

# Stratix 10 HyperFlex Architecture Overview

**HyperFlex**<sup>™</sup>  
ARCHITECTURE

**ALTERA**<sup>®</sup>

# Agenda

- ◀ Delivering the Unimaginable
- ◀ The HyperFlex Architecture Advantage
- ◀ HyperFlex Technical Overview
- ◀ Quartus II Design Flow
- ◀ Performance Benchmarks
- ◀ Early Access Software and Training



**HyperFlex**<sup>™</sup>  
ARCHITECTURE



# *Stratix 10 FPGAs and SoCs*



*Delivering the Unimaginable*

**ALTERA**

# Stratix 10 FPGAs and SoCs

*Delivering the Unimaginable*

**2X**  
Performance

Up to  
**70%**  
Lower power

**4X+**  
Logic  
Density

Quad Core  
**64** -bit  
ARM Cortex  
A53



**14nm Intel Tri-Gate**

+



**HyperFlex**<sup>TM</sup>  
ARCHITECTURE

**New Architecture**

# HyperFlex™ Architecture Overview

**HyperFlex**<sup>™</sup>  
ARCHITECTURE

**ALTERA**®

## Why Develop a New Architecture?

- ◀ Today's architectures will not hold up to tomorrow's performance demands
  - Making on-chip buses wider and wider is not sufficient, need to do more
- ◀ Need bigger step forward than we get with evolution
  - As geometries shrink, interconnect delays are dominating
- ◀ HyperFlex built on familiar concepts ...
  - Retiming, Pipelining, Optimization
- ◀ With an innovative new approach
  - Allows unimaginable levels of performance
  - Not possible with conventional architecture

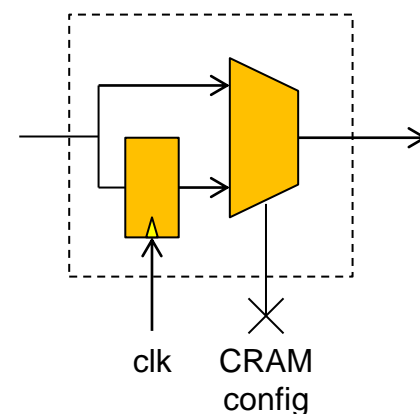


**HyperFlex**<sup>™</sup>  
ARCHITECTURE

*HyperFlex is New ...  
and It's a Big Improvement!*

## The HyperFlex Solution

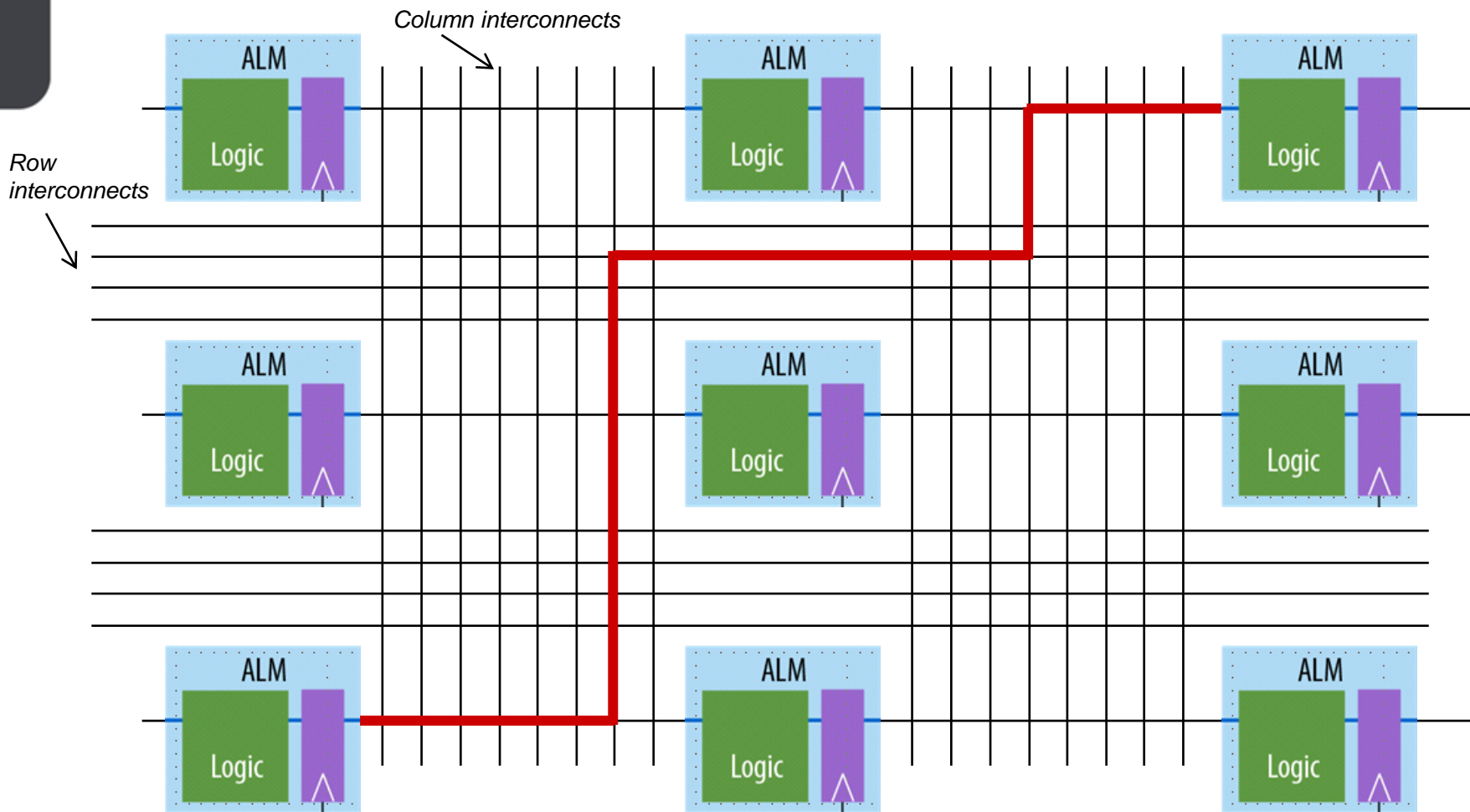
- HyperFlex has registers throughout the core fabric
- Bypassable Hyper-Registers in every routing segment
- Bypassable Hyper-Registers on all block inputs
  - ALMs, M20K blocks, DSP blocks, IO cells
- Register location is fine-grained
  - Throughout the interconnect
  - Available in optimal locations
- Allows new and better approach to
  - Retiming
  - Pipelining
  - Optimization



**Bypassable Hyper-Register**

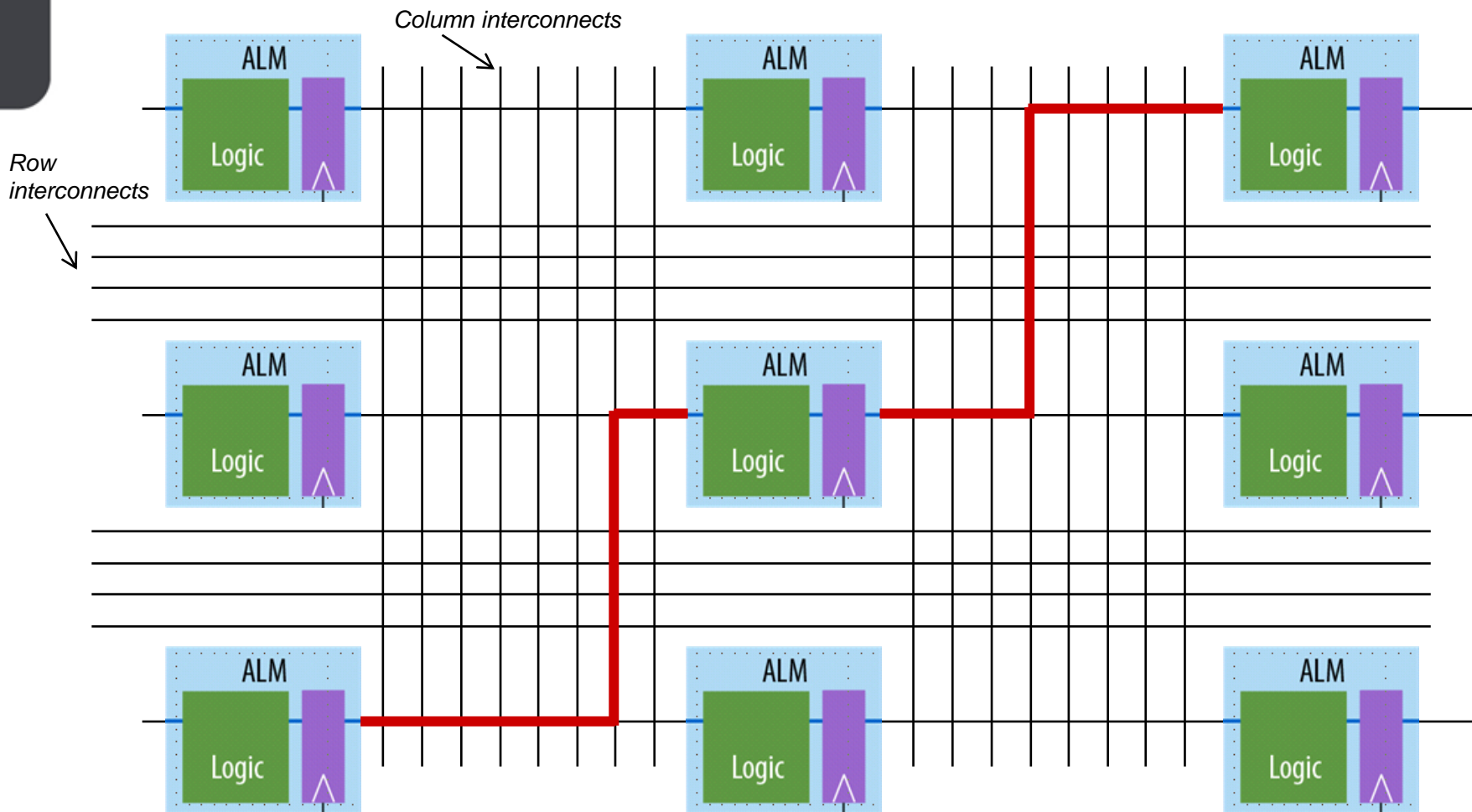
*Available “everywhere” throughout user logic and interconnect*

