WHITE PAPER

FPGA

Understanding How the New Intel® HyperFlex™ FPGA Architecture Enables Next-Generation High-Performance Systems

Intel Stratix® 10 FPGAs and SoCs leverage the Intel HyperFlex FPGA Architecture to deliver 2X the core clock frequency performance of previous generations.

Authors

Mike Hutton
Architect
Intel Programmable Solutions Group

Table of Contents

Introduction

To address the ever increasing bandwidth requirements of next-generation high-performance systems, FPGA vendors are continually making incremental improvements in their device architectures. Even with these advanced architectures, designers often resort to implementing their designs using very wide on-chip buses. In fact, on-chip buses of 512, 1,024 or 2,048 bits wide are increasingly common. Although this method improves data throughput in the FPGA core, these wide buses consume significant fabric resources and power. Also, as the FPGA fills up, routing resources can become congested and the core clock frequency may be limited.

Another way to increase bandwidth is to implement the design in an FPGA fabricated using the most advanced process node, hoping to benefit from the higher transistor switching speeds that are available with the latest technology. However, as geometries continue to shrink, the interconnect delays between the logic cells dominate the total delay in the FPGA and this limits the effectiveness of the higher transistor switching speeds. Fundamentally, conventional FPGA architectures cannot keep up with tomorrow's performance demands.

The need for bandwidth

Optical transport network (OTN), wireline, military, and high-performance computing applications require ever increasing bandwidth. The need to move greater quantities of data has resulted in increasing datapath widths inside the FPGA. The amount of data that can be moved through the routing architecture is a function of the number of wires used and the speed (f_{MAX}) of the wires. The number of wires available is a function of technology; it is derived from the size of the device and the minimum pitch of wires in the technology.

Routing architectures can make the wires more efficient by using hierarchy (for example, local routing in logic array blocks (LABs) and global routing on horizontal and vertical interconnect lines) and optimization techniques. However, doubling the number of wires simply adds die area and increases power dissipation. The speed of routing wires is technology driven (the RC delay on a wire), and is subject to the FPGA architecture and the design implementation. For example, pipelining a design can increase the clock speed without increasing the number of wires, which increases bandwidth for the same resources.

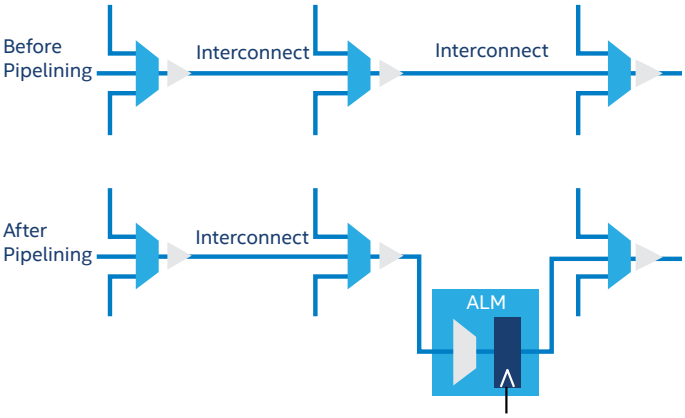

Figure 1. Added Delays with Conventional Pipelining

The need for efficiency

When designers pipeline a design for greater performance, they add registers to the design. The traditional methodology of building register-look-up table (LUT) pairs that is present in all existing FPGA core architectures means that logic is sacrificed to reach the added pipeline registers. Pipelining in conventional architectures also incurs a delay cost because a signal needs to be routed into and out of a logic block. The result is diminishing returns for the pipelining technique, especially when routing delays dominate the total delay. Figure 1 shows before and after examples of conventional pipelining, and the added delays due to routing into and out of the added register.

The need for improved clocking

As clock speeds increase, clock skew becomes increasingly important. Conventional FPGA core architectures have concentrated on balanced clock trees, which minimize deterministic skew. This method has served well for designs up to 500 MHz, but to break the 500 MHz barrier and reach speeds of up to 1 GHz, a next-generation clocking solution is needed. The solution must localize clocks to minimize local variation and skew, as well as provide a flexible network that services the numerous clocks that are common in high-performance designs.

