Zen and the Art of High Speed Design

Programmable Solutions Group (PSG)
mark.connor@intel.com

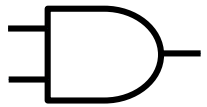
Agenda

- What you already know
  - Registers are free, levels of logic, fanout
- How we currently design
  - Design, Implement, Timing Analysis, goto 10
- How we have to change
  - Work with the tools, not against them
  - The fabric has to bend to the will of the designer, not the other way around
- What this looks like
  - Phys synthesis reports
  - Tighter feedback loop
  - Design for tools, not design for fabric

The FPGA world in 1996

- 10K30 was state of the art
  - 1,728 4-LUTs / Registers, 12kbits RAM blocks

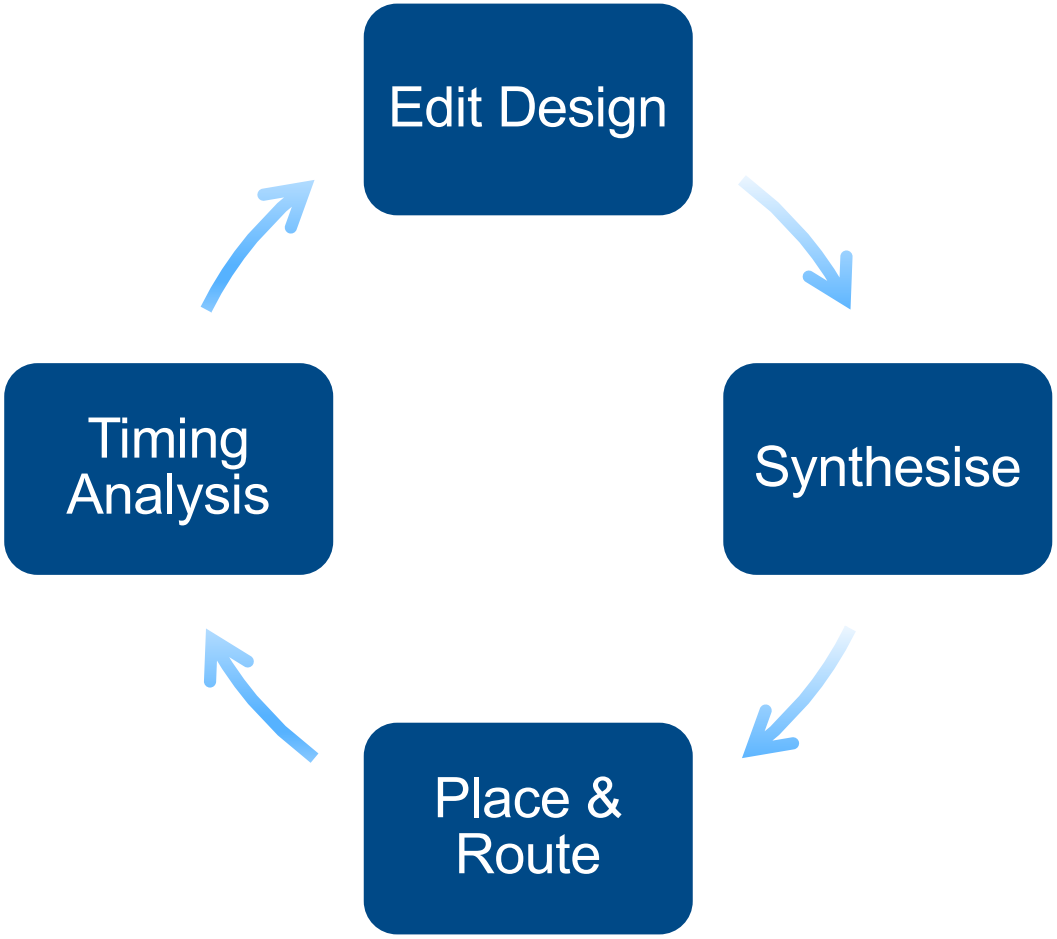
Schematics Rule OK!



- The three commandments
  - 1) Use lots of registers - Registers are free
  - 2) Logic depth is bad, see 1 - Because of the LUT delay, not the routing
  - 3) Fanout / distance is bad, see 2 and 1 - Because of the routing utilisation

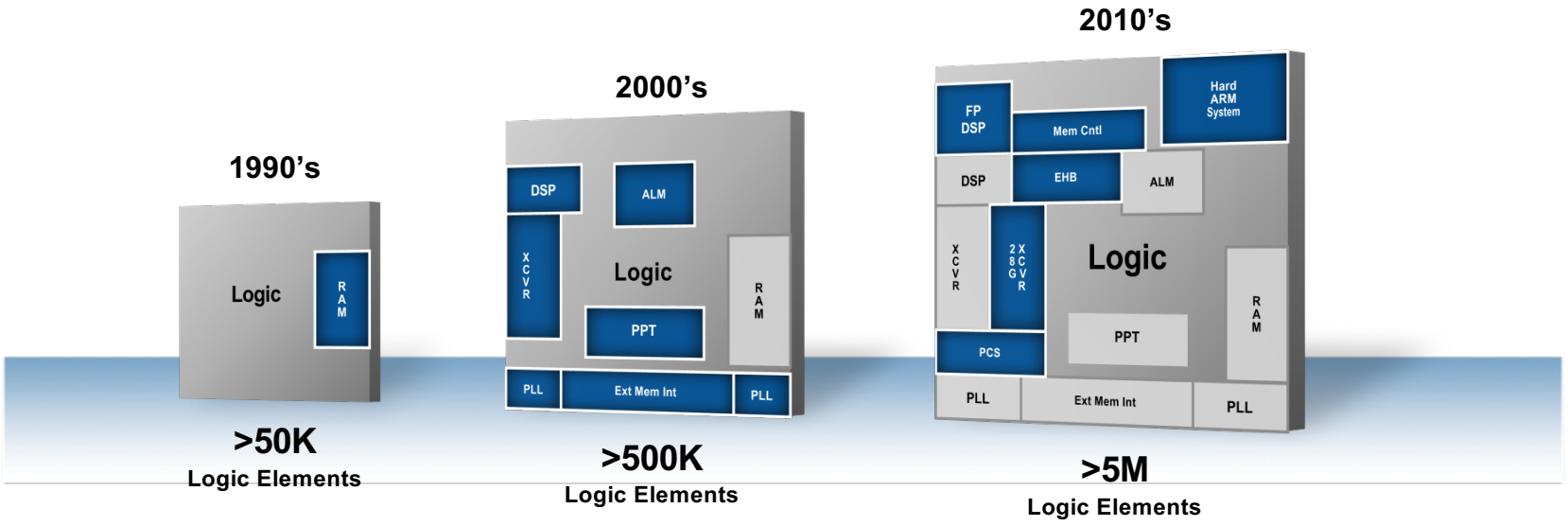
- Schematics ruled
- Some lazy, wasteful designers used VHDL, IP is for those that can't design.
- Cell delays dominated interconnect
- Timing Analysis was fmax and I/O delays.

