High Performance Design with Stratix 10

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

Agenda

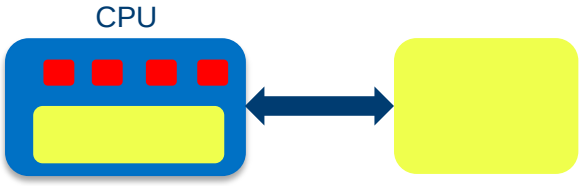
FPGA Processing

OpenCL – in brief

The Stratix 10 Fabric

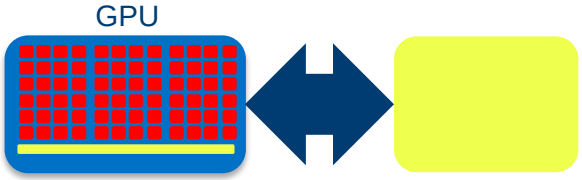
intel

Multiple Processing Architectures



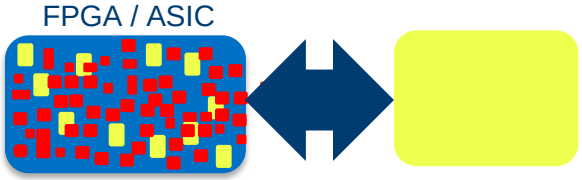
Up to tens of complex control processing elements

Large hierarchical cache



Thousands of simple streaming processing elements

High bandwidth GDDR / HBM



Millions of tiny processing elements, dataflow architecture

Embedded State

Heterogenous Systems Require Multiple Programming Environments

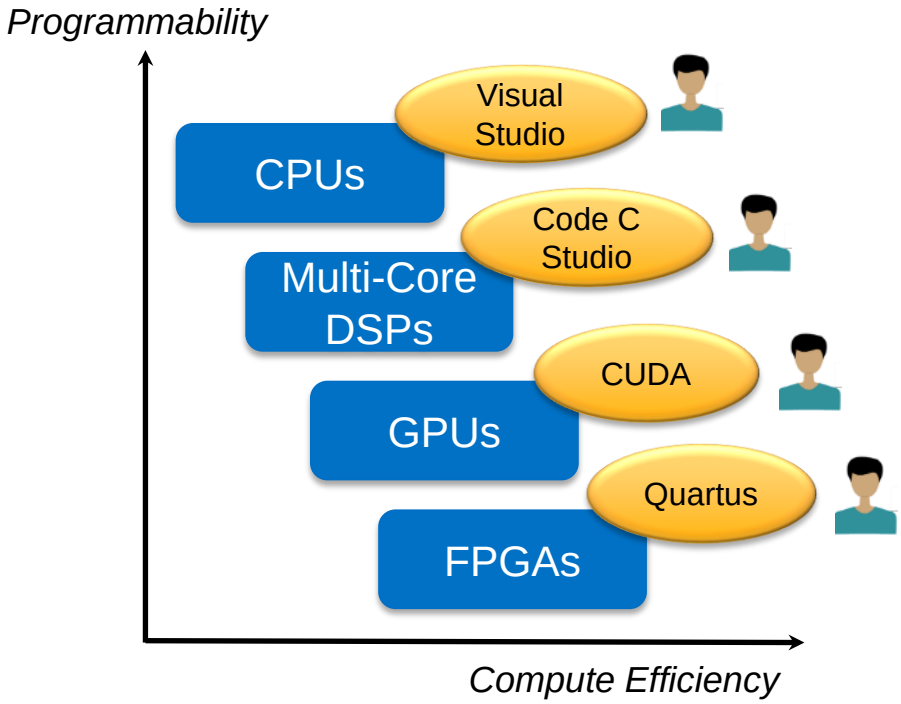
Different hardware platforms offer tradeoffs in functionality and compute efficiency

- CPUs for Complex multi-dimensional control
- GPUs for Image processing
- FPGAs for Data flow Processing

Each platform has a unique programming environment

- Different skillsets required

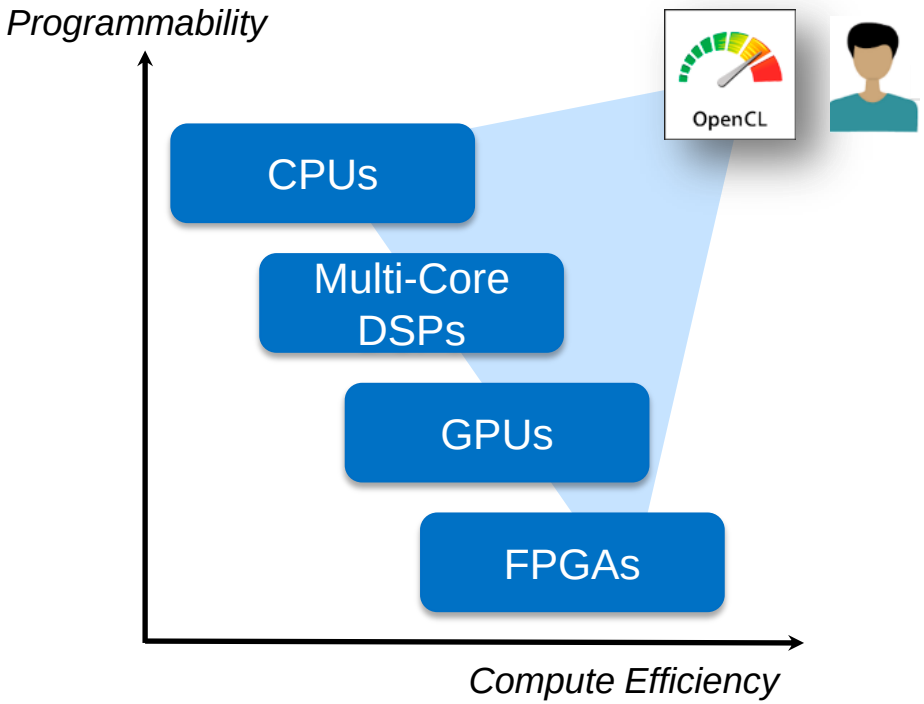
Difficult and ad-hoc system stitching processes



