InTime: A Machine Learning Approach for Efficient Selection of FPGA CAD Tool Parameters

Nachiket Kapre
Nanyang Technological University
Singapore
nachiket@ieee.org

Harnhua Ng, Kirvy Teo, Jaco Naude
Plunify Inc.
Singapore
harnhua@plunify.com

ABSTRACT

FPGA CAD tool parameters controlling synthesis optimizations, place and route effort, mapping criteria along with user-supplied physical constraints can affect timing results of the circuit by as much as 70% without any change in original source code. A correct selection of these parameters across a diverse set of benchmarks with varying characteristics and design goals is challenging. The sheer number of parameters and option values that can be selected is large (thousands of combinations for modern CAD tools) with often conflicting interactions. In this paper, we present InTime, a machine-learning approach supported by a cloud-based (or cluster-based) compilation infrastructure for automating the selection of these parameters effectively to minimize timing costs. InTime builds a database of results from a series of preliminary runs based on canned configurations of CAD options. It then learns from these runs to predict the next series of CAD tool options to improve timing results. Towards the end, we rely on a limited degree of statistical sampling of certain options like placer and synthesis seeds to further tighten results. Using our approach, we show 70% reduction in final timing results across industrial benchmark problems for the Altera CAD flow. This is 30% better than vendor-supplied design space exploration tools that attempts a similar optimization using canned heuristics.

Categories and Subject Descriptors

B.5.2 [**Design Aids**]: [Automatic Synthesis]; I.2.6 [**Learning**]: [Parameter Learning]

General Terms

FPGA CAD, Machine Learning, Tools

1. INTRODUCTION

Modern FPGA backend CAD tools compile behavioral and structural descriptions of circuits in Verilog/VHDL down to executable FPGA bitstreams. However, they are hard to configure correctly to satisfy intended design constraints. These CAD tools implement heuristics that tend to generally produce usable results, but often require careful configuration and setup to give you the desired QoR (quality of result). Designers today rely on ad-hoc tuning techniques based on intuition that is derived from years of experience. Intelligent tools like vendor-supplied design space exploration scripts attempt to guide the CAD process using vendor knowledge of the architecture-CAD interaction but fail to capture the full range of possibilities. Using the InTime framework, we hope to uncover the fundamental principles behind the selection of CAD tool options (called parameters in this paper) and deliver high-speed FPGA designs reliably across different circuit characteristics.

It is generally accepted that transformations at synthesis stage (high-level synthesis or RTL-level) have the maximum impact on circuit properties such as speed and area. However, verified RTL designs are often considered off-limits for optimization, particularly at late stages of the design process, to minimize risks of inserting functional errors in the design. This leaves us with the backend CAD flow as the only alternative for tuning and optimization. The control of the CAD flow is dictated to some extent by hard constraints such as pin location constraints, timing targets, and clock network choices, among others that are fixed. Beyond these fixed parameters, the correct setup and configuration of CAD tool parameters still requires managing a large space of possible option choices. Typically, designers will attempt to generate better QoR by varying placement and synthesis seeds. This is a scatter-shot approach that ignores the interactions with many other CAD tool parameters. Additionally, designers manage the CAD tool parameter selection complexity by heuristically making choices based on experience or intuition. CAD tools are often fussy and behave unpredictably when parameters are chosen incorrectly leading to wastage of designer effort. In most cases, the parameter choices are a tradeoff that can have unpredictable effects on an overall figure of merit. The correct selection of these parameters can give us freedom required to deliver better result quality without modifying RTL.

The key challenge in managing the fruitful operation of the FPGA backend CAD tools is addressing the magnitude of **parameter** choice combinations and automation of the search. Considering these effects, we may be tempted to consider a Monte-Carlo sampling based exploration of possible parameter combinations as a mechanism for achieving a high quality answer. We know this is not feasible due to

FPGA'15, February 22–24, 2015, Monterey, California, USA.

http://dx.doi.org/10.1145/2684746.2689081.

