PLUNIFY

Optimizing design performance with InTime and Xilinx tools

Executive Summary

This whitepaper describes how InTime™ works with Xilinx™ software to optimize FPGA timing performance by adjusting compilation parameters and running builds in parallel. InTime uses machine learning to determine the best combination of synthesis and place-&-route settings for an FPGA design. Combined with compute servers, InTime rapidly optimizes timing while simultaneously addressing limitations on the user's flow automation.

Introduction

The traditional approach to timing optimization is to check and improve the RTL or constraints. While this method works well, there are real-life situations where changes are restricted due to various technical and business limitations, for example, a potential design overhaul poses greater risk to the product release deadline. In this era of reusable design blocks, often there are third-party IP cores in the design which cannot be easily modified. The "worst-case scenario" solution is to simply upgrade the target device to a larger one or to one with a faster speed grade but both come at a cost.

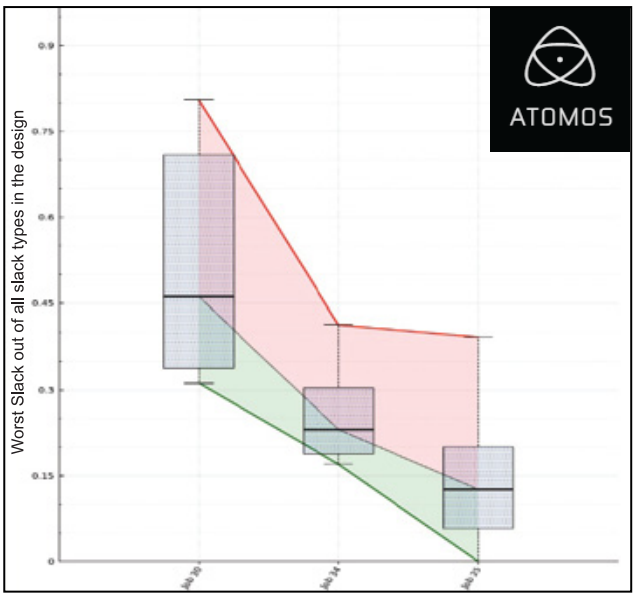


Figure 1

Luckily modern FPGA tools such as Xilinx's Vivado™ contain lots of knobs and switches to help close timing. InTime's approach is to solve the user's timing and other performance issues by tuning the compilation processes of the FPGA tools. ISE (also from Xilinx) and Vivado software contain many synthesis and place-and-route parameters, each with two or more possible values that can directly affect synthesis and place-and-route results. InTime helps designers tap into the full capabilities of these tools to get the required results.

In the customer provided design shown in Figure 1, the X-axis represents batches of compilations with different synthesis and place-and-route parameters and the Y-axis shows the absolute value of the failing Worst Slack (0 means timing pass) in nanoseconds. Here you can see that InTime successfully reduces the failing Worst Slack of a design from -450ps to 0ps, achieving the timing target by just changing compilation parameters, without any change in the design.

In many cases, the user usually leaves the synthesis and place-and-route parameters at default values. Not many users attempt to change these parameters as there is little understanding of their actual impact. This task is further complicated by the fact that the parameters are inter-dependent and may cause timing to get worse if the wrong ones are used.

Understanding the InTime Flow

InTime uses machine learning techniques to efficiently explore the effects of parameter set variations (strategies) in the FPGA compilation process. The techniques described below focuses on generating sufficient data points before converging on the performance peaks.

One key concept is the “recipe”. InTime classifies machine learning techniques into "recipes", which are further categorized into “Learning” and “Last Mile” recipes.

Learning Recipes
The more data, the better it is

Last Mile Recipes
The better the “parent” result, the better it is. More data does not help.

The reason for categorization is because compilations are highly intensive compute processes. The run time “cost” of obtaining new data is high (and human patience is in short supply). Each recipe can not run indefinitely, so there is a need to limit the number of learning runs based on result improvements. Once the results start to plateau (the ROI for improved results based on the time spent decreases), users switch to the “Last Mile” recipes. The “Last Mile” recipes are highly random techniques that work better the closer the design is to the target performance goals. For example, using the best result attained so far as a point of reference, “Last Mile” recipes will randomize the placements of the different logic elements.

