A Scalable and Robust Hierarchical Floorplanning to Enable 24-hour Prototyping for 100k-LUT FPGAs

Ganesh Gore, Xifan Tang and Pierre-Emmanuel Gaillardon
University of Utah, Salt Lake City, Utah, USA

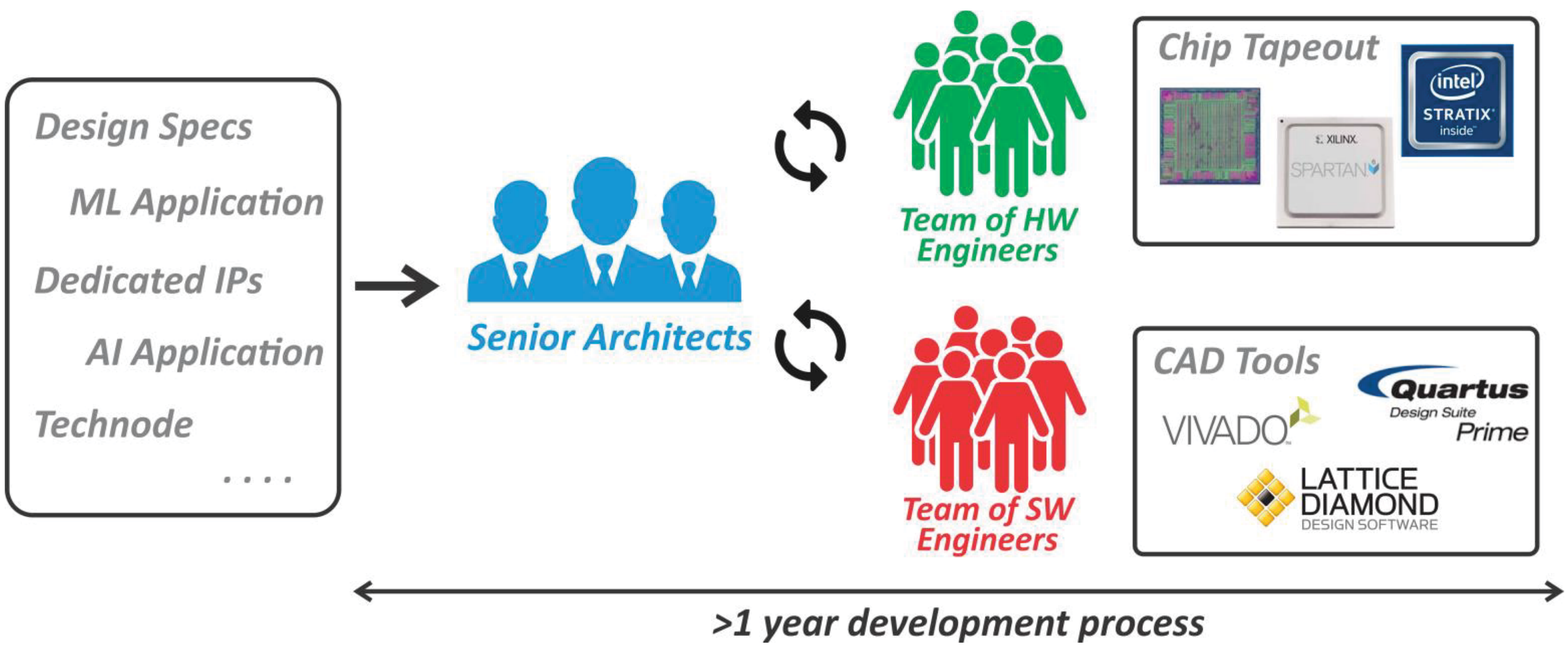
Research Sponsors:
AFRL and DARPA under agreement number FA8650-18-2-7855.

COLLEGE OF ENGINEERING
THE UNIVERSITY OF UTAH

ISPD 2021, March 22-24, 2021

Traditional FPGA design flow

- FPGA device provide higher design flexibility, by trading area and power
- To get better Power, Performance and Area the custom design is used
- Custom design requires careful design planning and longer development cycles
- The FPGA application domains are evolving rapidly, also the architectural required



Related work

- **2005, Ian Kuon et al.** 8×8 (256 LUTs) in ***180 nm*** technology, using ***custom tool GILES*** and bottom-up design approach.
- **2008, Chaudhuri et al.** 8×8 (256 LUTs) ***eFPGA in SoC*** using ***130nm*** technology using the commercial EDA tools.
- ***2014, Archipelago:*** opensource tool to generate FPGA fabric and implementation using the commercial ASIC
- ***2015, Kim et al.:*** *Synth Std. Cell FPGA* ***20x20*** *using VTR tool + Custom cell*
- **2020 Ang Li et al.-** Cycle free routing, ***4x4*** FPGA– Trades the routing flexibility to achieve top-down ***flat design flow***
- **2020 P. Mohan et al. –** Uses top-down ***flat level design*** using auto generated timing constraint

1. ***50-60% degradation in the performance but a significant reduction in runtime.***
2. ***Custom-designed cells TGATE and DFF can narrows the gap to 20-30%***
3. ***Most effort has been limited to small and medium scale FPGA (<20K LUTs)***
4. ***Flat PnR approach is used***

Contribution of this work

1. Leveraging standard ***hierarchical flow enabled by modern ASIC tools*** for FPGA design

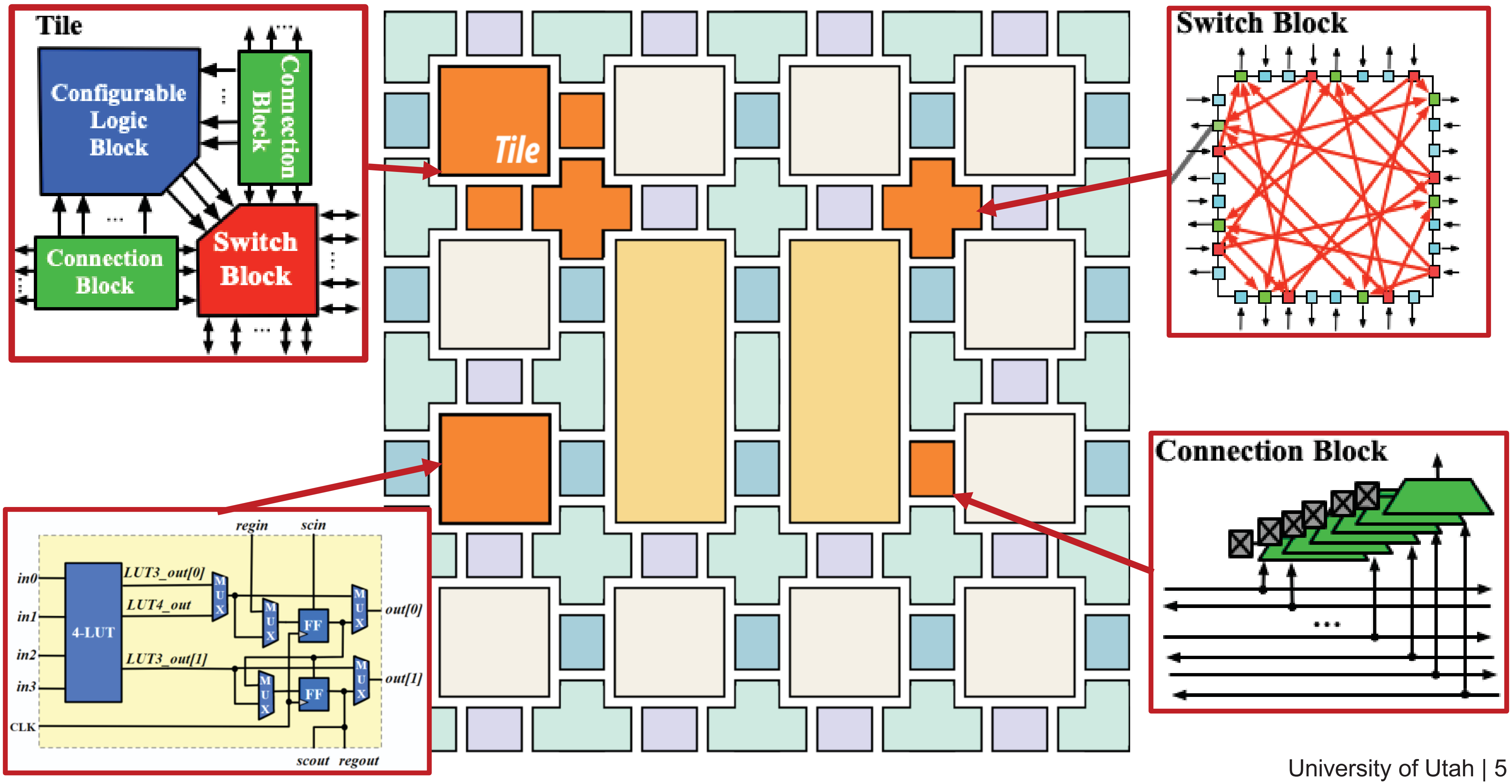
2. ***Optimizing hierarchical flow*** to design ***100k+ LUTs FPGA within 24-hours runtime*** by pre-planning the high-fanout nets and clock network

3. Improving the ***clock and global signal latency*** by Post-Placement and Routing (P&R) buffer sizing strategy

> ***1. The achieved runtime benefit allows the rapid FPGA fabric development***
> ***2. It enables comprehensive architecture exploration by considering the effect of physical design on architecture evaluation***