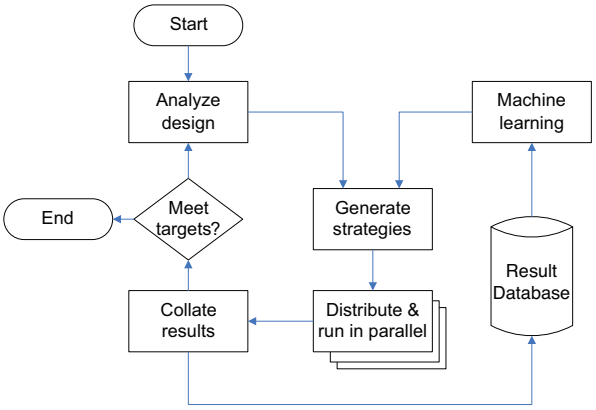

Figure 1: InTime Flow Diagram

long design times even for a single design instance which can take hours to days for large FPGAs. In this paper, we introduce a smarter, machine learning and cloud computing based approach for performing a series of CAD tool runs to determine an ideal combination of parameters to consider. This avoids the need for large-scale exploration while allowing us to reach close to the ideal answer with as few runs as possible.

The key contributions of this paper are:

- Development and design of machine learning and cloud computing infrastructure for FPGA CAD to deliver timing closure for industrial designs.
- Measuring and quantifying the quality and time gap between InTime and vendor-supplied design space exploration tools.
- Development and quantification of the benefits of a machine learning algorithm that identifies correlations between CAD tool parameters and solution quality.

2. THE INTIME FLOW

The problem of compiling circuits to FPGAs is a challenging problem that is managed by decomposing the compilation process into a series of sequential CAD stages *e.g.* synthesis, technology mapping, placement, routing. Each stage is typically an NP-complete problem and is associated with a *tunable heuristic*. Decisions made at each stage affect the quality of the solution by restricting/enhancing choice in the downstream stages. Furthermore, some of these heuristics use randomness that can add non-determinism in the solution if the user is careless with seeds. Circuit designers often spend weeks of design effort optimizing their circuit and to meet timing. If the correct set of compiler options are known in advance or discovered quickly, this tedious process can be eliminated and source code can stay unchanged. In this section we introduce InTime and discuss how it operates.

What is InTime?: To an FPGA developer, InTime is a plugin to the FPGA CAD flow that selects and tunes CAD tool parameters on behalf of the designer. After RTL verification, designers will often manually run their projects through different compiler parameters to see if performance can be optimized. Sometimes this struggle to meet timing requires weeks of expensive designer time which can be better spent elsewhere in the design stack. While RTL redesign may be necessary for critical timing failures, in many cases, with experienced developers, the RTL design is sufficiently well-pipelined and organized. Simply relying on placement and synthesis `SEED`s for exploring the design space only yields 5–10% variation in the timing results. However, `EFFORT_LEVEL` also introduces non-trivial randomness into the results, particularly for the Arria devices. In these instances, InTime takes over the tedious role of grinding the design through the CAD tools under various configurations. Additionally, InTime learns from the results to generate new strategies that help the design meet timing. While the FPGA vendors often supply tools that do these tasks, they are primitive in nature. InTime offers the following three advantages over vendor-supplied design space exploration products:

- **Multiple High-Quality Settings**: InTime analyzes and generates a broad range of synthesis and place-and-route settings based on pre-programmed recipes. In this configuration, InTime behaves similar to a vendor-provided design space exploration tools. However, a key difference, is that the goal of this initial exploration is to generate sufficient knowledge for use with our machine learning engine. Additionally, our exploration tools generate significantly more configurations for consideration.
- **Machine-Learning**: InTime's biggest asset is its machine-learning capacity. It learns from past results and subsequently adjusts the settings that it generates to go beyond the results produced from canned recipes. We use the Naive Bayesian approach for predicting good strategies while relying on Principal Component Analysis to isolate and focus on influential parameters.
- **Parallel Builds**: InTime is capable of operating sequentially on a single build machine, or more naturally in parallel on private cloud infrastructure (internal build clusters), or public clouds (Amazon EC2, Google Compute Engine) to parallelize builds and accelerate the search process. The use of cheap, affordable parallelism against expensive developer tinkering with the tools, allows InTime to be simultaneously faster and better in reducing implementation costs.