Steps to optimize a design

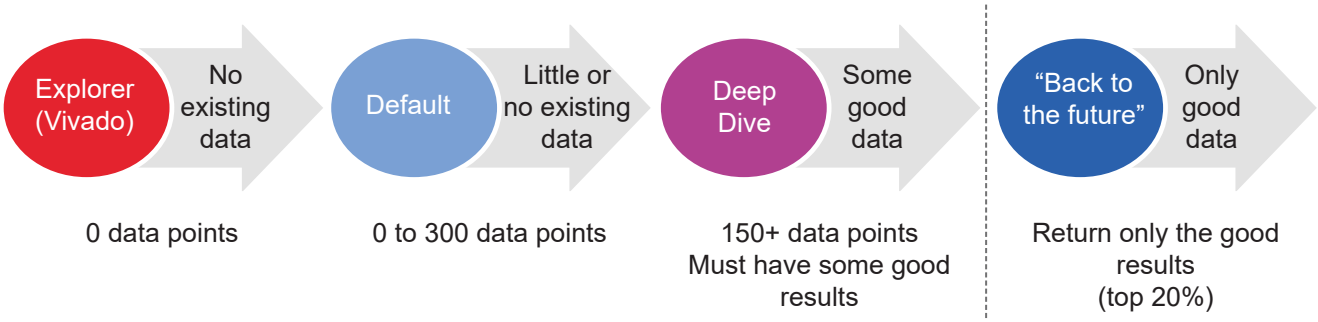
- **Step 0: Model the design**

To reduce the time required to converge on a result, InTime comes with an initial database that contains meta-data. This meta-data is built over time using different designs to help us determine what types of parameters are suitable for different designs. The aim is to narrow down the vast set of parameters to pick only significant parameters that will be most effective for a particular design.

- **Step 1: Generate data**

In this step, InTime generates compilation parameters (also known as “strategies”) in each round of execution. The designer should configure each round to run between 10 to 30 compilations. Different recipes are more suitable than others, based on the number of data points (compilation results) available.

Learning and analysis happen only at the end of each round and at the beginning of the next round. As a guideline, InTime needs to analyse around 100 data points over 3 to 5 rounds to reach a good local maxima (results closer to 0ns TNS or positive WNS).

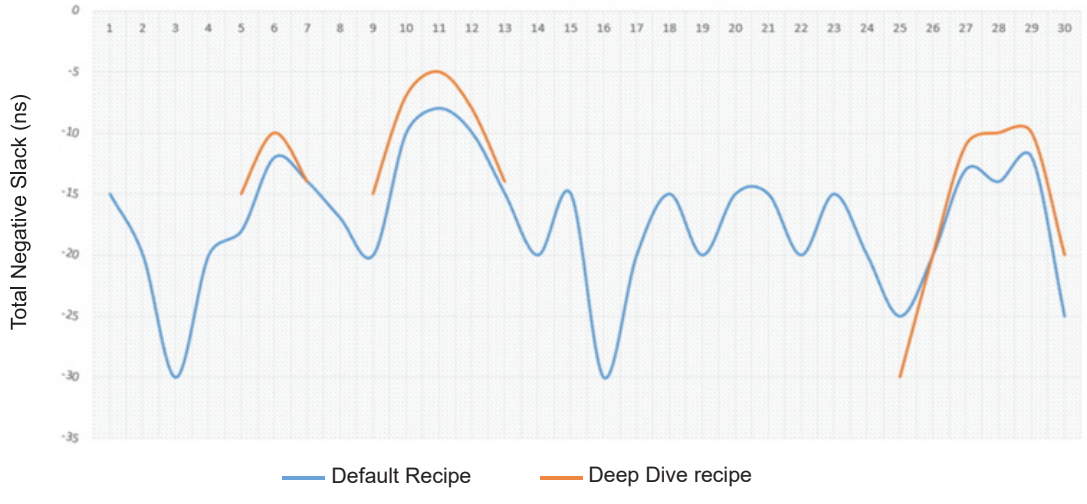


If the results do not show significant improvements, more compilations may be required as the recipe has not reached a local maxima. However, if timing has improved significantly (compared to the original results) and the improvements have plateaued, then it is time to switch recipes (see “Deep Dive”).