The Intel® FPGA HyperFlex™ solution

To address these challenges, Intel Stratix® 10 FPGAs and SoCs introduce (formerly Altera® Stratix 10 FPGAs and SOCs) an entirely new core architecture, the Intel HyperFlex FPGA Architecture. The innovative Intel HyperFlex FPGA Architecture supports previously unimaginable levels of performance: 2X the core performance compared to previous-generation high-performance FPGAs. This performance level is not possible with conventional architectures. To take advantage of the Intel HyperFlex FPGA Architecture, you use familiar techniques: register retiming, pipelining, and design optimization. These techniques can speed up designs on conventional architectures. However, when combined with the Intel HyperFlex FPGA Architecture, the result is designs that run at blazing fast speeds with core clock rates up to 1 GHz.

The Intel HyperFlex FPGA Architecture

Intel Stratix 10 devices have a redesigned core architecture that includes additional registers, called Hyper-Registers, everywhere throughout the core fabric. These registers are available in every interconnect routing segment and at the inputs of all functional blocks. The Hyper-Registers provide a fine-grained solution to the problem of how to increase bandwidth and improve area and power efficiency. With many more registers that are easy to access, you can retime registers to eliminate critical paths, add pipeline registers to remove routing delays, and optimize your design for best-in-class performance. When Hyper-Registers are used to implement these techniques, all FPGA logic resources are available for logic functions instead of being sacrificed as feed-through cells to reach conventional LUT registers.

To keep up with the high-performance of the core fabric, the dedicated function blocks in the FPGA core—such as M20K memory and floating-point digital signal processing (DSP) blocks—have been redesigned to support operation at clock speeds up to 1 GHz.

To make it easy to use the Hyper-Registers, the Intel Quartus® Prime software includes a Hyper-Aware design flow with:

- Post place-and-route performance tuning for accelerated timing closure
- Hyper-Aware synthesis and place-and-route for efficient pipelining
- Fast Forward Compilation to explore performance enhancement options

To address the need for a flexible clock network, Intel Stratix 10 FPGAs and SoCs include programmable clock tree synthesis. This ASIC-like clocking helps mitigate skew and uncertainty. It also lowers power dissipation by intelligently enabling clock network branches.

Intel Stratix 10 FPGAs and SoCs use Intel's 14 nm Tri-Gate (FinFET) process technology. The combination of the new Intel HyperFlex FPGA Architecture and the industry-leading FinFET process technology allows Intel Stratix 10 devices to achieve 2X the core performance compared to previous-generation high-performance FPGAs.

Hyper-registers

With 90 nm Stratix II FPGAs, Intel was the first FPGA vendor to shrink the critical path depth with 6-input LUTs. In 28 nm Stratix V FPGAs, Intel introduced time-borrowing latches to allow automatic micro-retiming of clock and data signals.

With 14 nm Intel Stratix 10 devices, Intel is the first FPGA company to introduce an entirely new "registers everywhere" core architecture filled with bypassable retiming and pipelining registers. This method breaks the link between the functional registers in the adaptive logic module (ALM) itself, and the Hyper-Registers used for retiming and pipelining critical paths and improving design efficiency.

The Intel HyperFlex FPGA Architecture is built for retiming and pipelining high-performance designs. All routing segments have an optional Hyper-Register built into the programmable routing multiplexer that allows the routing segment to be registered or combinational. These Hyper-Registers are available everywhere throughout the core