Timing Closure Loop

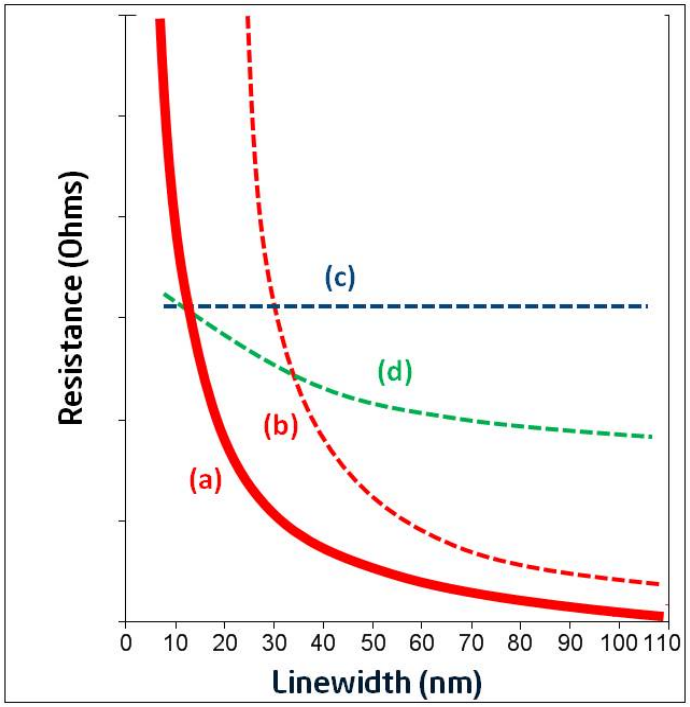


100x Increase in Complexity

- FPGAs delivering significantly more capabilities
  - **>5M** logic element density
  - **Up to 10** TFLOPs
  - **>4X** processor data throughput
  - **4X** serial transceiver bandwidth
  - **>50** serial protocols
  - **>2.5 Tbps** bandwidth for serial memory
  - **>2.3 Tbps** bandwidth for parallel memory (DDR4 @ 2666 Mbps)
  - **2X** core performance (HyperFlex)



Interconnect Dominates Delay

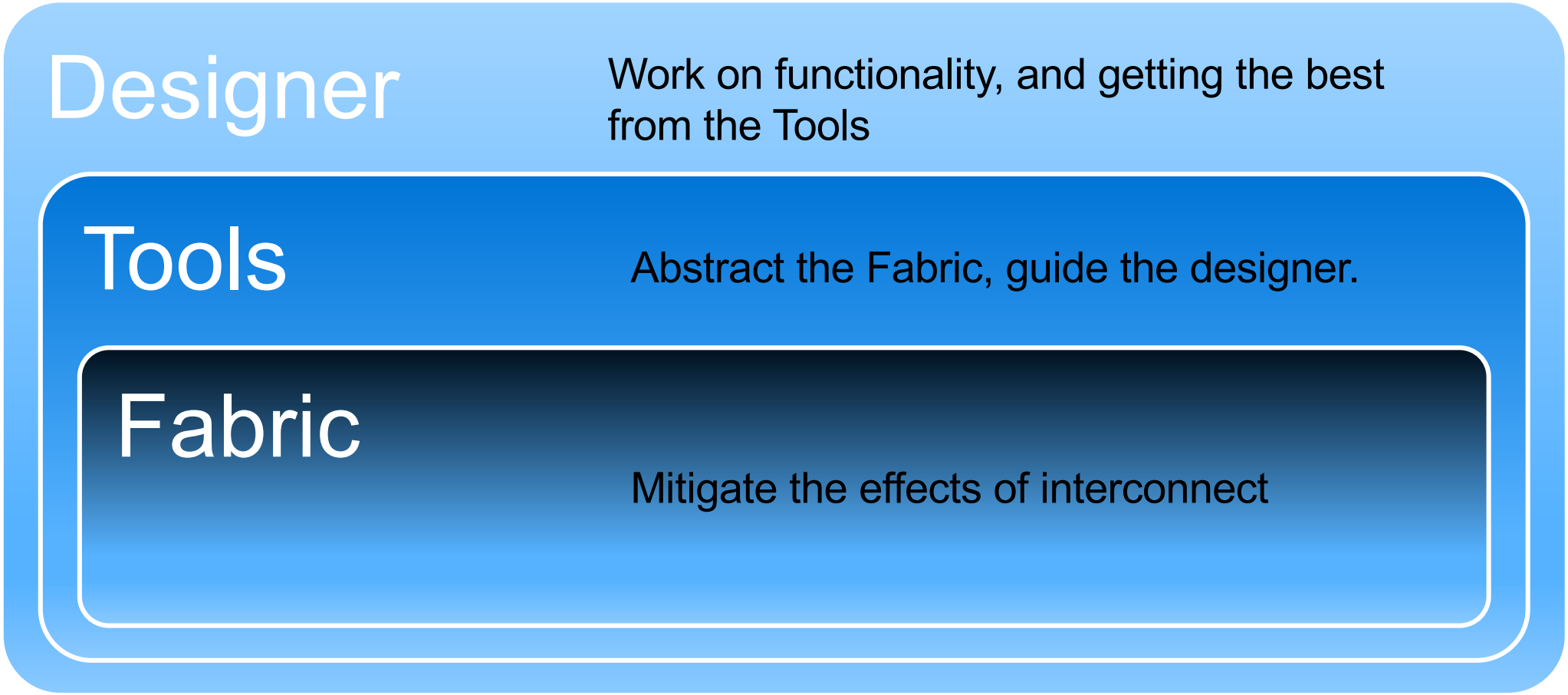


Process Technology Scaling in an Increasingly Interconnect Dominated World (James Clarke, Intel, VLSI 2014)

Problem Statement

- Our workload is 100x larger
- Performance expectations have scaled with Moore's Law
- The technology is fundamentally different
- We have fewer Engineers to do the job
- Those Engineers can't get to much of the source because it's in IP

