported Features • Vivado Design Suite User Guide: Dynamic Function eXchange (UG909) • Reader • AMD Technical Information Portal." [Online]. Available: https://docs.amd.com/r/en-US/ug909-vivado-partial-reconfiguration/Supported/Unsupported-Features?tocId=Ilw7PlHTVUODn9xBfPjdKA

[12] S. U. Bhandari, S. Subbaraman, S. Pujari, and R. Mahajan, "Internal dynamic partial reconfiguration for real time signal processing on fpga," *Indian Journal of Science and Technology*, vol. 3, no. 4, pp. 365–368, 2010.

[13] P. Manet, D. Maufroid, L. Tosi, G. Gailliard, O. Mulertt, M. Di Ciano, J.-D. Legat, D. Aulagnier, C. Gamrat, R. Liberati *et al.*, "An evaluation of dynamic partial reconfiguration for signal and image processing in professional electronics applications," *EURASIP Journal on Embedded Systems*, vol. 2008, no. 1, p. 367860, 2009.

[14] A. Alkamil and D. G. Perera, "Towards dynamic and partial reconfigurable hardware architectures for cryptographic algorithms on embedded devices," *IEEE Access*, vol. 8, pp. 221 720–221 742, 2020.

[15] F. Galea, S. Carpov, and L. Zaourar, "Multi-start simulated annealing for partially-reconfigurable fpga floorplanning," in *2018 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)*, 2018, pp. 1335–1338.

[16] "Chapter 10 - floorplanning," in *Electronic Design Automation*, L.-T. Wang, Y.-W. Chang, and K.-T. T. Cheng, Eds. Morgan Kaufmann, 2009, pp. 575–634.

[17] F. Mao, Y. Ma, N. Xu, S. Liu, Y. Wang, and X. Hong, "Congestion-driven floorplanning based on two-stage optimization," in *2009 IEEE 8th International Conference on ASIC*, 2009, pp. 1298–1301.

[18] A. Safir, B. Haroun, and K. Thulasiraman, "Floorplanning with datapath optimization," in *Proceedings of ISCAS'95 - International Symposium on Circuits and Systems*, 1995, pp. 41–44.

[19] M. Ibro and G. Marinova, "Fpga power consumption optimization methods analysis," in *2023 International Conference on Electromechanical and Energy Systems (SIELMEN)*, 2023, pp. 1–4.

[20] X. Zhang, H. Rabah, and S. Weber, "Dynamic slowdown and partial reconfiguration to optimize energy in fpga based auto-adaptive sopc," in *4th IEEE International Symposium on Electronic Design, Test and Applications (delta 2008)*, 2008, pp. 153–157.

[21] F. Sun, H. Wang, F. Fu, and X. Li, "Survey of fpga low power design," in *2010 International Conference on Intelligent Control and Information Processing*, 2010, pp. 547–550.

[22] P. Lysaght, B. Blodget, J. Mason, J. Young, and B. Bridgford, "Enhanced architectures, design methodologies and cad tools for dynamic reconfiguration of xilinx fpgas," in *2006 International Conference on Field Programmable Logic and Applications*. IEEE, 2006, pp. 1–6.

[23] C. Beckhoff, D. Koch, and J. Torresen, "Go ahead: A partial reconfiguration framework," in *2012 IEEE 20th International Symposium on Field-Programmable Custom Computing Machines*. IEEE, 2012, pp. 37–44.

[24] A. A. Sohanghpurwala, P. Athanas, T. Frangieh, and A. Wood, "Openpr: An open-source partial-reconfiguration toolkit for xilinx fpgas," in *2011 IEEE International Symposium on Parallel and Distributed Processing Workshops and Phd Forum*. IEEE, 2011, pp. 228–235.

[25] D. Koch, C. Beckhoff, and J. Teich, "Recobus-builder—a novel tool and technique to build statically and dynamically reconfigurable systems for fpgas," in *2008 International conference on field programmable logic and applications*. IEEE, 2008, pp. 119–124.

[26] K. Vipin and S. A. Fahmy, "Mapping adaptive hardware systems with partial reconfiguration using copr for zynq," in *2015 NASA/ESA Conference on Adaptive Hardware and Systems (AHS)*. IEEE, 2015, pp. 1–8.

[27] T. Haroldsen, B. Nelson, and B. Hutchings, "Rapidsmith 2: A framework for bel-level cad exploration on xilinx fpgas," in *Proceedings of the 2015 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, 2015, pp. 66–69.

[28] T. D. Nguyen and A. Kumar, "Prfloor: An automatic floorplanner for partially reconfigurable fpga systems," in *FPGA 2016 - Proceedings of the 2016 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*. Association for Computing Machinery, Inc, 2 2016, pp. 149–158.

