White Paper

Quartus® Prime

altera™
An Intel Company

Advanced Physical Synthesis in the Quartus® Prime Pro Edition Software

Authors

Mahesh A. Iyer
Intel Senior Fellow

Junaid Khan
Senior FPGA Development Tools Manager

Selvin Quadros
FPGA Development Tools Engineer

Nan Zhuang
Principal Engineer

Intel Corp.

Table of Contents

Abstract

Logic synthesis comprises techniques to optimize a logical representation of a user design to create an optimized logical netlist. In a typical design implementation flow, this logical netlist is taken through a place and route flow to physically place the logic elements and route all the connections with them in a manner that is legally compliant to the underlying FPGA architecture, layout, and design rules. Place and route operations perform concurrent optimization of multiple metrics such as wiring usage, timing, utilization, and routing congestion. Physical synthesis comprises techniques to optimize the logical netlist further for metrics such as timing and area during and after the place and route process. Physical synthesis optimizations perform surgical netlist modifications by considering the physical information that are readily available during and after the place and route process. The key objective of physical synthesis is to improve the Quality of Results (QoR) metrics, such as maximum frequency (f_{MAX}) and user design area, with minimal compile time overhead. In this paper, we discuss the advanced physical synthesis optimizations performed in the Intel® Quartus® Prime Pro Edition Software that achieve an average of about 13% f_{MAX} improvement with <10% compile time overhead on the Intel Agilex® FPGA family.

Introduction

Timing closure is a time-consuming process of any digital logic implementation flow and users spend a significant portion of their design cycles to achieve their desired timing goals. Key features in the Intel Agilex family of FPGAs, such as the register-rich Intel® HyperFlex® architecture and the flexible clocking architecture, were co-designed with the Intel Quartus software. To leverage these features, the Intel Quartus software compiler engines and flows were significantly revamped.

The high-level compilation flow in the Intel Quartus Prime Pro Edition Software is shown in Figure 1. The FPGA design implementation flow starts with logic synthesis, followed by a plan stage that performs periphery placement, clock allocation, and early global retiming. Next, in the place stage, the logic elements are placed and clustered to the adaptive logic modules (ALMs) and logic array blocks (LABs) on the FPGA. The place stage also legally positions all the block random access memory (RAM) and digital signal processing (DSP) elements in the user design on the underlying FPGA fabric. In the route stage, all the connections in the user design are routed using the programmable routing fabric of the FPGA.

With the Intel Hyperflex architecture in the Intel Agilex FPGA, the Intel Quartus software compiler flow was significantly revamped to be highly retiming-centric. Various parts of the flow such as synthesis, placement, and routing were made retiming-aware so they can fix critical paths that cannot be fixed by retiming. The Intel Quartus software performs late-stage global retiming and clock skew optimization with accurate delays very easily, without the need to reroute connections. The software performs physical synthesis optimizations throughout the place and route flow – during placement, after placement, and after routing and global retiming.

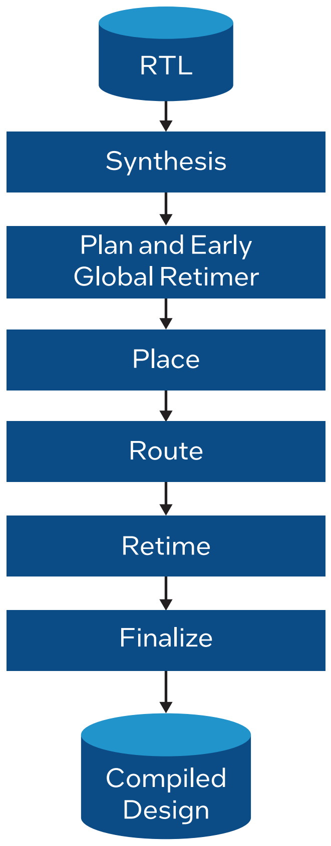


Figure 1. Quartus Pro high level compilation flow.

The Intel Quartus Prime Pro Edition software performs optimizations in all these compilation stages. Earlier in the compilation flow, the optimizations mostly focus on reducing the logical depth, while balancing the resources required to implement the design in terms of ALMs and LABs, block RAM, and DSP elements. As we progress further in the flow, more accurate physical information progressively makes the timing information more accurate. This progression in timing accuracy starts exposing the real critical paths during the place and route process. Physical synthesis is exactly targeted to optimize such paths during the place and route process through netlist optimizations, without perturbing the optimization convergence of the placement and routing engines.

For the rest of this paper, we present details where physical synthesis is performed in the overall place and route flow of the Intel Quartus software and the physical synthesis optimization techniques used in the Intel Quartus Prime Pro Edition Software. We also show our experimental results on a large set of customer designs that show significant and industry-leading improvements in f_{MAX} with minimal compile time overhead.