OpenCL Provides A Single Environment for Heterogeneous Platforms

Enables a single user with one skillset to target multiple platforms

- Enables developer to optimize for performance
- Automates heterogenous system stitching



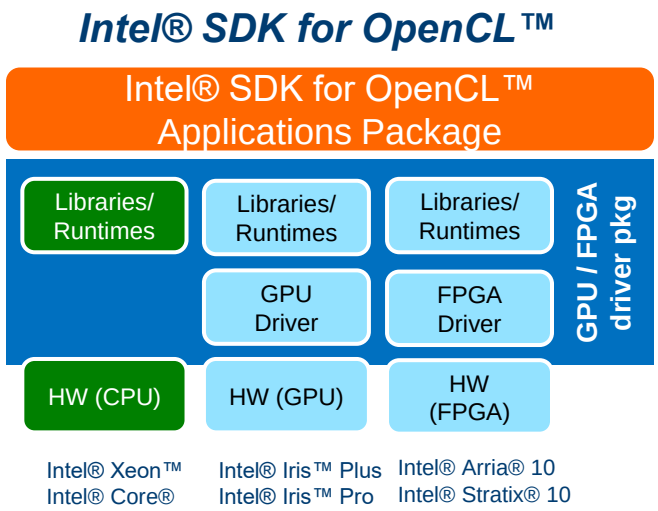
Intel SDK for OpenCL now with FPGA Support (Nov'17)

World class, professional environment for developing and optimizing OpenCL

- Eclipse based IDE for Linux
- Microsoft Visual Studio for Windows
- OpenCL Keyword Highlighting
- Kernel Analysis Tools and Report viewing

Optimize Host and Accelerator code from common environment

- Track events and queues on the host
- Optimize CPU to FPGA system performance



Common Environment for CPU, GPU and FPGA Programming

Scalable Compute

Preliminary specification – subject to change

520

High performance compute node with optical IO for creation of directly-coupled, dense FPGA clusters

Nallatech a molex company

EARLY ACCESS PROGRAM OPEN!

intel

OpenCL

Stratix 10 FPGA • SoC

- **GPU** form factor (3/4 length) - dual slot standard configuration with Active and Passive cooling options
- 16-lane PCIe Gen 3.0
- **(1) Intel Stratix 10 FPGA GX280** F1760 NF43 speed grade 2 (050 → 280 pin compatible)
- **(4)** QSFP28 Cages supporting 1G → 100G line rates
- **(4)** banks of DDR4 SDRAM @ 2400MTPS, up to 128GB total
- On-board USB hub for system control and monitoring
- Board Support Packages (BSP) for Intel OpenCL SDK

FPGA Compute Node

Maximize Acceleration – Increase Efficiency – Save Time

The **Nallatech FPGA Accelerated Compute Node** allows you to drive the most demanding HPC, data visualization and rendering workloads with a flexible, extremely dense 1U rack server optimized for accelerators.

Features:

- High Density 1U Rack Server Optimized for Energy-Efficient FPGA Acceleration
- (4) Nallatech 510T Cards with (8) Latest-Generation Altera / Intel Arria 10 FPGAs
- **OpenCL** Programming Support
- Nallatech Application Optimization and Custom Configurations Available

Benefits:

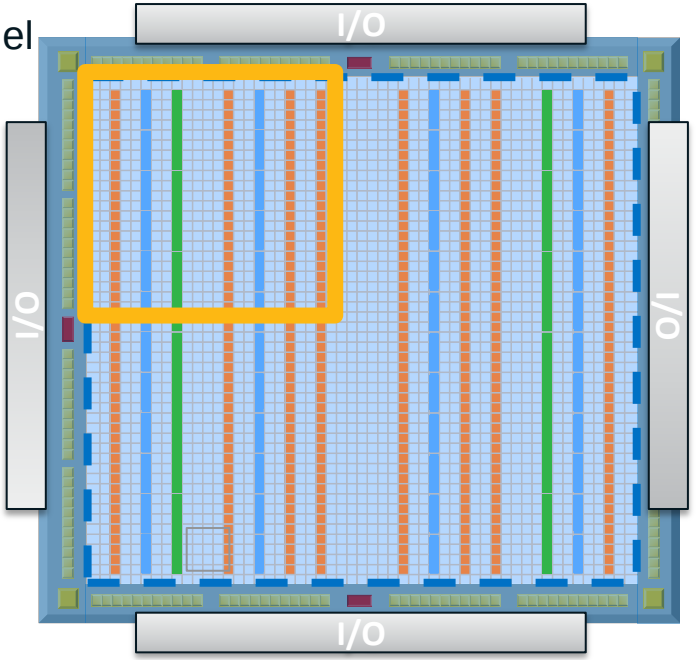
- Rich Blend of Processor, FPGA and Memory Resources Provides Ultimate Platform for FPGA Accelerated Computing
- Dense Form-Factor Provides Datacenter Scalability
- System, Power and Thermal Management so you can Focus on Application Development Instead of System Integration

The FPGA Fabric

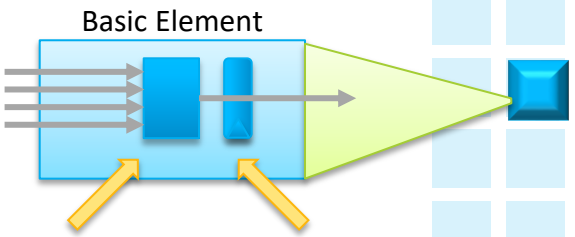
So Why Are FPGAs So Power Efficient?

