PLUNIFY
Whitepaper

Transformational ML-Based Approach to Tackle Severe FPGA Placement and Routing Failures

Hichem Belhadj, Plunify Inc.

165, University Street, Palo Alto, CA

Hichem.Belhadj@plunify.com

Abstract

Placement and routing failures during the FPGA backend flow are frequent with many FPGA designs especially when these involve multiple physical constraints. This paper introduces the Machine-Learning-based approach implemented in InTime toolset to transform these failures to a less complex problem of timing convergence. Thanks to machine learning, InTime can identify the root causes of the placement or the routing failures, mitigating them through adjustment of the user constraints, and setting the various options making the design placement and routing friendly. As the experimental results illustrate, the encountered placement and routing failures are all resolved. Additionally, timing requirements are met for over 85% of the case through this ML-based transformation. The other 15% of the designs are processed through the traditional InTime recipes to resolve timing convergence.

Introduction

FPGA designers face several types of failures while processing their designs. While most of these challenges are related to meeting the timing requirements, routing failures occur in many instances due to inherent or artificial congestion. Placement failures while less frequent than routing failures do occur in 5 to 10% of the FPGA designs when complex and numerous physical constraints are associated with these designs. Regardless of the root cause, the impact of these failures on the schedule of the projects, the human resources allocation, and the overall cost is not only high but also unpredictable.

The paper is organized in 4 sections: the first covers the inherent and artificial root causes of placement and routing failures, while the second section introduces the ML-based approach to radically tackle these failures. The third section provides actual results on real life designs. The final section summarizes the takeaways and introduces new venues Plunify is exploring to further the improvements of the outcome in terms of QoR (quality of results) and enhancement of the predictability of the schedule and human and compute resources allocation.

1. Inherent and Artificial Root Causes of Placement and Routing Failures

While dealing with FPGA design's challenges, designers are frustrated with the lack of predictability of the results and with the lack of hints the tools provide to solve these challenges. This frustration is at its highest level when these failures are placement or routing failures. In this section, the focus is on identifying the root causes that yield these challenges.

Among the main sources of the placement and routing challenges one can list the following:

i. High congestion level inherent to the design where the routing needs are extremely high compared to the actual logic used. For instance, large crossbar, deep CRC, large Mux/Demux, or designs with large busses between hierarchical blocks exhibit high congestion level inherently.
ii. High utilization of various resources including logic cells, embedded resources such as RAM, DSP, or high-speed IO blocks, clocking resources, and User IOs.
iii. Large number of high fanout nets that drive logic across multiple hierarchical blocks and/or IOs.
iv. Poor placement that artificially causes interacting hierarchical blocks to be placed in far away location of the die or even in adjacent of no adjacent dies in a multi-die FPGAs.
v. Poor routing may artificially cause fights over routing resources and could yield either artificial high congestion or unsuccessful routing such as high number of open nets or overlapping nodes.
vi. User constraints that separately may make sense but combined makes it hard for the backend toolset to place or to route the design. While timing constraints may cause the tools to choke on parts of the design due to their aggressive nature, physical constraints such as placement or floorplan requirements could be even harder to honor and usually lead to tools failure when the number of these physical constraints is high.

InTime, the Plunify ML-based toolset [1], distinguishes between the class of root causes i, ii, and iii and consider them inherent to the design profile and the class of root causes described in iv, v, and vi above and we call them artificial root causes. InTime recipes such as "Hot Start", "InTime Default", "SSI Exploration", "Extra Opt" recipes tackle the design inherent structural challenges [2]. The new Auto-ML approach provides a new recipe implemented in the new InTime release focuses on resolving the artificial root causes of placement and routing failures.

2. Auto ML-Based Approach to Tackle Placement and Routing Failures

Ultimately, the goal is to resolve primarily the placement and/or the routing failures and to transform the design and associated constraints to a timing convergence issue that InTime has a track record of resolving. It is a transformation from what is conventionally called "red ocean" to "blue ocean" type of problem as illustrated in the figure 1 below.