Physical Synthesis Stages in the Intel Quartus Software

Physical synthesis in the Intel Quartus software is performed in three different stages.

Post-Global Placement

Global placement performs flat analytic placement of the logical netlist comprising the basic logic elements (BLEs) such as the lookup tables (LUTs), flip-flops (FFs), block RAM, and DSP elements, concurrently optimizing for wiring usage, timing, density, and routing congestion. Placement locations after global placement are approximate and not fully legal to the underlying FPGA architecture. This is the first stage in the Intel Quartus software flow that has the physical information for all the core fabric elements in the user design. This allows for more realistic timing information considering the physical locations of the BLEs and the modeled physical connections between the BLEs, compared to a pure logical netlist as seen in the logic synthesis phase.

We apply physical synthesis optimizations after global placement to condition the netlist for further placement operations. Since the design is not fully legalized at this global placement stage, physical synthesis has more flexibility in optimizing the netlist for wiring usage, area, and timing. Approximate placement locations are also computed for any newly created BLEs during post-global placement physical synthesis, ensuring that the operations strictly improve the results using an elaborate costing mechanism, while maintaining the spirit of the global placement solution computed by the solver.

After global placement, the Intel Quartus software physical synthesis process applies several netlist optimization techniques, such as register duplication, retiming, resynthesis, shift register to RAM inferencing, DSP register unpacking, and more, details of which are discussed in a later section.

Post-Detailed Placement

After post-global placement physical synthesis, the netlist is optimally clustered into the physical elements of the underlying Intel FPGA architecture, through the creation of ALMs and LABs. All ALMs, LABs, block RAM, and DSP blocks are then legalized onto actual physical sites on the FPGA. This is followed by a fine-grained detailed placement step that further optimizes the physical placement of the various BLEs, ALMs, and LABs for wiring usage, timing, and routing congestion.

With a fully legalized and optimized placement after the detailed placement stage, the physical locations of the elements in the user design are near final, thereby further increasing the accuracy of timing analysis. We apply incremental physical synthesis at this stage to further optimize the netlist for any new critical paths that were not optimized during the global placement stage. Again, post-placement physical synthesis strictly improves the results using an elaborate costing mechanism and ensures that any netlist modifications are also completely legalized on the FPGA. Many optimization techniques including retiming, resynthesis, register duplication, placement adjustment, LUT rotation, and others are used in this stage of physical synthesis.

Finalize Stage

After detailed placement, the Intel Quartus software performs routing to optimally route all the wiring connections in the user design using the routing resources available on the underlying FPGA. The complete placed and routed netlist now provides very accurate timing information since all the wire delays are computed using the actual routed connections.

The Intel Quartus software leverages the 2nd generation register-rich Intel HyperFlex architecture in the Intel Agilex FPGA that provides programmable registers ubiquitously throughout the routing fabric. The software performs global retiming to reposition the registers in the design using the programmable Intel HyperFlex architecture registers on the routing fabric. There are several advantages to performing this late-stage global retiming. First, the timing information is very accurate at this stage, and the Intel Quartus software nicely optimizes the critical paths using global retiming. Second, the underlying FPGA architecture does not require rewiring or re-placing any logic or re-routing any connections – as the Intel HyperFlex architecture registers are already present on the routing fabric, and all the software needs to do is program these registers to indicate whether or not they are used in the design, as determined by the global retimer. This is an industry-leading software/hardware co-design methodology and is a unique optimization technique that is beyond the scope of even state-of-the-art application-specific integrated circuit (ASIC) Electronic Design Automation (EDA) tools. The hardware and software capabilities provided by the Intel HyperFlex architecture and global retiming are industry leading and contribute significantly to the leadership performance per watt advantages in Intel Agilex FPGAs.

The finalize stage in the Intel Quartus Prime Pro Edition Software executes after the global retimer. At this stage, the software leverages multi-corner, signoff-quality timing analysis and fixes any remaining hold violations in the design using routing optimization techniques. Next, with the accurate multi-corner signoff-quality timing information, we perform late-stage physical synthesis to further optimize the timing of the design. Like in pre-route physical synthesis, several optimization techniques are used at this late-stage physical synthesis, such as retiming, register duplication, DSP register unpacking, hold fixing, placement adjustment, routing adjustment, ALM and LUT input rotation, clock skew scheduling, and others. Like with the Intel HyperFlex architecture retiming, the Intel Agilex FPGA hardware architecture was enhanced to include a flexible and programmable clock skew architecture that was co-designed with the Intel Quartus software physical synthesis. This allows the software to leverage the combined power of retiming and clock skew scheduling to achieve superior fMAX. An important and unique highlight of the Intel Quartus software physical synthesis in the finalize stage is that it operates with highly accurate sign-off quality multi-corner timing information for setup and hold. It also ensures that the optimizations applied strictly improve the fMAX while keeping the design fully legalized and routed at any point in time during the optimizations. Furthermore, it accomplishes these optimizations using highly incremental engines during its elaborate costing mechanism thus providing industry-leading compile times.

