Planning for Timing Closure in FPGA Designs

Anyone who has implemented designs using FPGAs over the years knows how important timing closure is and how tough it sometimes can be to achieve. The inability to achieve timing closure can impact many things.

- It can delay your time to market as your engineering team struggles to achieve the design's timing goals which can be costly from an R&D and product market perspective.
- Sometimes a larger device or a faster speed grade must be used to achieve closure. This impacts the profitability of your product especially for higher volumes.
- New requirements can come in late to a project that requires additional logic. This will increase logic density which can negatively impact timing. A design that had once achieved its timing goals might now fail even if a small amount of logic was changed.

InTime

Website: www.plunify.com
Email: tellus@plunify.com

The vendor tools do a remarkably good job at design compilation for the general case but sometimes they can fall short

- **Compilation Options**
  There are a large number of options on the compilation tools because compilation algorithms sometimes need human help to determine the best way to compile a design. Even using the vendor supplied templates of compilation switches may not achieve the result you need. With scores of different compilation options it can be a daunting task for a designer to figure out which ones to use or change. This is further complicated since a set of compile options, that once worked for the design at one point in the development cycle, might not work as well in future iterations of the design.

- **Moore's law**
  Moore's law is still holding for increases in logic density from generation to generation but no longer does it hold for the traditional gains in fmax that had been seen for decades. The implication is that compile times will get longer since processors speeds are not increasing but FPGA logic capacity continues to increase. The vendor compilation tools continue to get more sophisticated to cope with this but it is an uphill battle. Certainly, multi-threading and parallel processing are a means to help with this but not all compilation algorithms easily lend themselves to parallelism.

Why is timing closure so difficult for the vendor FPGA compilation tools?

"In real life applications this is an impractical method since for an N as small as 20 there are 20! ≈ 10^{18} possible routes (search space) to analyze."

From a theoretical computer science perspective, optimization for synthesis, placement and routing fall into a class of algorithms that are called NP-complete. These are one of the most difficult classes of problems to resolve due to the overwhelming number of possible solutions you must sift through to find an optimal solution.

- Algorithmic cost or computational complexity is based on how many calculations are needed to complete a solution for a set of N elements. This helps to roughly determine the amount of real time it takes the algorithm to solve a problem. There are polynomial complete (P-complete) problems that have an approximate cost of N^X, a simple polynomial, where N is the variable number of elements and X is a fixed constant. An NP-complete algorithm has a cost that grows much larger than a P-complete algorithm for a large enough N. These have non-deterministic polynomial cost functions such as N! (factorial) or X^N (exponential) which grow very quickly in size with increasing N. For the case of FPGA compilation the cost is N! to find the optimal solution.

- As an aside, if you can find a means of converting an NP-complete to a P-complete problem you can make a million dollars since this is one of the Millennial Prize problems.

- A classic example of an NP-complete problem is the traveling salesman problem. A salesman must fly to N different cities. What is the best order to visit each of these cities once to minimize the amount of time he is traveling? There are N! (N factorial) possible solutions since there are N! different routes to visit the cities. To find the optimal solution you compute the travel time for each possible route then select the one with the lowest value. In real life applications this is an impractical method since for an N as small as 20 there are 20! ≈ 10^{18} possible routes (search space) to analyze. A more practical approach to this problem uses heuristic methods to more efficiently find a good route rather than using a brute force search to find the optimal route. The pitfall with this heuristic approach is you generally do not know how close you are to the optimal solution.

- Imagine a 10 by 10 array of 100 FPGA look-up tables (LUTs). To find the optimal placement for these, based on timing, you have to examine all possible placements of the LUTs within the array since there is no direct way to directly compute the best solution. This has a search space that is 100 factorial in size (100! = 100x99x98x97x...x6x5x4x3x2x1 ≈ 10^{157}). Sifting through each permutation of placements looking for the best placement would take longer to compute than the age of the universe by many orders of magnitude even if you used every computer in the world! Modern FPGAs have tens of thousands of LUTs so the possible search space for this is beyond belief. On top of that, throw in the search spaces for synthesis and routing and you soon realize a brute force solution is certainly not feasible.

- Knowing that finding the optimal solution for a given design is out of the question, the FPGA vendors must deploy a set of heuristic algorithms for design compilation that give a good solution but also do not take too long to run. For the largest FPGAs or for really tough designs the vendor software teams try to come up with a reasonable solution that typically requires less than 12 hours of compute time. This is a pragmatic tradeoff between getting the best timing score possible versus limiting the computation to a reasonable amount of runtime. However, this will sometimes compromise timing closure. Each and every year, one of the top goals for the vendor's software development team is to improve the timing closure capabilities of their compilation tools. Given the nature of this difficult problem, this has been a top goal for more than 20 years and will continue to be a top goal for many years to come.

What to do if you cannot achieve timing closure?

- Modify your design to reduce levels of logic, restructure clocking trees or introduce pipelining
- Remove functionality from your design
- Move to a larger device or faster speed grade
- Try to do some floor planning or add/change constraints to help influence the compilation
- Ship product with slower performance
- Tweak compiler options in the hope that you will find the right settings to close timing
- Try to get help from the vendor FAEs
- Use InTime from Plunify

How does the InTime timing closure tool from Plunify work?

- InTime is a plug-in module for the Xilinx and Intel (formerly Altera) FPGA development environments.
- InTime applies machine learning by running a sequence of parallel compilations, utilizing the vendor's compile tools, to intelligently converge on a set of optimized compile tool options that are specific to your design.
- The improved compiler options found by InTime are used as a template in subsequent usages of the vendor's compile tools so synthesis, place and route have a better chance of meeting your design's timing goals.
- InTime requires no manual intervention or changes to your RTL.
- InTime can run inside your firewall on your systems utilizing the FPGA tools you have already purchased from the vendor. It can reside in a local environment that is completely under your control.
- Once the initial RTL is created, you can use InTime at any point in your development cycle.
- Using InTime means you don't have to be an expert on what all of the compile tool options are giving you and your engineers more time to focus on the design itself.
- The more of your designs that you use with InTime the better it can enhance its learning database, which always remains protected inside your firewall, so it can be even more effective with your designs in the future.
- InTime cannot guarantee timing closure for all designs but it will almost always improve any design's timing score .
- Not only can InTime help you achieve your timing goals it can save you time and money as well.

InTime has worked successfully for scores of FPGA designers around the world

- InTime has established a successful track record with FPGA designers at both large and small companies and across many different types of FPGA designs. View customer testimonials here.
- Plunify is an official technology partner to Intel and Xilinx.

Try InTime for free!

- Apply for a trial version of InTime at http://www.plunify.com/en/free-evaluation/
- Additional information http://www.plunify.com/en/product/
- Contact Plunify at mailto:tellus@plunify.com if you have more questions or would like to purchase the product.