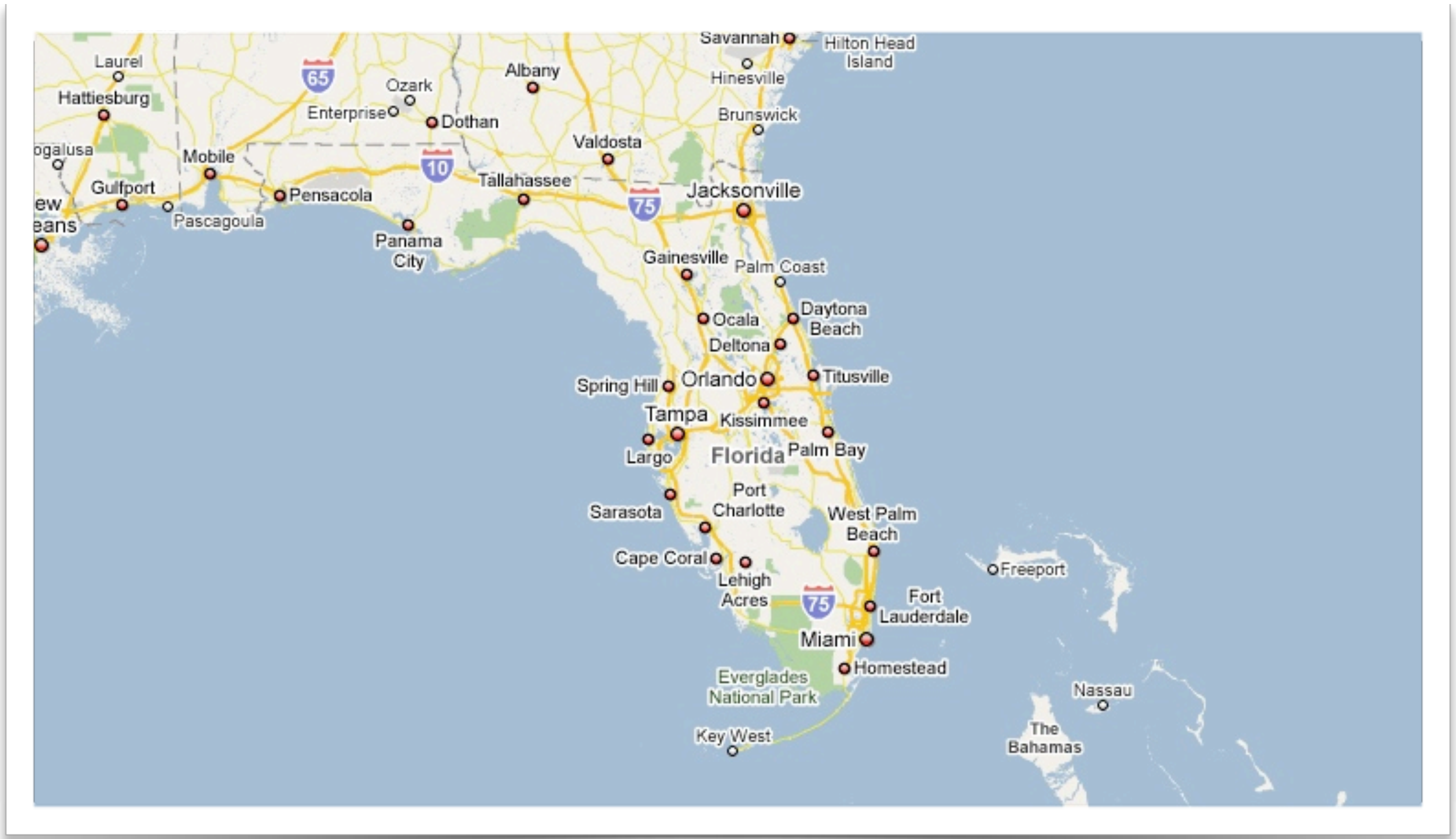
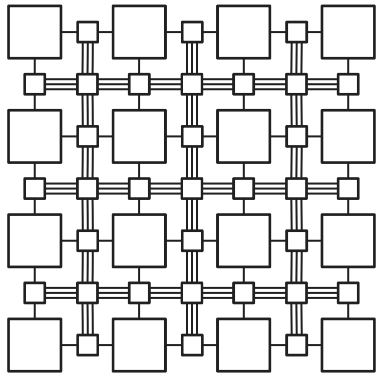
F2-09: Intermediate Fabrics