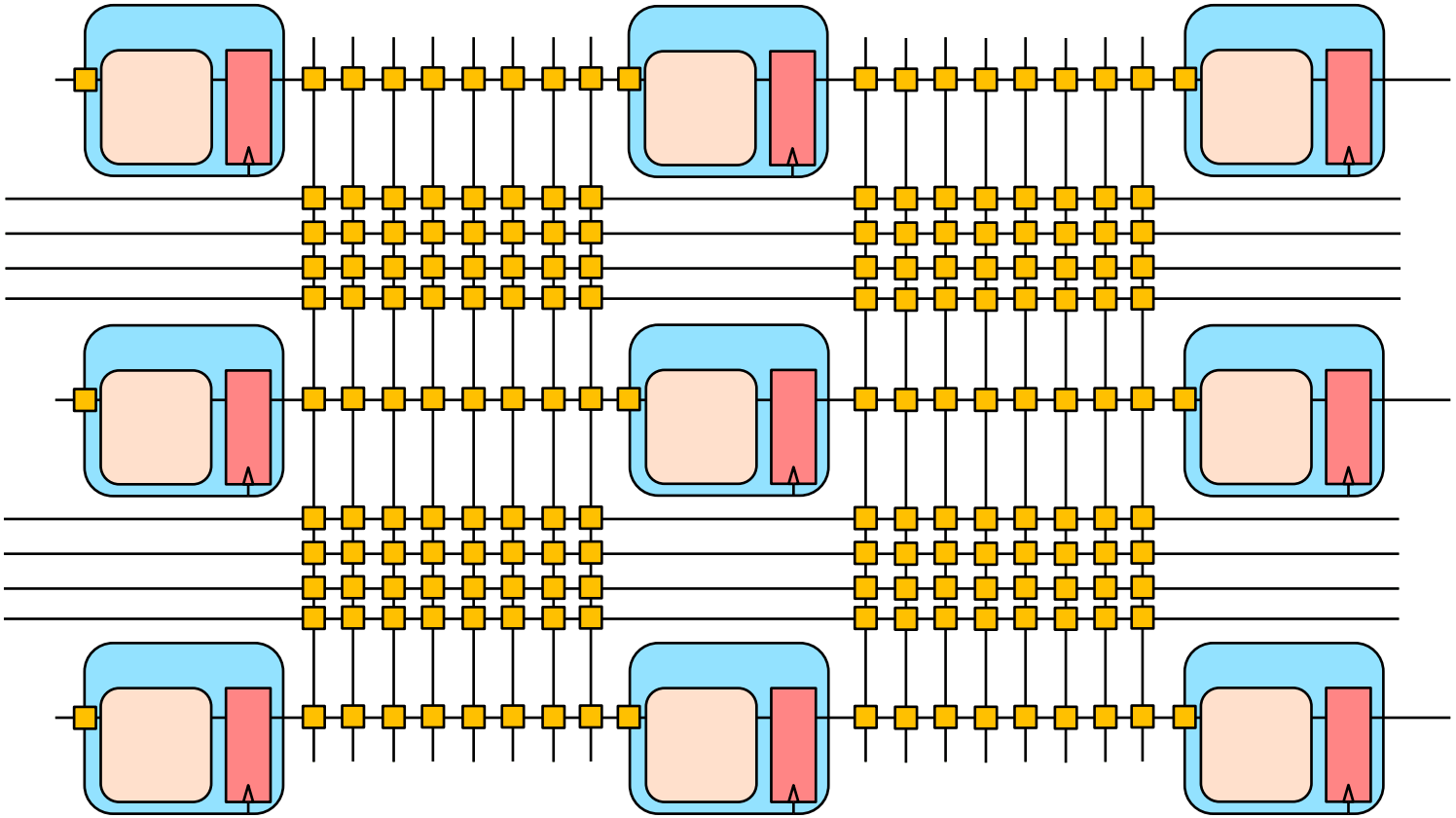
- We have to reduce interconnect delay
- We have to do it in a way that doesn't burden the designer



The Fabric

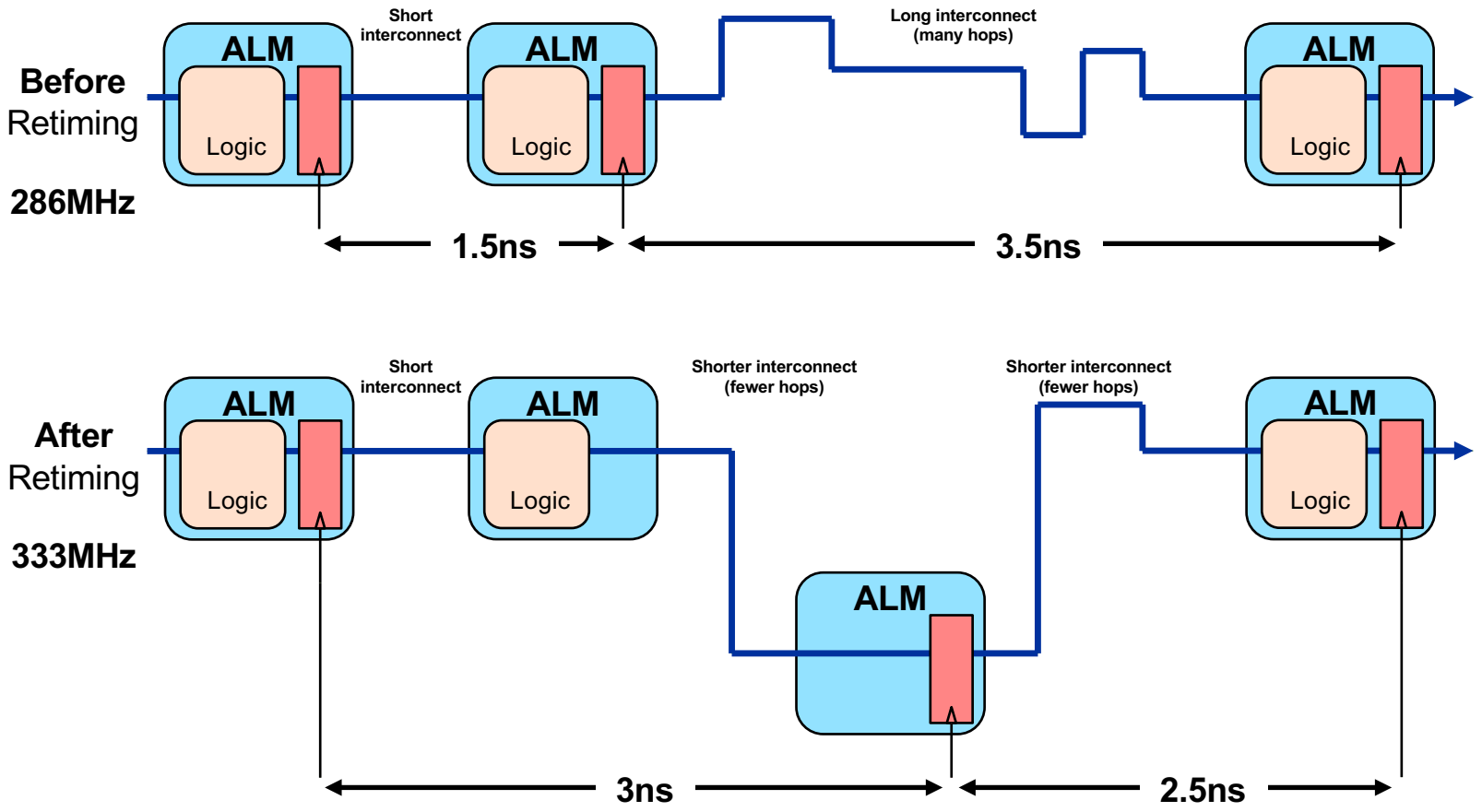
Mitigate the long delays from modern interconnect.
….at a viable cost.