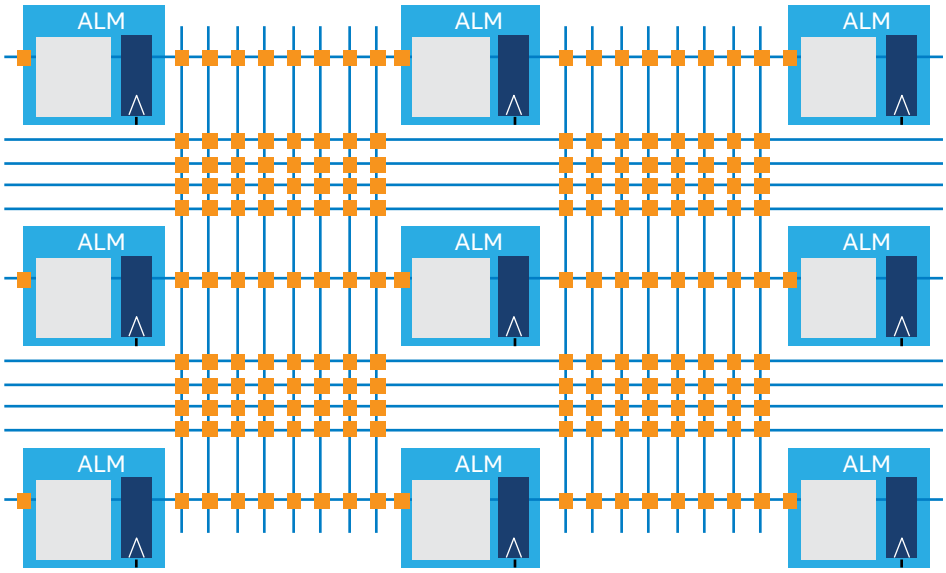

Figure 2. "Registers Everywhere" Intel HyperFlex FPGA Architecture

fabric as shown in Figure 2. The Hyper-Registers are represented by the small squares at the intersection of every horizontal and vertical routing segment.

With this architecture, there is no need to use an ALM to find a pipeline register. Every horizontal and vertical interconnect line in the device contains a Hyper-Register that can be turned on or off by configuring the FPGA.

Hyper-Registers are simple, one-input one-output bypassable registers without routing multiplexers on the input. You control these registers with configuration bits. They are inexpensive and do not add significant silicon area to the device. Because Hyper-Registers are ubiquitous in the core fabric, designers are not limited by the number of registers in their design. They can retime and pipeline as needed without consuming additional LAB resources. In many cases, the design uses fewer LAB resources because registers are implemented using the Hyper-Registers in the routing instead of partially consuming an ALM simply to use its register.

The HyperFlex Advantage

Because the Hyper-Registers are included in the interconnect routing architecture, timing optimizations can be done after place-and-route, without changing the design's routing. The Intel Quartus Prime software can easily find and use the Hyper-Registers during retiming operations. Figure 3 compares a conventional routing multiplexer and a HyperFlex routing multiplexer with the included Hyper-Register.

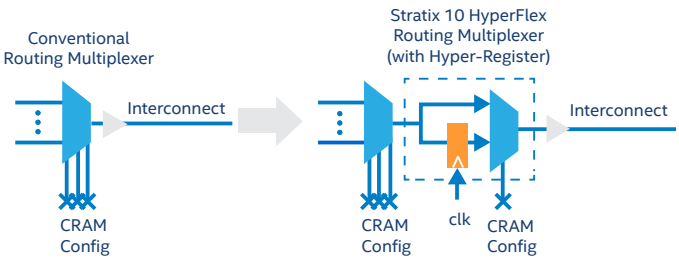

Figure 3. Comparing Conventional and Hyperflex Routing

The Hyper-Registers allow you to take advantage of traditional performance enhancement methods—retiming, pipelining, and optimization—implemented in a new and better way. When implemented using the Hyper-Registers instead of the ALM registers, these techniques are referred to as Hyper-Retiming, Hyper-Pipelining and Hyper-Optimization. Table 1 summarizes the performance gains achieved when these techniques are used in sequence, giving a three-step process to maximize performance using the Intel HyperFlex FPGA Architecture.

Step	Architecture Advantage	Effort Required	Core Performance *
1	Hyper-Retiming	No change or minor RTL changes	1.5X
2	Hyper-Pipelining	Added pipelining	1.65X
3	Hyper-Optimization	Design dependent	2X or more

Table 1. Three-Step Process to Maximize Performance Using the Intel HyperFlex FPGA Architecture

* vs. previous-generation high-performance FPGAs

Hyper-retiming

In conventional architectures, software performs retiming by finding a nearby, unused ALM register and including it in the circuit. This retiming method is limited by the granularity of the ALM register placement:

- The unused ALM may not be located conveniently, causing additional delay to include it in the design.
- There is a delay overhead to route through the ALM to the register.