Presented by: James Coole
April 13, 2009


Savannah
Hilton Head Island
Laurel
Hattiesburg
Ozark
Albany
Hinesville
Enterprise
Dothan
Brunswick
ogalusa
Mobile
Valdosta
Tallahassee
Jacksonville
Gulfport
Pensacola
ew
eans
Pascagoula
Panama City
Gainesville
Palm Coast
Ocala
Daytona Beach
Deltona
Spring Hill
Orlando
Titusville
Tampa
Kissimmee
Largo
Florida
Palm Bay
Port Charlotte
Sarasota
West Palm Beach
Cape Coral
Lehigh Acres
Freeport
Fort Lauderdale
Miami
Homestead
Everglades National Park
Nassau
The Bahamas
Key West
65
10
75

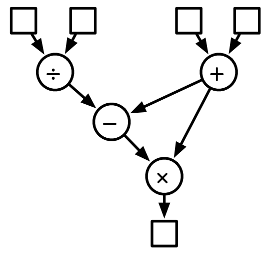
Overview

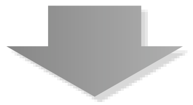
Placement and routing (PAR) is slow on existing fine-grained FPGAs

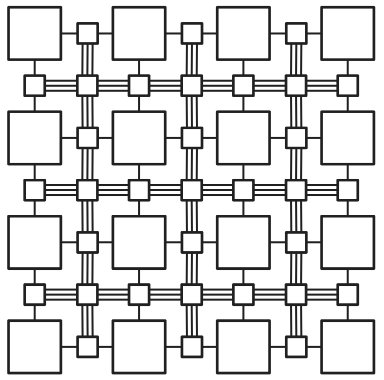

Intermediate Fabric

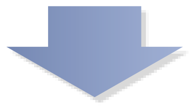
Implement a coarse-grain reconfigurable fabric on top of the FPGA

- device-independent VHDL
- implemented on the device using existing tools (ex. Quartus, ISE, etc.)