The Fabric - HyperFlex



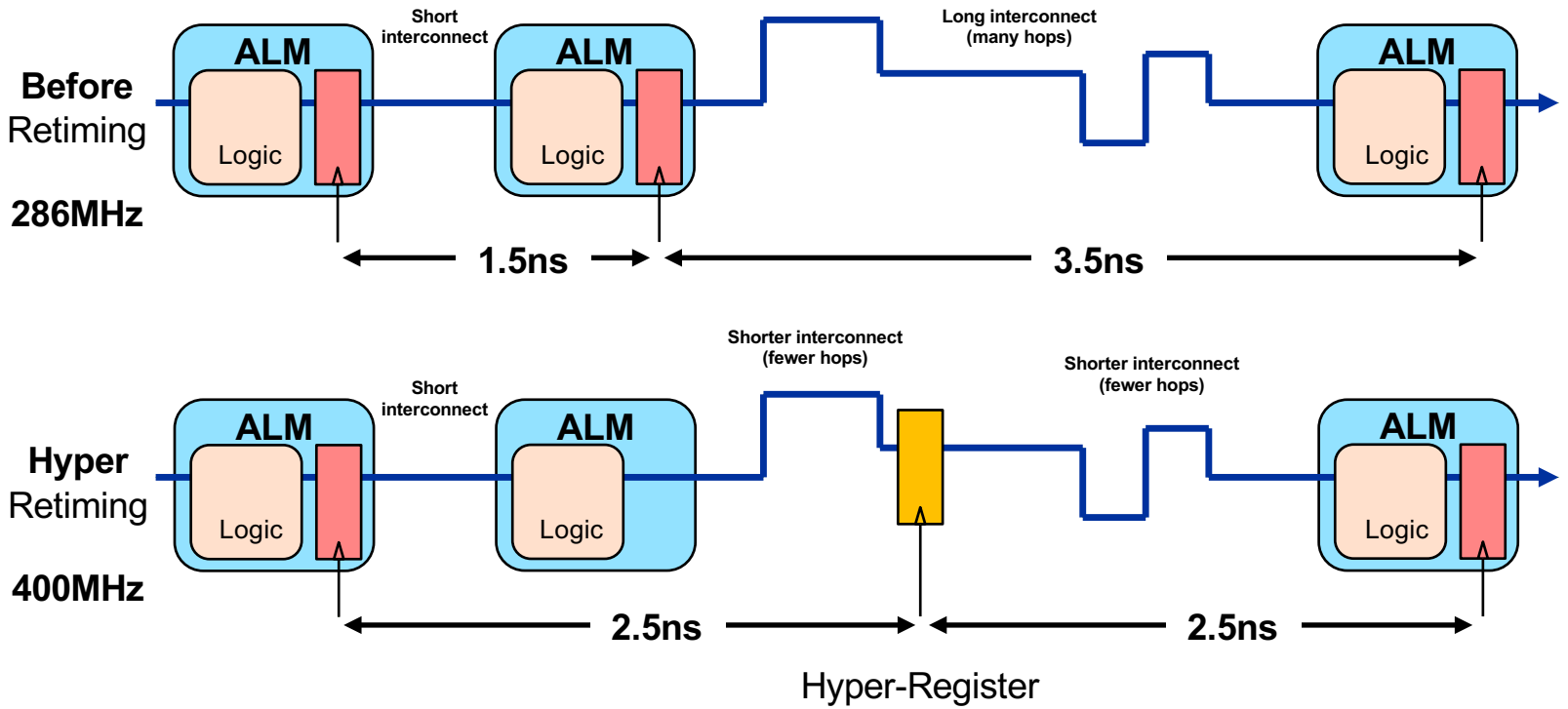
= Hyper-Register

Conventional Register Retiming



286MHz → 333MHz = 16% gain

Hyper-Retiming



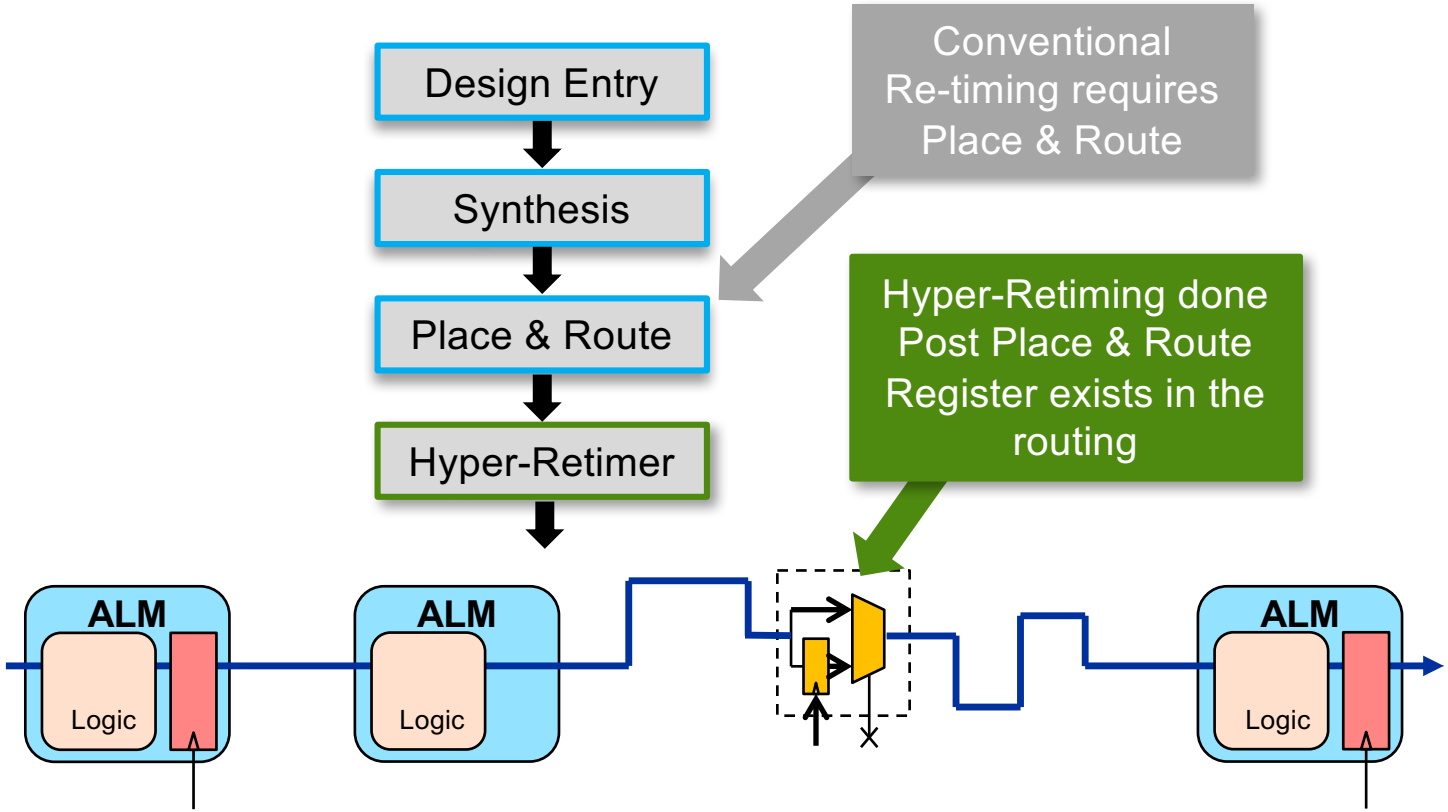
Hyper-Retiming step occurs AFTER place & route!