- If software is trying to retime a wide bus (512 bits, 1,024 bits, or wider), retiming requires a large number of additional logic cells.
- The algorithms required to determine the best location for a retimed register are difficult.

Figure 4 shows a routing example before and after retiming with conventional architectures.

In the new HyperFlex core architecture, the Hyper-Registers are used to enable fine-grained Hyper-Retiming. The Intel Quartus Prime software retimes the path by moving the register out of the logic cell and into the interconnect. Because there are Hyper-Registers available in every routing segment, there are many registers locations available, making the optimization easy.

With Hyper-Registers, the retiming granularity is extremely fine; it is the delay of an individual routing wire, which is a few tens of picoseconds. The compromises made when trying to locate the retiming registers in conventional architectures, shown in Figure 4, are unnecessary with the Intel HyperFlex FPGA Architecture. Therefore, paths that are a few nanoseconds long can be split perfectly during Hyper-Retiming as shown in Figure 5.

Hyper-Retiming does not affect existing LABs and ALMs, which means that there is no incremental placement or routing required and no significant impact on compilation time. To retime a register, the register location is simply pushed into the routing to its naturally balanced final location (see Figure 5) after place and route. This feature is a tremendous benefit for designs with wide data buses that require hundreds or thousands of additional ALMs to achieve retiming in conventional architectures, and typically require extensive rerouting.

- For more information about using the Intel Quartus Prime software to perform Hyper-Retiming, refer to the Using Intel Quartus Prime Software to Maximize Performance in the Intel HyperFlex FPGA Architecture technical white paper.