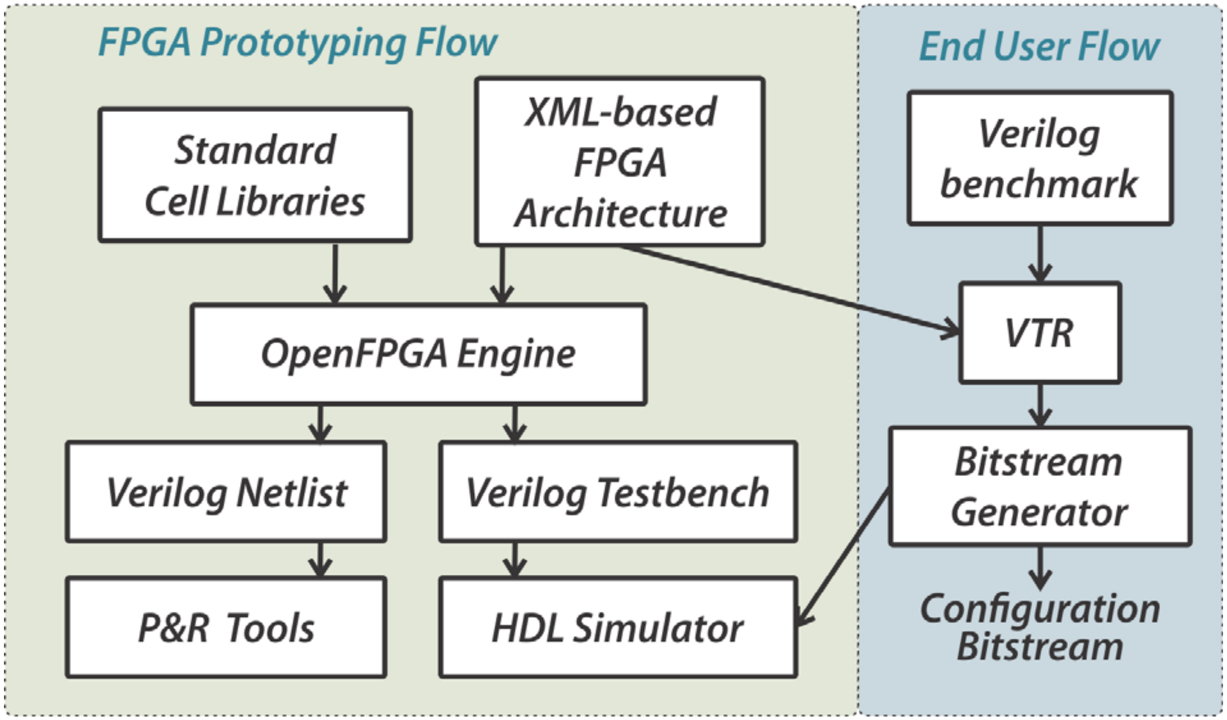
Modern FPGA architecture

- Island-style tillable homogeneous FPGA architecture
- This can be easily extended to heterogeneous Architecture

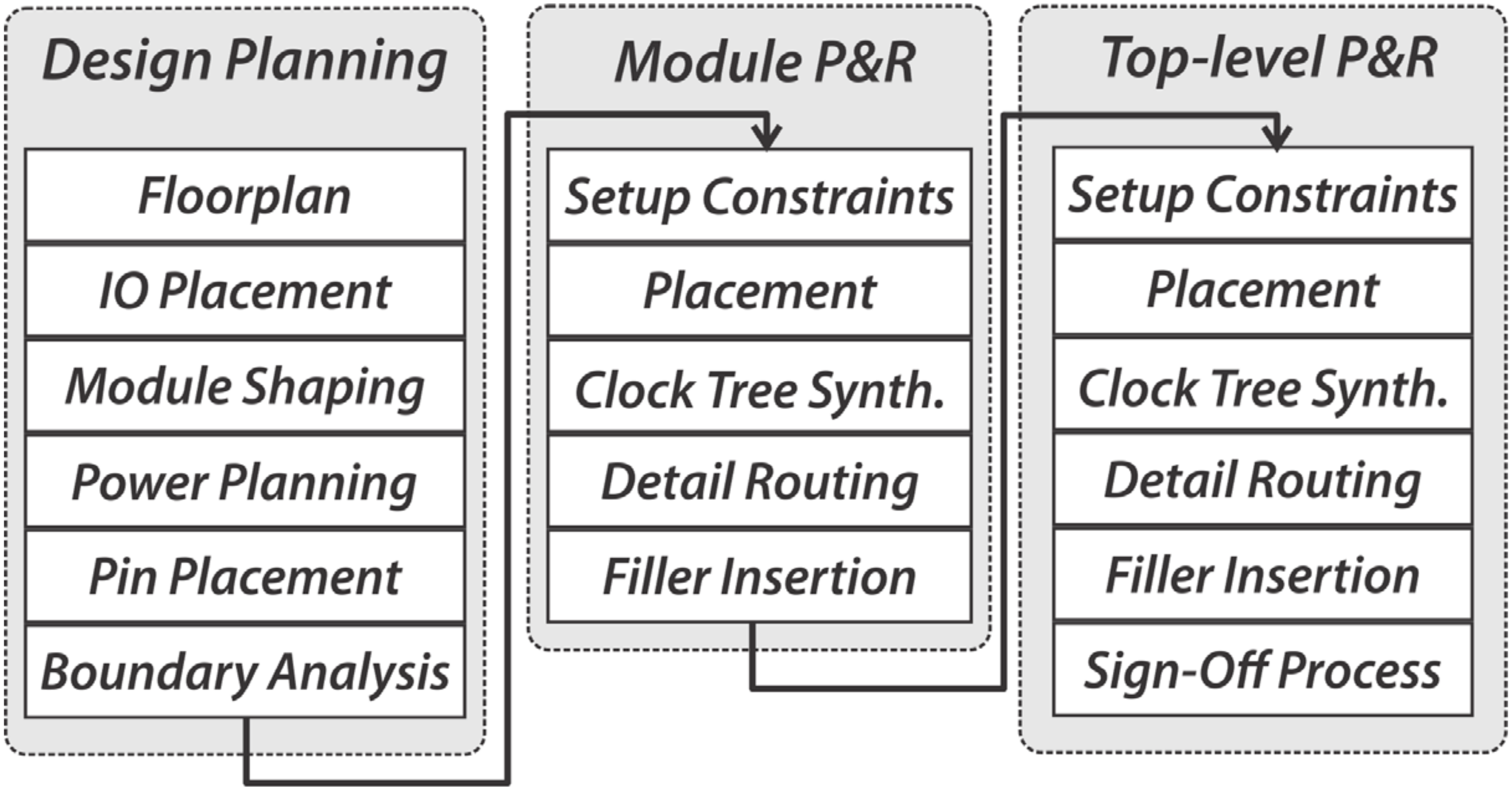


A OpenFPGA Framework

- The fabric is generated using ***OpenFPGA*** which is based on the popular ***VPR tool***, to enable our physical design flow.
- OpenFPGA generates a ***tileable Verilog netlist***. The tool can auto-generate a modular Verilog netlist along with the set of random and formal testbenches, which can be used for functional verification of Pre-P&R or Post-P&R netlist.



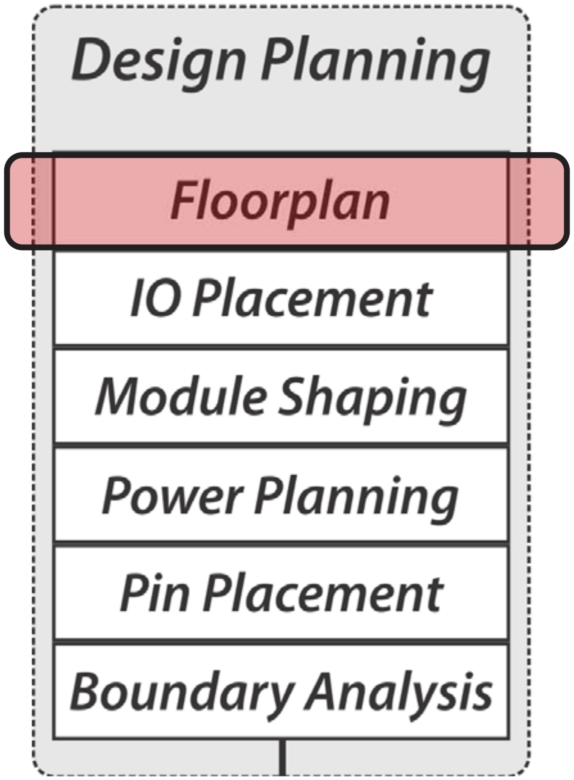
Standard hierarchical design Flow

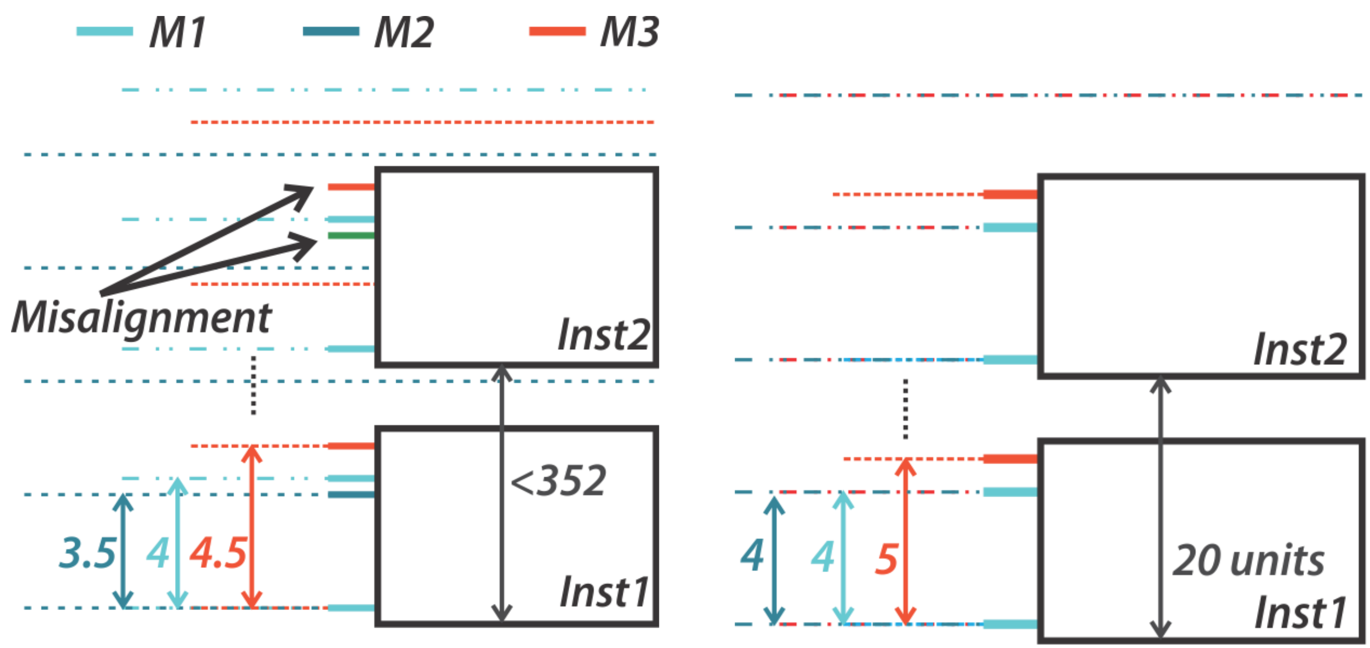


- The multilevel hierarchical flow is well supported in modern EDA toolchains
- The unplanned design may introduce multiple timing and design rule violations while performing top-level integration, resulting in multiple iterations for design closure