[29] J. Wang, Y. Kang, W. Wu, G. Xing, and L. Tu, "Duprfloor: Dynamic modeling and floorplanning for partially reconfigurable fpgas," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 40, no. 8, pp. 1613–1625, 2020.

[30] N. Deak, O. Cret, and H. Hedesiu, "Efficient fpga floorplanning for partial reconfiguration-based applications," in *2019 IEEE 27th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)*. IEEE, 2019, pp. 309–309.

[31] B. B. Seyoum, A. Biondi, and G. C. Buttazzo, "FLORA: FLoorplan Optimizer for Reconfigurable Areas in FPGAs," *ACM Transactions on Embedded Computing Systems*, vol. 18, no. 5s, pp. 1–20, Oct. 2019. [Online]. Available: https://dl.acm.org/doi/10.1145/3358202

[32] A. B. Abdelali, M. Hannachi, M. N. Krifa, H. Rabah, and A. Mtibaa, "High-level design flow and environment for fpga-based dynamic partial reconfiguration," *International Journal of Electronics*, vol. 104, no. 8, pp. 1254–1284, 2017. [Online]. Available: https://doi.org/10.1080/00207217.2017.1293172

[33] A. Sadeghi, M. Z. Lighvan, and P. Prinetto, "Automatic and simultaneous floorplanning and placement in field-programmable gate arrays with dynamic partial reconfiguration based on genetic algorithm," *Canadian Journal of Electrical and Computer Engineering*, vol. 43, no. 4, pp. 224–234, 2020.

[34] B. Ding, J. Huang, J. Wang, Q. Xu, S. Chen, and Y. Kang, "Task modules Partitioning, Scheduling and Floorplanning for Partially Dynamically Reconfigurable Systems Based on Modern Heterogeneous FPGAs," Dec. 2022, arXiv:2212.05397 [eess]. [Online]. Available: http://arxiv.org/abs/2212.05397

[35] Y. Xiao, A. Hota, D. Park, and A. Dehon, "Hipr: High-level partial reconfiguration for fast incremental fpga compilation," Tech. Rep. [Online]. Available: https://github.com/icgrp/hipr

[36] P. Goswami and D. Bhatia, "Automated floorplanning for partially reconfigurable designs on heterogenrous fpgas," Tech. Rep.

[37] K. Yeonjin and M. Kigarura, "Ai-augmented dynamic partial reconfiguration for adaptive edge intelligence in fpga-based systems," *SCCTS Journal of Embedded Systems Design and Applications*, vol. 3, no. 1, pp. 20–27, 2025.

[38] J. Muralidharan and D. Abdullah, "Ai-augmented runtime reconfiguration for energy-aware fpga-based edge computing systems," *SCCTS Transactions on Reconfigurable Computing*, vol. 3, pp. 48–59, 2026.

[39] G. Montanaro, A. Galimberti, and D. Zoni, "Omega: A hardware-software framework for complete design space exploration of fpga-based heterogeneous multi-core socs," *IEEE Transactions on Computers*, 2025.

[40] C. Guo and T. Zhao, "Resbench: Benchmarking LLM-Generated FPGA Designs with Resource Awareness," *Proceedings of International Symposium on Highly Efficient Accelerators and Reconfigurable Technologies (HEART '25)*, vol. 1, 2025, [Online]. [Online]. Available: http://arxiv.org/abs/2503.08823

[41] J. Boudjadar, S. U. Islam, and R. Buyya, "Dynamic FPGA Reconfiguration for Scalable Embedded Artificial Intelligence (AI): A Co-Design Methodology for Convolutional Neural Networks (CNN) Acceleration," *Future Generation Computer Systems*, vol. 169, p. 8, 2025.

[42] S. A. Fahmy and K. B. Iyer, "Dynamic and Partial Reconfiguration of FPGAs," in *Handbook of Computer Architecture*, A. Chattopadhyay, Ed. Springer Nature Singapore, 2022, pp. 1–24. [Online]. Available: https://doi.org/10.1007/978-981-15-6401-7_51-1

[43] J. Gu, H. Wang, X. Guo, M. Schulz, and M. Gerndt, "Versaslot: Efficient fine-grained fpga sharing with big.little slots and live migration in fpga cluster," in *2025 62nd ACM/IEEE Design Automation Conference (DAC)*, 2025, pp. 1–7.

[44] Y. Li, Y. Chen, Z. Xu, Y. Wang, H. Jiang, and K. Li, "Feditor: Consecutive Task Placement with Adjustable Shapes Using FPGA State Frames," *IEEE Transactions on Parallel and Distributed Systems*, vol. 37, pp. 1–14, 2026.