286MHz → 400MHz = 40% gain

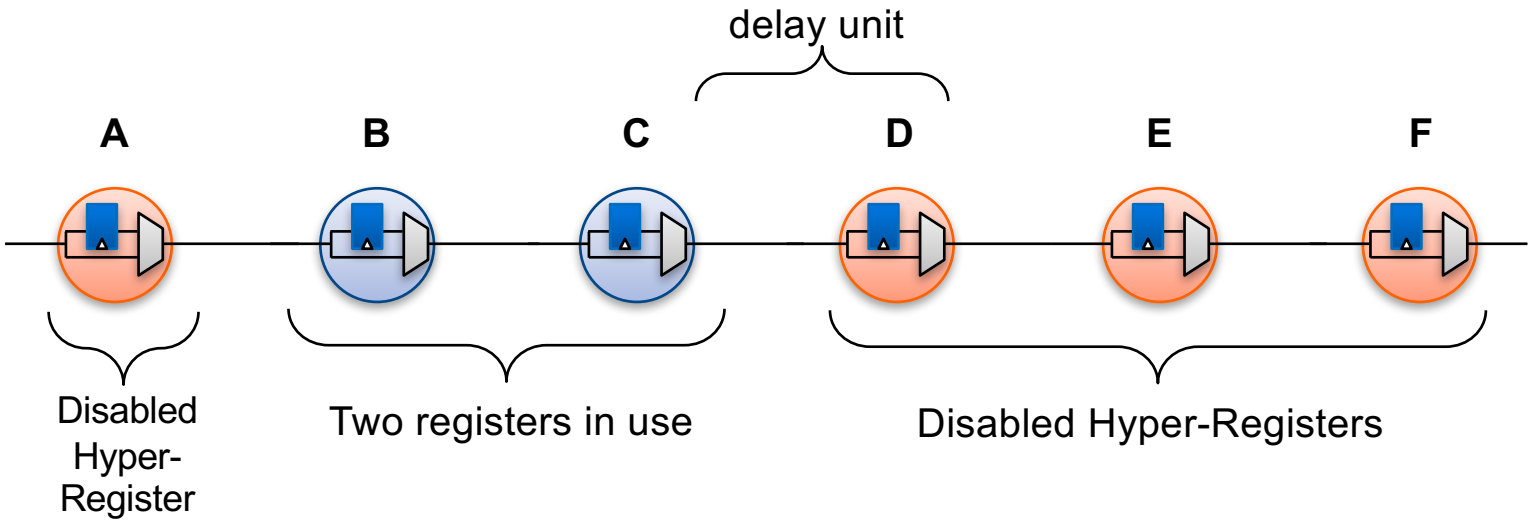
The Tools

Abstract the complex fabric away.
Guide the Designer.

Hyper-Retiming Tool Flow

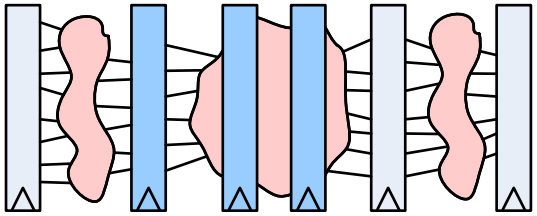


Retiming Example

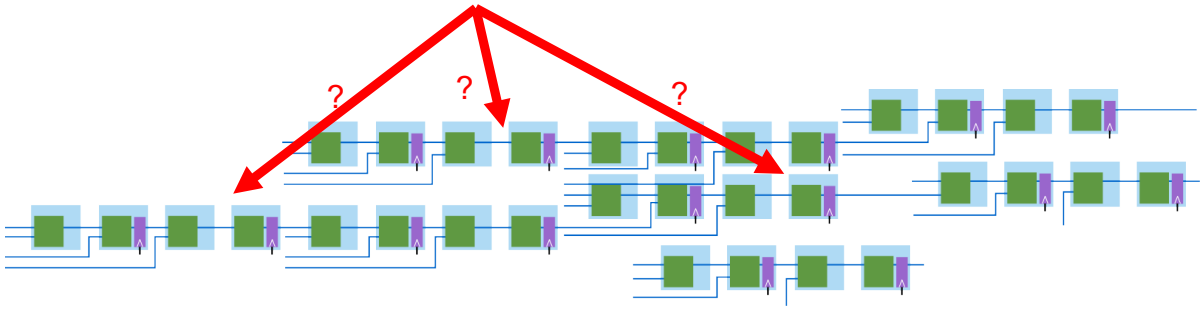


Implementing Hyper-Pipelining (the “hard” way)

- Conventional Pipelining
  - Manually pipeline a block, by inserting registers at just the right place
  - Can be **very difficult** to do for an **arbitrary combinational block**

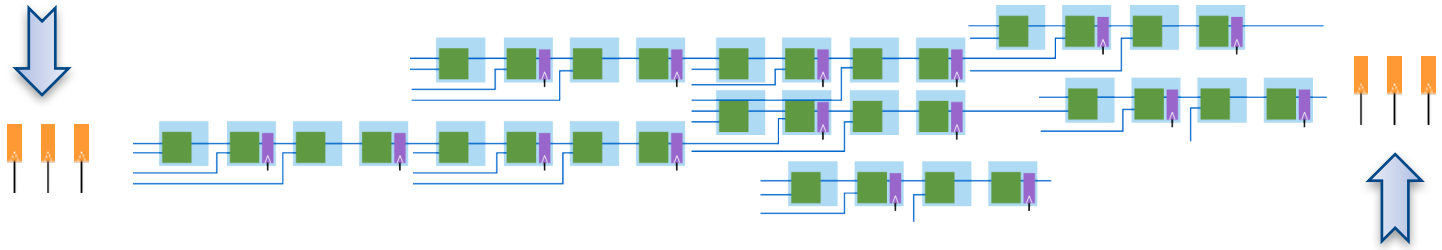


  - This effort can be complex and time-consuming (iterative by nature)
  - Requires designers to know **where** and **how many** registers to add