FPGA Architecture: Fine-grained Massively Parallel

- Millions of reconfigurable logic elements
- Thousands of 20Kb memory blocks
- Thousands of Variable Precision DSP blocks
- Dozens of High-speed transceivers
- Multiple High Speed Memory Controllers
- Multiple ARM© Cores

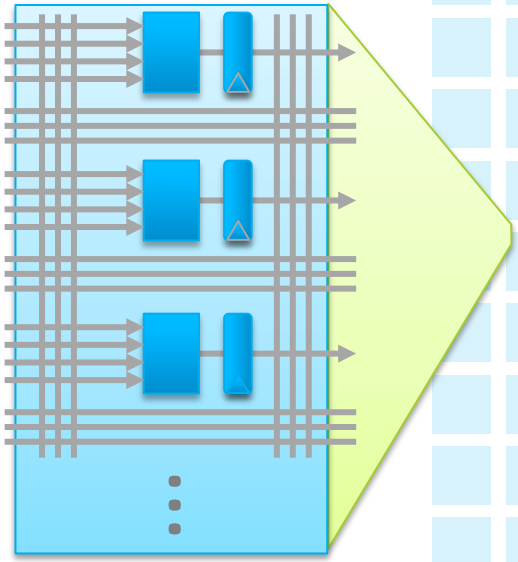


FPGA Architecture: Basic Elements



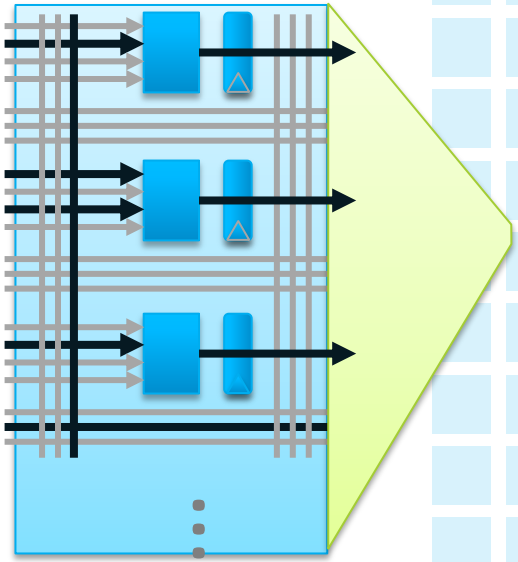
Configured *to perform any 1-bit operation: AND, OR, NOT, ADD, SUB*

FPGA Architecture: Flexible Interconnect



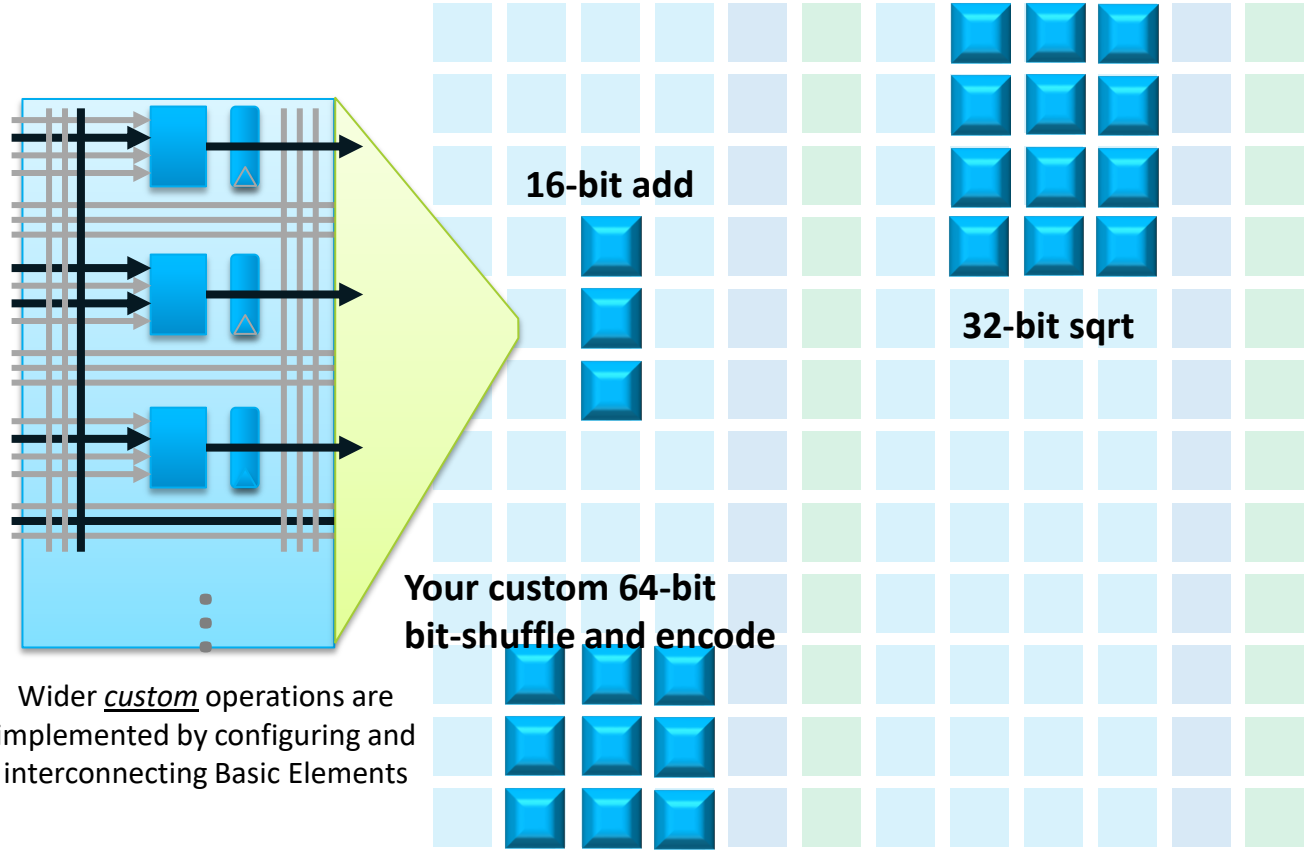
Basic Elements are surrounded
with a
flexible interconnect

