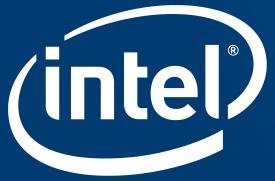


SpiderWeb high performance FPGA NOC

Martin Langhammer, Gregg Baeckler, Sergey Gribok

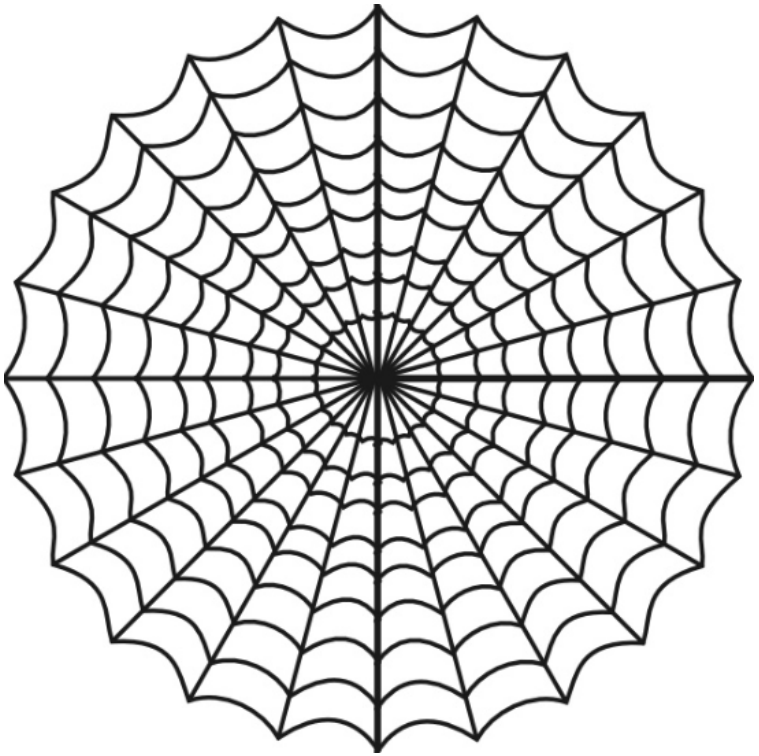
SpiderWeb – FPGA Soft NoC

The SpiderWeb NoC is named because it has a regular structure like a spiderweb

FPGA place and route is typically a minimization exercise

The approach of SpiderWeb is instead to force a regularization function

- Results in predictable timing
- Resource use
- Resource type



Conventional wisdom (i.e. prior work)

- You cannot build an effective high bus width, high speed soft NoC on FPGA
  - You can build a high bus width, low speed NoC
  - You can build a low bus width, high speed NoC
- This is a Place and Route problem
  - You never know how your NoC is going to end up on the device
  - You can only guess how it will interact with the rest of your design
- To fix this we need a different approach
  - We need a deterministic, repeatable NoC placement
  - The NoC has to have a deterministic resource usage profile

Method (leaving out many tricky details)

1. Using heuristics, fix the nodes of the NoC, for a reasonable target performance.
2. Do a trial place and route.
3. Keep the paths within your performance profile, drop the ones outside the profile.*
4. Repeat until all paths fit within performance profile.
5. Back-annotate to fix NoC

* it is just as important to drop paths that are too fast as the ones that are too slow. this converges the resource use to similar types