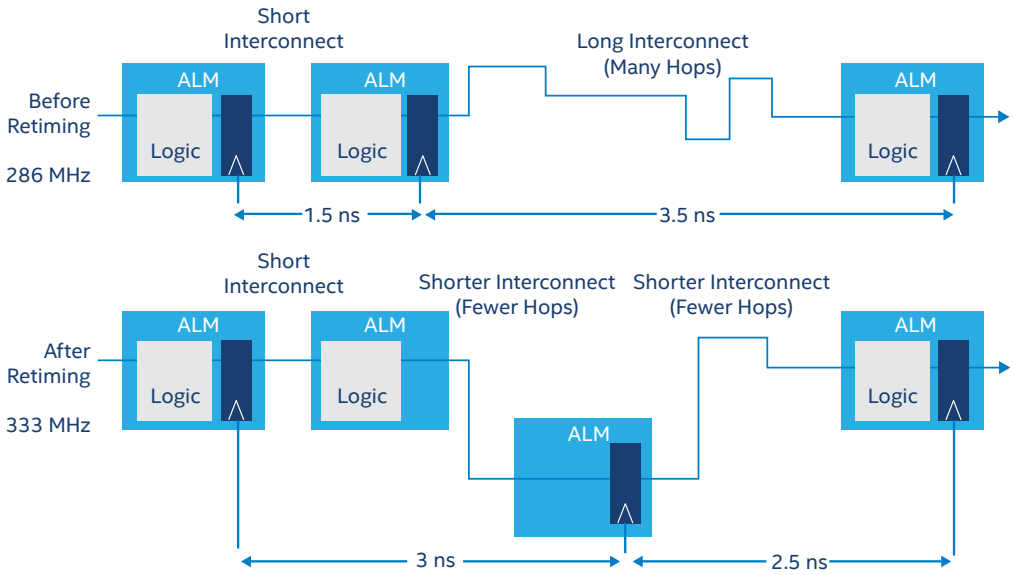

Figure 4. Retiming in Conventional FPGA Architecture

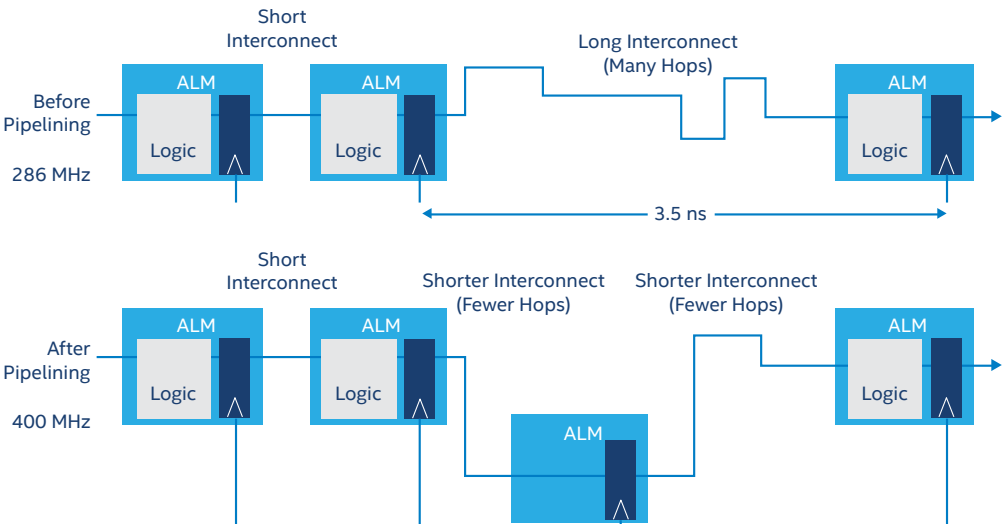

Figure 5. Hyper-Retiming in the Intel HyperFlex FPGA Architecture

Hyper-pipelining

Conventional pipelining suffers from the same drawbacks as conventional retiming, and the lack of register granularity reduces the effectiveness of this optimization. Conventional pipelining is inherently an iterative process because the number of pipeline stages required, and their optimum location, is unknown at the start of the process. Therefore, the design must be placed and routed several times while trying to converge on a pipelined solution that meets performance goals. Figure 6 shows a simple example before and after conventional pipelining.

When using the Intel HyperFlex FPGA Architecture, you can pipeline at will using the Hyper-Registers without bloating the size of the design. This process is known as Hyper-Pipelining. In many cases, a design with heavy register usage experiences a decrease in the number of ALMs required to implement the design because no "orphaned" registers are needed.

With what amounts to cost-free pipelining, you can use the technique aggressively, particularly in datapath and feed-forward logic. Figure 7 shows an example of Hyper-Pipelining. performance goals. Figure 6 shows a simple example before and after conventional pipelining.

When using the Intel HyperFlex FPGA Architecture, you can pipeline at will using the Hyper-Registers without bloating the size of the design. This process is known as Hyper-Pipelining. In many cases, a design with heavy register usage experiences a decrease in the number of ALMs required to implement the design because no "orphaned" registers are needed.

With what amounts to cost-free pipelining, you can use the technique aggressively, particularly in datapath and feed-forward logic. Figure 7 shows an example of Hyper-Pipelining.

Because the software can automatically retime the logic by moving registers into the interconnect, you only need to specify the required number of pipeline registers at the input to a clock domain or at a sub-design's pin logic. The Intel Quartus Prime software then moves the registers into the routing as required, after place and route, solving the