# Conventional FPGA Architecture - Interconnect



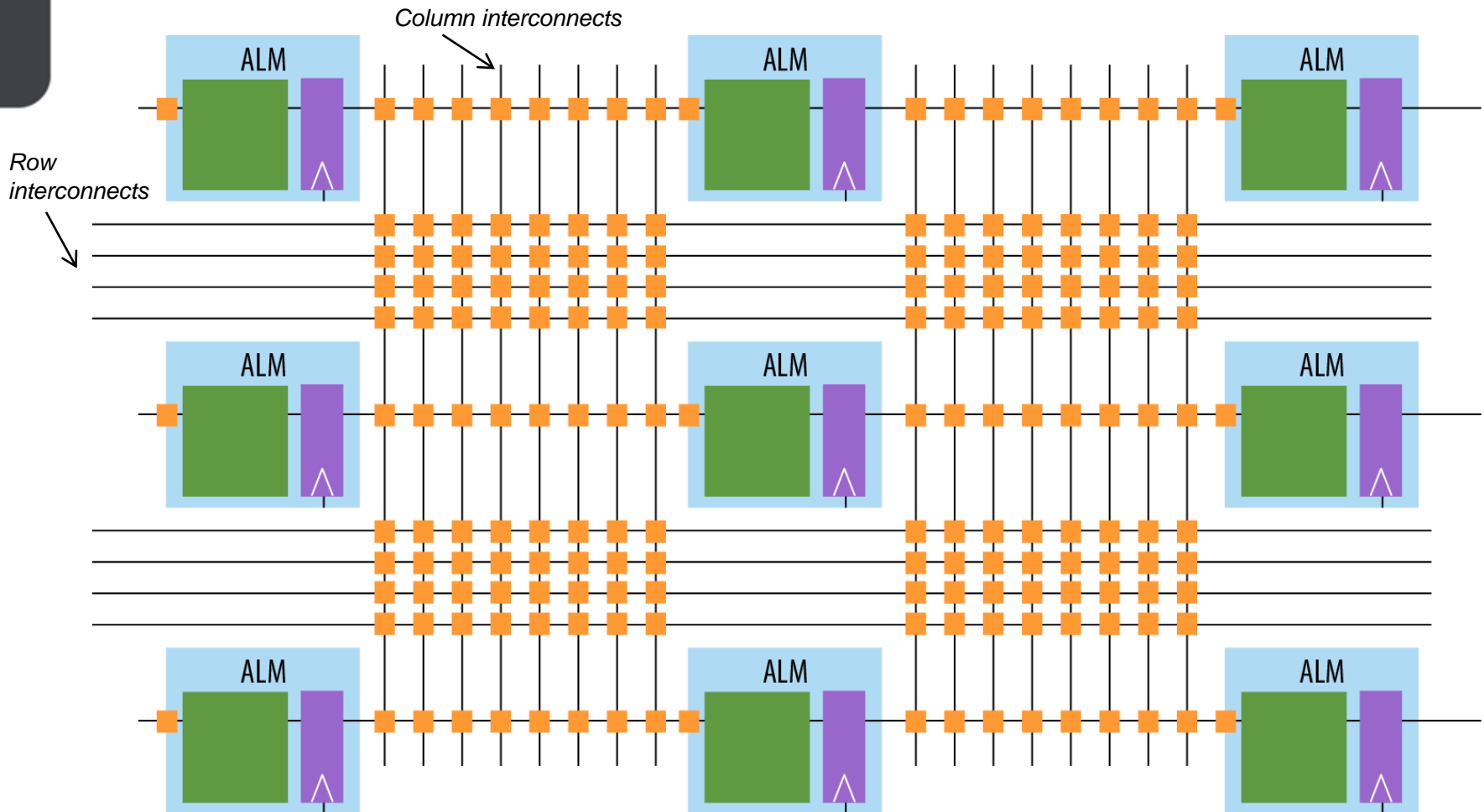
Conventional logic cell to logic cell path

# Conventional FPGA Architecture - Interconnect



Conventional pipelined path between logic cells

# The HyperFlex Architecture - Interconnect



*Number of Hyper-Registers >10X  
Number of ALM Registers!*

■ = Hyper-Register



# All New Stratix 10 HyperFlex Architecture

- ◀ Hyper-Registers throughout the FPGA fabric enable
  - Fine grain **Hyper-Retiming** to eliminate critical paths
  - Zero latency **Hyper-Pipelining** to eliminate routing delays
  - Flexible **Hyper-Optimization** for best-in-class performance
- ◀ Hyper-Aware design flow for accelerated timing closure with
  - Post place & route performance tuning
  - Hyper-register enabled synthesis and place & route for efficient pipelining
  - Fast Forward compilation enabling performance exploration
- ◀ Programmable clock tree synthesis offers
  - ASIC-like clocking to mitigate skew & uncertainty
  - Lowers power through intelligent clock enablement

**HyperFlex**<sup>TM</sup>  
ARCHITECTURE

## How Do We Get to 2X Performance?

Step	Architecture Advantage	Customer Effort	Stratix 10 versus Stratix V (Average Gain)
1	Hyper-Retiming	No change, or minor RTL changes	1.4X
2	Hyper-Pipelining	Added Pipelining	1.6X
3	Hyper-Optimization	More Effort	2X or more

- Three-step process to achieve maximum performance
- Most of the gain comes from the first two steps
  - Uses well understood retiming and pipelining techniques
  - Large performance gains come from relatively small effort
- More effort required to implement the third step
  - May be required to achieve 2X or more performance gain

# HyperFlex Optimization Strategies

## ◀ Hyper-Retiming (effort level: low)

- Moving **existing registers** in the design to optimize performance
- Remove restrictions that prevent registers from retiming
- Eliminates critical paths

## ◀ Hyper-Pipelining (effort level: medium)

- Add **new registers to the design** to pipeline paths better
- New registers need to be **inserted in the user RTL**
- Eliminates routing delays

## ◀ Hyper-Optimization (effort level: high)

- Optimizations to **improve the design further** beyond Hyper-Retiming and Hyper-Pipelining alone – **restructuring loops** in the design
- Enables maximum performance

# Core Performance is More Than Just Performance

## More Performance

- Enabling higher performance applications

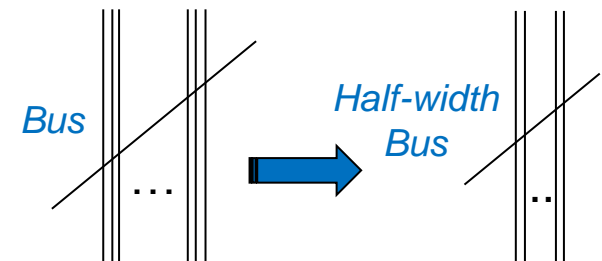


## Higher Productivity and Time to Market

- Reduce engineering development time
- Close timing faster

## Reduce Device Cost

- Choose a less-expensive **slower** device
  - With HyperFlex 2X performance, can you use a slower speed grade device?
- Choose a less expensive **smaller** device
  - Can you use a smaller device now that you have Hyper-Registers throughout the fabric?
  - Could you run your bus at 1/2 the width and twice the frequency?