300Gbps Stratix10 NoC

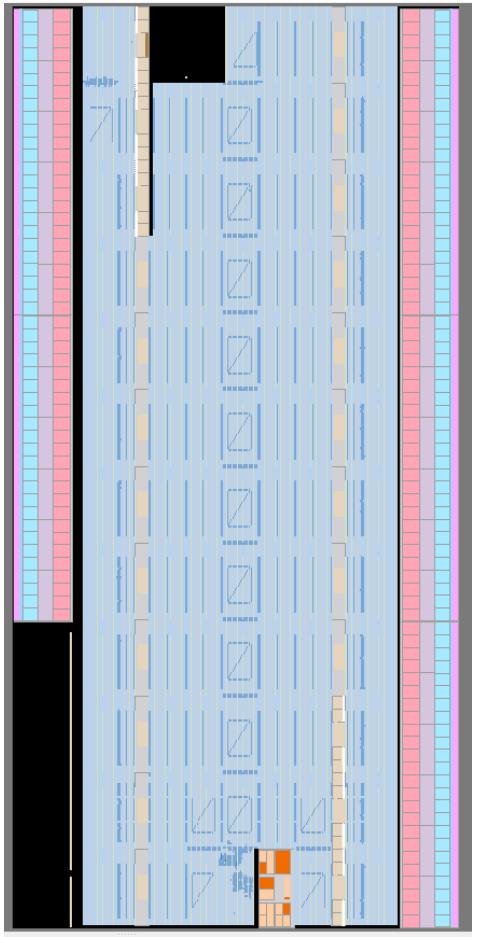
Can the chip fabric do it? Yes.

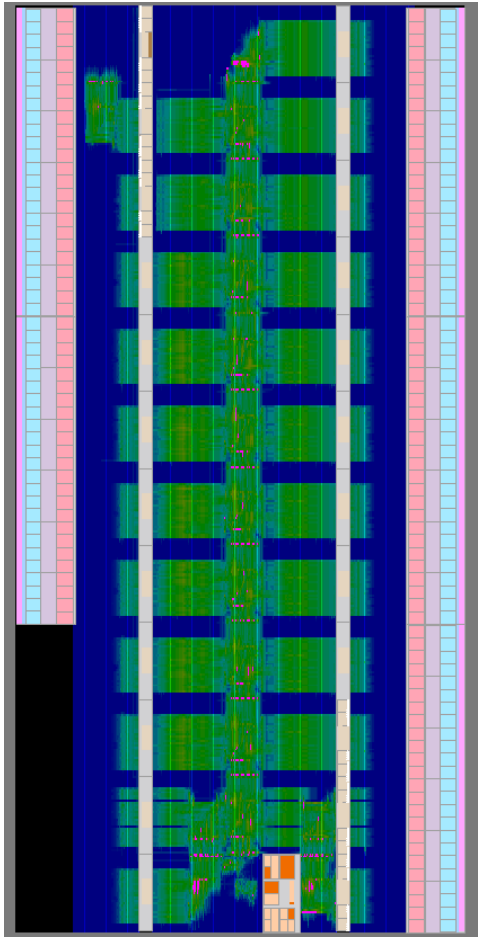
512 bit broadcast from bottom to all ~100 sectors of the device.

Register view on left. Routing on right. Minimal congestion.

587 MHz on mid-speed grade device

3	333.78 MHz	333.78 MHz	reconfig_clk
4	587.54 MHz	587.54 MHz	iopll_sclk_inst|iopll_0_outclk0