FPGA Architecture: Flexible Interconnect



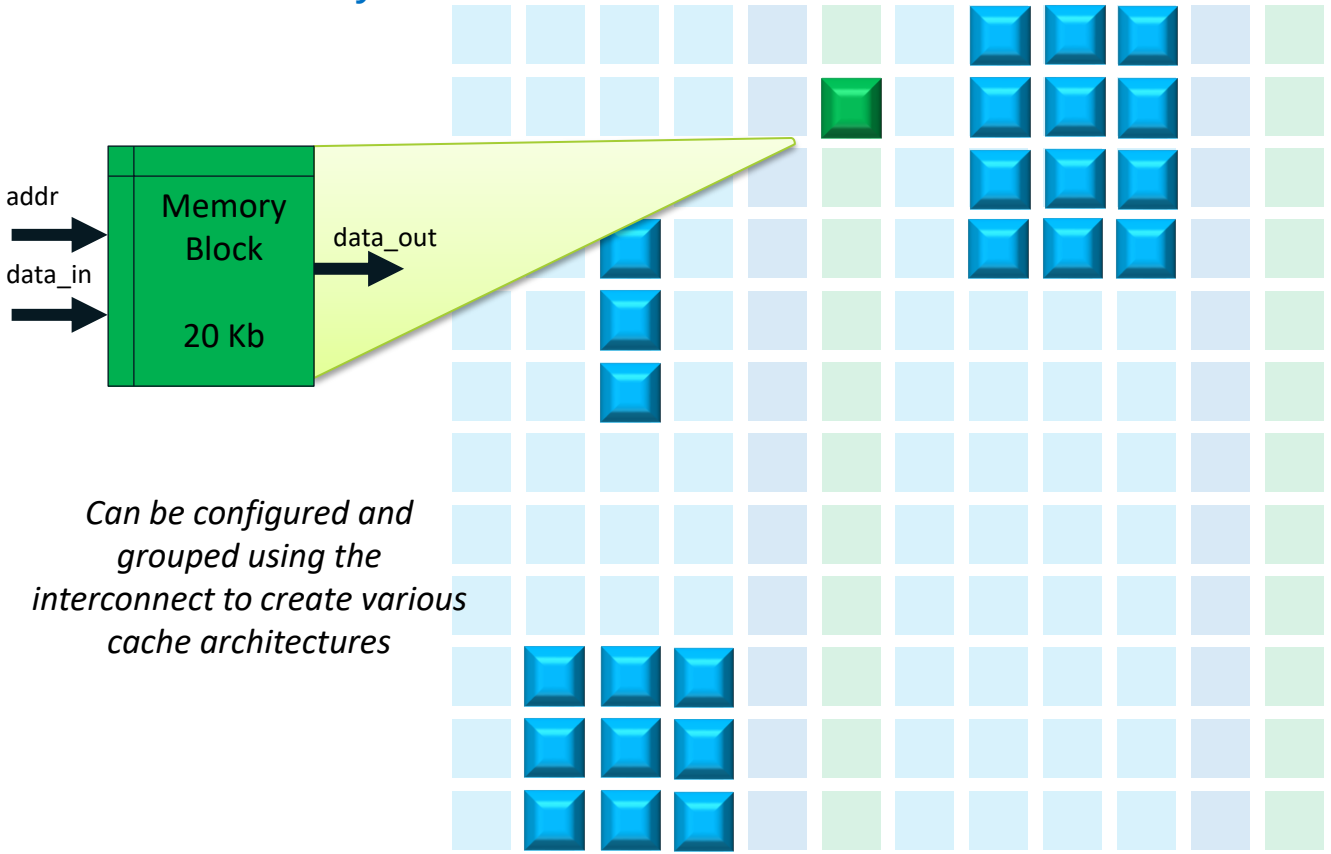
Wider *custom* operations are implemented by configuring and interconnecting Basic Elements