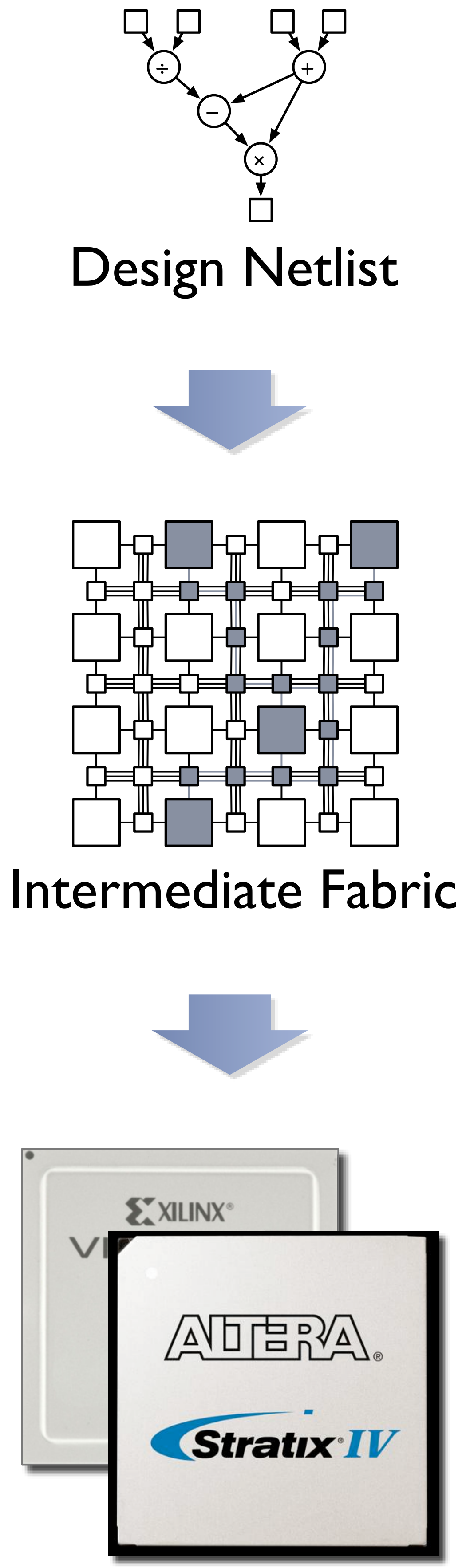

Design Netlist




Intermediate Fabric

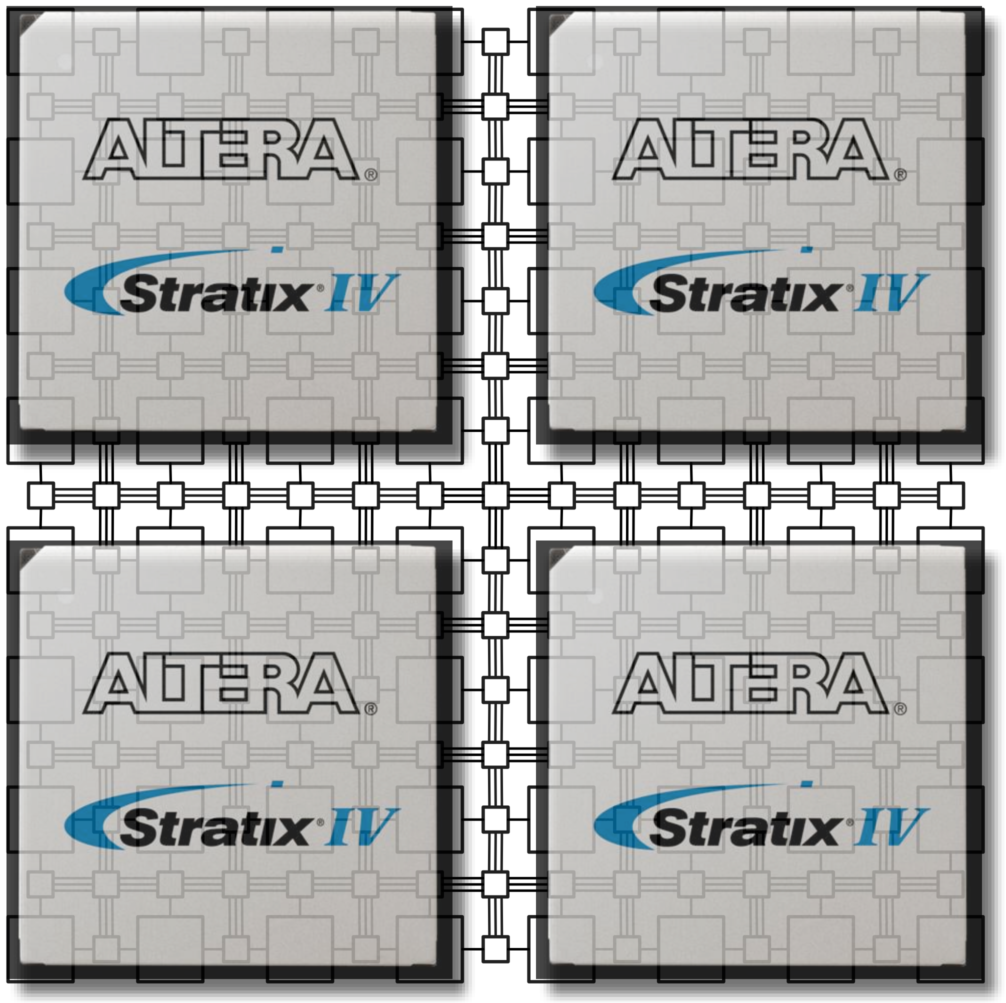


User design implemented on top of coarse-grain (intermediate) fabric

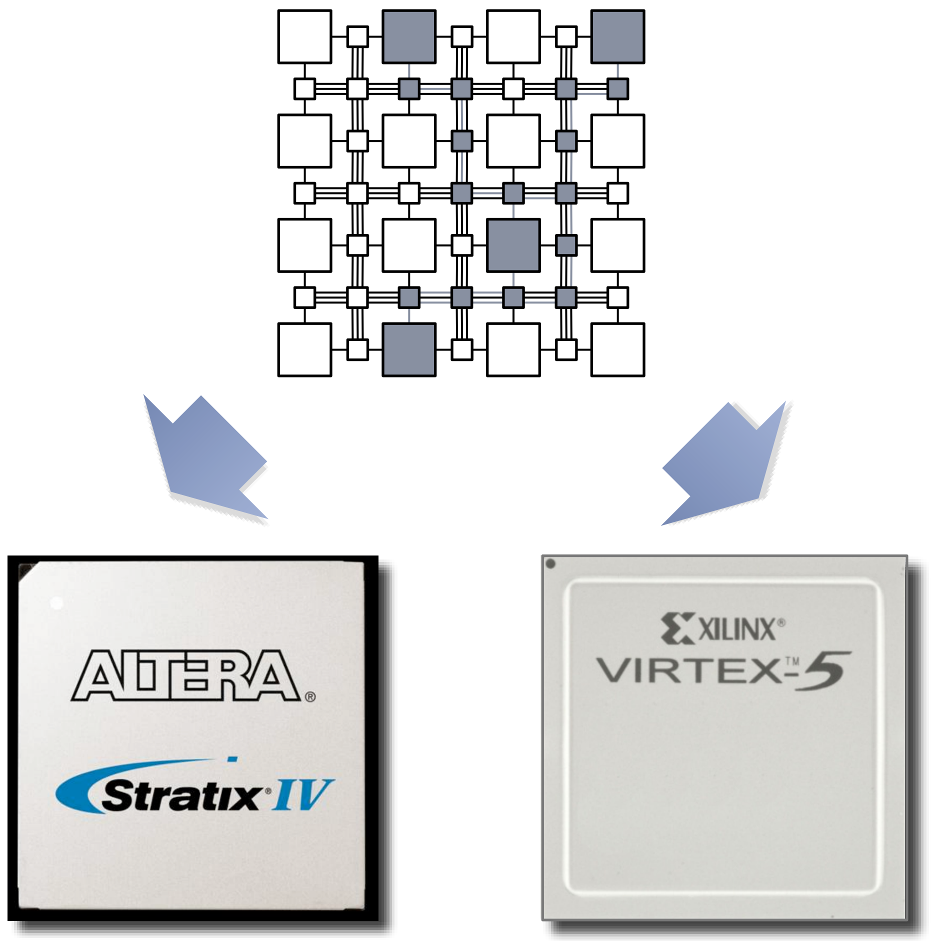


Custom PAR to Intermediate Fabric (IF) is considerably faster