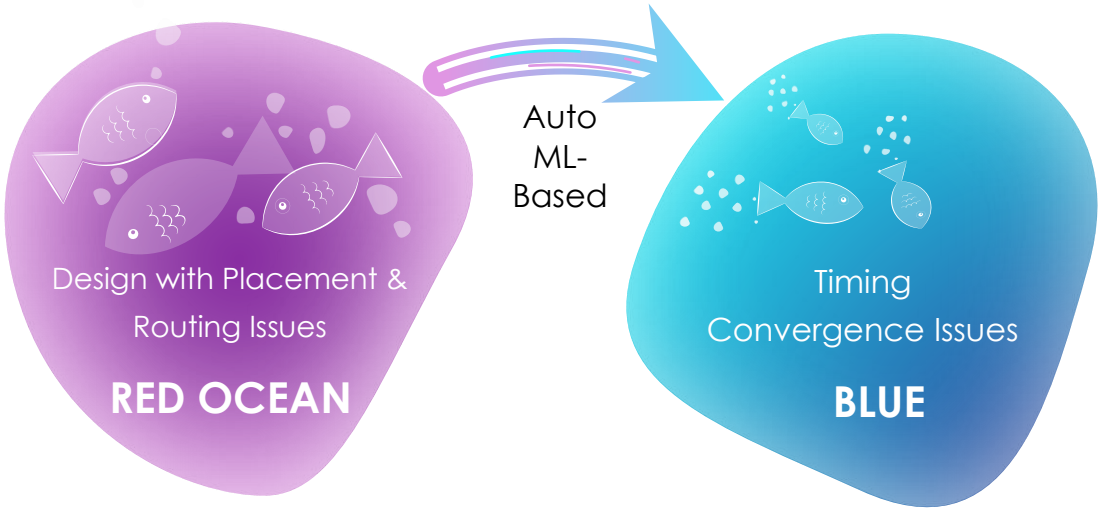

Figure 1: Transformational Auto ML-Based Approach Tackling Complex Placement and Routing Failures.

While sophisticated, the auto ML-based approach derives its strength from the learning it acquires from the in-depth analysis of the design profile as well as the "adequacy" of the physical and timing constraints. The concept of "adequacy" is a holistic profiling of the constraints that not only considers each constraint separately, but also the overall impact of the combination of these constraints. Added to the profiling, InTime uses the contextual ML-based insight relative to the FPGA architecture, the specifics or the die/package combination, and the characterization of the release of the FPGA vendor tools to derive adjusted set of timing and physical constraints that transform the design and associated constraints to be a lot more placement and/or routing friendly.

Moreover, the newly context-aware generated constraints are limited in number to reduce the need for compute resources and shorten the overall compile time. To illustrate this statement, floorplan constraints define regions where functional block or portions of the design need to be assigned to or placed. The shape and size of these regions (pblocks in the Vivado terminology of constraints) matter and there are several combinations. Limiting the number of combinations of size/shape of these regions (or pblocks) and ensuring successful placement and routing is a paramount to reducing the compile time and the compute resources required for the parallel jobs InTime spawns to run concurrently.

This approach turned out to be highly effective and efficient in the sense it also contributes to the resolution of severe timing challenges while keeping the overall compile time and compute resources usage reasonable.

3. Auto ML-Based Approach Experimental Results

The proposed approach has been put to task on several real-life designs coming from HPC (high performance compute), storage, test and measurement, comms., and accelerated trading applications. The illustrative yielded results are split in 2 categories:

- Designs with routing failures
- Designs with placement failures

3.1. Experimental Results for Designs with Routing Failures

The Table 1 presents the various benchmarks, the associated end application, as well as the actual failure and the relative error message. Most of these designs are complex and with high utilization, and the associated timing and physical constraints are stringent to say the least. The newly introduced Auto ML-based recipe yielded a complete resolution of the routing failures for all these benchmarks except for two designs, highlighted in yellow, where the routing failure has been drastically tamed with the new recipe. Additionally, the timing requirements were met for sixteen (16) out of the eighteen (18) processed designs and within very reasonable compile time as opposed to a manual and iterative approach based on changes of RTL, physical or timing constraints, or tools' settings that will consume an order of magnitude more time and compute resources and adds a tremendous pressure on the designers involved in this optimization effort and their managers.