Physical Synthesis Optimizations Details

In this section, we discuss the key optimization techniques that are used in the various stages of physical synthesis in the Intel Quartus Prime Pro Edition Software.

Shift-Register to RAM Inferencing

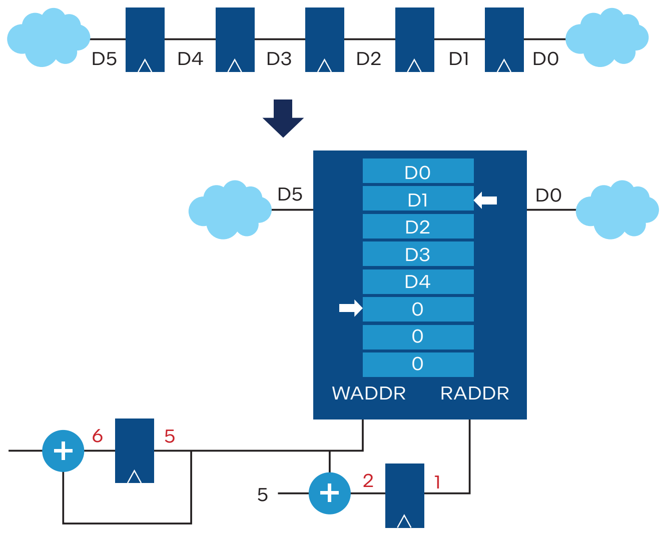


Figure 2. Shift Register to RAM Inference

Many user designs have shift registers that are generally inferred and optimized by the Intel Quartus software logic synthesis. One such key optimization to recover area is to transform deep and wide shift registers into a RAM-based implementation using the memory logic array block (MLAB) resources on the FPGA. With the retiming centric Intel Quartus software flow in the Intel Stratix® 10 and Intel Agilex FPGA families, we defer some of these optimizations of shift registers with depth ≤ 32 to be part of the physical synthesis optimizations. This allows the early global retimer in the plan stage of the Intel Quartus software to reposition these registers in the netlist and create balanced paths prior to global placement. Post-global placement physical synthesis optimizes the remaining shift registers into a RAM-based implementation, including the read-write address logic, using the MLAB resources on the FPGA. This is shown in Figure 2.

DSP Register Unpacking

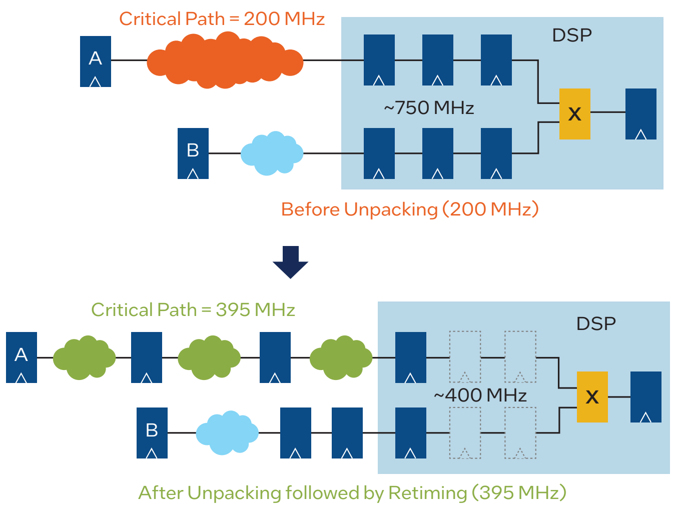


Figure 3. DSP Register Unpacking

In the plan phase of the Intel Quartus software flow, registers are aggressively packed into DSP blocks to save ALM and LAB area. However, later in the fitter flow when timing information becomes more accurate with progressively refined placement and routing information, the DSP blocks may run at a higher clock speed than the logic around them. In such cases, it is beneficial to selectively unpack registers from the DSP block into the logic that can be further retimed. The physical synthesis process in the Intel Quartus software performs these selective DSP register unpacking and retiming operations during various stages of physical synthesis when the input/output paths to/from the DSP blocks start becoming critical. Such optimizations are only accepted if they strictly improve the overall timing of the design, as illustrated in Figure 3.

Register Retiming

Logic retiming is a sequential optimization technique to improve performance. The objectives are to maximize performance with the minimum number of registers. Retiming modifies the netlist structurally through forward and backward moves across combinational and interconnect elements. It can be shown that, in general, logic retiming does not strictly preserve sequential equivalence, particularly related to the behavior of the original and retimed circuits under certain initial power-up conditions of the registers.