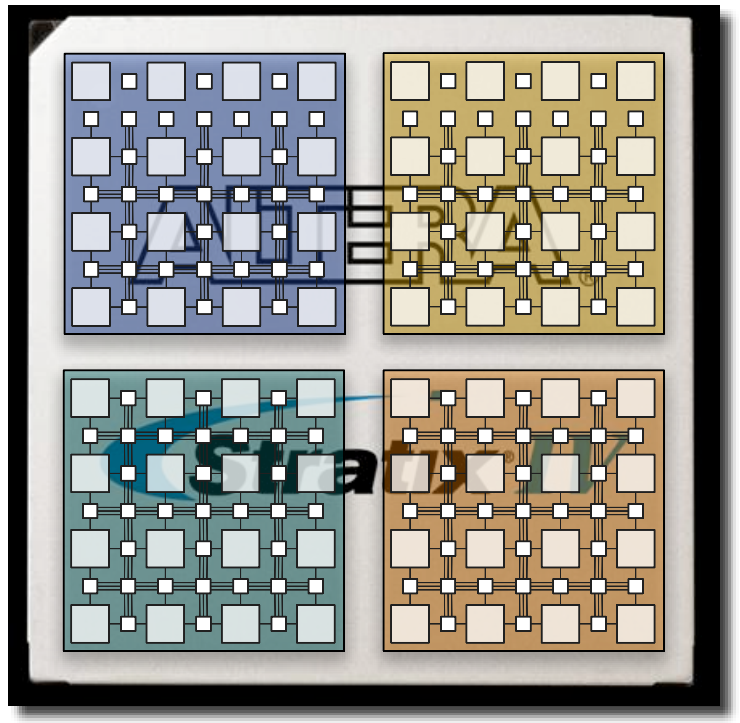
- coarse-grained means smaller solution space for PAR and prevents netlist size explosion after mapping
- early results suggest as much as 600x speedup compared to ISE



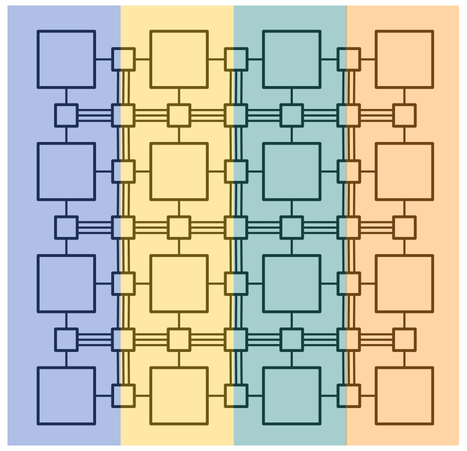
Abstract multiple devices as a single IF