Fully Bidirectional NoC Version

Routing is arranged to travel opposite directions also

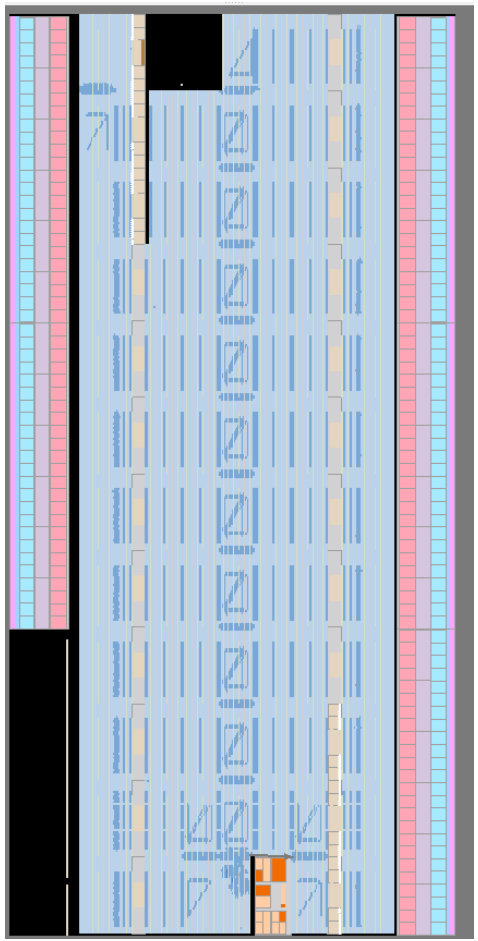
Minimal additional congestion

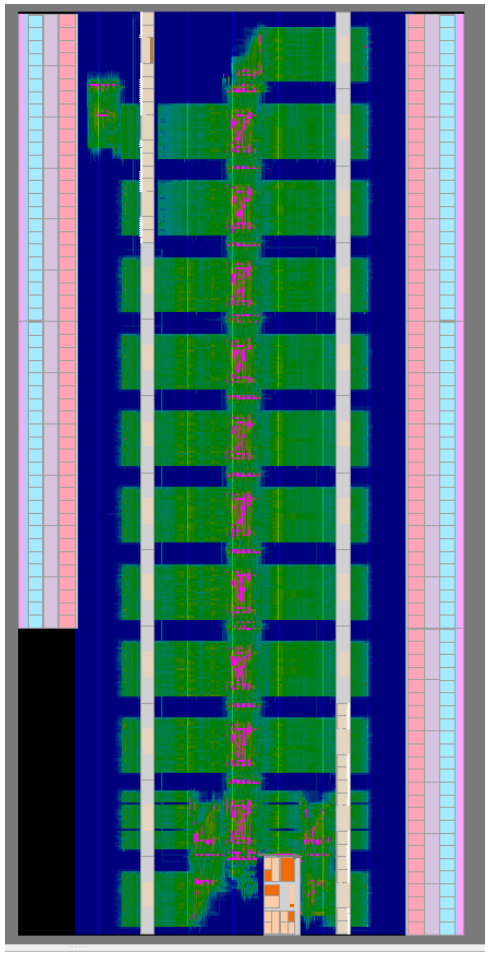
- only @ diagonal turns on the center spine (pink color)

Performance (Fmax) unchanged

Typically, return bandwidth is a fraction of outbound bandwidth

- real world area and routing impact minimal





Physical vs. Iterative Placement

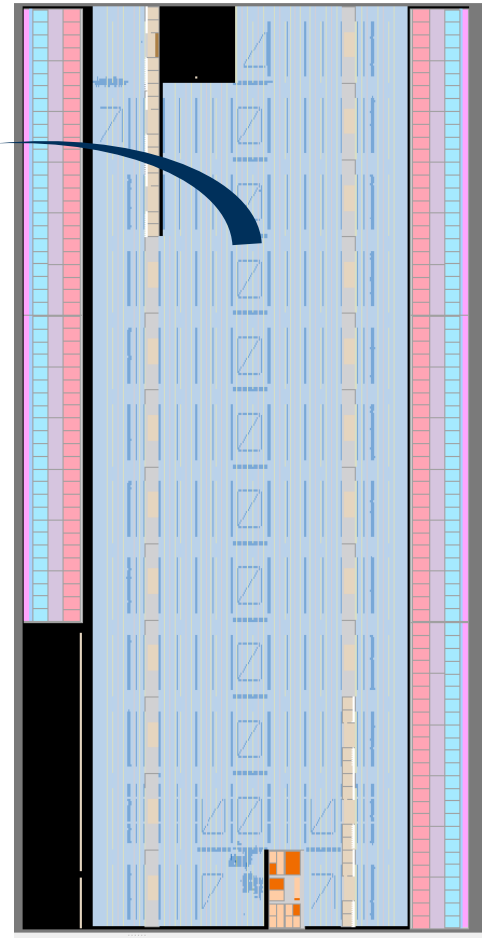
Some topologies (or subsets of the NoC) can be automatically placed