FPGA Architecture: Custom Operations Using Basic Elements



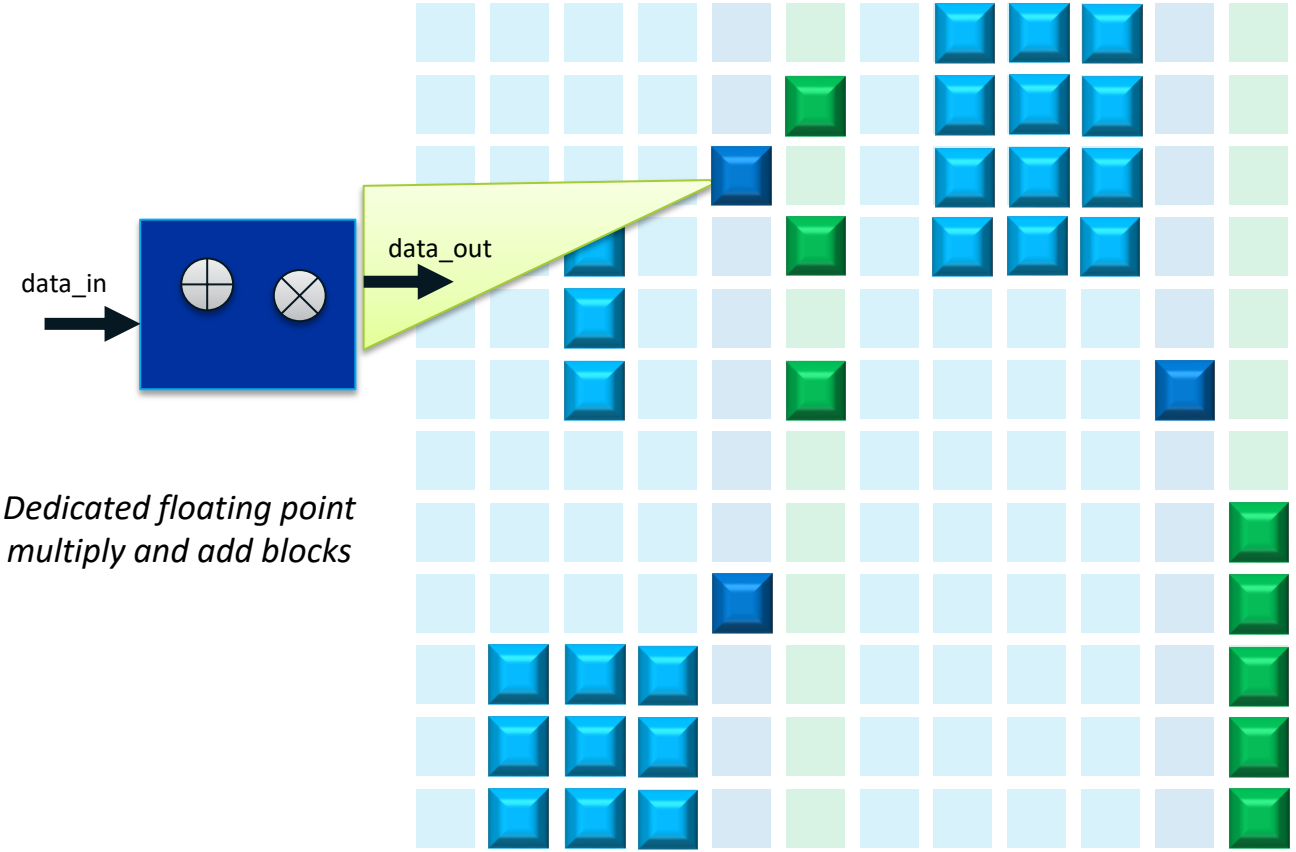
Wider *custom* operations are implemented by configuring and interconnecting Basic Elements

FPGA Architecture: Memory Blocks



Can be configured and grouped using the interconnect to create various cache architectures

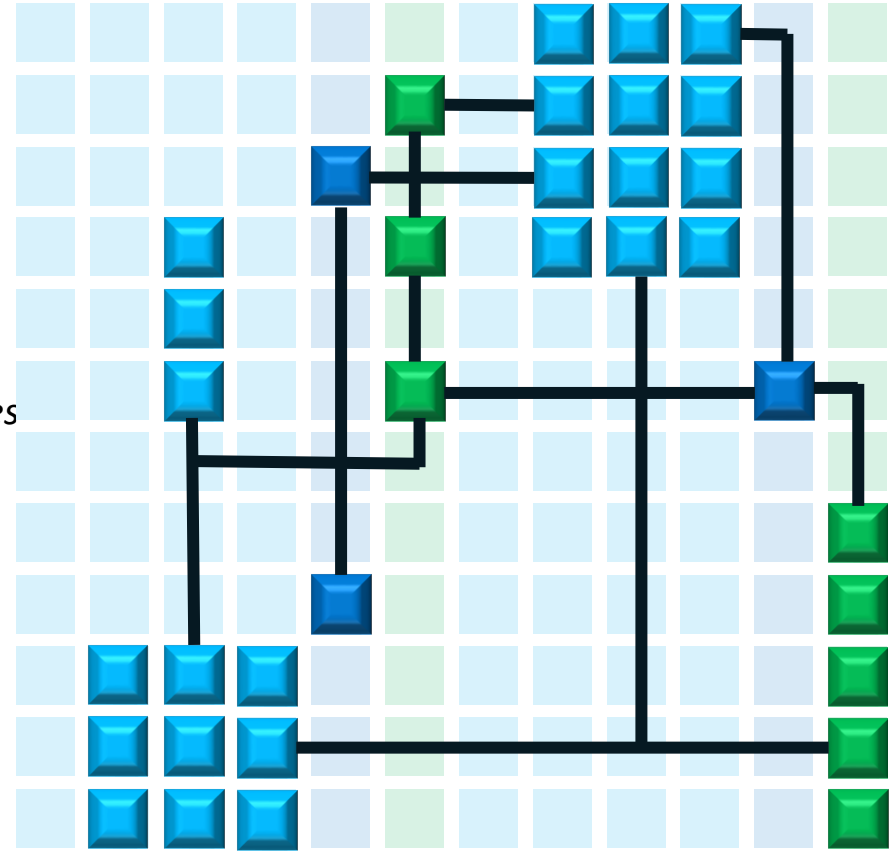
FPGA Architecture: Floating Point Multiplier/Adder Blocks