Through software/hardware co-design, we created two solutions to solve this sequential equivalence problem in the Intel Stratix 10 and Intel Agilex FPGA families. The first one introduced programmable initial states in the hardware architecture. The retimer computes the initial states as it moves the registers in compliance with the functionality of the combinational nodes. The second solution introduces the notion of c-cycle retiming in software. Here, a retimed circuit is shown to be a c-cycle delayed replacement of the original circuit, where 'c' is pessimistically computed by the retimer. The reset sequence of the retimed circuit is computed by pre-pending 'c' empty clock cycles to the reset sequence of the original circuit.

The Intel Quartus Prime Pro Edition Software supports both solutions for retiming operations. In addition to the global retiming stages, physical synthesis also surgically fine-tunes the retiming solution combined with other optimization techniques, with capabilities to retime in and out of ALM and Intel HyperFlex architecture registers, including c-cycle retiming for maximum flexibility and f_{MAX}. Figure 4 illustrates how retiming can improve circuit performance. Our results on a large set of industrial benchmarks show significant f_{MAX} improvements from retiming on Intel Agilex FPGAs.

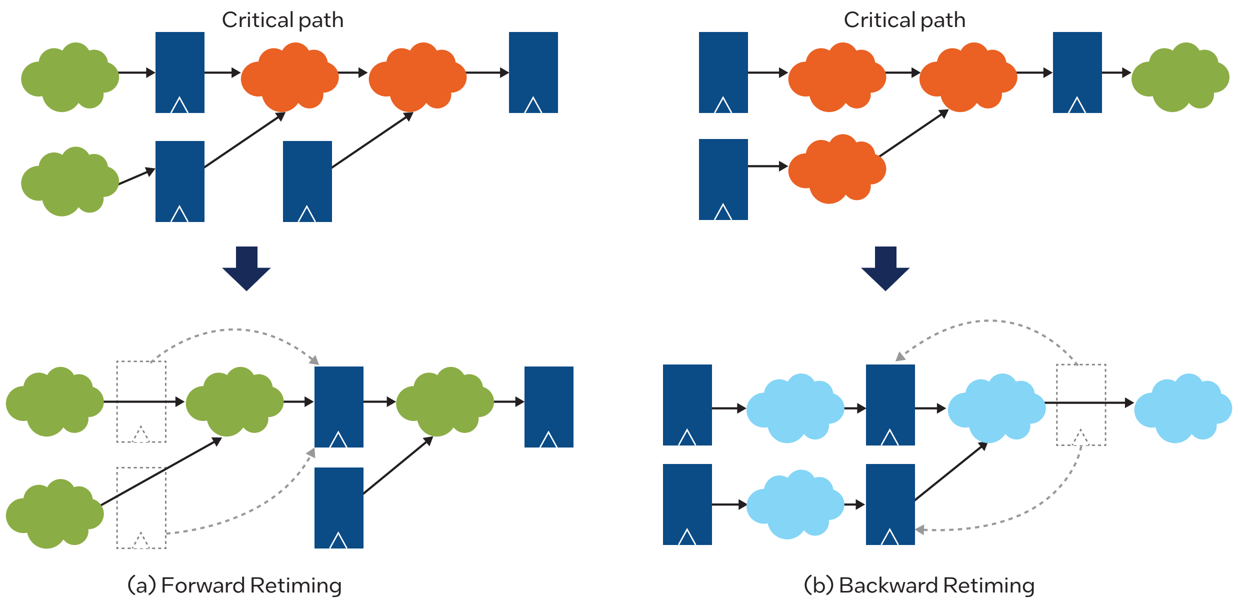


Figure 4. Register Retiming

Register Duplication

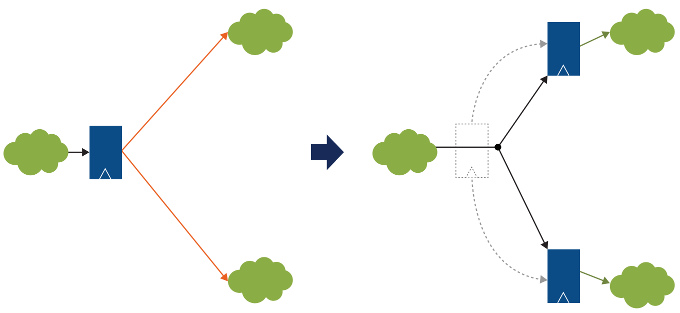


Figure 5. Register Duplication

Sometimes there are instances in the design where a register feeds two or more fanout logic blocks that are placed far away from each other due to other pulling forces in the global placement of those fanout logic blocks. In such cases, the interconnection wires between the register and one or more of these fanout logic blocks can become critical due to large wire delays. Such critical paths are solved by physical synthesis by simply duplicating the register and moving the critical registers closer to the fanout logic blocks they feed, as illustrated in Figure 5. This operation ensures that the timing of the design is improved before physical synthesis accepts the moves.