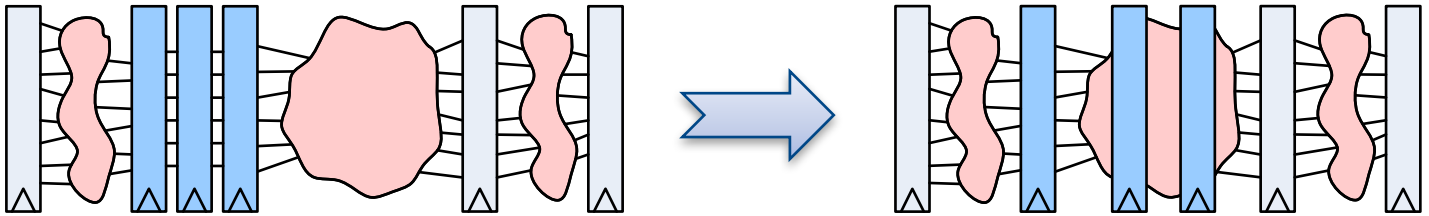


Implementing Hyper-Pipelining (the "easy" way)

- Automatic Hyper-Pipelining
  - Run Fast-Forward Compile to learn which paths require more registers
  - Add pipeline registers to the path (in HDL) before and after clock domains



  - Allow the Hyper-Retimer to automatically distribute the register(s) to the optimal place



Retiming Restrictions

- Asynchronous Clears
- Synchronizers
- Clock Crossings
- False Paths and Multicycle Constraints
- Dual-Clock RAMs

The Designer

Implement Functionality
Work with the tool to let it do its job

HyperFlex Performance Limit = Retiming Limit

- Conventional designs: timing closure is “local” problem

  - Timing closure issues between X and Y addressed locally (a critical path)

- Retimed designs: timing closure is “global” problem

  - Retiming gives freedom to balance across multiple register stages
  - Performance is limited by (approximate) average delay, less quantization

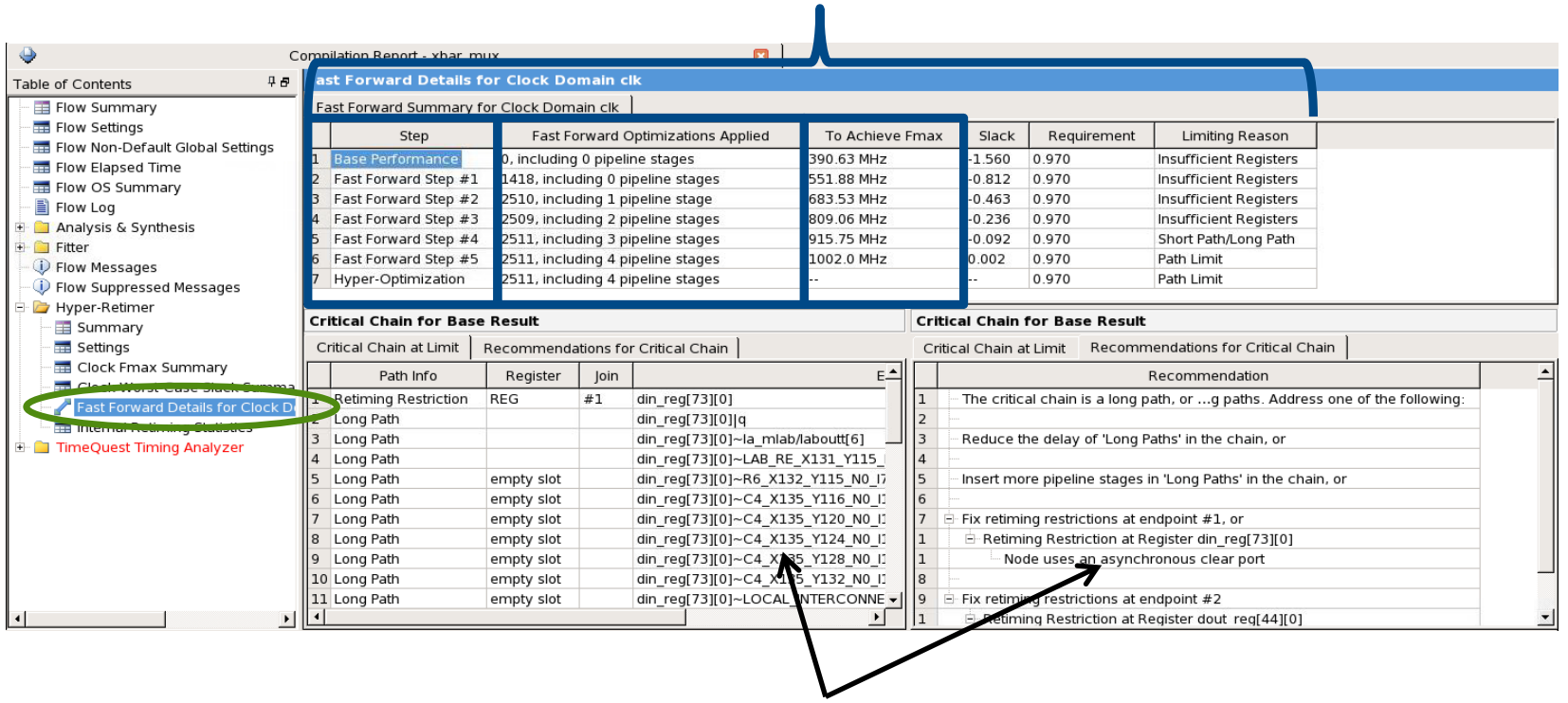
The Critical Chain

- In the presence of retiming, a “critical chain” is

- From **Register A** through multiple registers to **Register B**
  - Register A can’t be retimed forward any more
  - Register B can’t be retimed backward any more
  - Sometimes A and B are the same, indicating a loop
- Performance is limited by the **average delay of a register hop**
  - Less additional delay quantization effect
- ***A critical path abstracted to a higher level***