Floorplanning

- Adjusting routing metal track to allow multi-instantiate block design





Misaligned Track

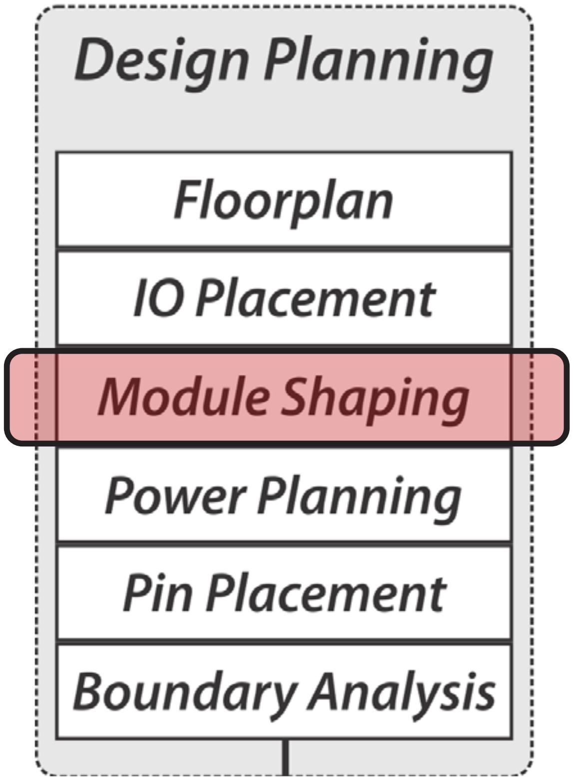
Track alignment at every 352 units

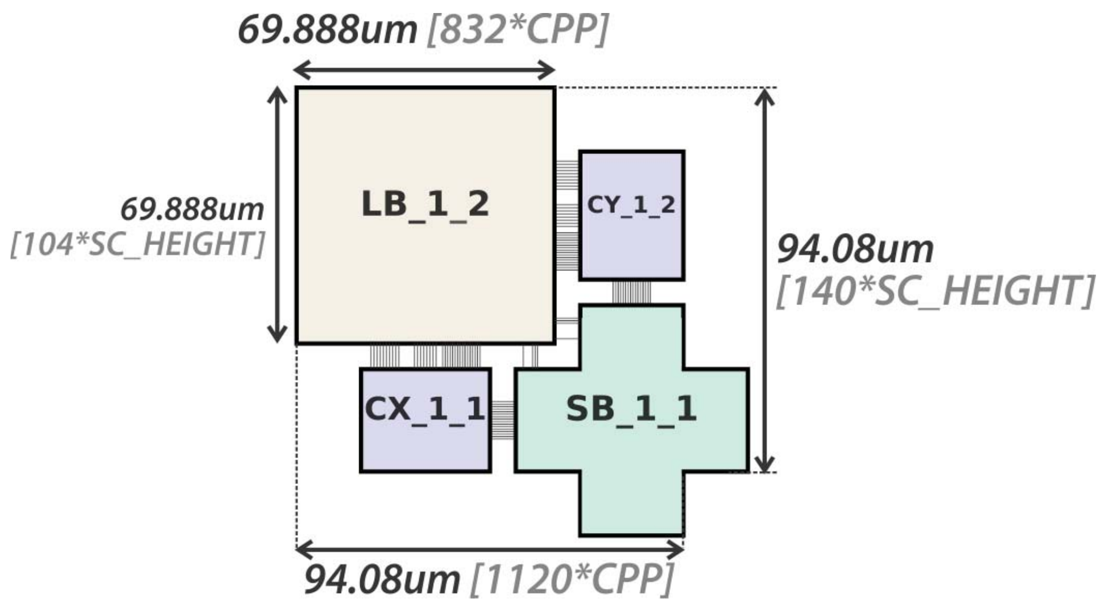
Aligned tracks

Track alignment at every 20 units

Block Shaping and Placement

- Utilization based and aspect ration-based shaping
- Narrow channel floorplan
- Parametrized script-based placement flow

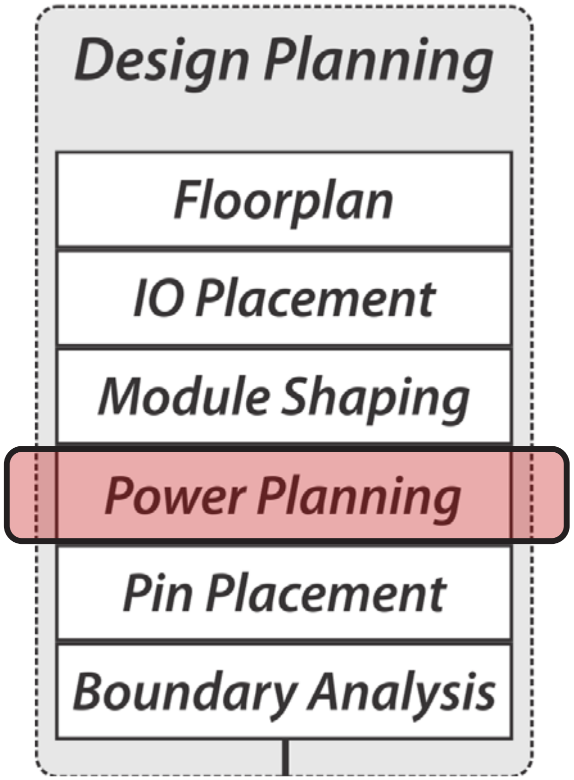


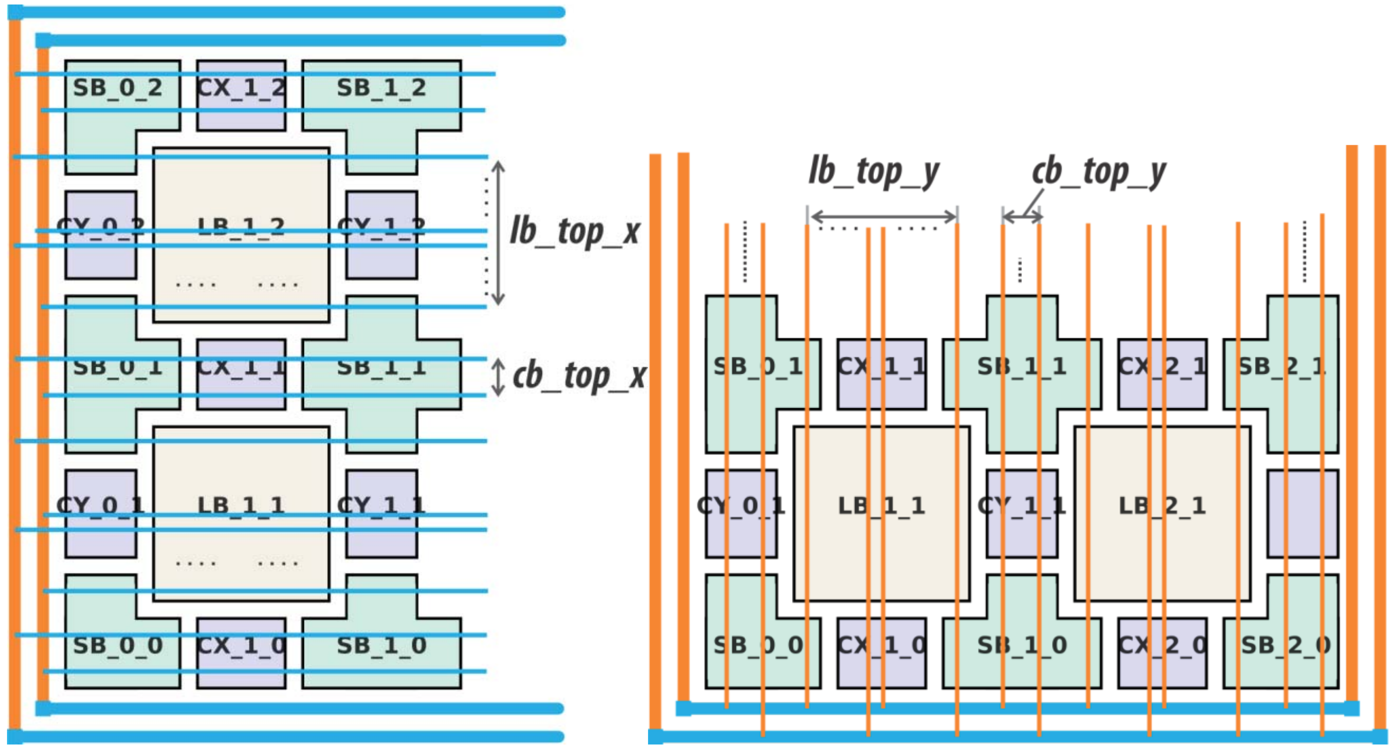


Shaping and Placement with narrow channel

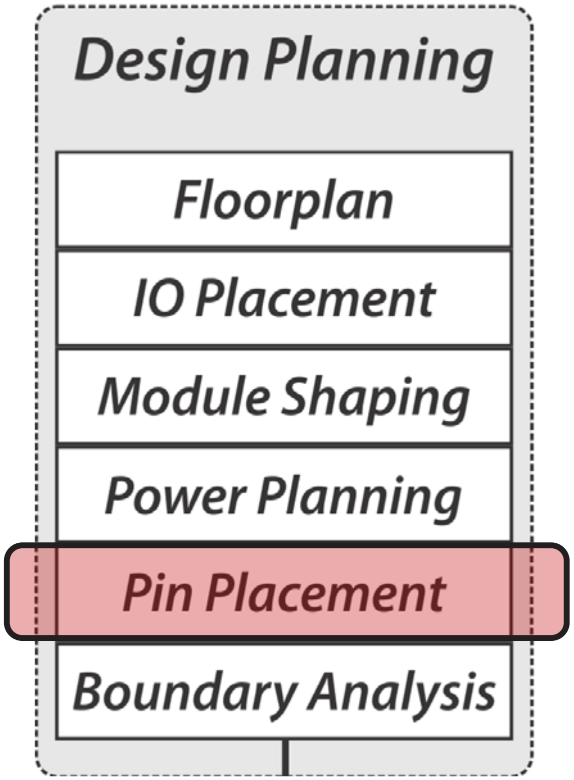
Highly regular power Grid

- FPGA grid-based power planning
- Highly repeatable across height and width of the fabric