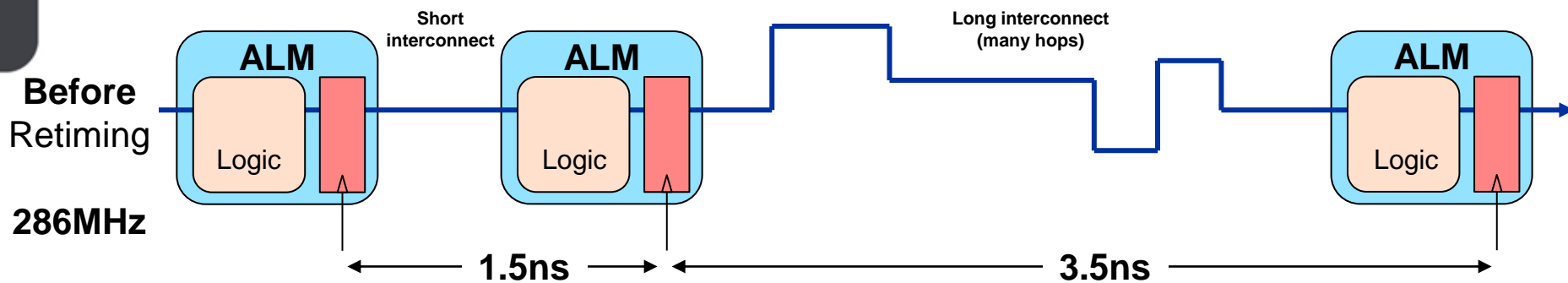


# Hyper-Retiming

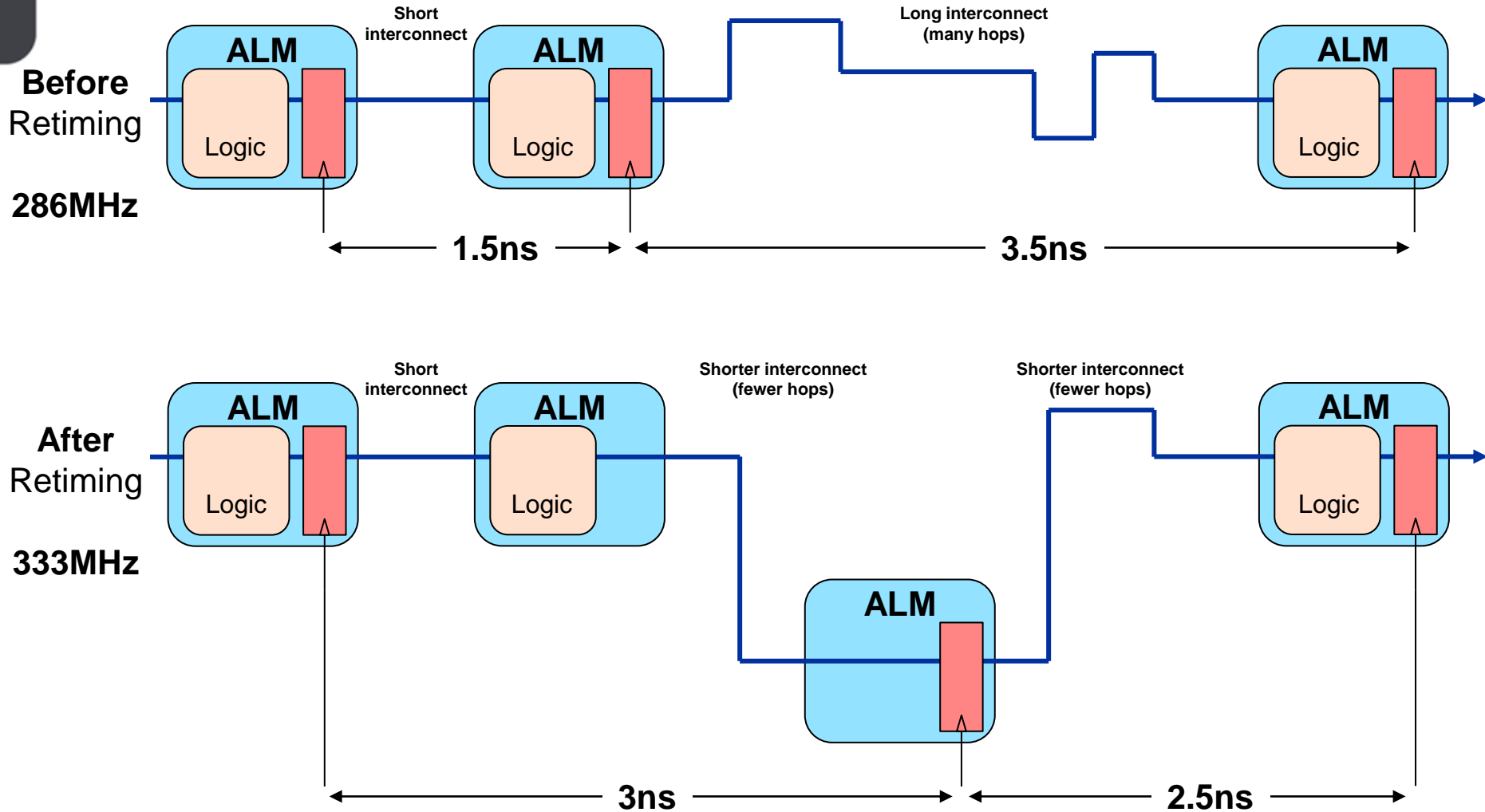
The Altera logo is displayed in a stylized, outlined font. It is positioned on a white, rounded rectangular background that is part of a larger blue graphic element at the bottom of the slide. The logo itself consists of the word "ALTERA" in all caps, with a registered trademark symbol (®) to its right.