Abstract different devices as the same IF

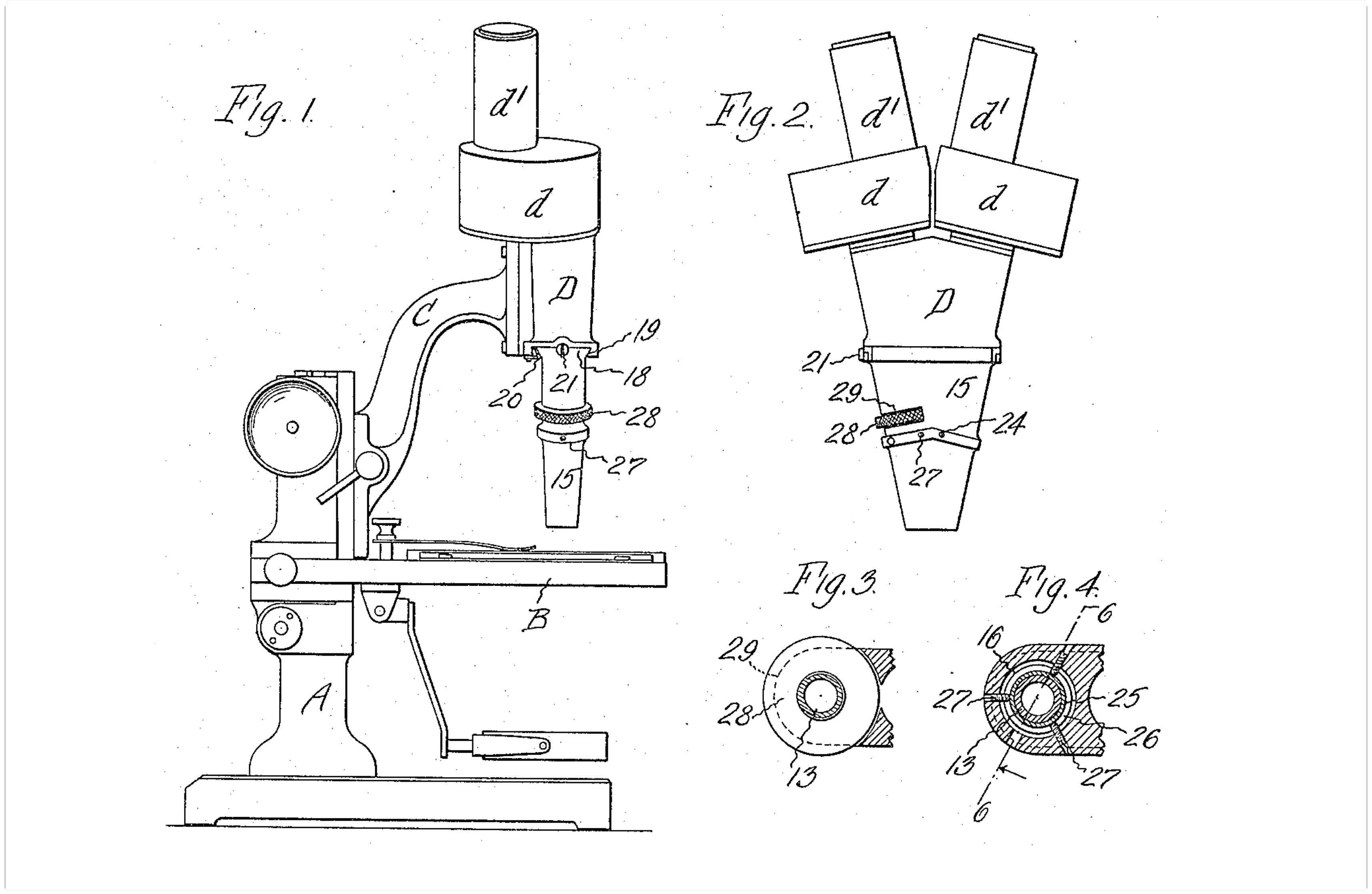


by hosting multiple
IFs on the device



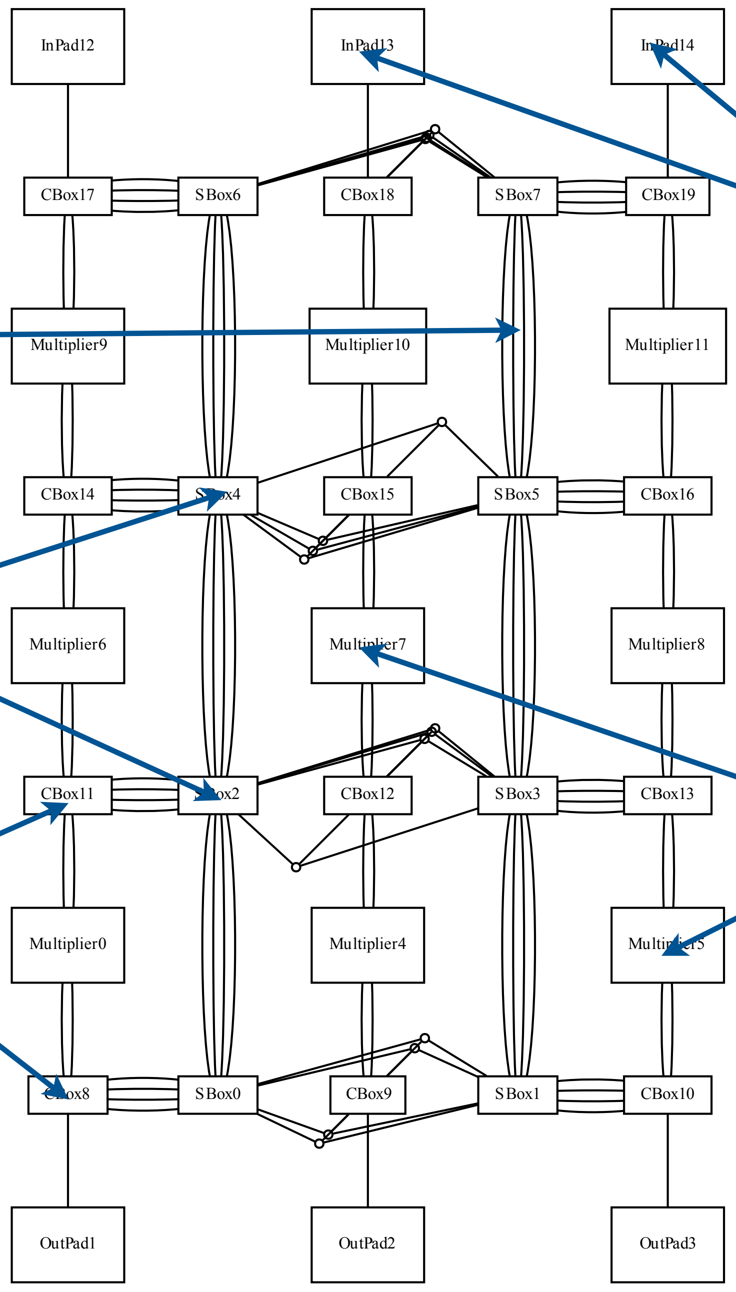
by using an IF with multiple
configuration chains

Enable partial reconfiguration without device support



Implementation

Fabric Structure



pads: signals in/out of the fabric

tracks: n-bit busses

switch boxes: connects tracks in one channel to tracks in another

nodes: adders, multipliers, etc.

connection boxes: connects nodes to channel tracks

currently exploring FPGA-like, island-style meshes