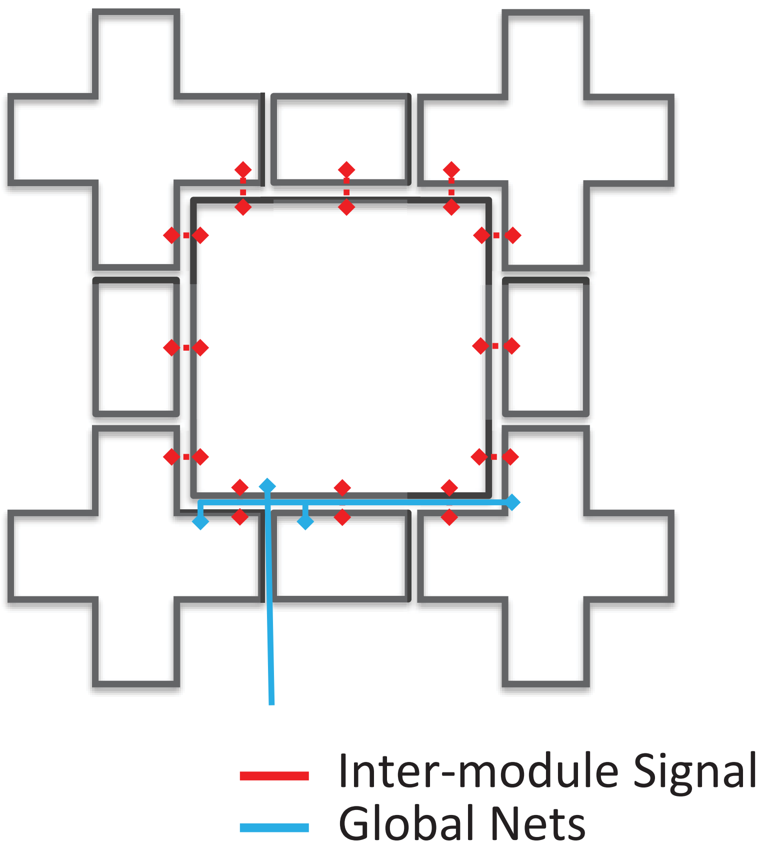




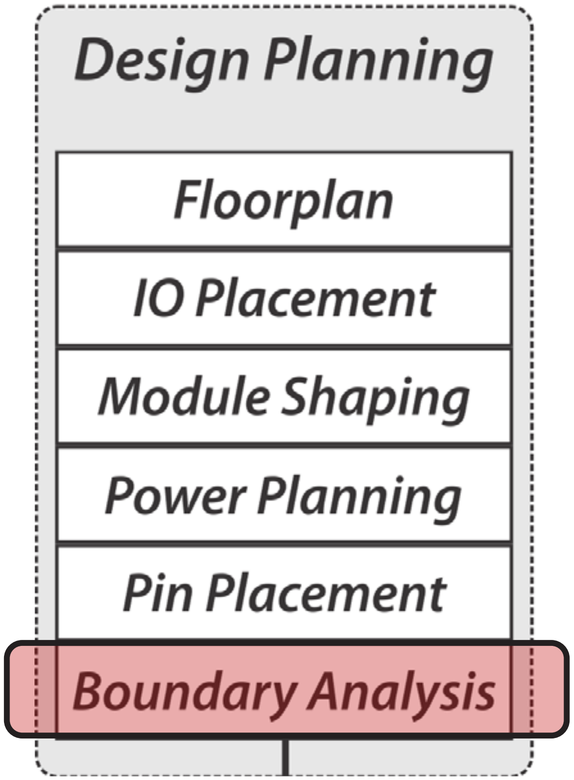
Pin placement



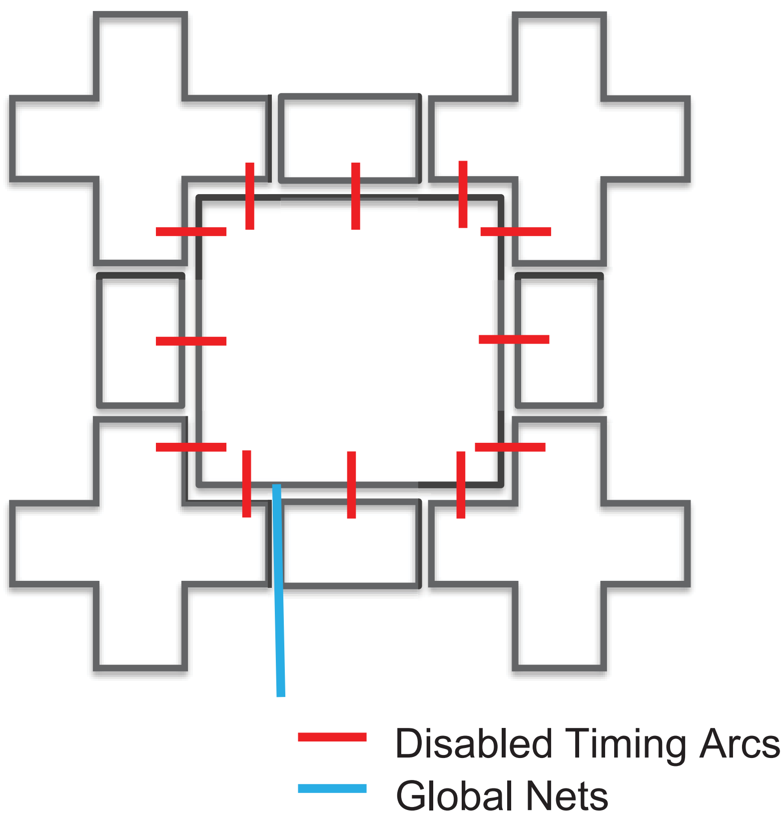
- Strict aligned pin placement
- Strict same metal layer pin placement on neighboring blocks
- Auto feed trough creation for the global signal



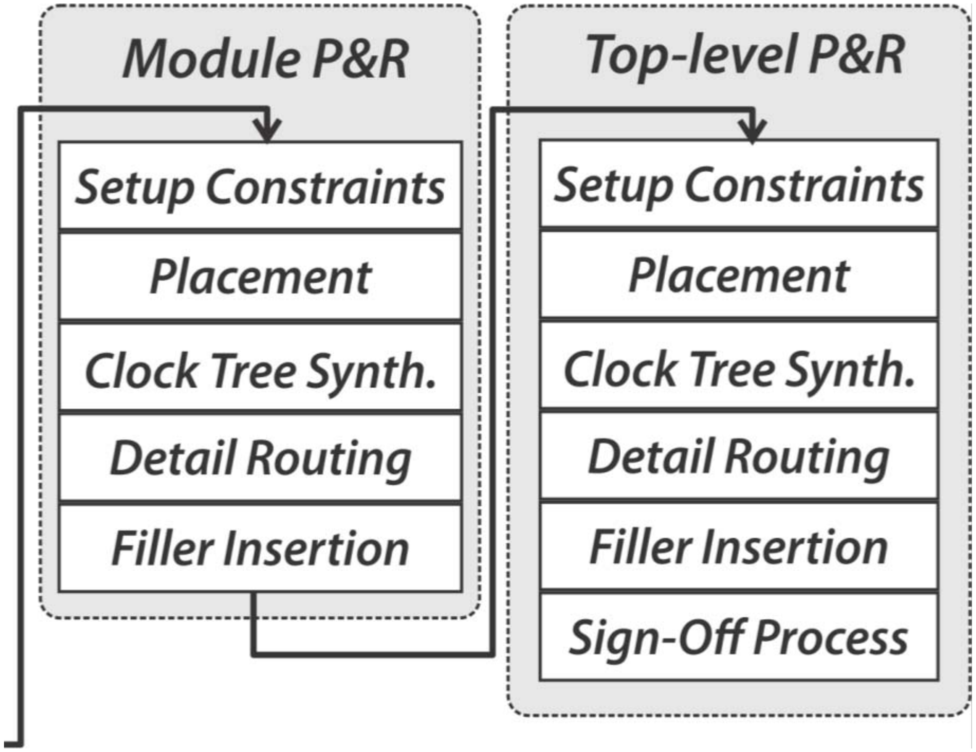
Boundary analysis



- All the combinational loops are terminated at every input of the switch box MUX
- The ***driving cell selection for input***, ***Cap load for the output*** *and* ***Input/Output Delays*** of the signal is derived