FPGA ID	FPGA Vendor Tool Results	Backend Tool's Error Message	InTime Results	Run Time
Benchmark HPC (1)	Severe Routing Failure	High Congestion Level 7	WNS=0.058, TNS=0	5h51m
Benchmark HPC (2)	Routing Failure	High Congestion Level 6	WNS=0.004, TNS=0	10h47m
Storage Benchmark (1)	Severe Routing Failure	High Congestion Level 7	WNS=0, TNS=0	24h18m
Storage Benchmark (2)	Severe Routing Failure	High Congestion Level 7	WNS=0, TNS=0	18h31m
TME Benchmark (1)	Routing Failure	High Congestion Level 6	WNS=0, TNS=0	21h17m
TME Benchmark (2)	Severe Routing Failure	High Congestion Level 6	WNS=0, TNS=0	18h48m
HPC Benchmark (3)	Severe Routing Failure	High Congestion Level 7	WNS=0, TNS=0	29h10m
HPC Benchmark (4)	Routing Failure	295567 Nodes overlap	124 Nodes overlap	9h24m
Edge Benchmark (1)	Routing Failure	High Congestion Level 6	WNS=0, TNS=0	16h38m
Edge Benchmark (2)	Severe Routing Failure	High Congestion Level 7	WNS=0, TNS=0	12h35m
Acc. Trading Benchmark (1)	Severe Routing Failure	High Congestion Level 7	WNS=0, TNS=0	6h56m
Accelerated Trading Benchmark (2)	Severe Routing Failure	High Congestion Level 6	WNS=0.074, TNS=0	23h21m
TME Benhmark (3)	Routing Failure	High Congestion Level 6	WNS=0.074, TNS=0	22h22m
Digital Beam Forming Benchmark	Routing Failure	54067 Nodes Overlap	WNS=0, TNS=0	27h57m
HPC Benchmark (5)	Severe Routing Failure	High Congestion Level 7	WNS=0, TNS=0	25h13m
Wireless Benchmark (1)	Severe Routing Failure	High Congestion Level 7	WNS=0.025, TNS=0	11h32m
Wireless Benchmark (2)	Routing Failure	High Congestion Level 6	WNS=0, TNS=0	22h38m
Wireline Benchmark	Routing Failure	1649514 Nodes Overlap	Routing Failue1102 Nodes Overlap	13h22m

Table 1: Efficiency and Effectiveness of Auto-ML on Routing Failures

3.2. Experimental Results for Designs with Placement Failures

Table 2 presents the various benchmarks, the associated end application, as well as the actual placement failure, and the results obtained using the new InTime recipe.

FPGA ID	FPGA Vendor Results	InTime Timing Results	Run Time
Wireline Benchmark (2)	Placement Failure	WNS=-3.17, TNS=-5806	32h12m
HPC Benchmark (6)	Placement Failure	WNS=0, TNS=0	21h46m
TME Benchmark (4)	Placement Failure	WNS=0.14, TNS=0	10h52m
HPC Benchmark (6)	Placement Failure	WNS=-1.693, TNS=-1621.7	25h50m

Table 2: Efficiency and Effectiveness of Auto-ML on Placement Failures

A brief exploration of the data provided in Table 2 demonstrates that the Auto ML-based recipe solved not only the placement and routing problems for all the designs, but also met the timing constraints associated with half of the considered benchmarks.

4. Final Look and Ongoing Near Future and Mid-Term Development

To put things in a broader context, previous releases of InTime, the ML-Based toolset, were focused and successful at talking the complex problem of timing convergence with several underlying technologies as illustrated in Figure 2.