How does InTime work?: InTime uses a combination of machine learning, search-space pruning, and limited random exploration to guide and drive the optimization process. We show an operational diagram of InTime in Figure 1. The InTime flow starts by analyzing the characteristics and existing results of the design under test. We derived mathematical models that related design characteristics like target device family and logic structure; compiler settings like synthesis and place-and-route options; to the design performance metrics like timing, area and power estimates. These models are used to determine the statistical relevance of the CAD settings as a function of design characteristics. Such properties are then applied to answer questions like, which of the 70-plus compiler settings have the most influence on timing? Which compiler setting values are the most influential ones for a particular device family? We use device

and toolchain specific initial machine learning estimates to guide the strategy generation engine. After each strategy is applied to the underlying FPGA toolchain, the strategies are distributed and evaluated on the selected run target. The InTime framework abstracts the IT infrastructure from the user and automatically analyzes and collates the generated results before they are incorporated into InTime's runs database. An iterative approach allows repeating the flow a configurable number of times. Each new round will use previously stored results to generate machine learning outcomes which guide the strategy generation engine to produce smarter strategies as the amount of training data increases. We use statistics to analyze the correlations between the design, the tool settings and the synthesis/implementation results. We discovered that as long as the median Timing Score improves from build to build, there is a good chance of improvement until Timing Closure is achieved. These correlations are saved so that knowledge of what works well is accumulated and more quickly applied onto subsequent designs

3. EVALUATION

For our experiments, we are interested in minimizing timing scores (*i.e.* critical path delay) at the end of the FPGA CAD process. Timing score is the sum of timing slacks of all paths that fail to meet timing. Timing score is a better, more insightful measure of which portions of the design need attention. A timing score of 0 indicates that the design meets timing, while a larger score indicates failing nets. We use a combination of open-source and industrial benchmarks that occupy between 50-90% of the FPGA capacity. We pick these benchmarks to cover various application domains and characteristics while also ensuring they stress the limits of the CAD tools through high occupancies.

How well does InTime perform when compared to vendor-supplied design space exploration tools? In Figure 2, we measure the final timing scores across our benchmark circuits when using InTime as well as the vendor tools. For this experiment, we collect benchmarks from open-source and industrial sources that are mapped across a variety of Altera devices taking up between 50–100% (average 73%) of the FPGA area. As you can see, the raw, original design generally has high timing scores. When using vendor-supplied exploration tools, we notice a reduction in these scores as expected but InTime consistently delivers routed designs with the lowest timing scores across our benchmark set.

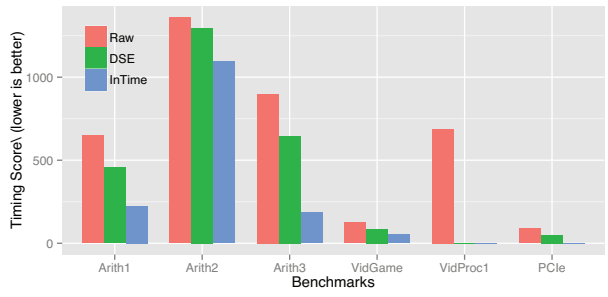

Figure 2: Comparing InTime with vendor design space exploration tool

How does machine-learning affect the quality of InTime's results? Machine-learning allows us to discover the best settings for CAD parameters with consistently better timing scores. In Figure 3, we show a distribution of timing scores obtained from 100 CAD compilation trials with and without InTime. After statistical analysis of the results, we can see that a large majority of InTime runs results in lower timing scores. This is despite starting with a higher timing score at the beginning (raw column in Figure 3). It is worth noting that for this benchmark, machine learning is not strictly necessary to meet timing, but very useful to reduce the CAD iterations and CAD runtime required to achieve timing closure.