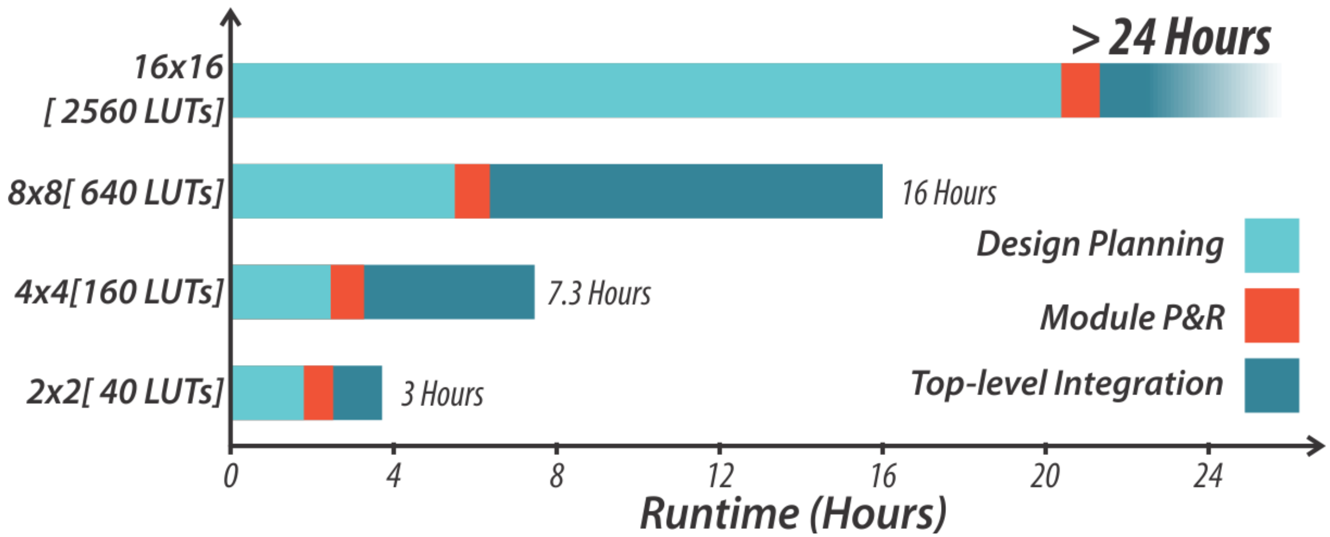


Module and Top-level P&R



- At the end of the design planning flow individual modules are export in the DEF and LEF format
- A standard P&R flow is followed for each individual modules followed by the integration on the top level
- Global signal like reset, clock and enable signals are optimized during placement and, clock tree synthesis

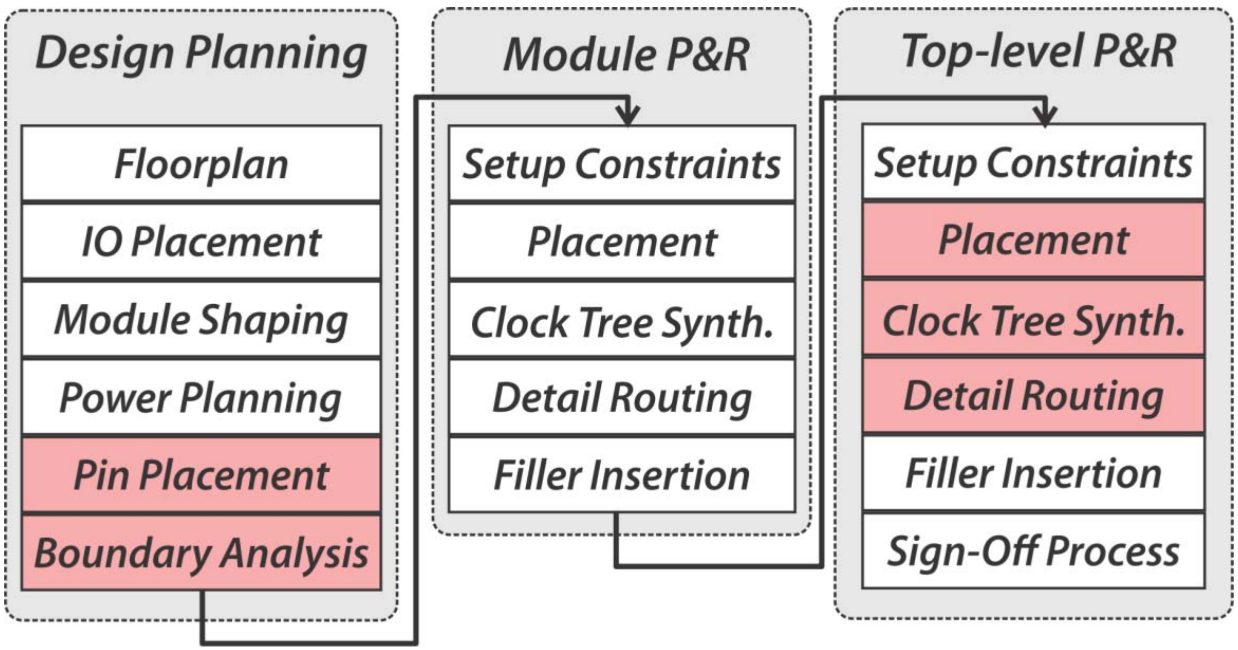
Runtime-Standard Hierarchical-flow



Std. hierarchical flow runtime Faster prototyping but do not to scale to larger fabrics

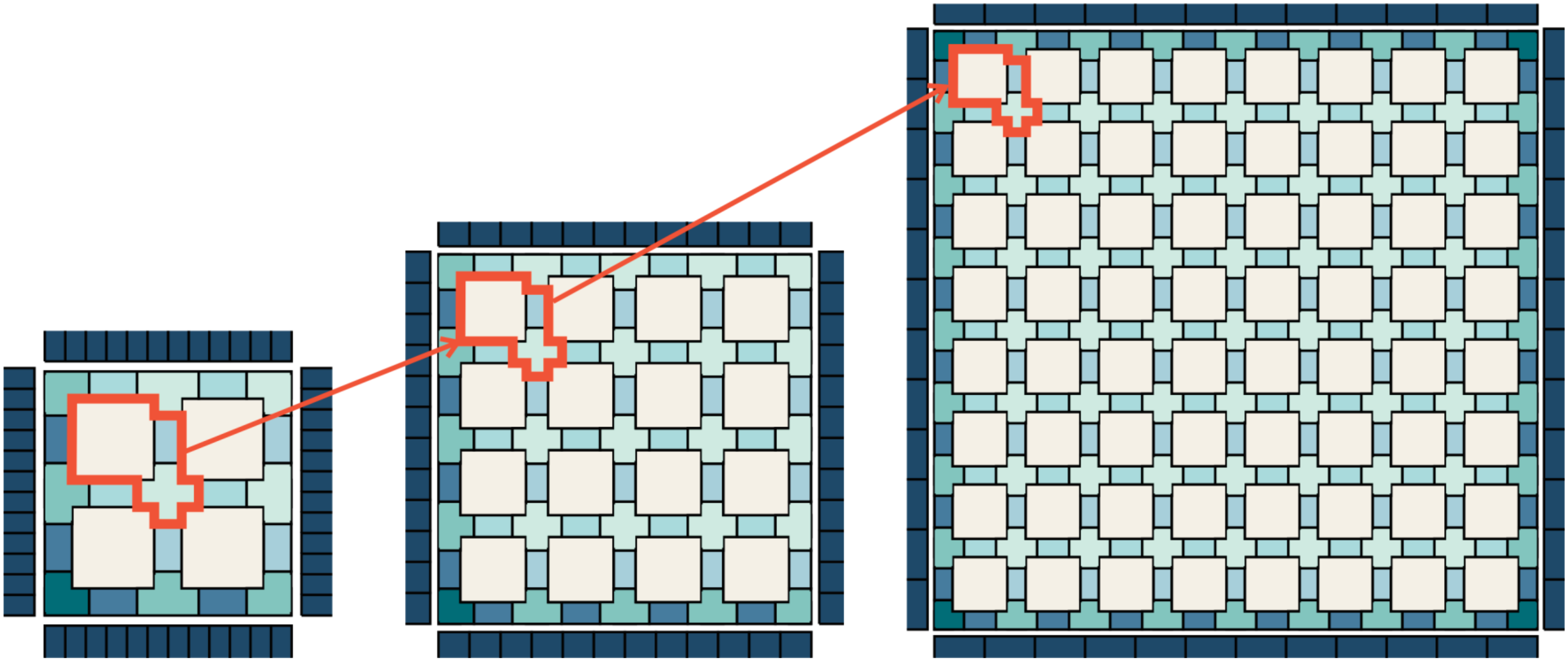
Runtime Bottlenecks

1. Number of instances increases quadratically with FPGA grid size
2. During top level P&R, slowdown is a result of global optimization step (highlighted in the figure)



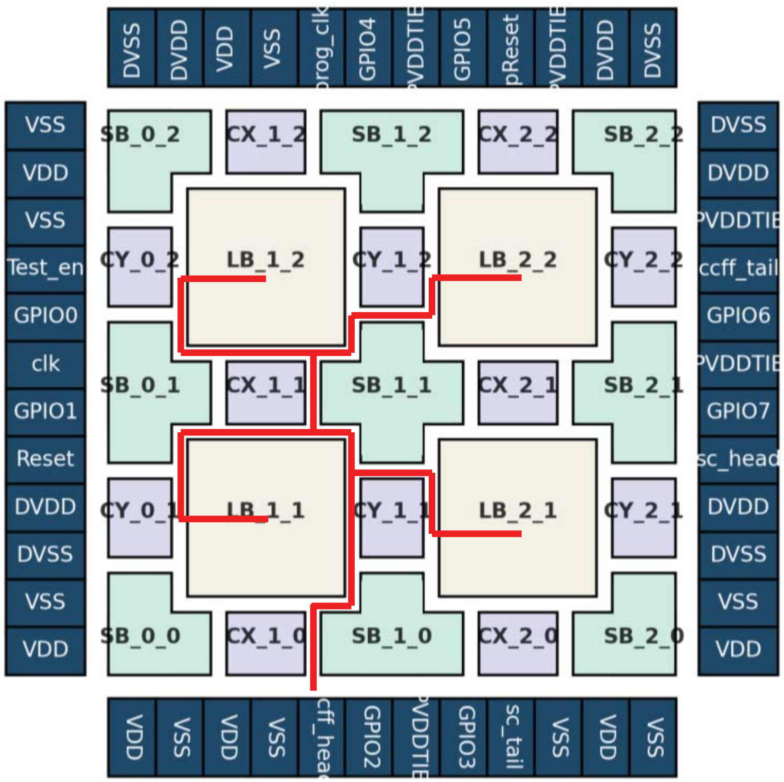
Scaling design planning

1. Scaled fabric uses same number of Unique blocks and structure as the smaller (FPGA22) design
2. Considered FPGA22 and FPGA44 as a model design to for design planning and reuse modules for bigger design