- **Step 2: Use “Deep Dive” Recipe**

Once a few good results have been obtained, or as soon as the improvements have slowed down, the “Deep Dive” recipe is the next recipe to use. “Deep Dive” examines the current crop of results and does an in-depth analysis of the local maxima as well as its surrounding points, yielding about 10% improvement in results in a shorter amount of time compared to the recipes before it. Of course, without the results from earlier recipes, Deep Dive will not be as effective. (Figure 2)

Figure 2: Default versus Deep Dive recipe



- **Step 3: Auto Placement or Extra Optimization**

Finally, the Last Mile category of recipes uses only specific Vivado settings that are pseudo-random and highly sensitive to code changes. Depending on the design, such recipes can generate only a few or a large number of compilations. For example, Placement Exploration can easily go up to 100 compilations whereas Extra Optimization is limited to 9.

InTime & Vivado in the cloud

Reduce the time needed to reach your timing targets using InTime within the Amazon Web Services (AWS) cloud. By doubling your concurrent runs, you can halve the time required to complete your optimization.

InTime partners with Xilinx to provide AMIs with all software licenses pre-installed. This allows you to quickly start an instance and run your FPGA projects in the cloud without any prior installation.

Conclusion

Choosing the right set of synthesis and place-and-route parameters is a powerful technique to achieve the target design performance, and obtain the most benefits from FPGA tools like Vivado. However, it is impossible to try every single set of parameters. Converging quickly on the right combination of parameters can yield drastic results as shown in Figure 3 (–3000ns to -3ns for Total Negative Slack). Using the cloud can also reduce the total time required to achieve ideal results.

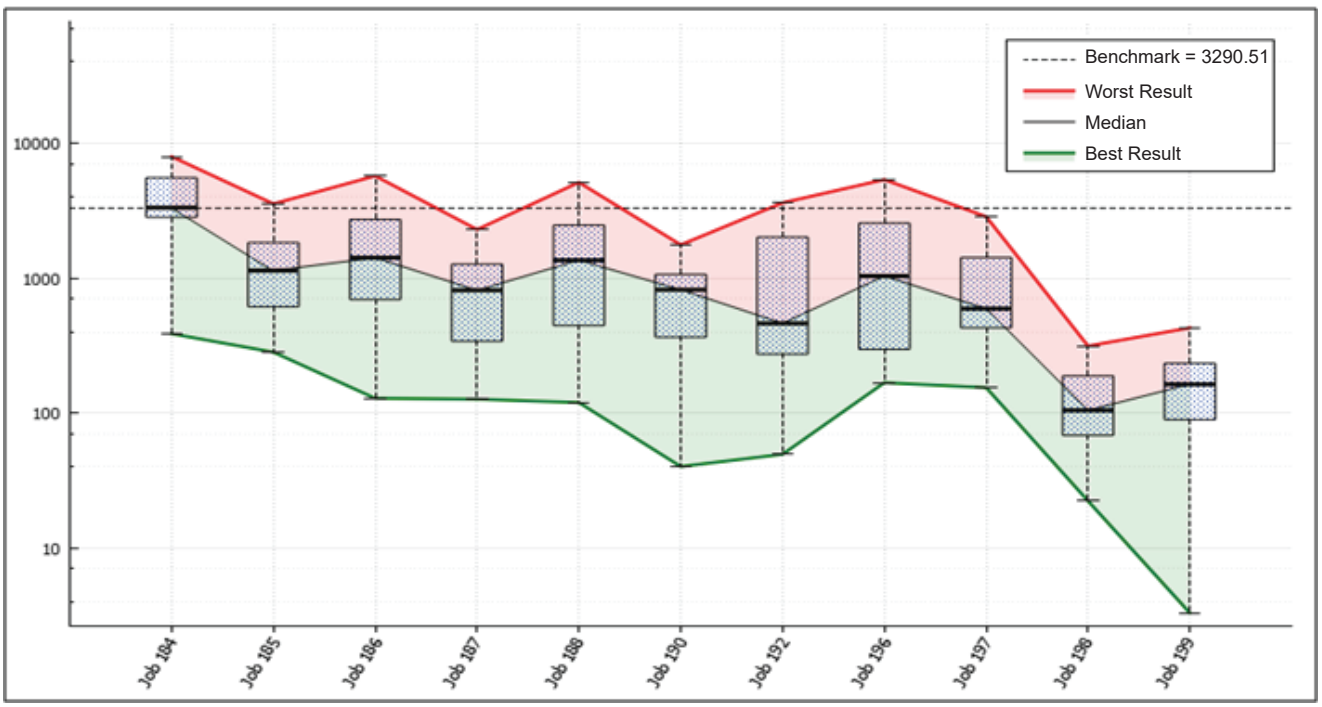


Figure 3