Configuration

Configuration is accomplished by setting the value of registers spread throughout fabric

- registers in nodes are node-specific but similar to FPGA nodes (ex. LUTs and IO select in CLBs)
- registers in switch boxes and connection boxes are similar to their FPGA counterparts, for the same topology

Configuration at the level of Tracks

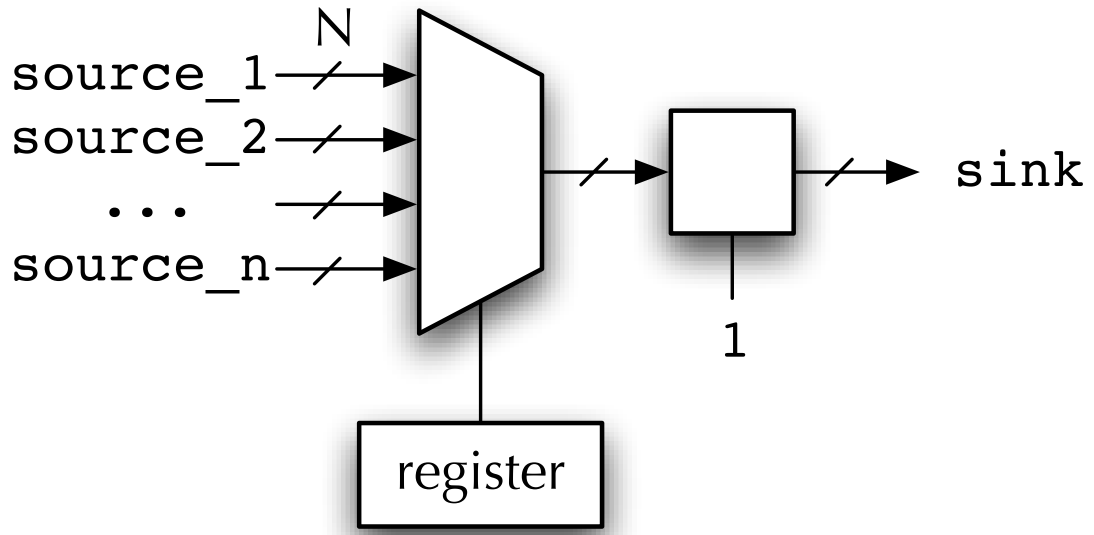
- unlike wires, signals can only have a single source, so...
- multiple sources MUXed down to a single sink
- configuration registers drive MUXes