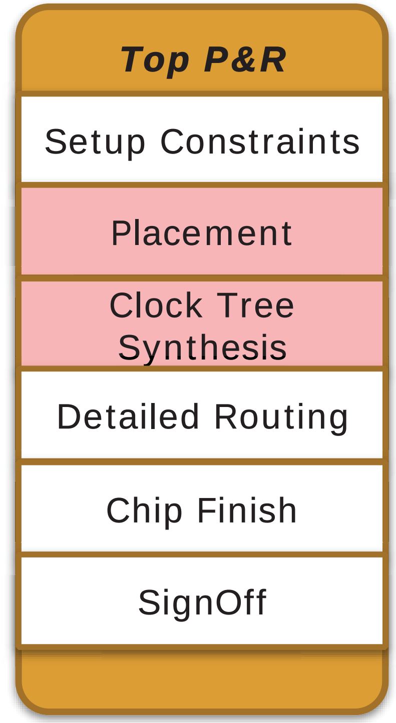


Top Level P&R

- **PlaceOpt** takes time to resolve high fanout nets, like ***pReset***, ***Reset*** and ***TestEn***
- **ClockOpt** takes time to place/route clock buffers in narrow channels between the blocks



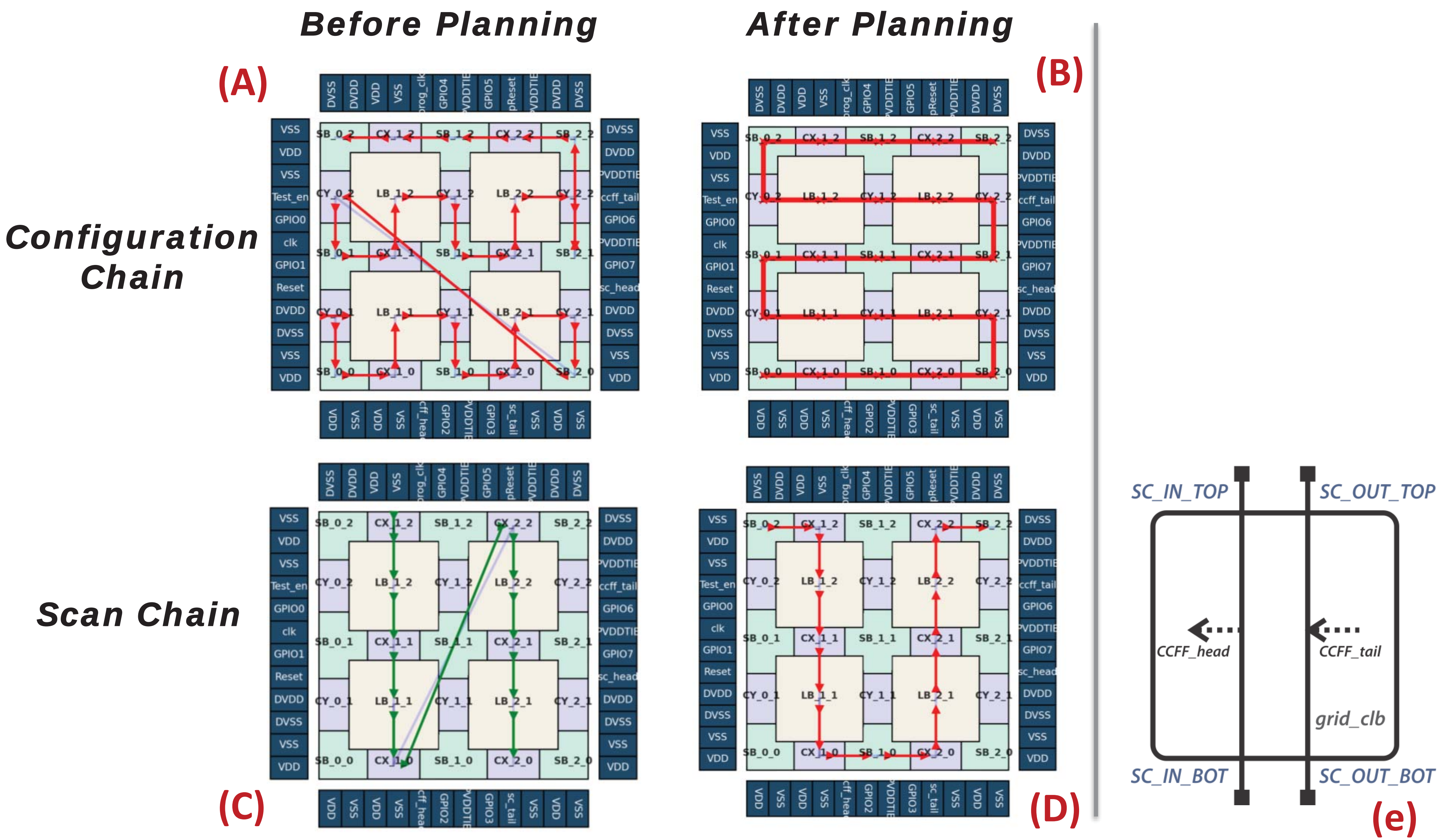
Routing through narrow channels



Solution

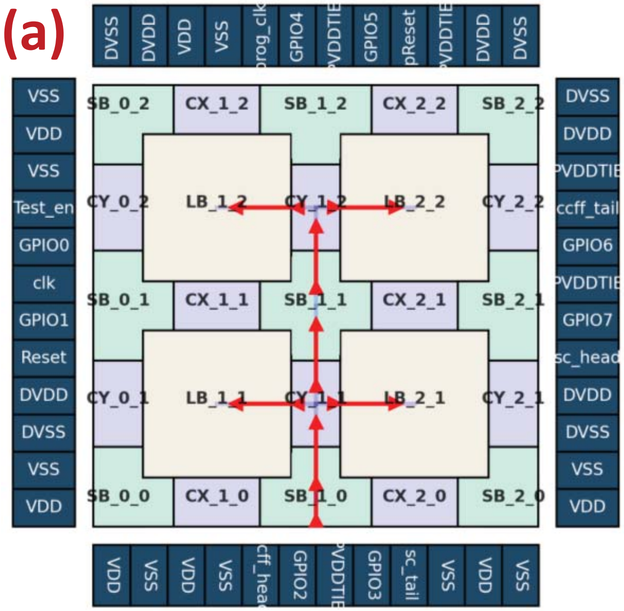
1. Pre route clock and top level high fanout signals with feedthroughs
2. *ScanChain, pReset, Reset, TestEn, clock, pClock*

Scan-chain rerouting



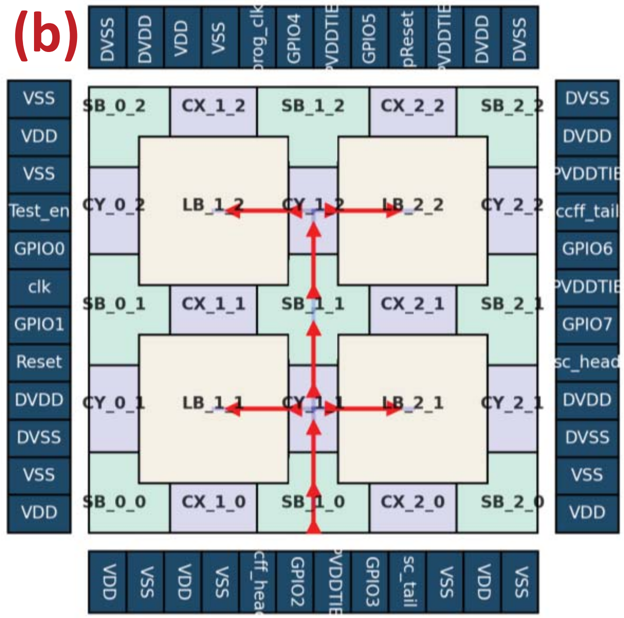
Global Signals

(a)



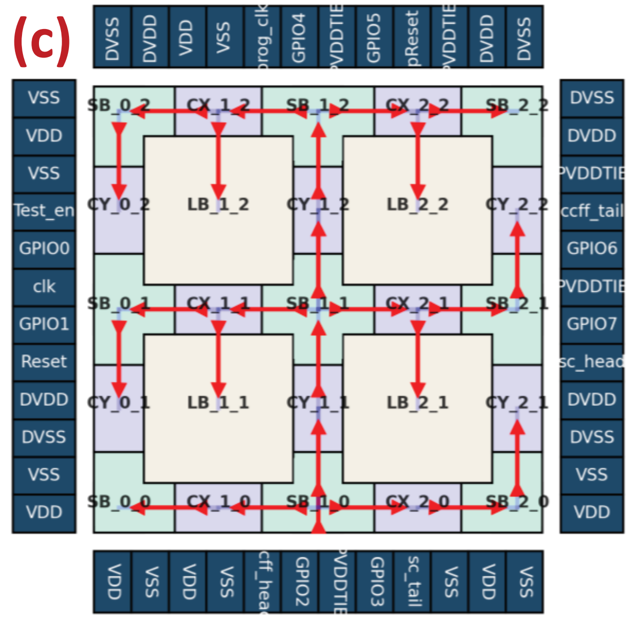
Reset Signal

(b)