**ALTERA®**

# Conventional Register Retiming

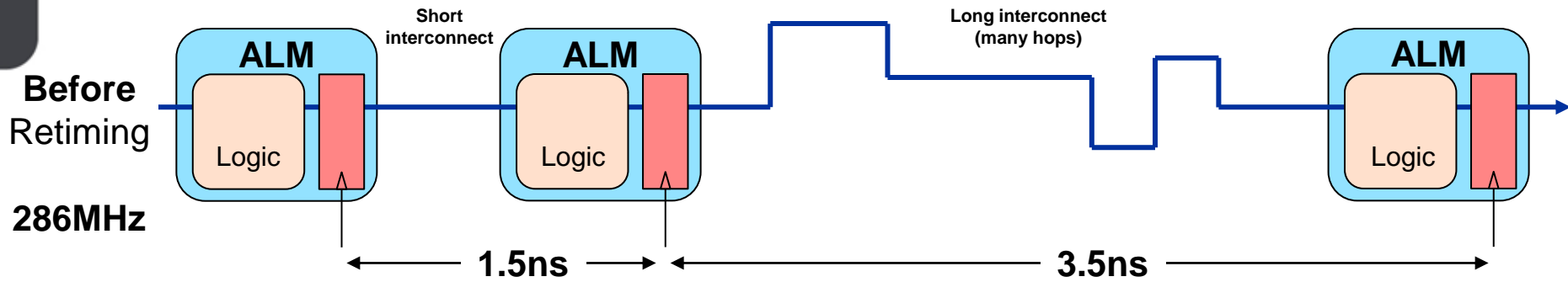


# Conventional Register Retiming

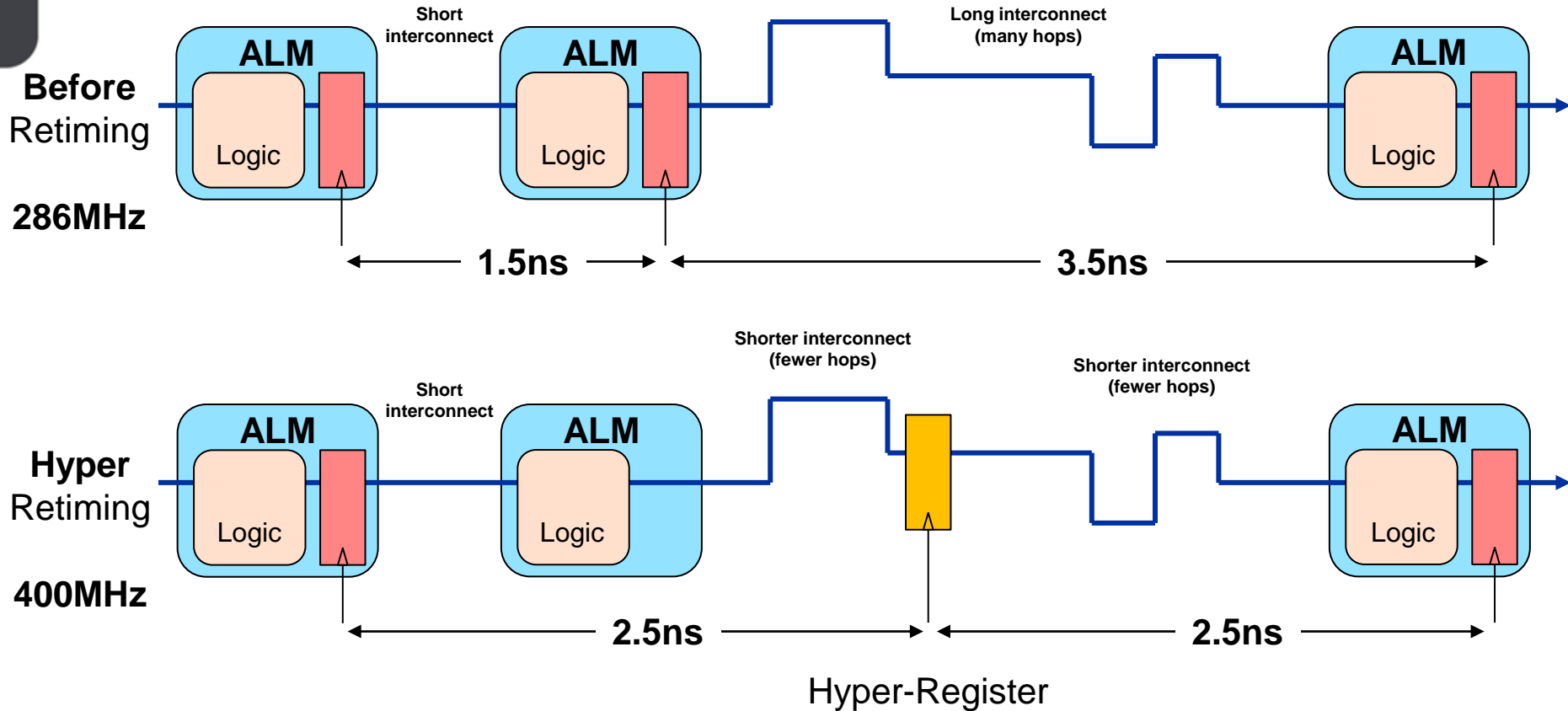


286MHz → 333MHz = 16% gain

# Hyper-Retiming



# Hyper-Retiming



*Hyper-Retiming step occurs AFTER place & route!*

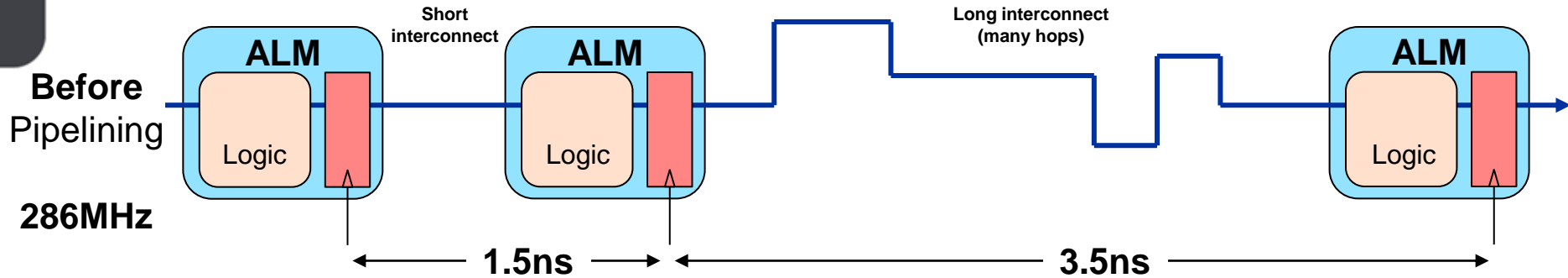
**286MHz → 400MHz = 40% gain**

# Hyper-Pipelining

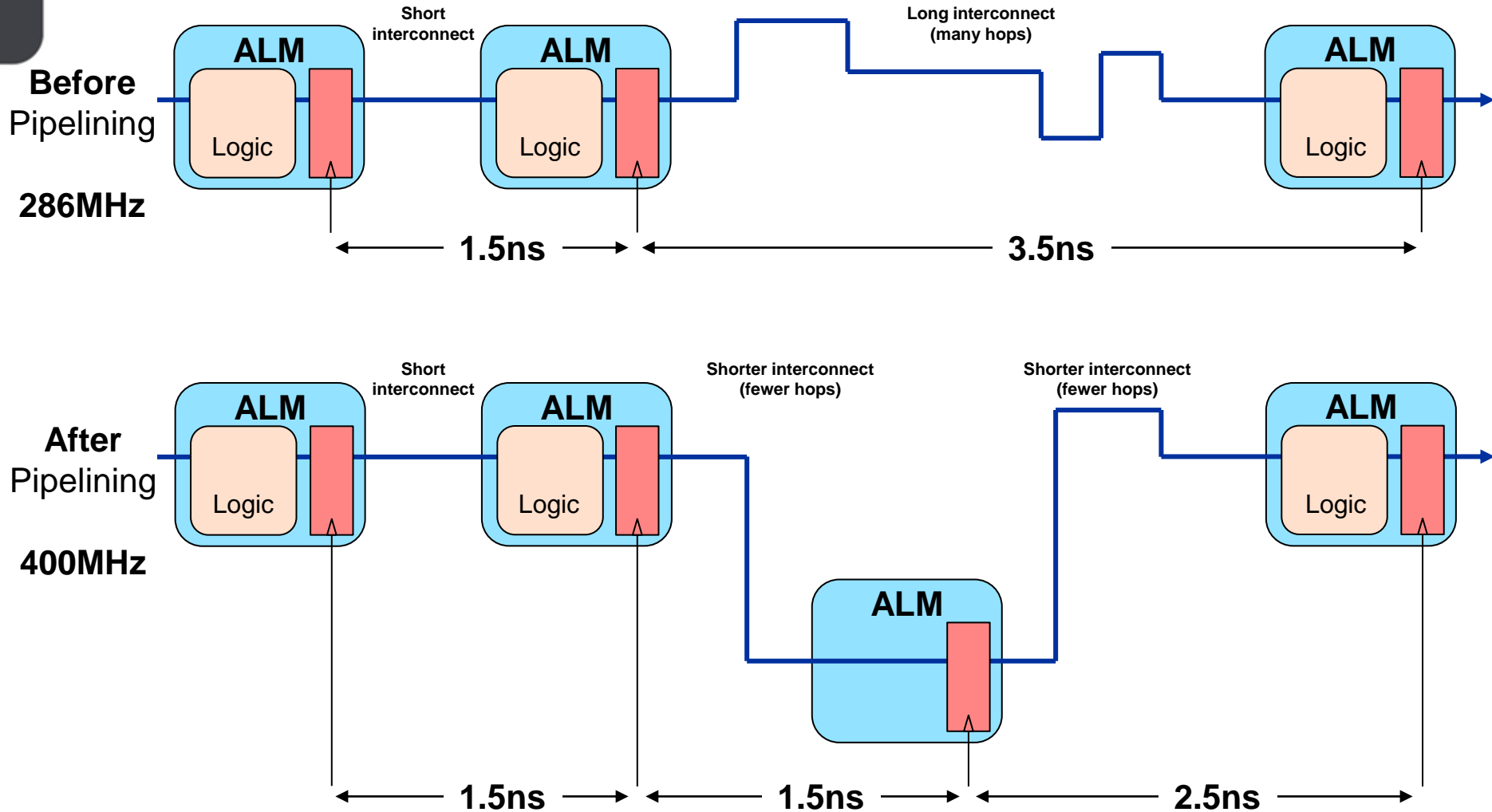
The Altera logo is displayed in a stylized, outlined font. It is positioned on a white, rounded rectangular background that is part of a larger blue graphic element at the bottom of the slide. The logo itself is blue with a white outline.

**ALTERA**®

# Conventional Pipelining



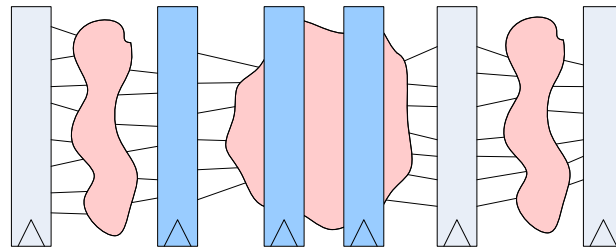
# Conventional Pipelining



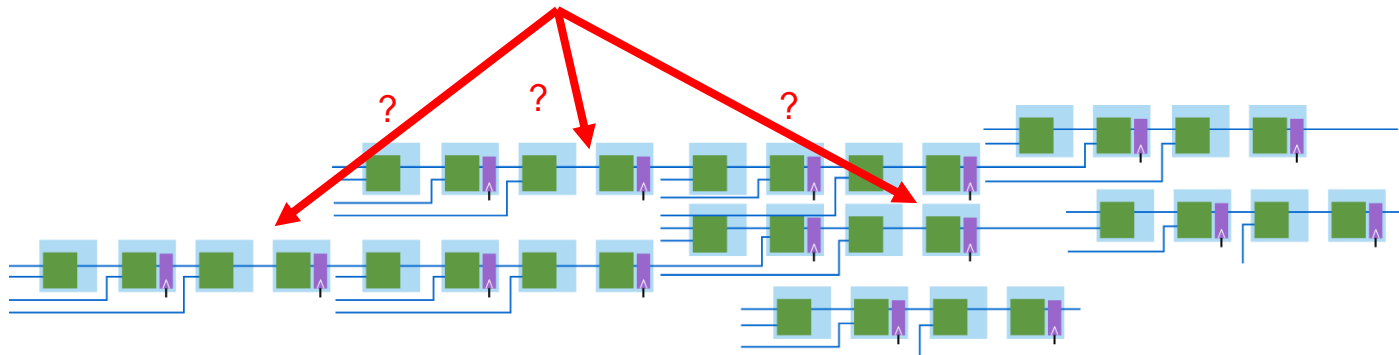
286MHz → 400MHz = 40% gain

# Conventional Pipelining

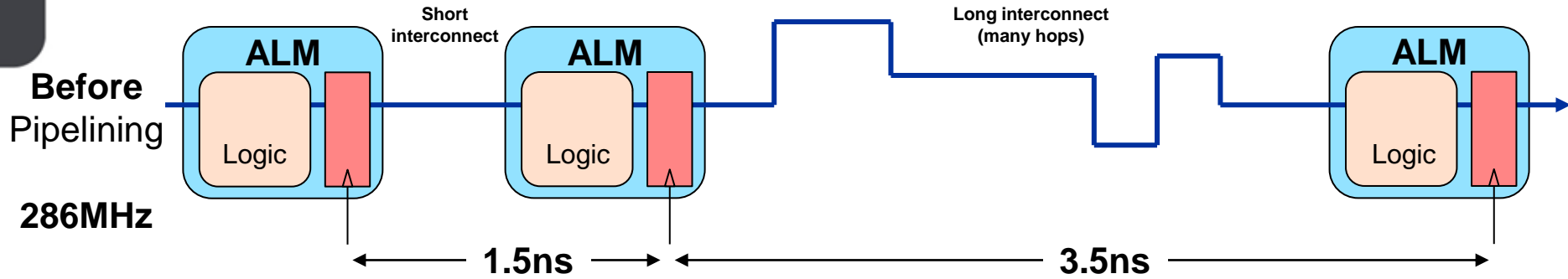
- Manually pipeline a block, by inserting registers at just the right place
- Can be **very difficult** to do for an **arbitrary combinational block**