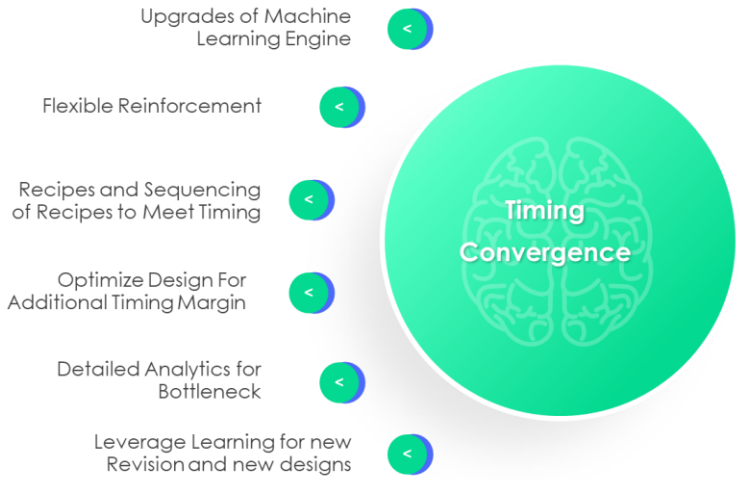

Figure 2: Previous Focus of InTime and Underlying Technologies to Achieve Timing Closure

The newly developed auto ML-based approach and associated recipes expanded the scope and the complexity of issues encountered by an increasing number of designs spanning across multiple applications pushing the performance envelope. Figure 3 provides an overview of the underlying technologies that enables a successful outcome when tackling placement and routing failure regardless of their severity.

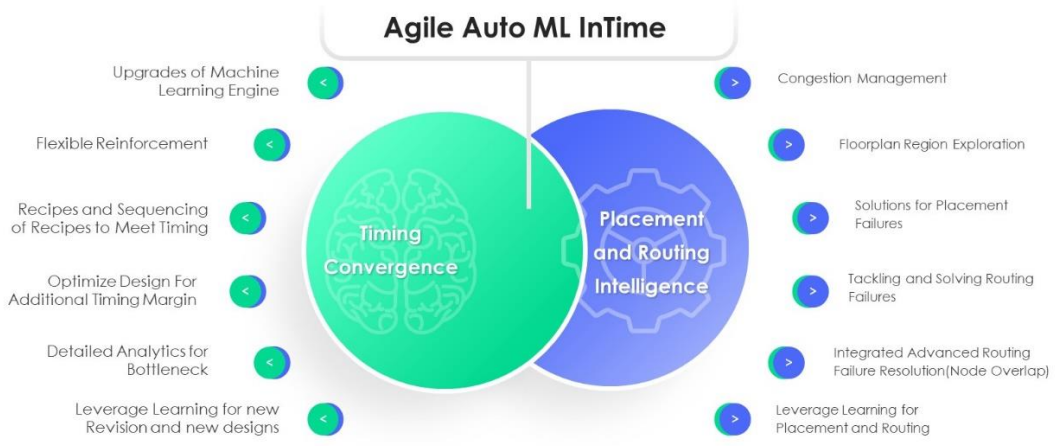

Figure 3: The New Agile InTime with Embedded Auto-ML and Underlying Technologies

The experimental results obtained on many real-life and complex designs and their associated stringent constraints illustrate the effectiveness of this approach. The compile time and the compute resources deployed during these experiments show the efficiency of using the resources and the short cycle-time to achieve not only successful place and route but also meet the stringent constraints.

Ongoing development is underway to further the improvement of the quality of results, the ease of use, and broadening the scope of InTime to tackle new problems such as optimizing designs power consumption while tackling timing convergence failures.

For the latest product information, please contact or visit www.plunify.com

About Plunify

Plunify helps businesses and organizations build better FPGA products. Our solutions enable shorter time-to-market, better design processes and predictable outcomes. Hardware and software developers can focus on delivering their FPGA-based applications without worrying about infrastructure and tools.

Email: tellus@plunify.com
Address: 165, University Street, Palo Alto, CA