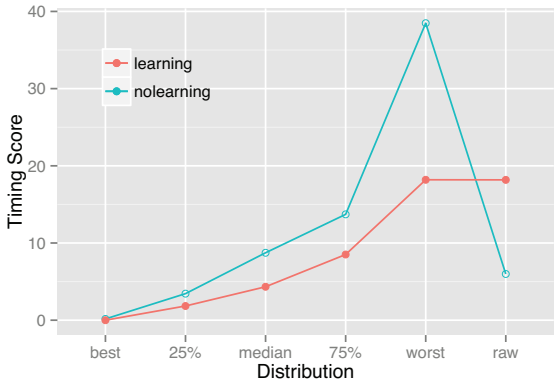

Figure 3: Understanding the impact of machine learning in InTime execution

How does InTime affect CAD parameter choices? The various CAD tool parameters explored by InTime eventually converge to steady-state values after a sufficient number of CAD trials. We show a representative sample set of 15 important CAD parameters in Figure 4 across 23 iterations of CAD jobs. The initial stages are seeded with strategies from our existing knowledge-base, but subsequent CAD jobs will try parameter values that are increasingly closer to their ideal results. The machine-learning algorithms help refine and revise the choices in a manner that helps us descend to the final solution.

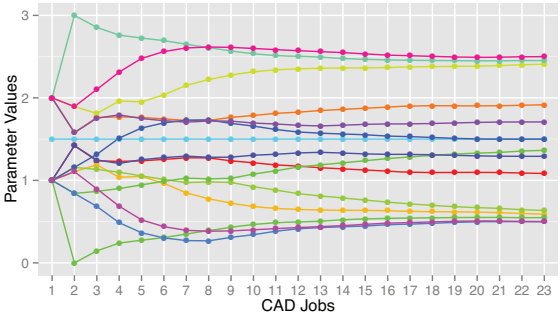

Figure 4: InTime exploring various CAD tool parameters over lifetime of one design run

Which CAD tool parameters are most influential in calculation of resulting timing score? In Figure 5, we show