Logic Resynthesis

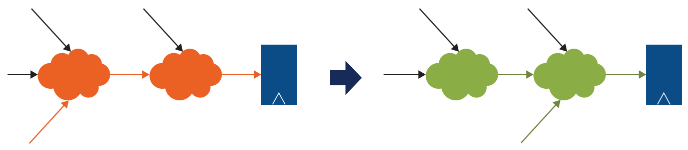


Figure 6. Re-Synthesis

Logic synthesis optimizes the depth of the user logic using the primitives of the underlying FPGA architecture. As these logic elements get placed and routed, some signals become more critical than others. To address this problem, physical synthesis resynthesizes certain cones of logic surgically using a Boolean formulation to move critical signals forward in a cone of logic to reduce their overall delay on the critical path. The operations are deeply interfaced with physical timing analysis, and only moves that improve the critical path delay are accepted. The newly created logic is also placed and routed legally on the FPGA device.

LUT, ALM, and Routing Adjustments

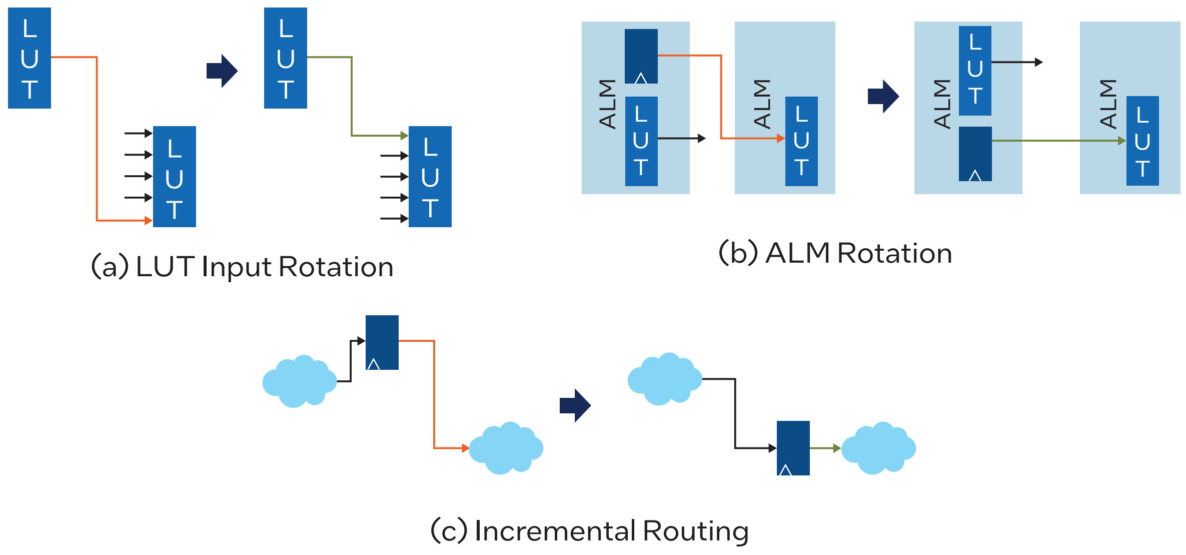


Figure 7. Routing Adjustment

Routing plays a significant role in determining the interconnect delays, which dictate the overall timing of the design. Physical synthesis performs several optimizations to readjust the routing and interconnections between logic elements. As shown in Figure 7(a), LUT input rotation is used to reassign critical signals to faster LUT inputs, since there is a variance of delays between each of the LUT inputs and outputs. ALM rotation is used to reassign critical ALMs to other ALMs within the same LAB to be closer to their connections outside the LAB. Physical synthesis also selectively and incrementally adjusts routing connections between logic blocks to improve their delays on critical paths. During this incremental re-routing process, any Intel HyperFlex architecture registers may also be rebalanced to further improve the interconnect delay, as shown in Figure 7(c).the moves.

Hold Fixup

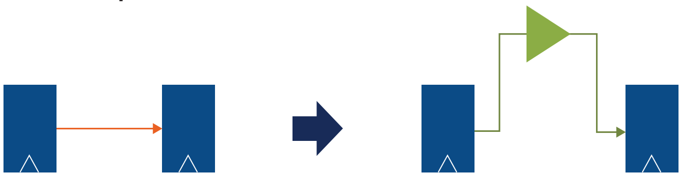


Figure 8. Hold Fixup

The Intel Quartus software ensures that the final design does not have any hold violations. To ensure this, the software has a special hold fixing step in the finalize stage so that such hold fixing is done with accurate sign-off quality multi-corner timing analysis. Two key techniques used to fix hold violations is to create extra delay through detoured routing, as well as to optionally insert buffer LUTs along very fast connection to meet minimum delay requirements. These techniques are illustrated in Figure 8. A key feature is that the Intel Quartus software physical synthesis in the finalize stage is setup and hold-aware when performing all its optimizations and in accepting moves that improve the overall design performance.