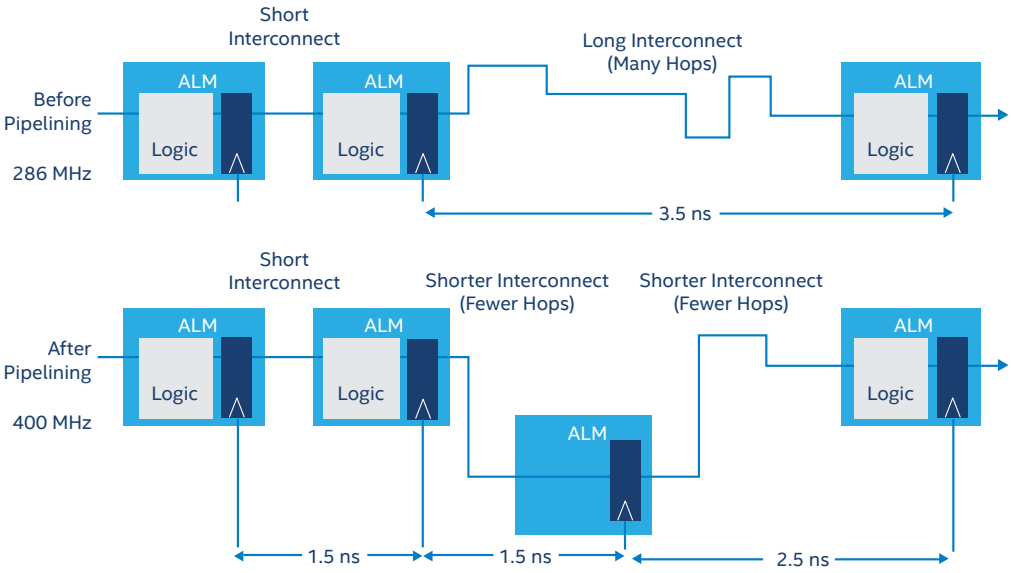

Figure 6. Pipelining in Conventional FPGA Architectures

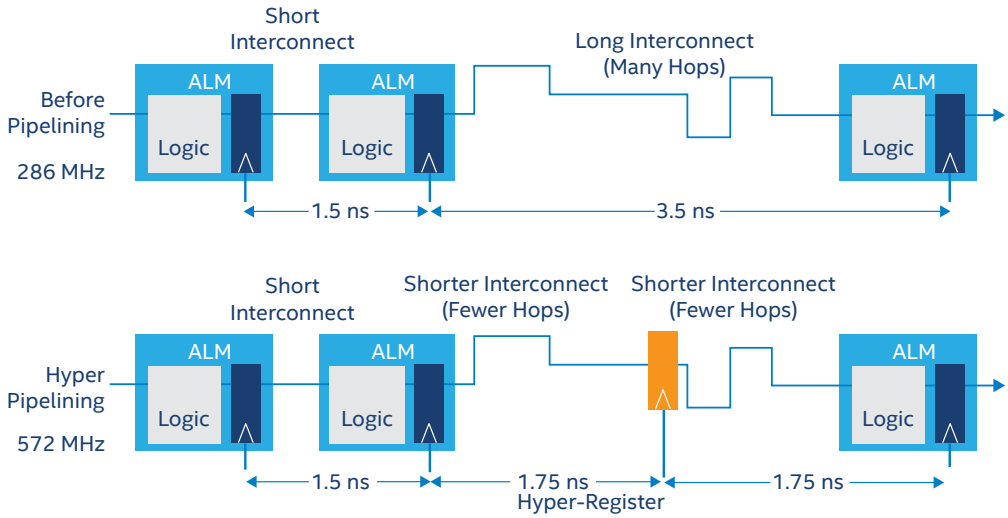

Figure 7. Hyper-Pipelining in the Intel HyperFlex FPGA Architecture

multiple iteration problem that exists with pipelining in a conventional architecture. Placing registers together in the RTL also allows for easy logic parameterization when intellectual property (IP) libraries target more than one clock frequency (fMAX). Figure 8 shows an example of placing additional pipeline registers at the input of a clock domain and the resulting movement of these registers into the optimum position in the interconnect routing.

Designers who make their design pipeline-friendly will experience the greatest benefit from the Intel HyperFlex FPGA Architecture. For example, latency-tolerant forms of flow control, such as using data-valid signals instead of high-fanout clock enables, allow the software to move registers easily through the FPGA core fabric.

For more information on design optimizations that take advantage of the Intel HyperFlex FPGA Architecture, refer to the Tailoring RTL Designs for Optimum Performance in the Intel HyperFlex FPGA Architecture technical white paper.

Hyper-optimization

After Hyper-Retiming and Hyper-Pipelining are complete, the performance gains may be so great in some sections of the design that other areas are exposed as bottlenecks that prevent further gains. These bottlenecks may be circuits such as long feedback loops or complex state machines that need to be evaluated on every clock cycle.