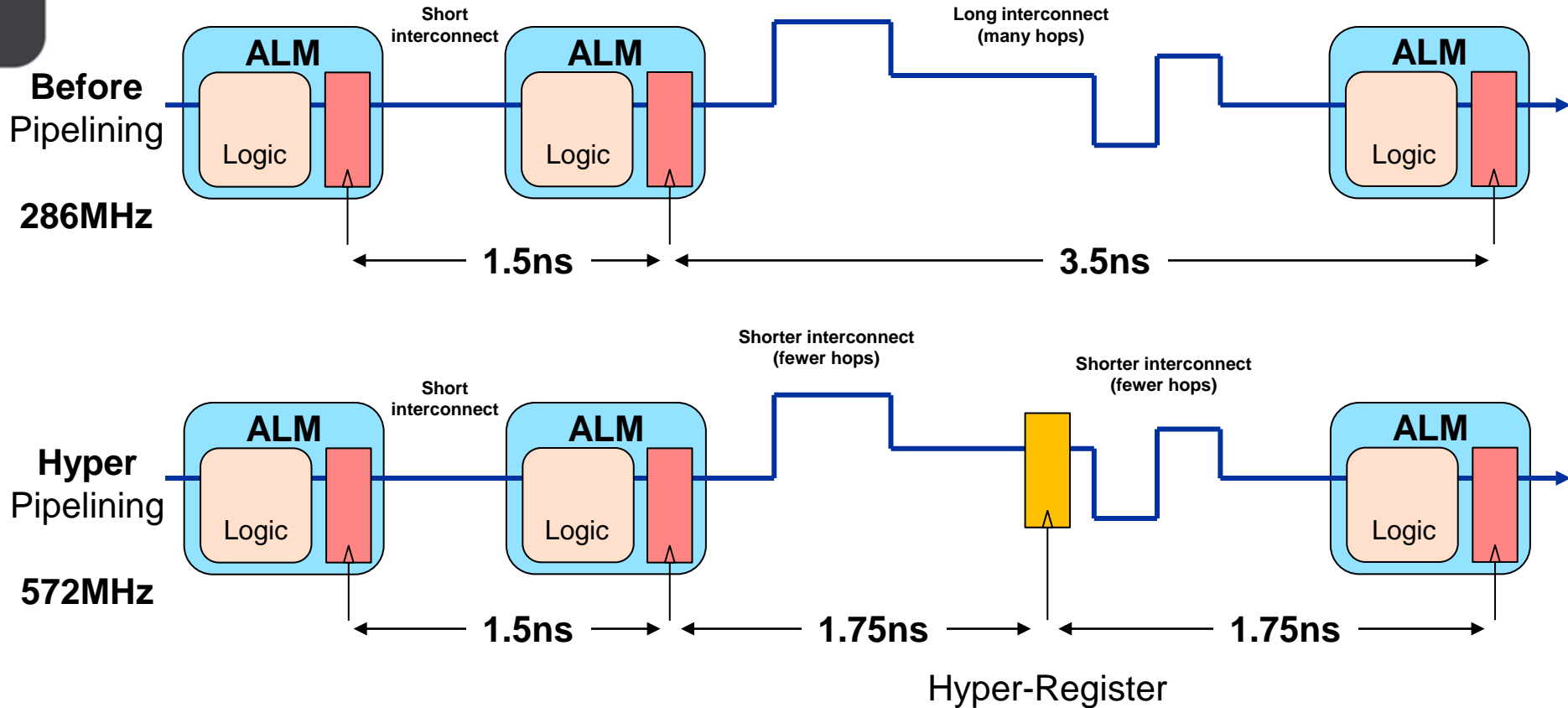
- This effort can be complex and time-consuming (iterative by nature)
- Requires designers to know **where** and **how many** registers to add



# Hyper-Pipelining



# Hyper-Pipelining



*Final placement of Hyper-Pipeline register occurs AFTER place & route!*

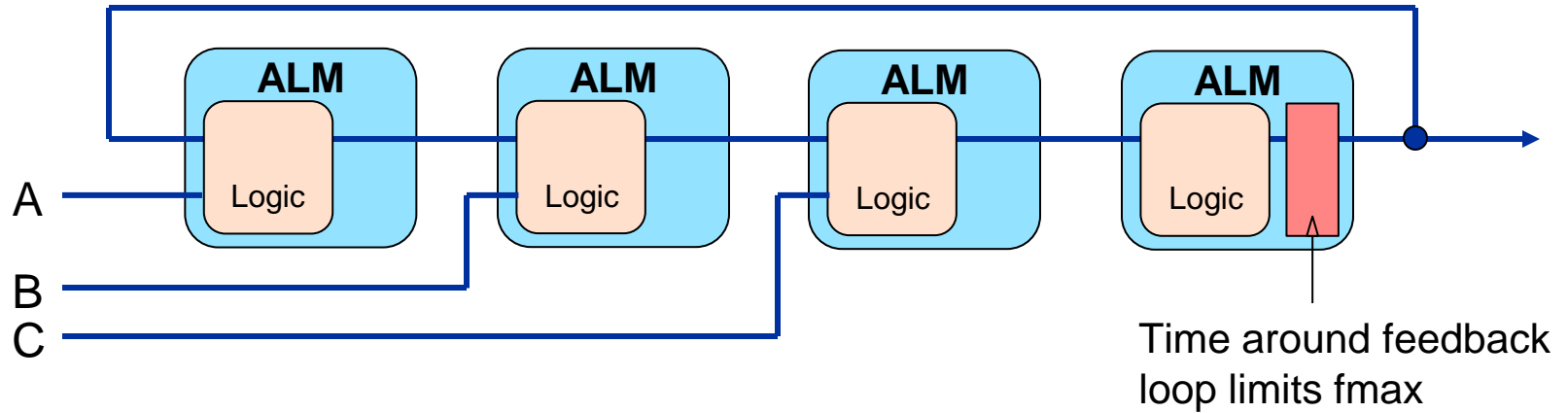
**286MHz → 572MHz = 100% gain**

# Hyper-Optimization

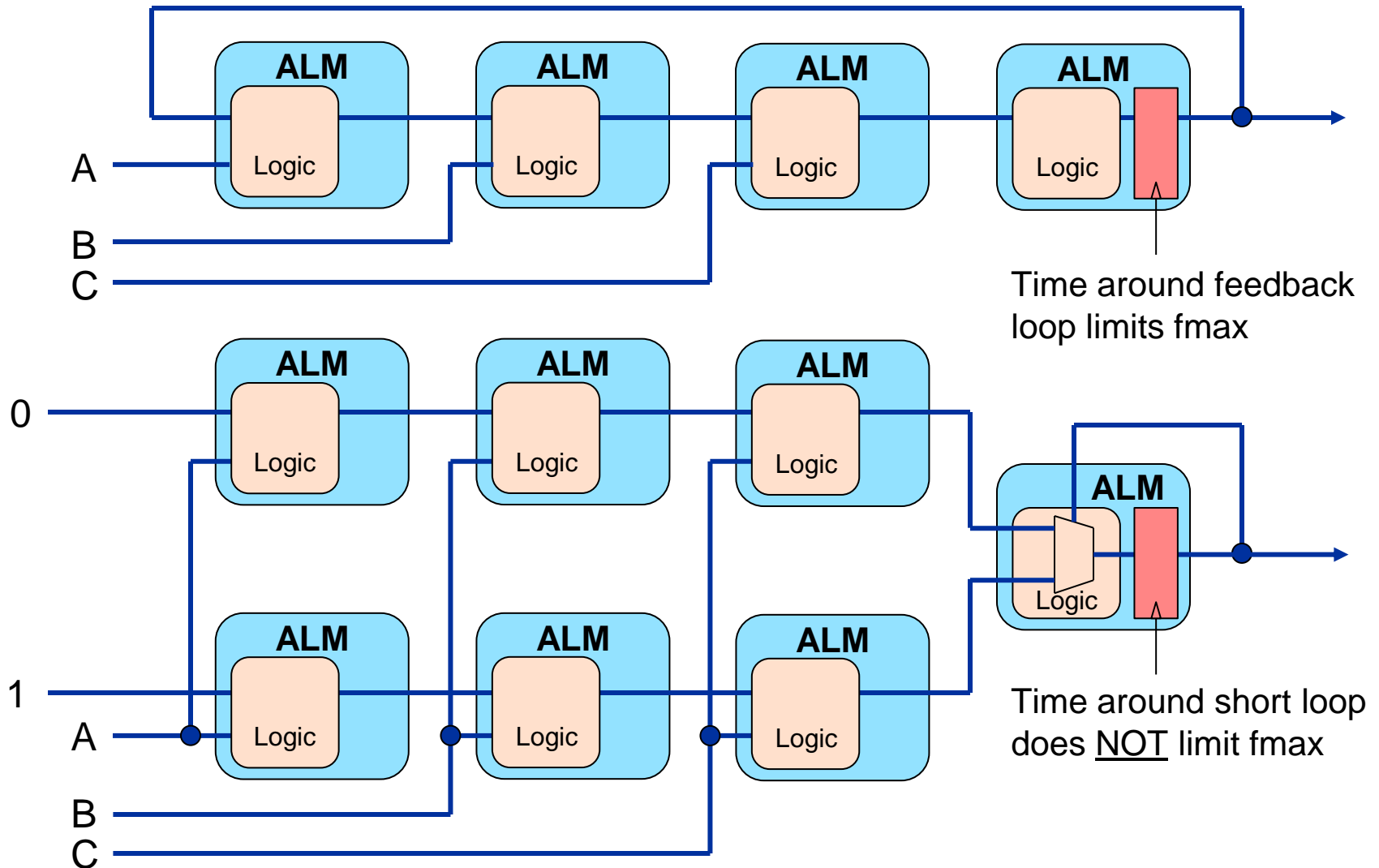
The Altera logo is displayed in a stylized, outlined font. It is positioned on a white, rounded rectangular background that is part of a larger blue graphic element at the bottom of the slide. The logo consists of the word "ALTERA" in all caps, with a registered trademark symbol (®) to its right.