Dedicated floating point multiply and add blocks

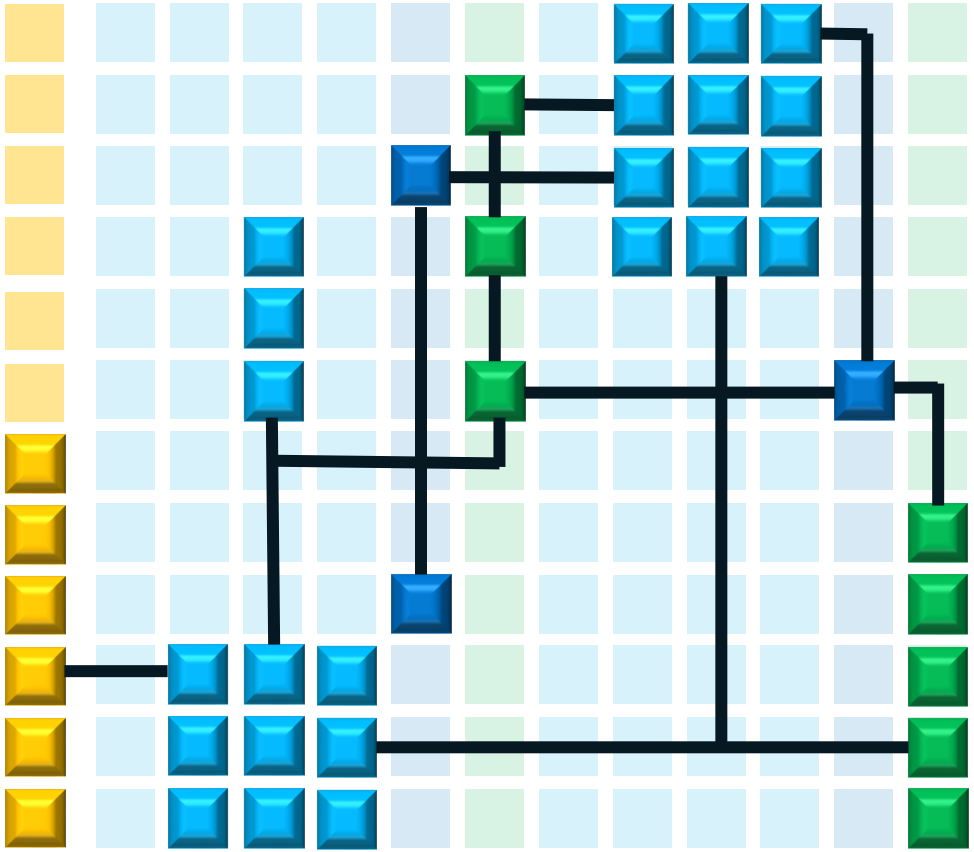
FPGA Architecture: Configurable Routing

*Blocks are connected into a **custom data-path** that matches your application.*

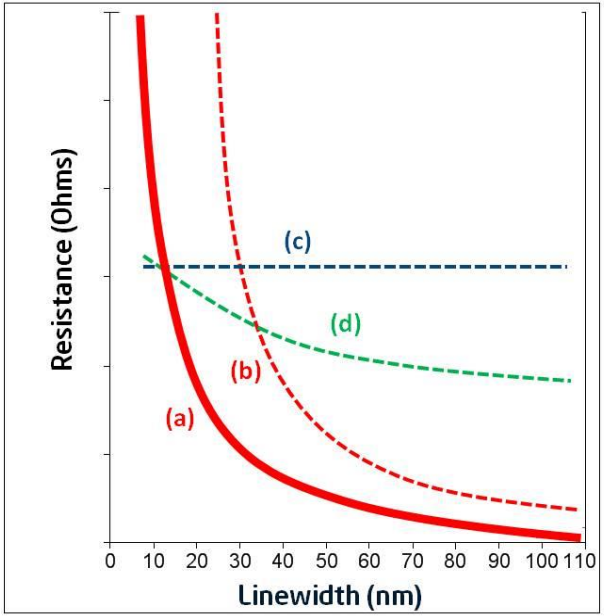


FPGA Architecture: Configurable IO

*The **Custom data-path** can be connected directly to **custom or standard IO interfaces** for inline data processing*

