Stratix 10 HyperFlex Architecture Overview

Delivering the Unimaginable

Tom Spyrou
Distinguished Architect
TAU 2016

Now part of Intel
2X Core Performance

5.5M Logic Elements

Up to 70% Lower Power

Up to 10 TFLOPS

Heterogeneous 3D SiP Integration

Intel 14 nm Tri-Gate

Most Comprehensive Security

Quad-Core Cortex-A53 ARM Processor
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
  - Not possible with conventional 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

Available “everywhere” throughout user logic and interconnect
The HyperFlex Architecture – A Fine Grained Approach

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
Why Stratix 10 is Fast

**Conventional architectures**
- Using register stages incurs significant additional delay
- Limits number of pipeline stages that can be added

**HyperFlex architecture**
- Significantly reduce cost of adding pipeline stages to a design
Why Stratix 10 is Fast

HyperFlex architecture

- Significantly reduce cost of adding pipeline stages to a design

Routing Wire
LUT
Routing Wire
Routing Wire
Routing Wire
Background: Routing Muxes

- Large portion of die area is routing muxes

- Each routing mux selects one signal to be output on routing wire
  - H3, H6, V4, etc, or into LAB

- Routing muxes interconnected ("routing pattern")
Stratix 10 HyperFlex Routing Muxes

- Extend routing muxes to include “register” stage

- 1 or 2 extra CRAM bits programmed to select a clock for the “register”
HyperFlex HW: Extra Register Locations

Add extra register locations

1. Bypassable registers in routing muxes

Routing muxes feeding programmable wires (H-wires, V-wire) can optionally be registered
HyperFlex HW: Extra Register Locations

Add extra register locations

1. Bypassable registers in routing muxes
2. Bypassable inputs to LUTs, FFs, DSPs, etc.

Inputs to FFs (shown) have optional bypassable registers
HyperFlex HW: Extra Register Locations

Add extra register locations

1. Bypassable registers in routing muxes
2. Bypassable inputs to LUTs, FFs, DSPs, etc.

LUT Inputs have bypassable registers
HyperFlex HW: Extra Register Locations

Add extra register locations

1. Bypassable registers in routing muxes
2. Bypassable inputs to LUTs, FFs, DSPs, etc.

**DSP / RAM Inputs have bypassable registers**
### How Do We Get to 2X Performance?

<table>
<thead>
<tr>
<th>Step</th>
<th>Architecture Advantage</th>
<th>Customer Effort</th>
<th>Stratix 10 versus Stratix V (Average Gain)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Hyper-Retiming</td>
<td>No change, or minor RTL changes</td>
<td>1.4X</td>
</tr>
<tr>
<td>2</td>
<td>Hyper-Pipelining</td>
<td>Added Pipelining</td>
<td>1.6X</td>
</tr>
<tr>
<td>3</td>
<td>Hyper-Optimization</td>
<td>More Effort</td>
<td>2X or more</td>
</tr>
</tbody>
</table>

- 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
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
Conventional Register Retiming

Before Retiming

286MHz

ALM
Logic

Short interconnect

1.5ns

ALM
Logic

Long interconnect
(many hops)

3.5ns

ALM
Logic
Conventional Register Retiming

Before Retiming

286MHz

1.5ns

3.5ns

After Retiming

333MHz

3ns

2.5ns

286MHz \rightarrow 333MHz = 16\% \text{ gain}
Hyper-Retiming

Before Retiming

286MHz

Short interconnect

Long interconnect (many hops)

1.5ns

3.5ns

Logic

Logic

Logic
Hyper-Retiming

Before Retiming

286MHz

Hyper Retiming

400MHz

Hyper-Register

Hyper-Retiming step occurs AFTER place & route!

286MHz → 400MHz = 40% gain
Unique challenges for STA

- In clock crossing the retimed register may be moved to a different clock but still achieve identical sequential behavior.
- Incremental timers often assume no change to the clock network and are not incremental with this type of change.
- CRPR credits must also be recalculated incrementally.
- Reconverge points updated incrementally.
- FPGA’s have large clock latency compare to ASICs.
- Increased latency already increases cost of CRPR.
- Now there are many more latch start points which need crpr tags with which to calculate the credit at the endpoint.

-TimeQuest 2 STA solves both of these problems.
HyperFlex Performance Benchmarks
### Benchmark Results From Real Designs

<table>
<thead>
<tr>
<th>Benchmark</th>
<th>Data Path</th>
<th>Control Logic</th>
<th>Co-Processor</th>
</tr>
</thead>
<tbody>
<tr>
<td>Design Target</td>
<td>&gt; 700 MHz</td>
<td>&gt; 550 MHz</td>
<td>300 MHz</td>
</tr>
<tr>
<td>Baseline</td>
<td>302 MHz (1X)</td>
<td>132 MHz (1X)</td>
<td>156 MHz (1X)</td>
</tr>
<tr>
<td>+ Hyper-Retiming</td>
<td>426 MHz (1.4X)</td>
<td>185 MHz (1.4X)</td>
<td>205 MHz (1.3X)</td>
</tr>
<tr>
<td>+ Hyper-Pipelining</td>
<td>518 MHz (1.7X)</td>
<td>276 MHz (2.1X)</td>
<td>305 MHz (1.96X)</td>
</tr>
<tr>
<td>+ Hyper-Optimization</td>
<td>745 MHz (2.4X)</td>
<td>623 MHz (4.7X)</td>
<td>Not required</td>
</tr>
</tbody>
</table>
Thank You