**ALTERA®**

# Non-Optimized Feedback Loop

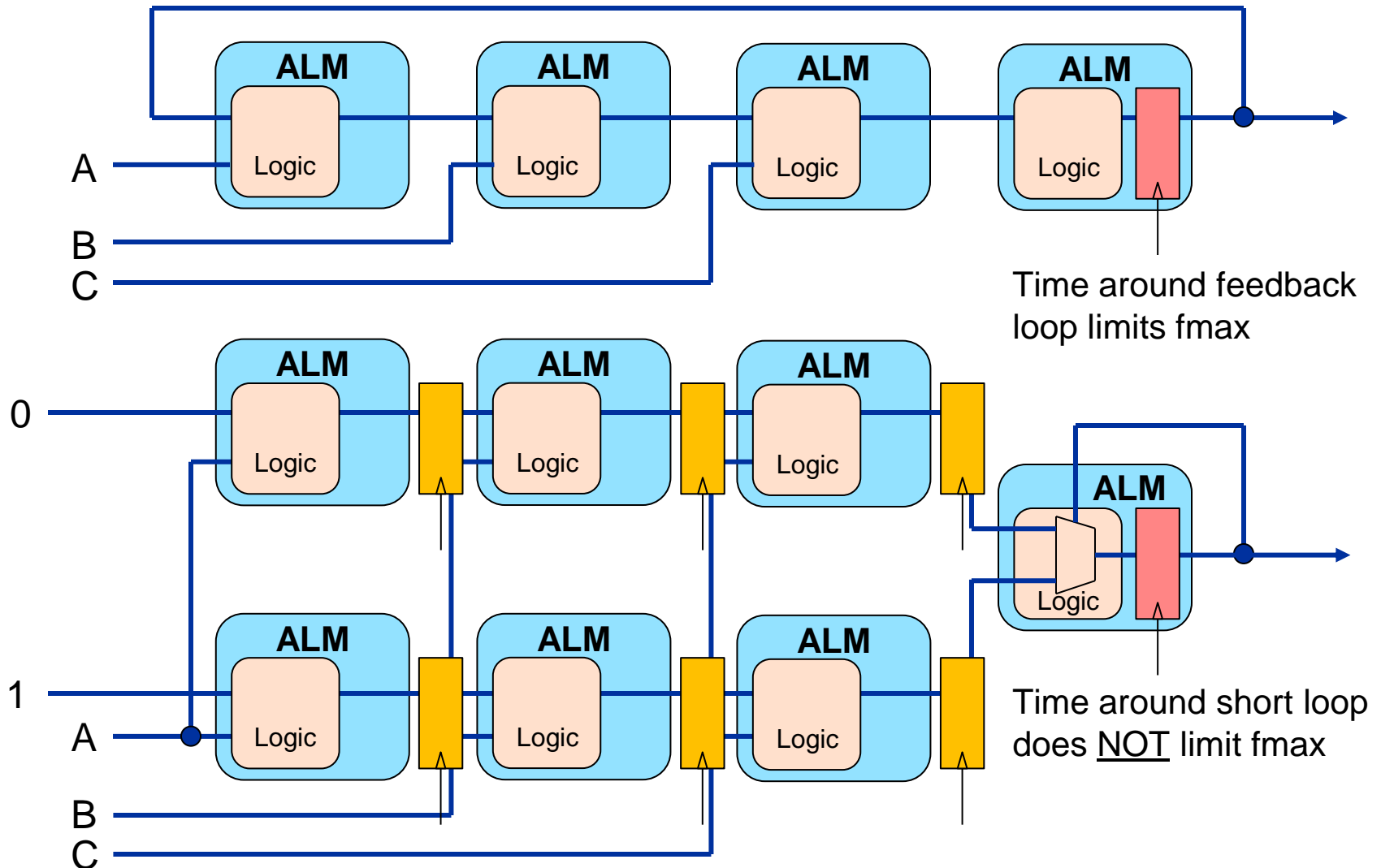


# Optimization



*Pre-compute to shorten feedback loops*

# Hyper-Optimization



*Pre-compute to shorten feedback loops*

# Programmable Clock Tree Synthesis



# Programmable Clock Tree Synthesis

## ◀ Reduced clock tree insertion delay

- Clock trees synthesized locally, precisely where needed
- Less variability, tighter clock skew between end points
- Allows higher fmax, up to 1GHz core clocking

## ◀ Greater clocking flexibility

- Uses dedicated clock routing instead of predefined clock trees
- Allows for greater number of clocks
- Up to 32 independent clocks per sector (X rows by Y columns)

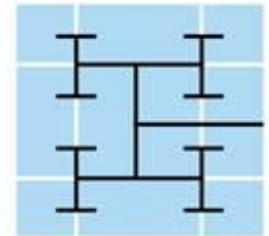
## ◀ Lower power dissipation

- Smaller clock trees, fine-grained approach
- Less switched capacitance
- Smart branch disabling