Like FPGAs, configuration registers are chained together and configured by shifting in a bitstream

Tracks

Combinatorial loops exist for some MUX select values

- ISE, Quartus, etc. can't implement the fabric with the loops intact
- can't simulate with zero-delay loops (limited to slow post-PAR)
- currently, we break the loops by inserting latches before sink



Mapping, PAR

Mapping, placement, and routing problems are similar to same problems in FPGAs

Currently...

- Mapping – one-to-one
- Placement – VPR
- Routing – PathFinder

Cost

Area Overhead
reduction in resources available to user design

+

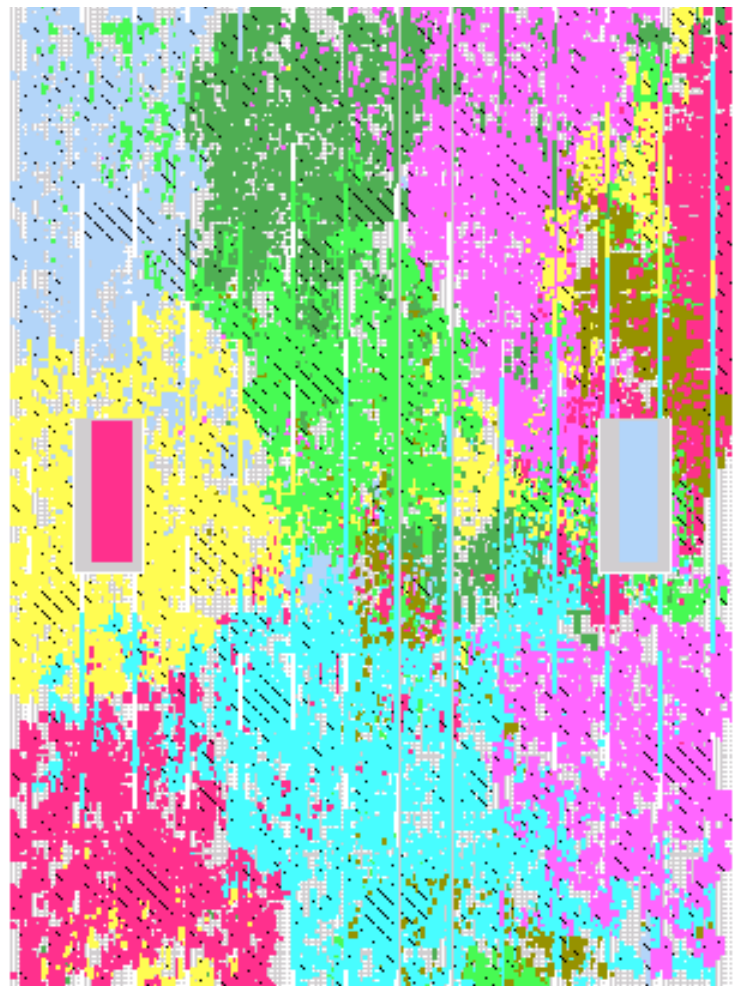
Clock Overhead
reduction in achievable clock rate of user design

Area Overhead

Cause: FPGA resources left unused by the IF

Make sure IFs provided are as big as possible for the device

- area can be maximized automatically by a fabric-generating tool



Area Overhead

Cause: Mismatch between mix of node types in IF and cell types in netlist

Provide a variety of application-specialized IFs so good matches are usually available

- hand-design a library based on identification and analysis of application domains (ex. DSP, block ciphers, bioinformatics, etc.)
- automatically generate based on analysis of design

Area Overhead

Cause: Resources used by logic in IF, but not necessary in direct implementation

- ➡ logic implementing routing resources
- reconfiguration logic

Don’t count resources that wouldn’t have been used by direct implementation anyway

Minimizing Cost

Minimizing Area Overhead

Study the area impact of properties of the fabric routing resources

- using a script to generate and map a bunch of fabrics, varying different properties to assess their impact on area

Study the impact of the same properties on routability

- need a method of assessing routability
  - Benchmarks (used to assess PAR algorithms) exist for traditional FPGA fabrics, but not for coarse-grain fabrics
  - Test on a set of randomly-generated netlists? (over a distribution of connectedness, heterogeneity, etc.)