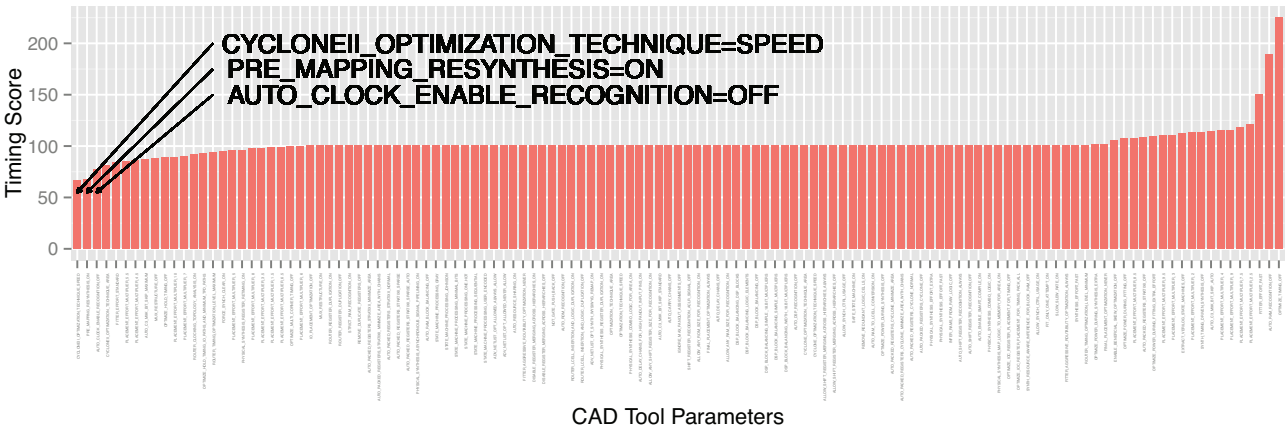

Figure 5: Sensitivity Analysis of CAD Parameter Settings

the impact of varying each individual CAD parameter (90 settings) while keeping the rest in their default settings for one benchmark. This allows us to do a limited sensitivity analysis to help isolate the parameters that matter. We use Principal Component Analysis on dataset produced in this fashion across various benchmarks to help build a pruned model of CAD tool behavior. While the exact settings vary with benchmark, the top-3 influential settings in this case were `CYCLONEII_OPTIMIZATION_TECHNIQUE=SPEED`, `PRE_MAPPING_RESYNTHESIS=ON`, and `AUTO_CLOCK_ENABLE_RECOGNITION=OFF`

4. RELATED WORK

Several academic studies have attempted to understand the impact of exploring multiple implementation parameters of the design space to better understand opportunity and potential.

For FPGA academic CAD tools in [3], the authors quantify the variation in solution quality when using VPR 5.0.2 under different timing targets and input net ordering. They report a critical path delay gap in the range of 17-110% when compared to nominal behavior. InTime moves beyond a trivial brute-force exploration of targets to employ machine-learning strategies to converge to the best answer sooner. In [4], the authors develop a strategy inspired by Design of Experiments (DoE) to customize the parameters of the soft processor design space. They do this by carefully selecting a subset of the parameters and their associated ranges for experimentation. Unlike DOE-techniques, we introduce learning-based techniques to allow tuning and customization of CAD runs particular to each design requirement. In [2], the authors consider the impact of ordering of LLVM passes on the quality of hardware solution for high-level synthesis. They observe a variation in excess of 10% by composing various compiler *passes* in different ways. InTime tackles the FPGA backend CAD flow choices rather than HLS flows and is aimed at developers who do not want to modify verified RTL code (or minimize the need for such modifications). In [1], for logic-synthesis, an area quality gap of between 70× was observed for industrial circuits when relying on the CAD tool heuristics to discover and exploit patterns in the input.

5. CONCLUSIONS

The correct choice of FPGA CAD tool parameters requires intelligence and learning for different combinations of FPGA platforms and applications. While human designers are capable of developing this intuition through years of practice, InTime complements this wisdom through an automated machine-learning based approach. When FPGA architectures and CAD tools evolve along with the characteristics of circuits that are mapped to FPGAs, the intuition is subject to continuing refinement and improvement that are better handled through automated tools like InTime. For modern Altera FPGAs and industrial circuit benchmarks, we are able to improve timing quality by as much as 70% for designs that often occupy most of the FPGA. More broadly, when coupled with the compute prowess of cloud-based technologies, the process of letting thousands of machines discover the best answer for CAD problems appears less wacky.

6. REFERENCES

[1] J. Cong and K. Minkovich. Optimality study of logic synthesis for lut-based fpgas. *Computer-Aided Design of Integrated Circuits and Systems, IEEE Transactions on*, 26(2):230–239, Feb 2007.

[2] Q. Huang, R. Lian, A. Canis, J. Choi, R. Xi, S. Brown, and J. Anderson. The Effect of Compiler Optimizations on High-Level Synthesis for FPGAs. *Field-Programmable Custom Computing Machines (FCCM), 2013 IEEE 21st Annual International Symposium on*, pages 89–96, 2013.

[3] R. Y. Rubin and A. M. DeHon. Timing-driven pathfinder pathology and remediation: quantifying and reducing delay noise in VPR-pathfinder. In *FPGA '11: Proceedings of the 19th ACM/SIGDA international symposium on Field programmable gate arrays*. ACM Request Permissions, Feb. 2011.

[4] D. Sheldon, F. Vahid, and S. Lonardi. Soft-core Processor Customization using the Design of Experiments Paradigm. In *Design, Automation & Test in Europe Conference & Exhibition, 2007. DATE '07*, pages 1–6, 2007.