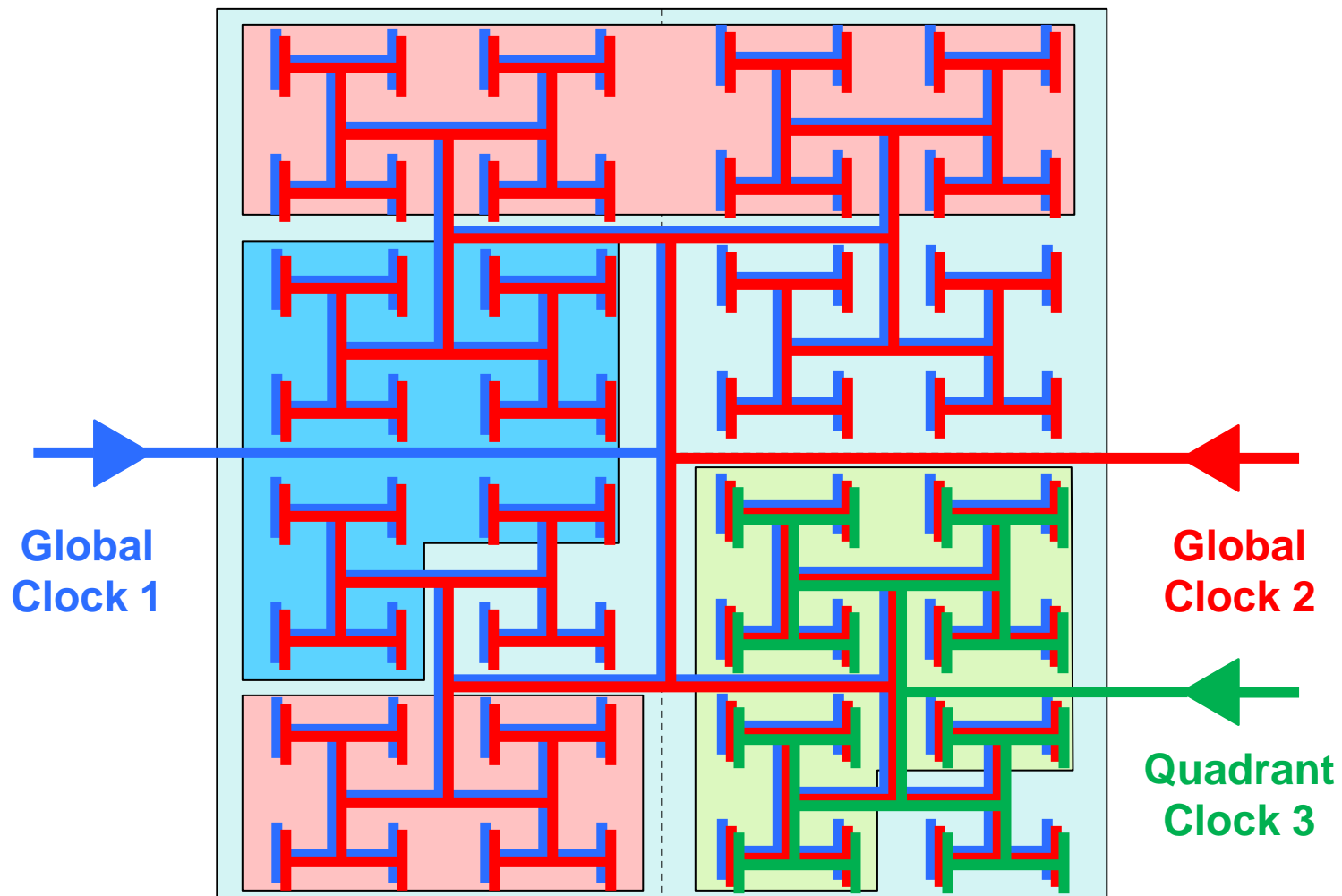
## ◀ Backwards compatible

- Easy to synthesize global, quadrant, regional clocks

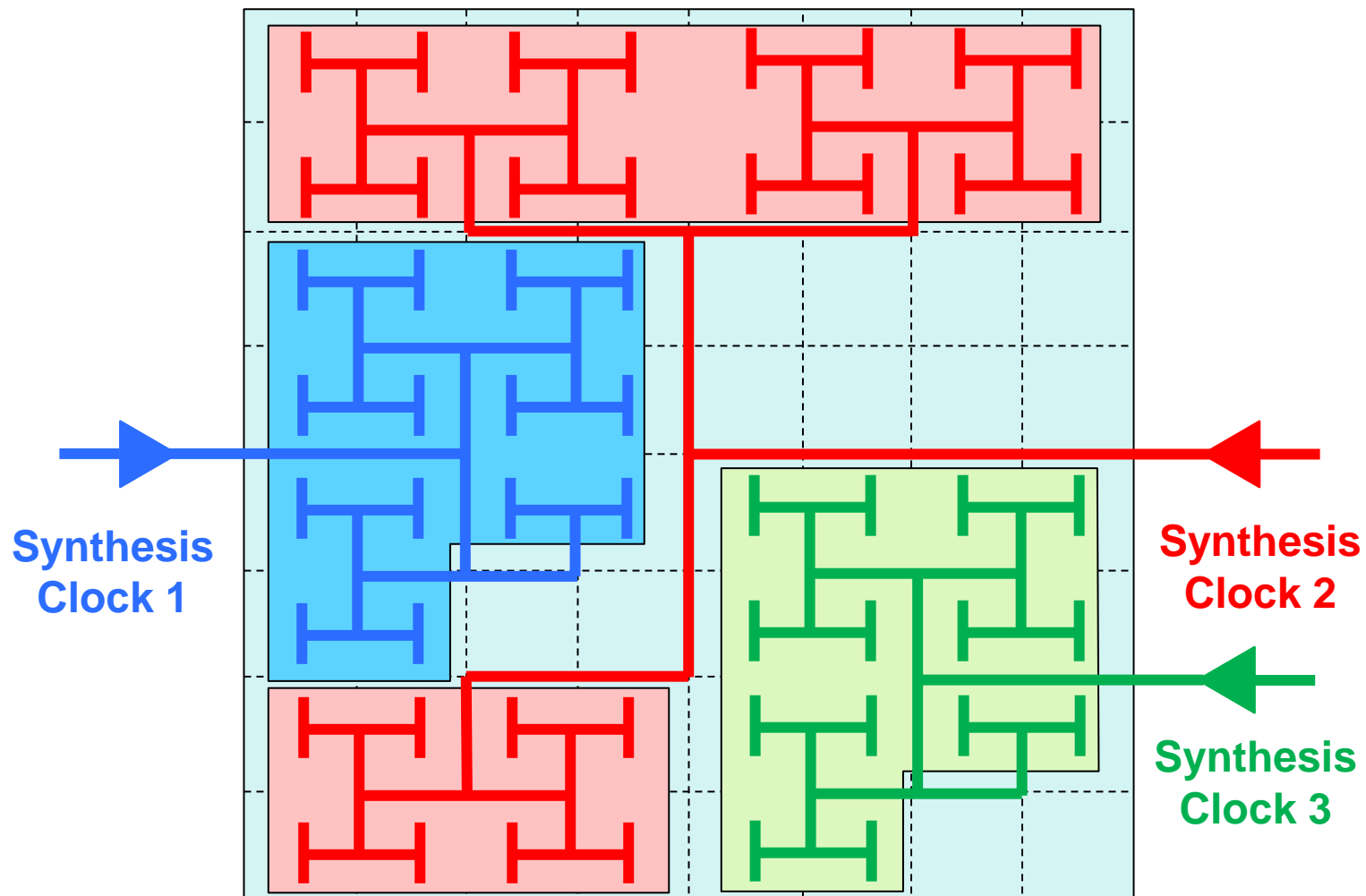
*Programmable Clock Trees Support  
“Registers Everywhere”*



# Conventional Global/Regional Clock Architecture



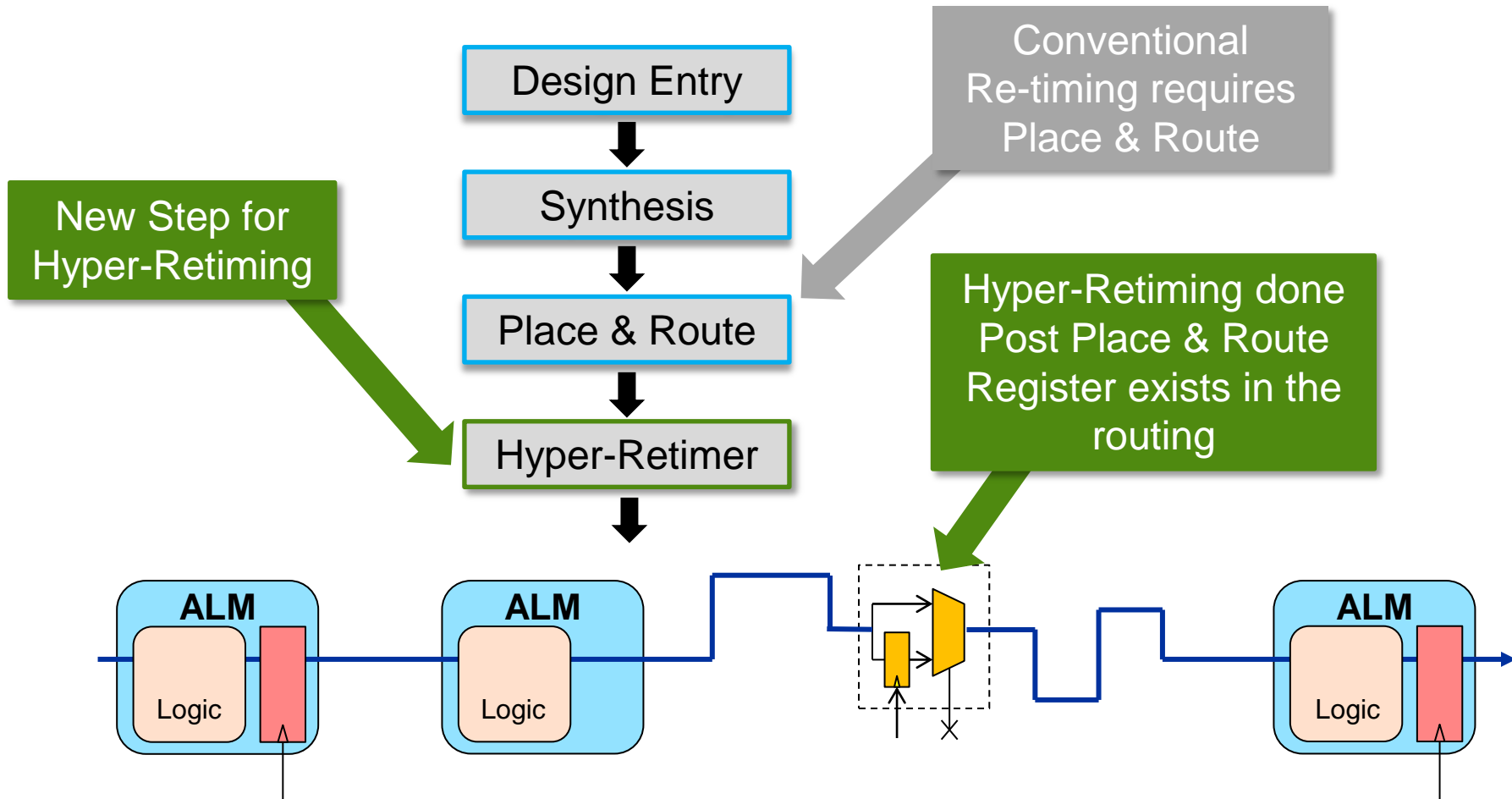
# Stratix 10 Programmable Clock Tree Synthesis



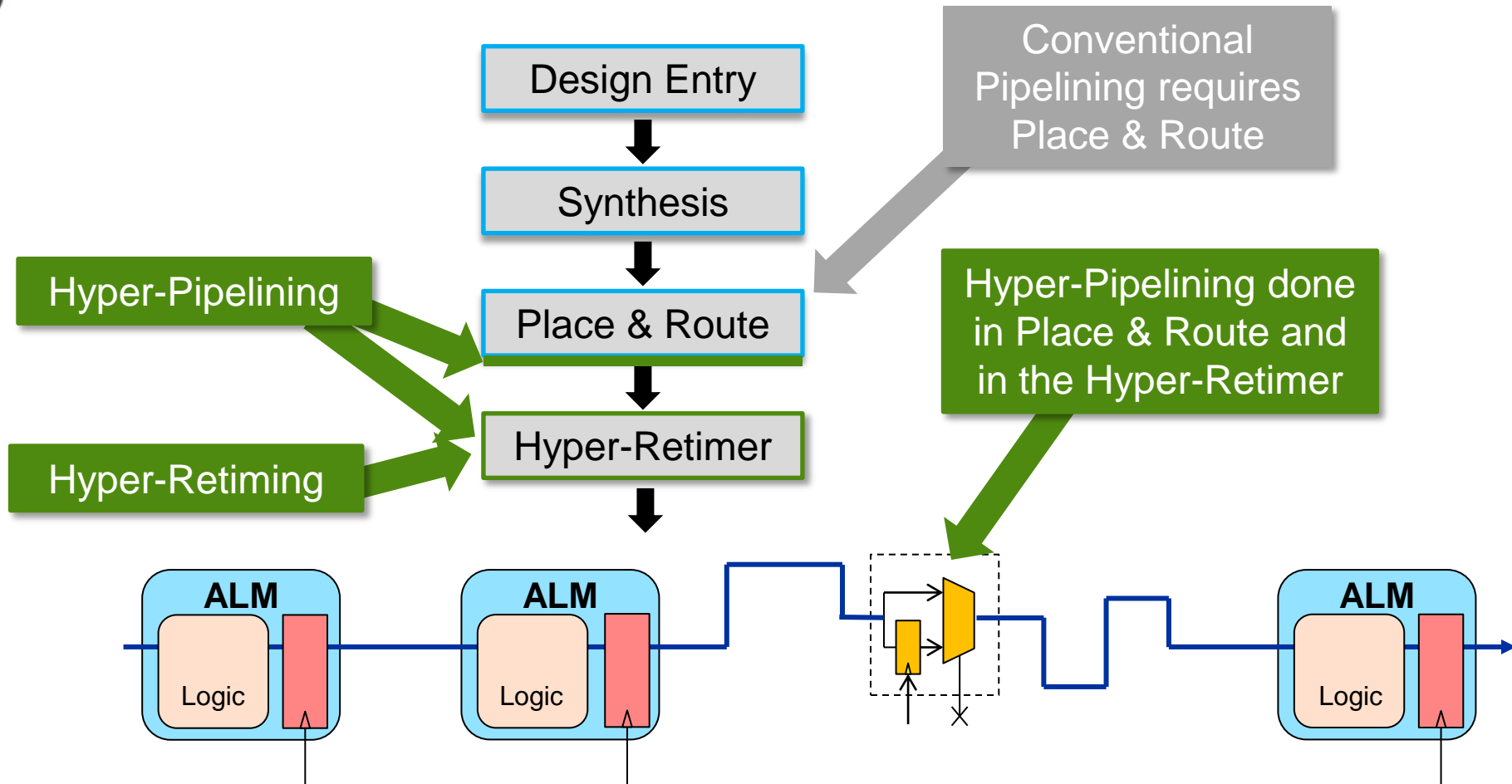
# Hyper-Aware Design Flow



# Hyper-Retiming Design Flow



# Hyper-Pipelining Design Flow



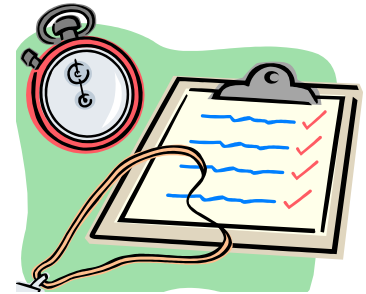
# Stratix 10 Early Access Software



# What's Available in Stratix 10 Early Access Software

## ◀ Stratix 10 Early Access SW Capabilities

- **Performance estimation tool** for Stratix 10 core fmax
- **Design Performance exploration tool** with HyperFlex
  - ◀ Hyper-Retiming
  - ◀ Hyper-Pipelining
  - ◀ Hyper-Optimization
- Based on an Arria 10 baseline design & Fast Forward Compile
  - ◀ Requires porting the design to Arria 10