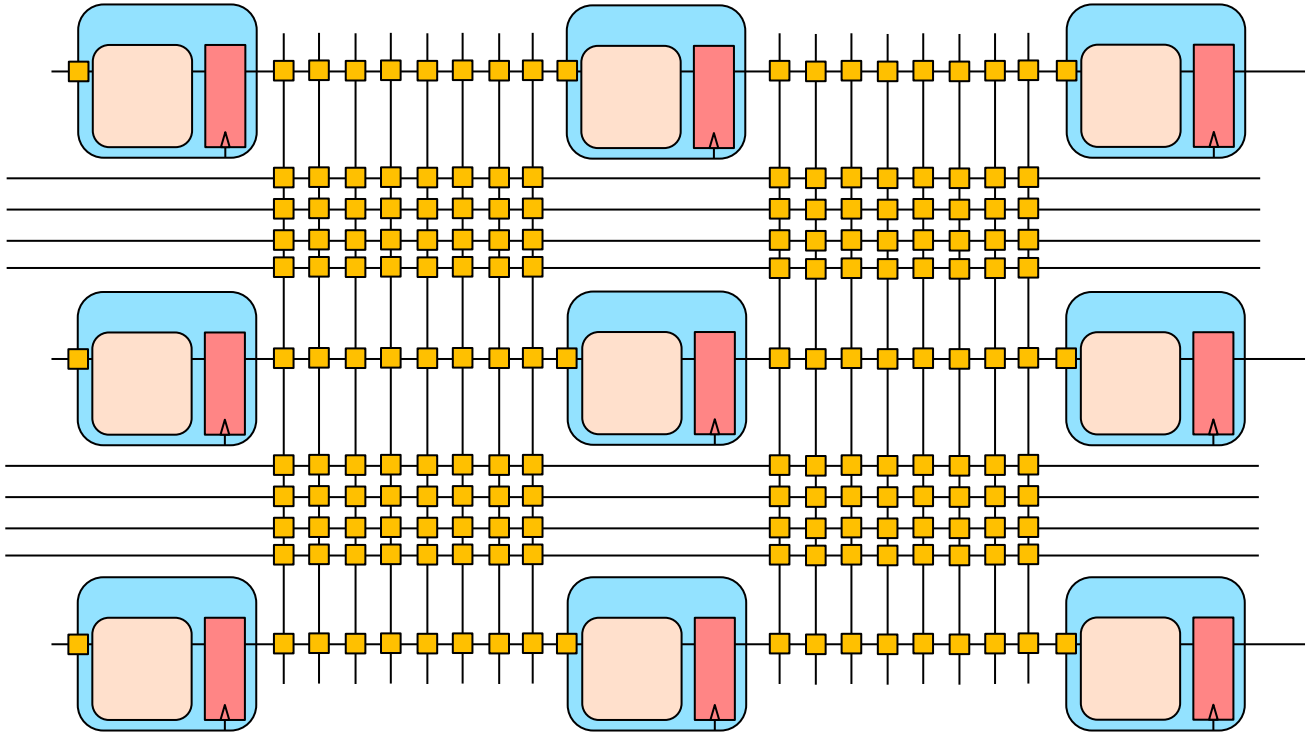


Interconnect Dominates Delay



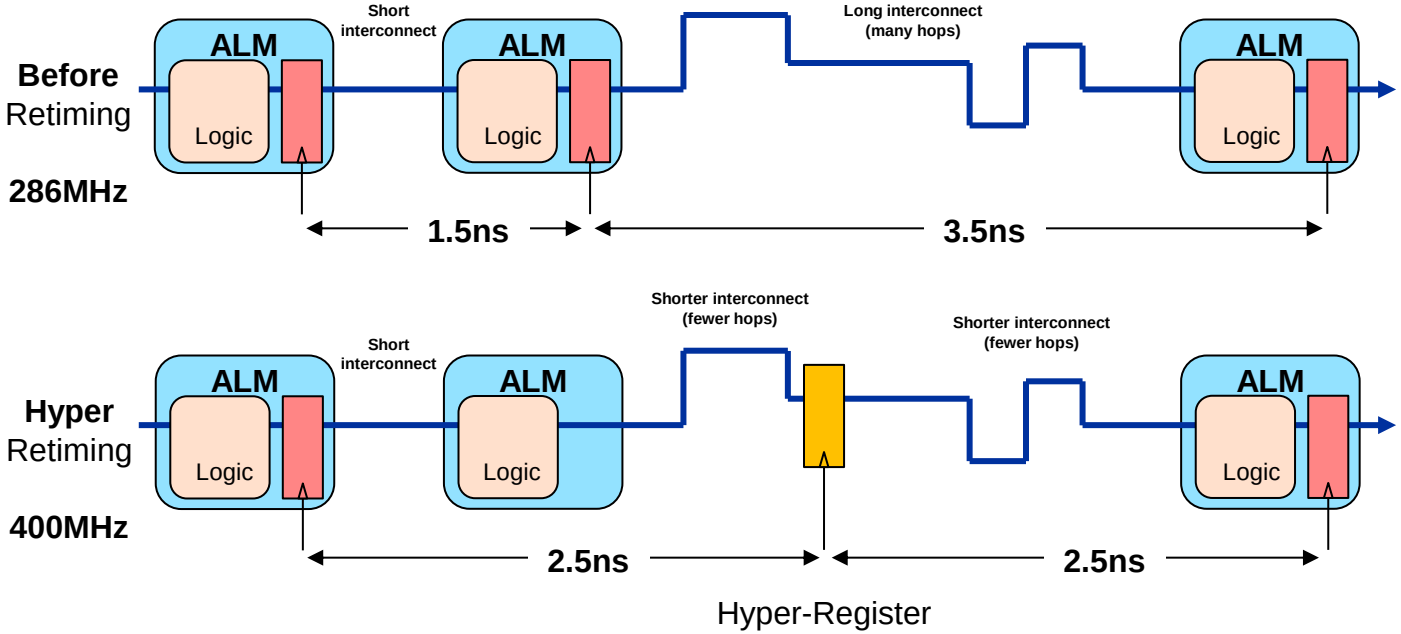
Process Technology Scaling in an Increasingly Interconnect Dominated World (James Clarke, Intel, VLSI 2014)

The Fabric - HyperFlex



= Hyper-Register

Hyper-Retiming



Hyper-Retiming step occurs AFTER place & route!