Clock Skew Optimizations

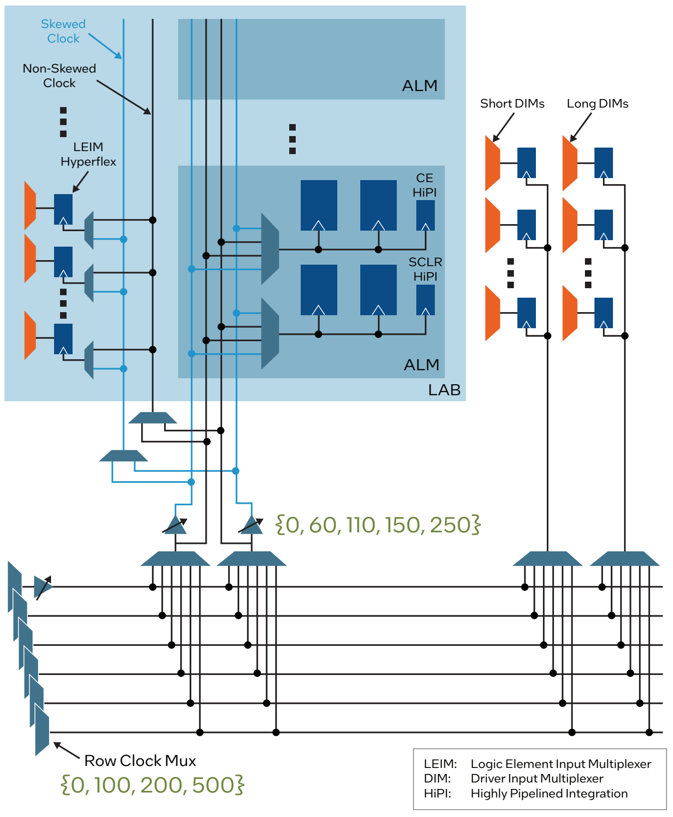


Figure 9. Intel Agilex Clock Skew Architecture

Clock skew optimization intentionally introduces skew on the clock input of certain registers to improve the overall setup timing of the design without introducing any new hold violations. Through software/hardware co-design for Intel Agilex FPGAs, we introduced several clock skew capabilities in the hardware architecture of LAB clocks and row clocks (as shown in Figure 9), and companion clock skew optimization technologies in the Intel Quartus software physical synthesis.

With accurate sign-off quality multi-corner timing analysis, the software determines optimal skew values for all register, DSP, and RAM clocks to improve the overall timing of the design. These clock skew optimizations have been architected to work in tandem with other physical synthesis operations, particularly retiming, such that they are complementary to each other in how they improve the timing of the design considering other trade-offs on logic utilization, routing resources, and compile time.

Incremental Engines

Physical synthesis in the Intel Quartus software has a detailed costing infrastructure that is used to determine if the attempted moves improve the design. This costing infrastructure heavily relies on the delay annotation and timing engines to get accurate timing information depending on which part of the flow is running physical synthesis. Physical synthesis performs surgical netlist modifications to improve the timing of the design. Since these modifications are done during the place and route flow, it is imperative that physical synthesis also determines placement and routing for the modified logic elements (depending on which part of the flow physical synthesis is running) before timing and costing the changes.

One of the key objectives of the Intel Quartus software physical synthesis is to improve the design's QoR, while having fast compile times. To enable this, the Intel Quartus software physical synthesis is powered by highly incremental engines listed below which allows it to try many optimization moves without bloating the compile time. The efficient implementation of these engines ensure that they only perform the necessary computations, while also taking advantage of parallelism.

- Incremental Placement: Places modified logic legally on the FPGA without perturbing the globally converged solution of the various placers
- Incremental Routing: Routes modified connections between logic blocks without perturbing the globally converged solution of the router
- Incremental Delay Annotation and Timing Analysis: Computes the updated delays and timing of all modified logic elements and connections as netlist, placement, and routing modifications are made. This timing analysis gets progressively more accurate as we progress through the Intel Quartus software flow.
- Incremental Netlist Management: Performs fast netlist modifications with the ability to undo the change if physical synthesis determines as such.
- Incremental Costing Infrastructure: Computes all the costs necessary for physical synthesis to determine if an attempted move should be accepted or rejected. These costs model many metrics such as timing, area, utilization, and routing congestion. This costing infrastructure uses the incremental delay annotation and timing analysis engines.