## ◀ Not a full Quartus Release for Stratix 10

- No EMIF Support
- No Pin Assignments
- No Transceiver Support
- No Device Selection

## ◀ Available Today through your FAE

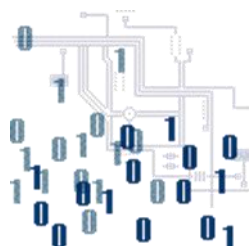
- Requires additional free license on top of valid Quartus II license

# Stratix 10 Early Access Software Flow

```
module binary_adder_tree (a, !
parameter width = 16;
input [width-1:0] a, b, c,
input clk;
output [width-1:0] out;

wire [width-1:0] sum1, sum
reg [width-1:0] sumreg1, s
// Registers

always @ (posedge CLK)
begin
sumreg1 <= sum1;
sumreg2 <= sum2;
sumreg3 <= sum3;
sumreg4 <= sum4;
end
```



Design  
Entry  
(RTL)

Standard  
Arria 10  
Compile

Fast  
Forward  
Compile

Report  
Predicted  
Stratix 10  
Fmax  
+  
Report how to  
increase  
performance  
further

**HyperFlex**<sup>TM</sup>  
ARCHITECTURE

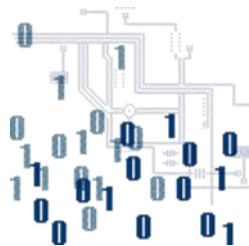
# Stratix 10 Early Access Software Flow

```

module binary_adder_tree (a, b, c, clk, out);
    parameter width = 16;
    input [width-1:0] a, b, c;
    input clk;
    output [width-1:0] out;

    wire [width-1:0] sum1, sum2, sum3, sum4;
    reg [width-1:0] sumreg1, sumreg2, sumreg3, sumreg4;
    // Registers

    always @ (posedge CLK)
    begin
        sumreg1 <= sum1;
        sumreg2 <= sum2;
        sumreg3 <= sum3;
        sumreg4 <= sum4;
    end
end
    
```



Report  
 Predicted  
 Stratix 10  
 Fmax  
 +  
 Report how to  
 increase



Clock Fmax Summary				
	Clock Name	Fmax	Achieved with Hyper-Retiming	Achieved with Hyper-Pipelining
		Fmax	Achieved with Hyper-Retiming	Achieved with Hyper-Pipelining
		425.89 MHz	687.76 MHz	811.03 MHz

**Base  
Stratix 10  
Fmax**

**Fmax Achieved After:  
Virtual Hyper-Retiming  
Virtual Hyper-Pipelining**

# Step-by-Step Guidance to Increase Performance

## Performance Improvement Recommendations



Compilation Report - /data/gchiu/hipi/14.0\_147\_20140

File Edit Tools Window Help

Table of Contents

- Flow Summary
- Flow Settings
- Flow Non-Default Global Settings
- Flow Elapsed Time
- Flow OS Summary
- Flow Log
- Analysis & Synthesis
- Fitter
- Flow Messages
- Flow Suppressed Messages
- HyperRetimer
  - Summary
  - Clock Fmax Summary
  - Clock Worst-Case Slack Sum
  - Fast Forward Summary for Clock Domain clk
  - Fast Forward Details for Clock Domain clk
  - Internal Retiming Statistics
  - TimeQuest Timing Analyzer

Options Summary Fast Forward Summary for Clock Domain clk

Step	Limiting Reason	Recommendations Applied	Pipelining Required	To Achieve Fmax
1 Base Performance	Insufficient Registers	0 recommendations applied	0 stages (0 paths)	425.89 MHz
2 Fast Forward Step #1	Short Path/Long Path	1152 recommendations applied	1 stage (1152 paths)	687.76 MHz
3 Fast Forward Step #2	Short Path/Long Path	1152 recommendations applied	2 stages (1152 paths)	811.03 MHz
4 Hyper-Optimization	Short Path/Long Path	1152 recommendations applied	2 stages (1152 paths)	--

4 Hyper-Optimization Short Path/Long Path 1152 recommendations applied 2 stages (1152 paths) -- -0.233 0.970

**Critical Chain for Base Result**

Critical Chain Recommendations at Retiming Limit

**Critical Chain for Base Result**

Critical Chain Recommendations at Retiming Limit

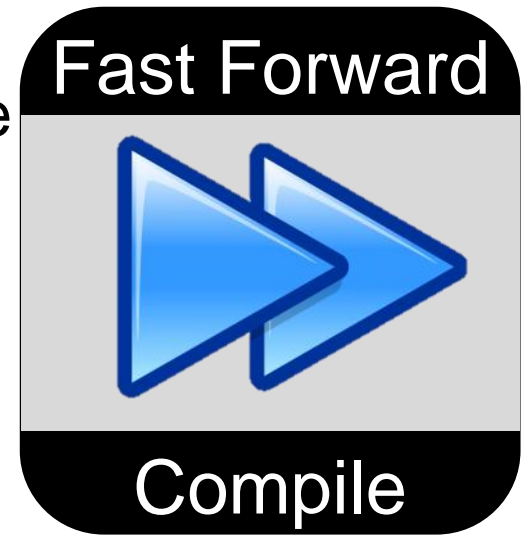
Recommendation	
1	The critical chain is a long path, or sequence of long paths.
1	Address restrictions at one of the two chain endpoints, or
2	Reduce the length of 'Long Paths' in the chain, or
3	Insert more pipeline latency in 'Long Paths' in the chain.
2	
3	Add Additional Pipeline Stages at Endpoint #1
1	Add additional pipeline stages
1	From din_reg[4][1]
2	To din_reg[4][1]

0% 00:00:00

### Steps to Achieve Performance Increase

# Fast-Forward Compile – Performance Exploration

- ◀ Analyzes how fast a design can go with the HyperFlex Architecture
- ◀ How it works?
  - Virtually Implements Hyper-Retiming
  - Virtually Implements Hyper-Pipelining
  - Identifies bottlenecks for Hyper-Optimization
- ◀ Fast-Forward Compile generates an accurate “what-if” Fmax
  - Provides detailed guidance to maximize performance using Hyper-Retiming and Hyper-Pipelining
  - Identifies critical paths for further Hyper-Optimization



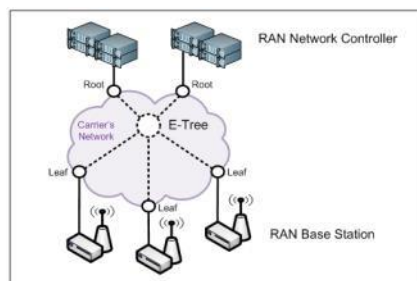
Available Now!  
Stratix 10  
Early Access  
Software

# HyperFlex Performance Benchmarks



# Benchmark Results From Real Designs

Benchmark	Data Path	Control Logic	Co-Processor
Design Target	> 700 MHz	> 550 MHz	300 MHz
Baseline	302 MHz (1X)	132 MHz (1X)	156 MHz (1X)
+ Hyper-Retiming	426 MHz (1.4X)	185 MHz (1.4X)	205 MHz (1.3X)
+ Hyper-Pipelining	518 MHz (1.7X)	276 MHz (2.1X)	305 MHz (1.96X)
+ Hyper-Optimization	745 MHz (2.4X)	623 MHz (4.7X)	Not required



# Stratix 10 HyperFlex Application Notes

2014.12.15  
AN714 | Subscribe | Send Feedback

## Hyper-Retiming for Stratix 10 Designs

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, setting 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 combinational logic. Retiming across node splits and merges may involve register duplications or merges. By balancing the register 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 clocks, 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 reinserting 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, EMPIRON, MAX, MEGACORE, NIOS, QUANTUS 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 of their respective holders are identified as such in this document. 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.

101 Innovation Drive, San Jose, CA 95134

## AN714

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

2014.12.15  
AN715 | Subscribe | Send Feedback

## Hyper-Pipelining for Stratix 10 Designs

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, EMPIRON, MAX, MEGACORE, NIOS, QUANTUS 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 of their respective holders are identified as such in this document. 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.

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

2014.12.15  
AN716 | Subscribe | Send Feedback

## Hyper-Optimization for Stratix 10 Designs

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 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  $t_{MAX}$  of the critical chain is limited by the average delay of a register-to-register path, and quantization delays of individual 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, EMPIRON, MAX, MEGACORE, NIOS, QUANTUS 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 of their respective holders are identified as such in this document. 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.

101 Innovation Drive, San Jose, CA 95134

## AN716

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

# Thank You

© 2015 Altera Corporation—Public

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/legal](http://www.altera.com/legal).

The Altera logo is rendered in a bold, blue, blocky font with a registered trademark symbol (®) to the right. It is positioned in the bottom right corner of the slide, partially overlapping a large, light blue, curved graphic element that sweeps across the bottom of the page.

**ALTERA**®