See “Method” slide

Others benefit from physical placement

Corner turn from Spine to Leaf

Many different topologies can be built with the described method

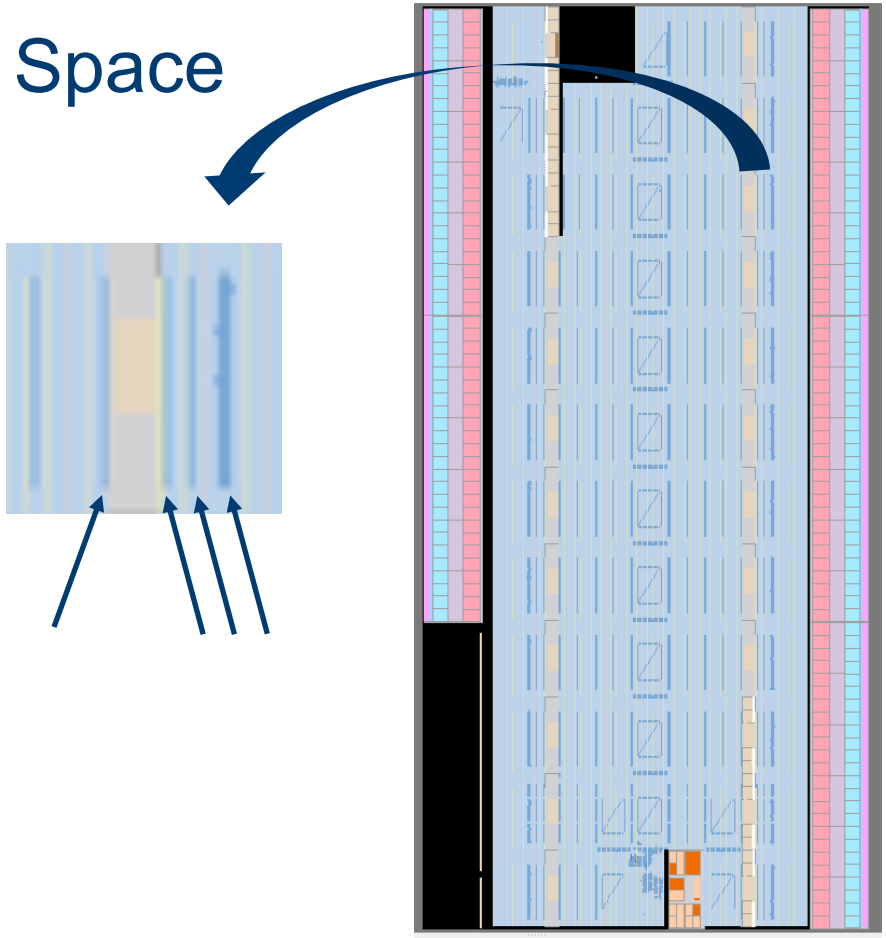


Constant Time != Constant Space

Distance is not the same as delay

Crossing I/O column long t_{pd}

This is where the iterative method – rejecting both too fast and too slow paths – finds the optimal solutions



Scalabilty and Flexibility

SpiderWeb is not a fixed NoC – the 512b 300 Gbps design in the paper is just an example

Once a SpiderWeb has been designed, the resource usage (both logic and wires) is fixed. It will then always fit the exact same way onto a device, with the exact same performance

The 512b NoC requires 10% device logic. If you want a smaller NoC (*e.g.* 128b) then it will use less logic – 2%.

The iterative design approach means that you don't need to know architectural details to design a good NoC.

- If you are too aggressive in speed or other specs it won't converge – will be obvious quickly

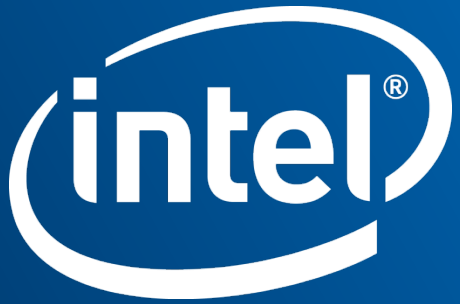
Conclusions

SpiderWeb is more than a soft NoC – consider it as a firm NoC

- With near hard NoC performance
- With soft NoC flexibility

A really big NoC (512b @ 300Gbps) takes 10% of the device. A smaller NoC will take a lot less.

Iterative method within range (throw away both too slow and too fast paths) means that resources usage known and interaction with the rest of the design is understood before you begin the architecture phase.


intel®