Functional Verification

Functional verification plays a significant role in ensuring that the functionality of the netlist implementations from the Intel Quartus software is identical to the user's register transfer level (RTL) design. Since physical synthesis performs many netlist transformations, Intel uses several techniques to validate the functional correctness of such transformations. While these techniques are routinely used by Intel across a broad set of design suites to ensure functional quality of the designs produced by the Intel Quartus software, we have also developed some partner flows for customers to perform some of these verifications. These techniques include:

- Formal verification with third-party verification tools
- Random design simulations (internal only)
- Sub-netlist simulations (internal only)
- Rewind retiming verification, that also performs initial state verification and c-cycle retiming verification (internal only)

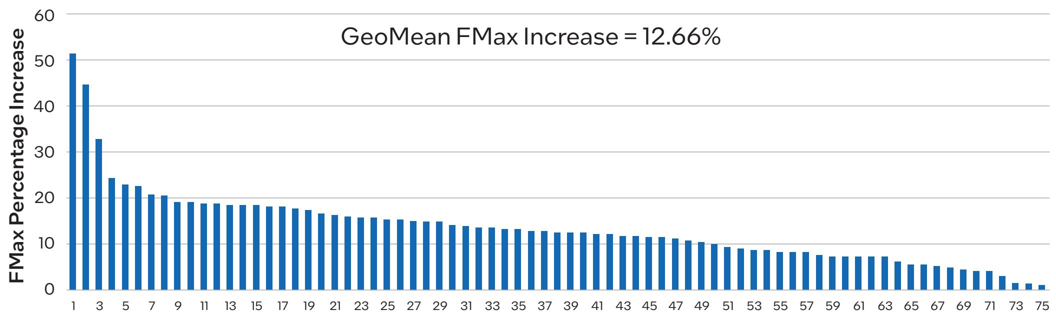
Results

Physical synthesis has been productized in the high-effort flow of the Intel Quartus Prime Pro Edition Software and is supported for Intel Arria® 10, Intel Stratix 10, and Intel Agilex 7 FPGAs. This flow is geared towards achieving maximum clock frequency, while maintaining good compile times.

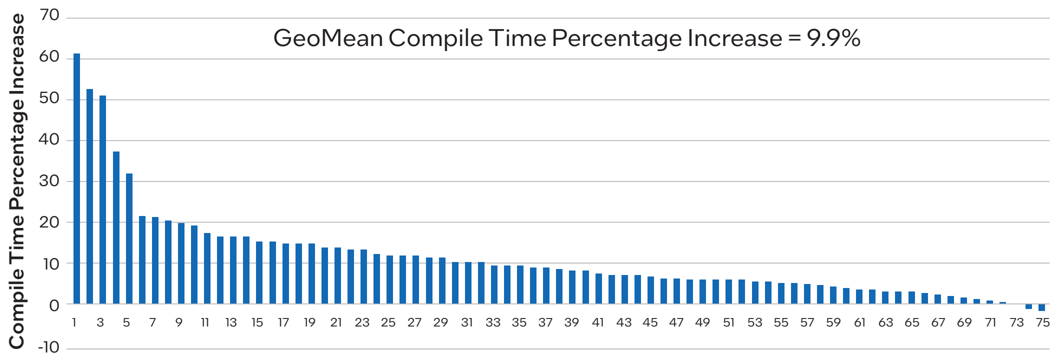
Figure 10 shows results from the Intel Quartus software physical synthesis on a large set of industrial benchmarks using Intel Agilex FPGAs. As shown in Figure 10(a), every design improved in fMAX, which is a very statistically significant result. This is also a validation of the accurate costing mechanism in physical synthesis that ensures the overall design is strictly improved, without causing signification optimization trajectory changes in downstream engines. The average improvement in fMAX was 12.66%, which is a speed-grade advantage. Figure 10(b) shows that the compile time increase from physical synthesis is <10% on the average. That is, for every % increase in fMAX, physical synthesis consumed <1% compile time.

Conclusion

In this paper, we presented the advanced physical synthesis technology in the Intel Quartus Prime Pro Edition Software. Physical synthesis is performed pervasively in the software flow and employs several advanced optimization techniques. The Intel Quartus software physical synthesis uses a formalized and elaborate costing mechanism to ensure that every move that it attempts through netlist, placement, or routing modifications strictly improve the overall quality of the design. The Intel Quartus software physical synthesis has the unique and differentiated capabilities in its use of highly incremental placement, routing, netlist modification, delay annotation, and timing analysis engines. These capabilities ensure that the compile time of the Intel Quartus software physical synthesis is low and the fMAX improvements it provides is large and statistically significant. The Intel Quartus software physical synthesis is a key technology that provides fMAX leadership in Intel Agilex FPGAs while maintaining very low compile time overheads, both of which are critical to boost designer productivity for timing closure.