A common method for improving the design performance is to optimize specific portions of the design. For example, a design with a long feedback loop can limit the maximum frequency ($\mathbf{f_{MAX}}$). Redesigning the circuit to pre-compute the possible feedback values, and using a short feedback loop to select between them, increases the maximum frequency. With Hyper-Registers, this process can achieve higher speeds than are possible with conventional architectures because the pre-compute paths can be optimized using Hyper-Retiming and Hyper-Pipelining. Figure 9 shows an example of Hyper-Optimization; performing a Shannon decomposition (or Boolean factorization) to shorten the loop thus increasing the maximum frequency. Typically, you target these optimizations at control loops in which the performance benefit greatly outweighs any area cost of the additional logic required to achieve the factorization.

Flexible, high-speed programmable clock tree synthesis

Clocking in high-performance FPGA designs is becoming more challenging for designers. Conventional FPGAs have fixed global clock tree networks that are designed to support high-fanout, chip-wide, global clock domains. At

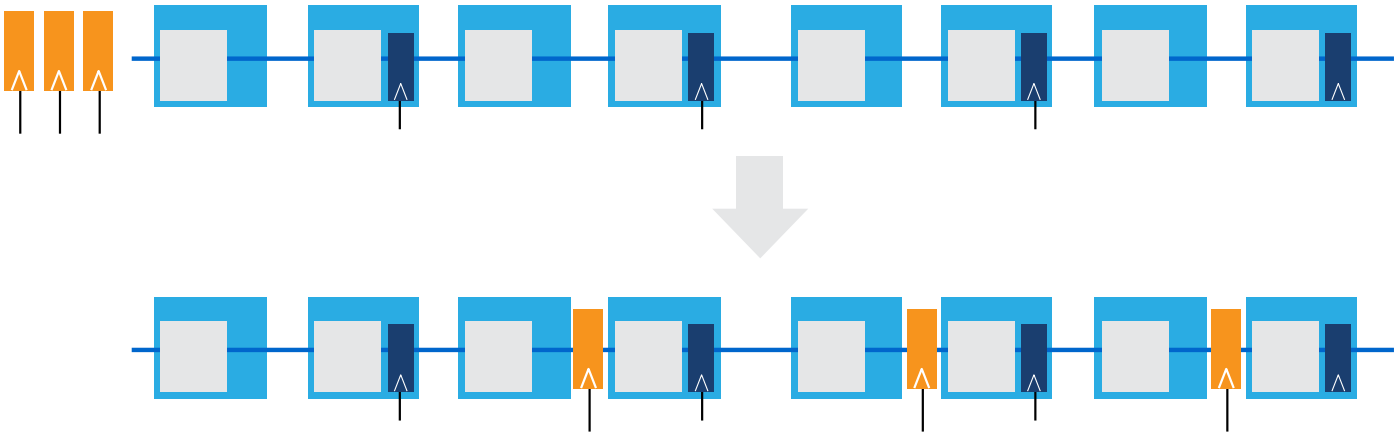

Figure 8. Placing Additional Pipeline Registers at the Input of a Clock Domain

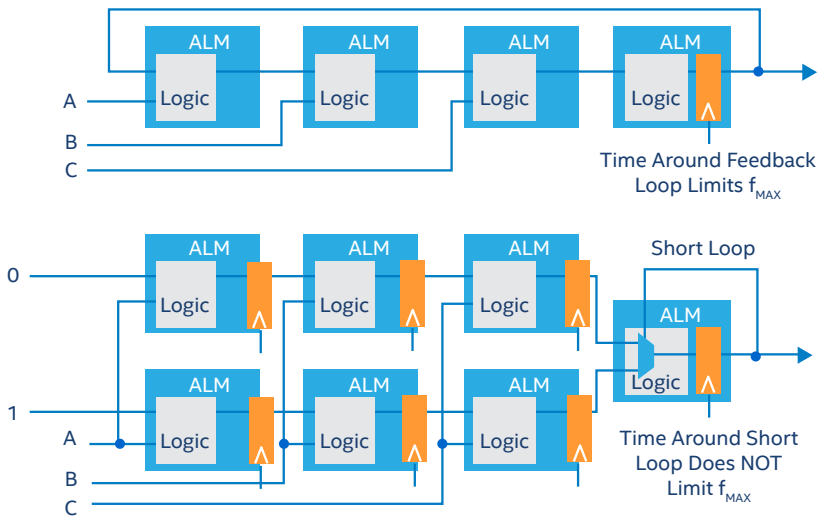

Figure 9. Hyper-Optimization of a Long Feedback Loop

GHz performance, however, clock networks require greater flexibility. Designers want to create time-shifted clocks for performance balancing and clock crossing, and generate dynamically gated clocks for rate-matching and system power management.