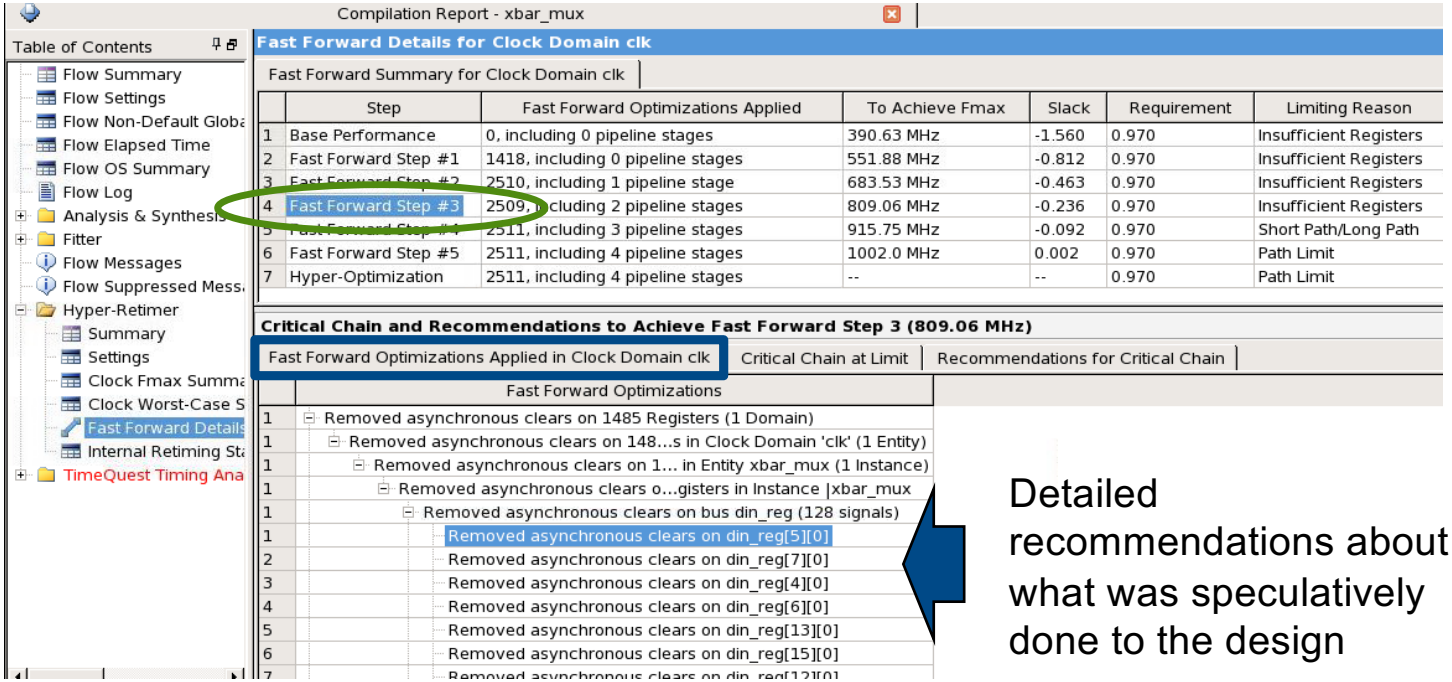
Example Fast-Forward Compile Details Report

List of speculative modifications (shown as steps) with fmax predictions for each



Critical Chain at Limit and Recommendations for Critical Chain for each speculation

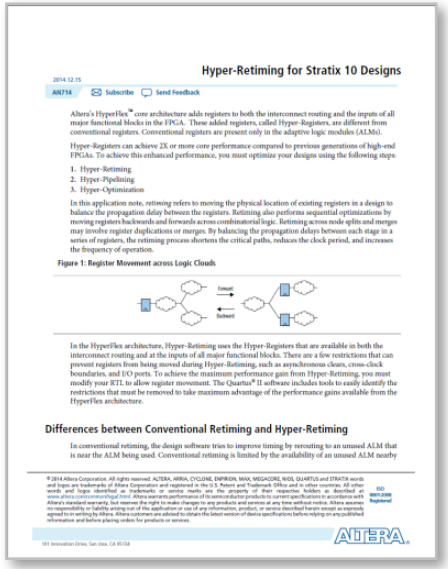
Fast Forward Optimizations Applied Tab



Detailed recommendations about what was speculatively done to the design

- Detailed Recommendations about what was done to the design (and what you can do) to reach this step in performance
- No need to look at ALL recommendations; designer can handle busses/common inputs simultaneously

Stratix 10 HyperFlex Application Notes


Hyper-Retiming for Stratix 10 Designs

2014.12.15
AN714 Subscribe Send Feedback

Altera's HyperFlex™ core architecture adds registers to both the interconnect routing and the inputs of all major functional blocks in the FPGA. These added registers, called Hyper-Registers, are different from conventional registers. Conventional registers are present only in the adaptive logic modules (ALMs).

Hyper-Registers can achieve 2X or more core performance compared to previous generations of high-end FPGAs. To achieve this enhanced performance, you must optimize your designs using the following steps:

1. Hyper-Retiming
2. Hyper-Pipelining
3. Hyper-Optimization

In this application note, *retiming* refers to moving the physical location of existing registers in a design to balance the propagation delay between the registers. Retiming also performs sequential optimizations by moving registers backwards and forwards across combinatorial logic. Retiming across node splits and merges may involve register duplications or merges. By balancing the propagation delays between each stage in a series of registers, the retiming process shortens the critical paths, reduces the clock period, and increases the frequency of operation.

Figure 1: Register Movement across Logic Clouds

In the HyperFlex architecture, Hyper-Retiming uses the Hyper-Registers that are available in both the interconnect routing and at the inputs of all major functional blocks. There are a few restrictions that can prevent registers from being moved during Hyper-Retiming, such as asynchronous clears, cross-clock boundaries, and I/O ports. To achieve the maximum performance gain from Hyper-Retiming, you must modify your RTL to allow register movement. The Quartus® II software includes tools to easily identify the restrictions that must be removed to take maximum advantage of the performance gains available from the HyperFlex architecture.

Differences between Conventional Retiming and Hyper-Retiming

In conventional retiming, the design software tries to improve timing by rerouting to an unused ALM that is near the ALM being used. Conventional retiming is limited by the availability of an unused ALM nearby

© 2014 Altera Corporation. All rights reserved. ALTERA, ARRIA, CYCLONE, ENPIRION, MAX, MEGACORE, NIOS, QUARTUS and STRATIX words and logos are trademarks of Altera Corporation and registered in the U.S. Patent and Trademark Office and in other countries. All other words and logos identified as trademarks or service marks are the property of their respective holders as described at www.altera.com/common/legal.html. Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

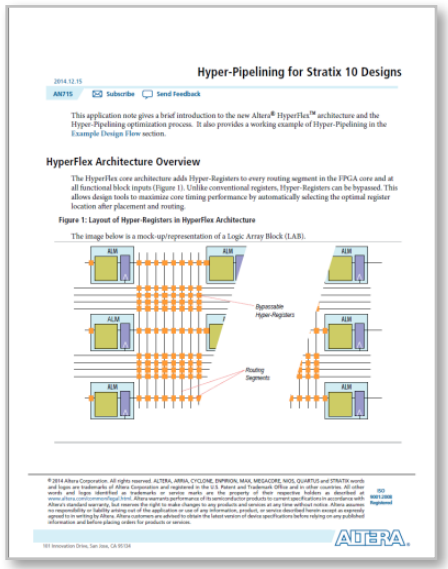
ISO 9001:2008 Registered

ALTERA

101 Innovation Drive, San Jose, CA 95134

AN714

- **Hyper-Retiming** for Stratix 10 Designs
- How it works
- Understand what to do with register controls (async and sync clears, clock enables)


Hyper-Pipelining for Stratix 10 Designs

2014.12.15
AN715 Subscribe Send Feedback

This application note gives a brief introduction to the new Altera® HyperFlex™ architecture and the Hyper-Pipelining optimization process. It also provides a working example of Hyper-Pipelining in the Example Design Flow section.