(a) Intel Agilex® FMax Increase with Pysical Synthesis (ordered highest to lowest)



(b) Intel Agilex® Compile Time Increase with Physical Synthesis (ordered highest to lowest)

Figure 10. Physical Synthesis Results (a) FMax Improvement (b) Compile Time Increase

References

- S. Adya, L. Singhal, D. Grant, and M. A. Iyer, "Analytic Fitter", Proceedings Altera Technical Symposium, November 2015.
- M. A. Iyer, "Are You Ready to Re-Time Your Design?", Keynote talk, ACM International Workshop on Timing Issues in the Specification and Synthesis of Digital Systems, March 2017.
- M. A. Iyer, "CAD Opportunities with Hyper-Pipelining", Invited talk, Proceedings ACM International Symposium on Physical Design, March 2017.
- L. Singhal, M. A. Iyer, and S. Adya, "LSC: A Large-Scale Consensus-Based Clustering Algorithm for High-Performance FPGAs, Proceedings of the ACM/EDAC/IEEE Design Automation Conference, June 2017.
- M. A. Iyer, "Symbiosis in Action: Reconfigurable Architectures and EDA", Keynote talk, Proceedings of the 28th ACM/SIGDA International Symposium on Field-Programmable Gate Arrays, February 2020.
- I. K. Ganusov, M. A. Iyer, N. Cheng, A. Meisler, "Agilex™ Generation of Intel® FPGAs", Hot Chips: A Symposium on High Performance Chips, August 2020.
- C. Ebeling, H. Schmit, D. How, M. Iyer, S. Adya, "Sector-Based Clock Routing Methods and Apparatus", US Patent Number 9,922,157 issued on March 20, 2018.
- S. Adya, M. A. Iyer, and L. Singhal, "Method and Apparatus for Performing Clock Allocation for a System Implemented on a Programmable Device", US Patent Number 10,303,202 issued on May 28, 2019.
- M. A. Iyer, V. M. Kamath, and R. L. Walker, "Integrated Circuit Retiming with Selective Modeling of Flip-Flop Secondary Signals", US Patent Number 10,162,918 issued on December 25, 2018.
- M. A. Iyer and R. L. Walker, "Methods for Incremental Circuit Design Legalization During Physical Synthesis", US Patent Number 10,339,241 issued on July 2, 2019.
- M. A. Iyer, V. M. Kamath, and R. L. Walker, "Retiming with Fixed Power-up States", US Patent Number 10,296,701 issued on May 21, 2019.
- M. A. Iyer, V. M. Kamath, and R. L. Walker, "Retiming with Programmable Power-up States", US Patent Number 10,255,404 issued on April 9, 2019.
- M. A. Iyer, R. L. Walker, and V. M. Kamath, "Methods for Incremental Circuit Physical Synthesis, US Patent Number 10,936,772 issued on March 2, 2021.
- M. A. Iyer, "Methods for Delaying Register Reset for Retimed Circuits", US Patent Number 10,169,518 issued on January 1, 2019.
- M. A. Iyer, "Methods for Bounding the Number of Delayed Reset Clock Cycles for Retimed Circuit", US Patent Number 10,354,038 issued on July 16, 2019.
- D. Le, M. A. Iyer, and I, Milton, "Method for Retiming with Hybrid Initial States", US Patent Application filed on May 31, 2017.
- L. Singhal, M. A. Iyer, and S. Adya, "Method and Apparatus for Performing Large-Scale Consensus-Based Clustering", US Patent Number 10,162,924 issued on December 25, 2018.
- M. A. Iyer, "Method and Apparatus for Verifying Structural Correctness in Retimed Circuits", US Patent Number 9,824,177 issued on November 21, 2017.
- M. A. Iyer, "Method and Apparatus for Verifying Initial States Equivalence of Unchanged Registers in Retimed Circuits", US Patent Application filed on March 24, 2016.
- M. A. Iyer, "Method and Apparatus for Verifying Initial States Equivalence of Changed Registers in Retimed Circuits", US Patent Number 10,706,203 issued on July 7, 2020.
- M. A. Iyer, "Methods for Verifying Retimed Circuits with Delayed Initialization", US Patent Number 10,372,850 issued on August 6, 2019.
- M. A. Iyer and V. M. Kamath, "Method and Apparatus for Reducing Constraints During Rewind Structural Verification of Retimed Circuits", US Patent Number 10,489,535 issued on November 26, 2019.
- M. A. Iyer and V. M. Kamath, "Method and Apparatus for Performing Rewind Structural Verification of Retimed Circuits Driven by a Plurality of Clocks", US Patent Number 10,157,247 issued December 18, 2018.
- M. A. Iyer, "Method and Apparatus for Verifying Structural Correctness in Retimed Circuits", US Patent Number 10,671,790 issued on June 2, 2020.
- M. A. Iyer and V. M. Kamath, "Method and Apparatus for Performing Rewind Structural Verification of Retimed Circuits Driven by a Plurality of Clocks", US Patent Number 10,922,461 issued February 16, 2021.

altera™
An Intel Company

Intel technologies may require enabled hardware, software or service activation.
No product or component can be absolutely secure.
Your costs and results may vary.
© Intel Corporation. Intel, the Intel logo, and other Intel marks are trademarks of Intel Corporation or its subsidiaries. *Other names and brands may be claimed as the property of others.

WP-01327-1.0