To address these needs, the Intel HyperFlex FPGA Architecture contains an entirely new clock structure with pre-routed clock paths onto which a design's clock region is synthesized (as is common for ASIC clock tree synthesis). This structure allows unprecedented flexibility to create small, localized clock domains. It also lets the software manage skew: taking advantage of beneficial skew when available and minimizing skew when necessary. Additionally, when required, this clock structure can be used to synthesize traditional global and regional balanced H-tree clocks for backwards compatibility.

The Intel Quartus Prime software manages the programmable clock tree synthesis; it synthesizes clock trees in an integrated fashion during place-and-route. Figure 10 shows an example of balanced and unbalanced clock trees synthesized by this approach.

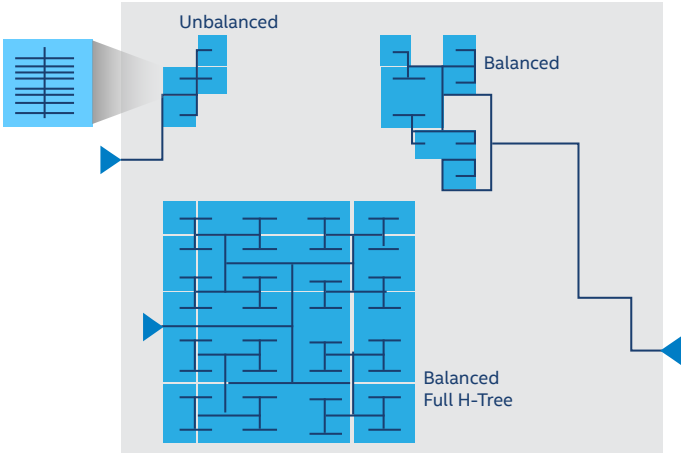

Figure 10. Balanced and Unbalanced Clock Tree Synthesis

Power efficiency

Intel Stratix 10 FPGAs and SoCs offer a significant power improvement over previous families in large part due to the use of Intel's 14 nm Tri-Gate (FinFET) process technology to fabricate the devices. Additionally, the Intel HyperFlex FPGA Architecture facilitates dramatic power savings. The increased performance of the Intel HyperFlex FPGA Architecture enables a 1,024 bit datapath clocked at 350 MHz to be implemented as a 512 bit datapath clocked at 700 MHz. As a result, the design fits into a device half the size. This change is neutral for dynamic power but reduces static power by half and also results in significant cost savings by using a smaller device. Alternatively, the designer has the freedom to use part of the performance dividend to improve clock speed, and convert the remaining performance dividend into power savings through reduced core power supply voltage or use a slower speed grade device.

Productivity

The increased core performance available with the Intel HyperFlex FPGA Architecture provides benefits that go beyond simply running the core at a faster clock rate. The additional performance results in easier timing closure, improved design team productivity, and shorter time-to-market for the product.

Conclusion

Meeting the needs of next generation, high-performance designs is a challenge with conventional FPGA core architectures. The value of techniques such as retiming, pipelining, and optimization are limited by the architecture itself. The Intel Stratix 10 Intel HyperFlex FPGA Architecture, with its "registers everywhere" approach, takes these optimizations to the next level, resulting in 2X the core performance compared to previous generation high-performance FPGAs.

Where to Get More Information

For more information about Intel and Stratix 10 FPGAs, visit **https://www.altera.com/products/fpga/stratix-series/stratix-10/overview.html**

[1] White Paper: A New FPGA Architecture and Leading-Edge FinFET Process Technology Promise to Meet Next-Generation System Requirements
[2] Technical White Paper: Using Intel Quartus Prime Software to Maximize Performance in the Intel HyperFlex FPGA Architecture
[3] Technical White Paper: Tailoring RTL Designs for Optimum Performance in the Intel HyperFlex FPGA Architecture