286MHz → 400MHz = 40% gain

The FPGA Designer

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**

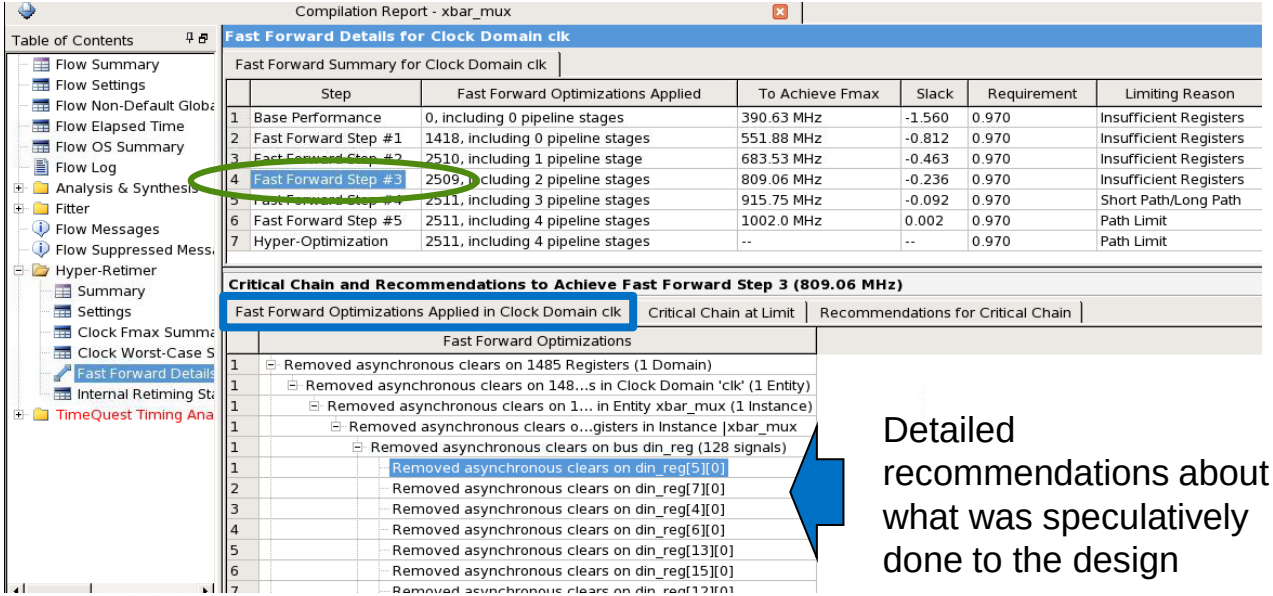
- Register A can’t be retimed forward any more
- Register B can’t be retimed backward any more
- Sometimes A and B are the same, indicating a loop

Performance is limited by the **average delay of a register hop**

  – Less additional delay quantization effect

A critical path abstracted to a higher level

Fast Forward Optimizations Applied Tab

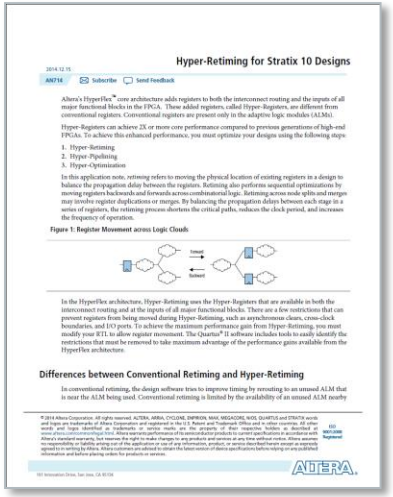


Detailed recommendations about what was speculatively done to the design

Detailed Recommendations about what was done to the design (and what you can do) to reach this step in performance

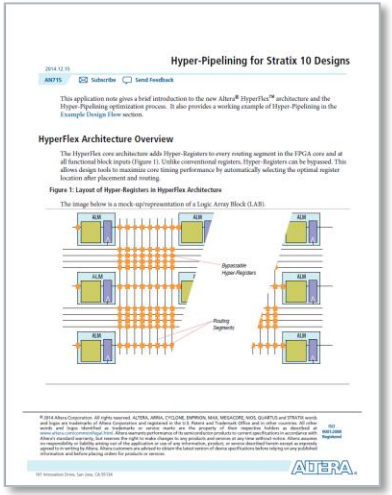
No need to look at ALL recommendations; designer can handle busses/common inputs simultaneously

Stratix 10 HyperFlex Application Notes



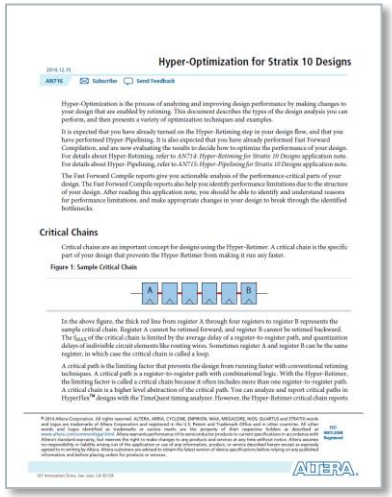
AN714

- **Hyper-Retiming**
- How it works
- Understand what to do with register controls



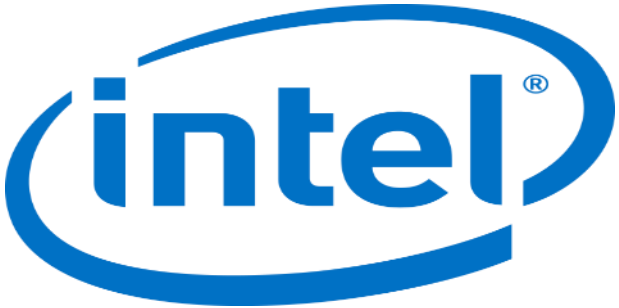
AN715

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



AN716

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


intel