HyperFlex Architecture Overview

The HyperFlex core architecture adds Hyper-Registers to every routing segment in the FPGA core and at all functional block inputs (Figure 1). Unlike conventional registers, Hyper-Registers can be bypassed. This allows design tools to maximize core timing performance by automatically selecting the optimal register location after placement and routing.

Figure 1: Layout of Hyper-Registers in HyperFlex Architecture

The image below is a mock-up/representation of a Logic Array Block (LAB).

© 2014 Altera Corporation. All rights reserved. ALTERA, ARRIA, CYCLONE, ENPIRION, MAX, MEGACORE, NIOS, QUARTUS and STRATIX words and logos are trademarks of Altera Corporation and registered in the U.S. Patent and Trademark Office and in other countries. All other words and logos identified as trademarks or service marks are the property of their respective holders as described at www.altera.com/common/legal.html. Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

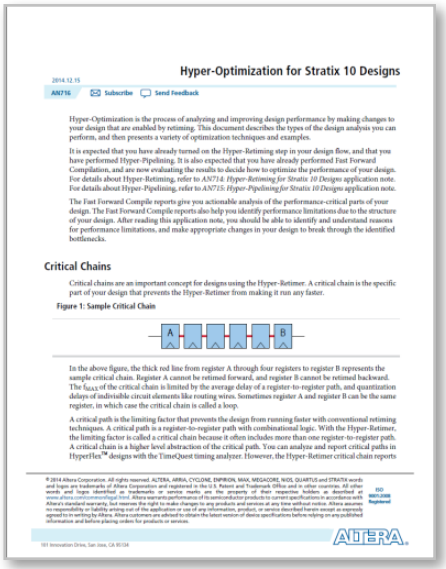
ISO 9001:2008 Registered

ALTERA

101 Innovation Drive, San Jose, CA 95134

AN715

- **Hyper-Pipelining** for Stratix 10 Designs
- How it works
- Understand how to use pipelining to optimize speed of your design
- Example design flow


Hyper-Optimization for Stratix 10 Designs

2014.12.15
AN716 Subscribe Send Feedback

Hyper-Optimization is the process of analyzing and improving design performance by making changes to your design that are enabled by retiming. This document describes the types of the design analysis you can perform, and then presents a variety of optimization techniques and examples.

It is expected that you have already turned on the Hyper-Retiming step in your design flow, and that you have performed Hyper-Pipelining. It is also expected that you have already performed Fast Forward Compilation, and are now evaluating the results to decide how to optimize the performance of your design. For details about Hyper-Retiming, refer to *AN714: Hyper-Retiming for Stratix 10 Designs* application note. For details about Hyper-Pipelining, refer to *AN715: Hyper-Pipelining for Stratix 10 Designs* application note.

The Fast Forward Compile reports give you actionable analysis of the performance-critical parts of your design. The Fast Forward Compile reports also help you identify performance limitations due to the structure of your design. After reading this application note, you should be able to identify and understand reasons for performance limitations, and make appropriate changes in your design to break through the identified bottlenecks.

Critical Chains

Critical chains are an important concept for designs using the Hyper-Retimer. A critical chain is the specific part of your design that prevents the Hyper-Retimer from making it run any faster.

Figure 1: Sample Critical Chain

In the above figure, the thick red line from register A through four registers to register B represents the sample critical chain. Register A cannot be retimed forward, and register B cannot be retimed backward. The f_{MAX} of the critical chain is limited by the average delay of a register-to-register path, and quantization delays of indivisible circuit elements like routing wires. Sometimes register A and register B can be the same register, in which case the critical chain is called a loop.

A critical path is the limiting factor that prevents the design from running faster with conventional retiming techniques. A critical path is a register-to-register path with combinational logic. With the Hyper-Retimer, the limiting factor is called a critical chain because it often includes more than one register-to-register path. A critical chain is a higher level abstraction of the critical path. You can analyze and report critical paths in HyperFlex™ designs with the TimeQuest timing analyzer. However, the Hyper-Retimer critical chain reports

© 2014 Altera Corporation. All rights reserved. ALTERA, ARRIA, CYCLONE, ENPIRION, MAX, MEGACORE, NIOS, QUARTUS and STRATIX words and logos are trademarks of Altera Corporation and registered in the U.S. Patent and Trademark Office and in other countries. All other words and logos identified as trademarks or service marks are the property of their respective holders as described at www.altera.com/common/legal.html. Altera warrants performance of its semiconductor products to current specifications in accordance with Altera's standard warranty, but reserves the right to make changes to any products and services at any time without notice. Altera assumes no responsibility or liability arising out of the application or use of any information, product, or service described herein except as expressly agreed to in writing by Altera. Altera customers are advised to obtain the latest version of device specifications before relying on any published information and before placing orders for products or services.

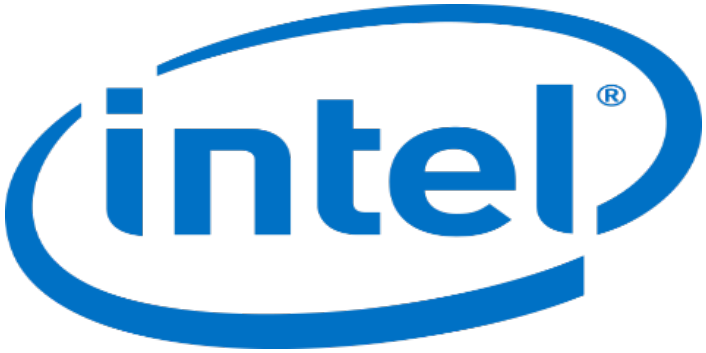
ISO 9001:2008 Registered

ALTERA

101 Innovation Drive, San Jose, CA 95134

AN716

- **Hyper-Optimization** for Stratix 10 Designs
- Interpret fast forward report file
- Understand bottle-necks limiting further performance gains
- Know what to do about them


intel