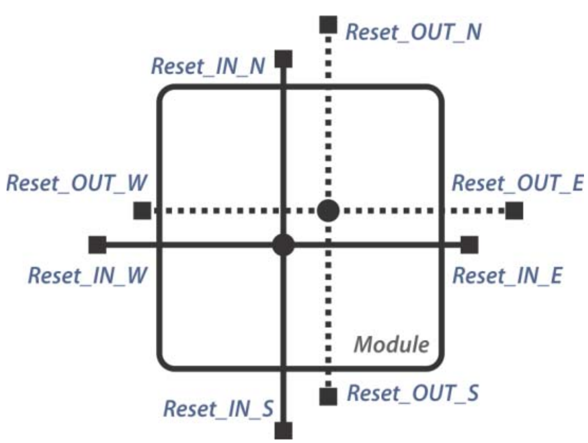


Test_en Signal

(c)



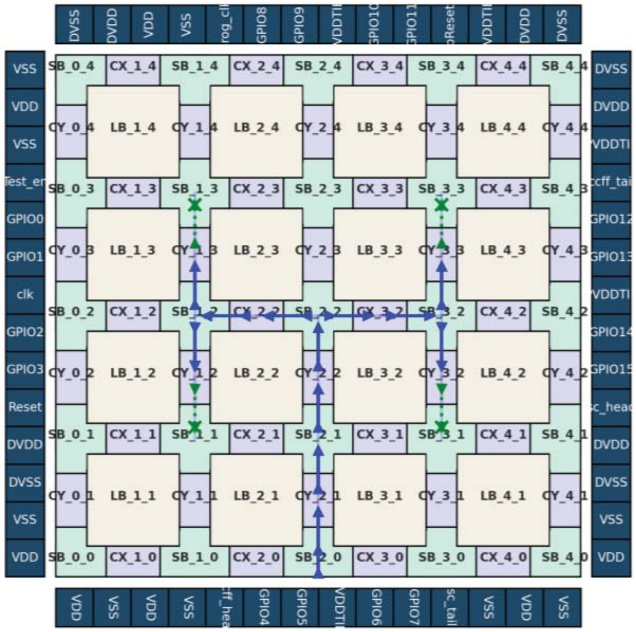
pReset Signal



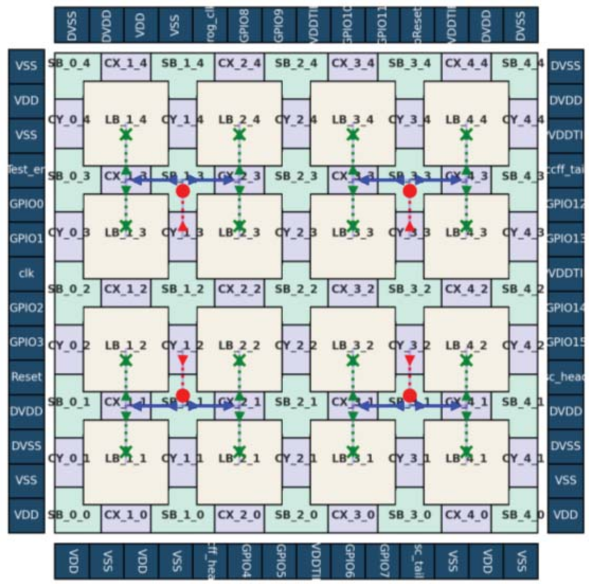
(D) Feedthrough ports

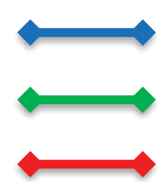
Clock Routing

Level 2

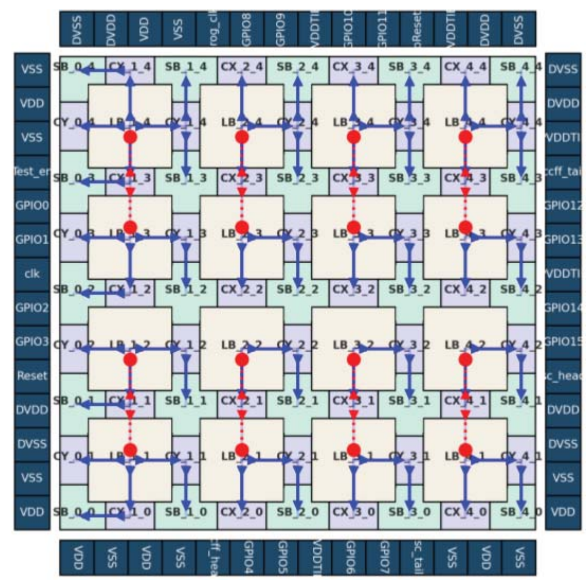


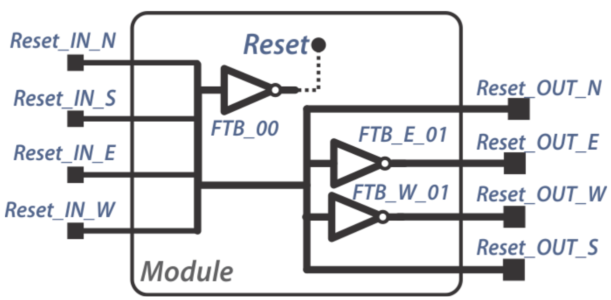
Level 1





Level 0





Buffered Feedthrough

Restructuring with connect file

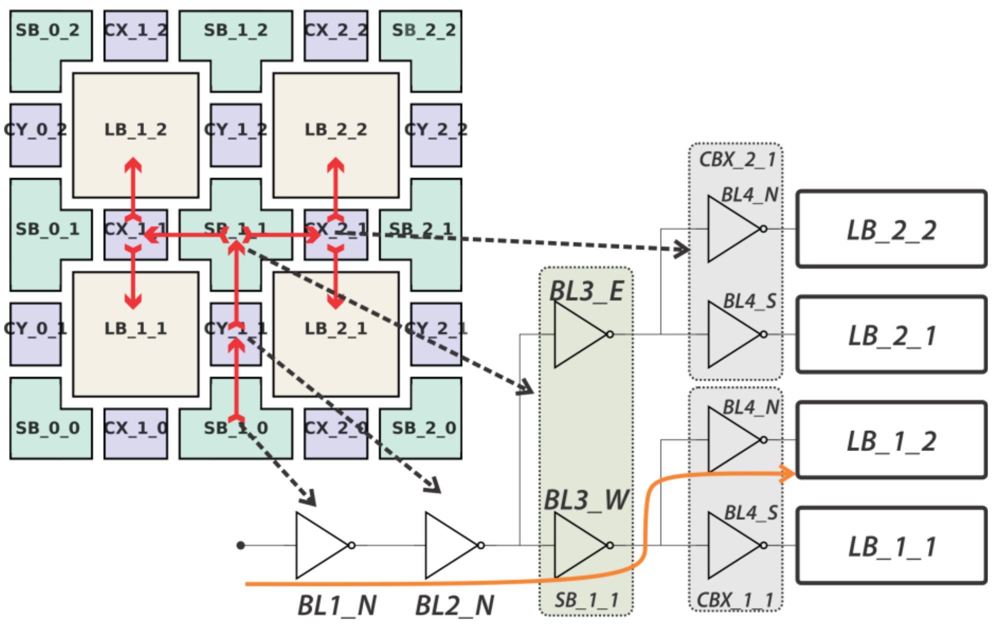
- Feedthrough generation and pre-routing is achieved by automated connect file
- Connect file contains connections information between each module and direction of connections

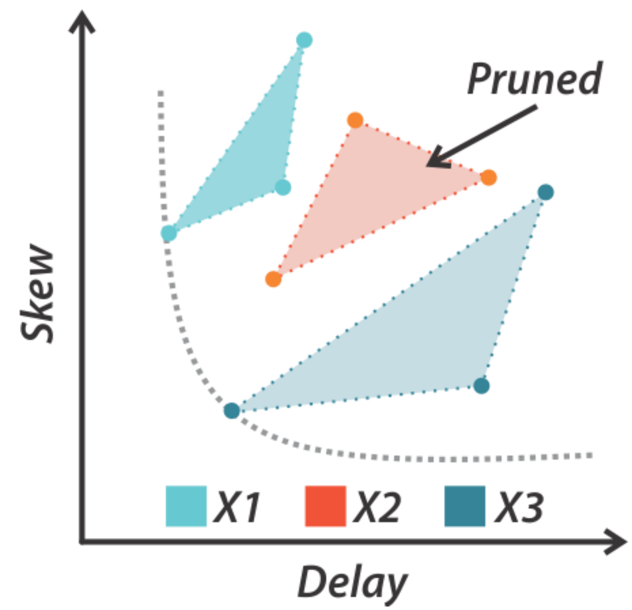
```
# SrcInstance ,DestInstance   ,Direction
cbx_1__1_     ,grid_clb_1__1_ , N
cbx_1__1_     ,grid_clb_1__2_ , S
cbx_1__3_     ,grid_clb_1__3_ , N
cbx_1__3_     ,grid_clb_1__4_ , S
....
...
..
.
```

Clock tree buffer sizing

- Optimal buffer sizing to meet the latency criteria without breaking regularity of the fabric
- Modified version of Van-Ginneken buffer insertion algorithm with a solution pruning policy

BL1_N	BL2_N	Delay	Slew
X1	X1	581.7	30.78
X1	X2	623.5	32.83
X1	X3	633.5	34.99
X2	X1	525.6	27.57
X2	X2	540.3	29.84
X2	X3	560.1	31.28
X3	X1	510.2	26.18
X3	X2	514.3	27.56
X3	X3	520.2	29.45





Experimental Results

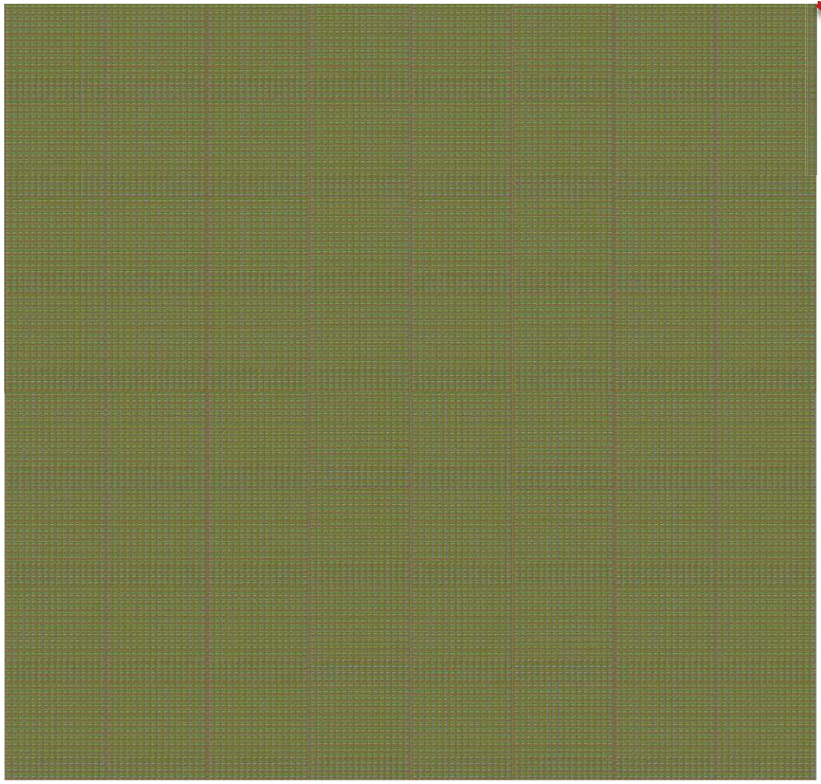
Experiment setup

- ***Stratix IV*** resembling architecture
- ***200 routing tracks/channel*** *(L1=20%, L2=20%, L3=60%)*
- ***Multi-configuration chain*** protocol to configure the FPGA
- ***Scan-chain*** for functional test
- ***OpenFPGA framework*** to generate the technology mapped Verilog netlists
- FPGA grid sized swept from ***2x2*** (40 LUTs) to ***128x128*** (164K LUTs)
- ***12nm GlobalFoundry*** tech-node

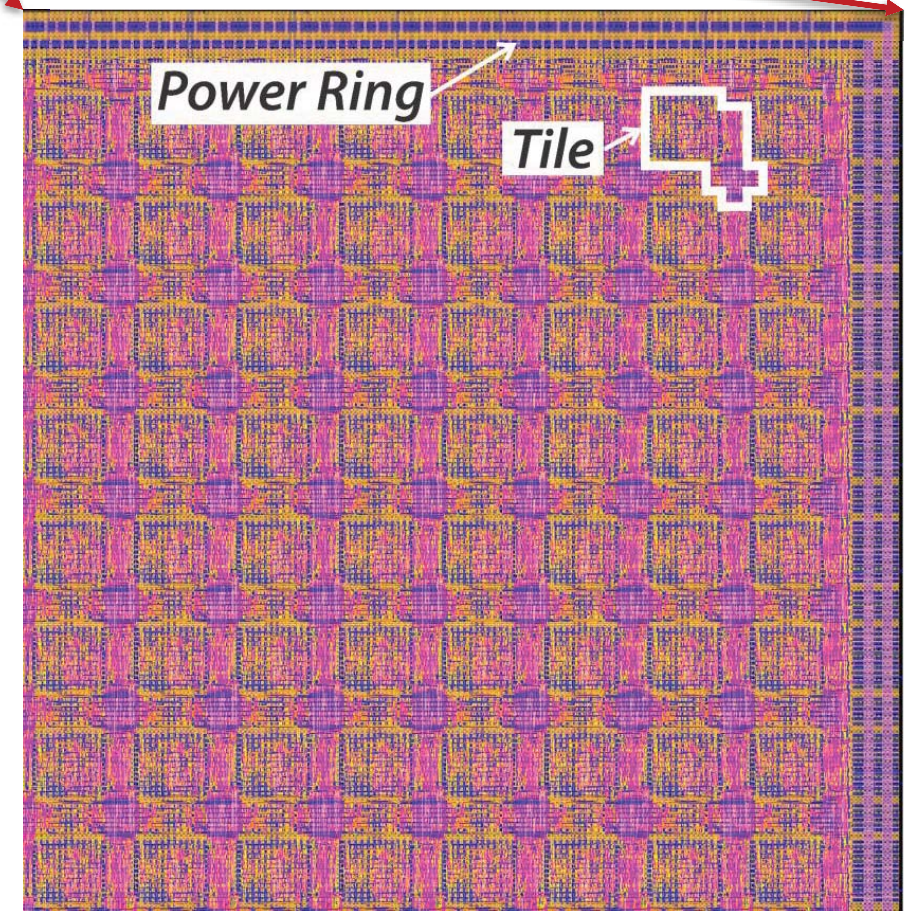
Table 1: Architecture parameters

Architecture Parameters	Value
LUT Inputs	6
LE/Logic Blocks	10
Crossbar connectivity	50%
$F_{c_{in}}$	0.055
$F_{c_{out}}$	0.1

128x128 fabric GDS view



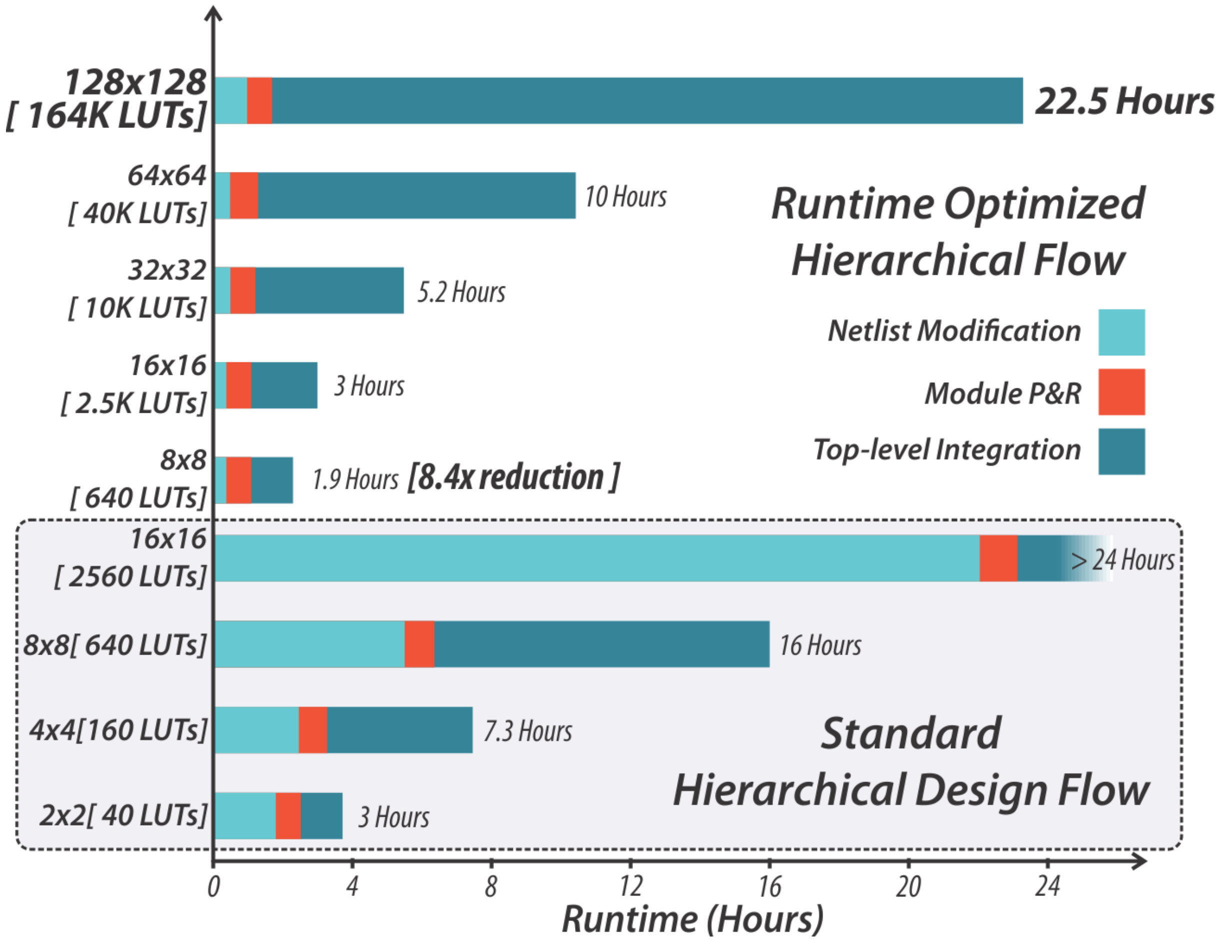
GDS view of 128x128 (164k LUTs) Device



Top right 8x8 grid of 128x128 GDS view

Runtime Improvement

- Over 8.4x runtime improvement for small fabrics (8x8 with 640 LUTs)
- 100K+ LUTs design achievable in <24hours



Clock skew and latency

- 4x improvement in clock skew
- 15% increase in the clock latency

Table 2: Clock network Latency and Skew Comparison

Design	Standard Hier. (ρs)		Opt. Hier. (ρs)		length
	Latency	Skew	Latency	Skew	(μm)
2×2	122.52	30.6	134	10.11	95.08
4×4	252.56	50.4	304	12.45	380.32
8×8	380.4	70.4	437	15.4	950.8
16×16	-	-	2658	19.87	2091.76
32×32	-	-	4566	32.2	4373.68
64×64	-	-	8954	50.5	8937.52
128×128	-	-	21012	70.56	18065.2

Conclusion

- We presented a scalable and robust hierarchical floorplanning technique to design a 100k LUTs FPGA and demonstrated the design runtime of fewer than 24-hours is achievable.
- We leveraged the standard ASIC toolchain hierarchical flow to perform Floorplanning on small-size FPGA architecture and reuse information to perform larger FPGA floorplanning.
- To optimize the runtime, we perform feedthrough creation to bypass the need for any time-consuming top-level optimization, i.e., clock tree synthesis.
- The proposed post-P&R sizing strategy allows latency optimization of pre-planned global signals.

The achieved 24-hour prototyping allows the rapid development of FPGA fabric and enables comprehensive architecture exploration by considering the effect of physical design on architecture selection.

Thank You

For any further questions please contact:

ganesh.gore@utah.edu