...and strike a balance between area and routability

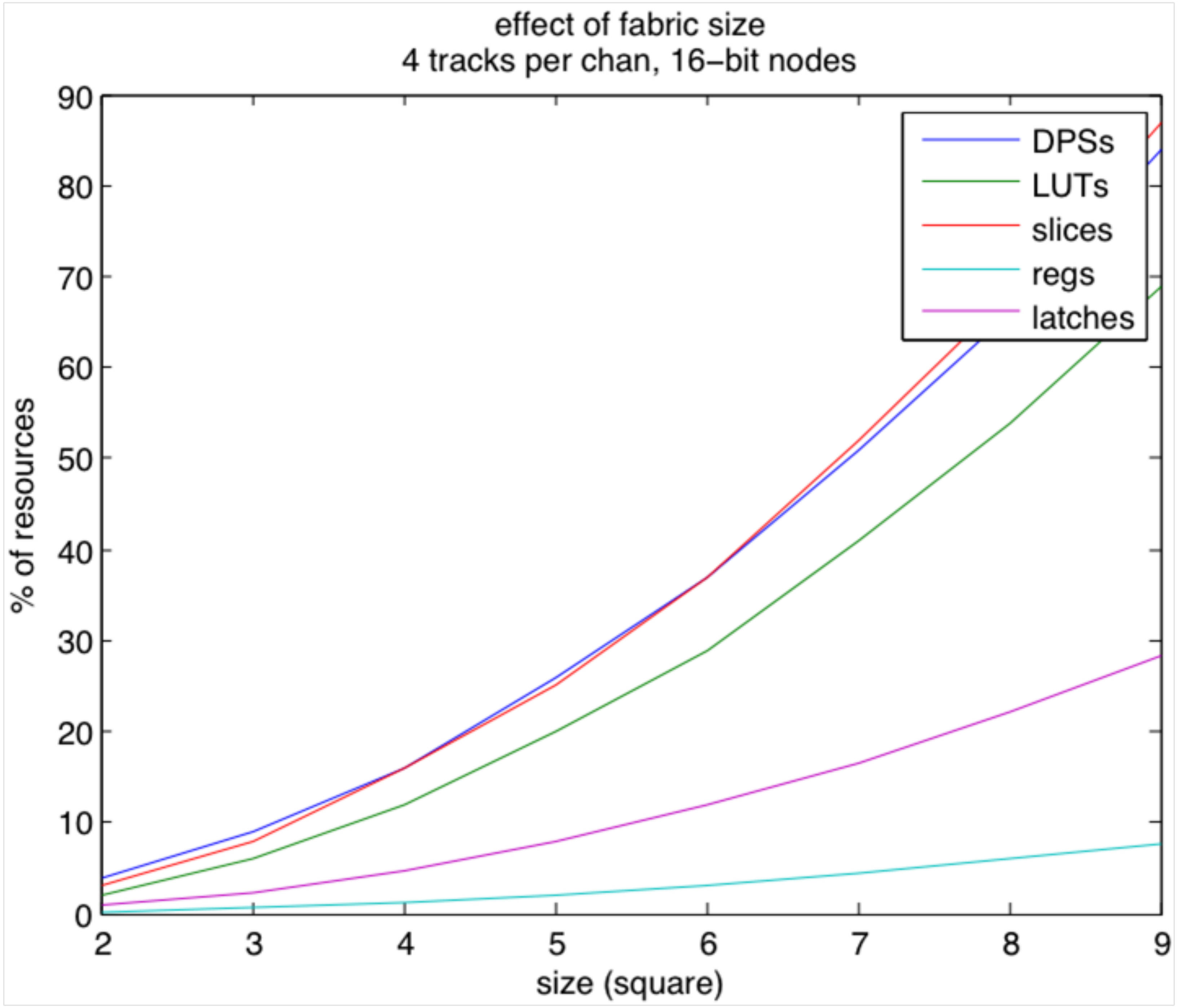
Area Results: Example

Example (from XST)

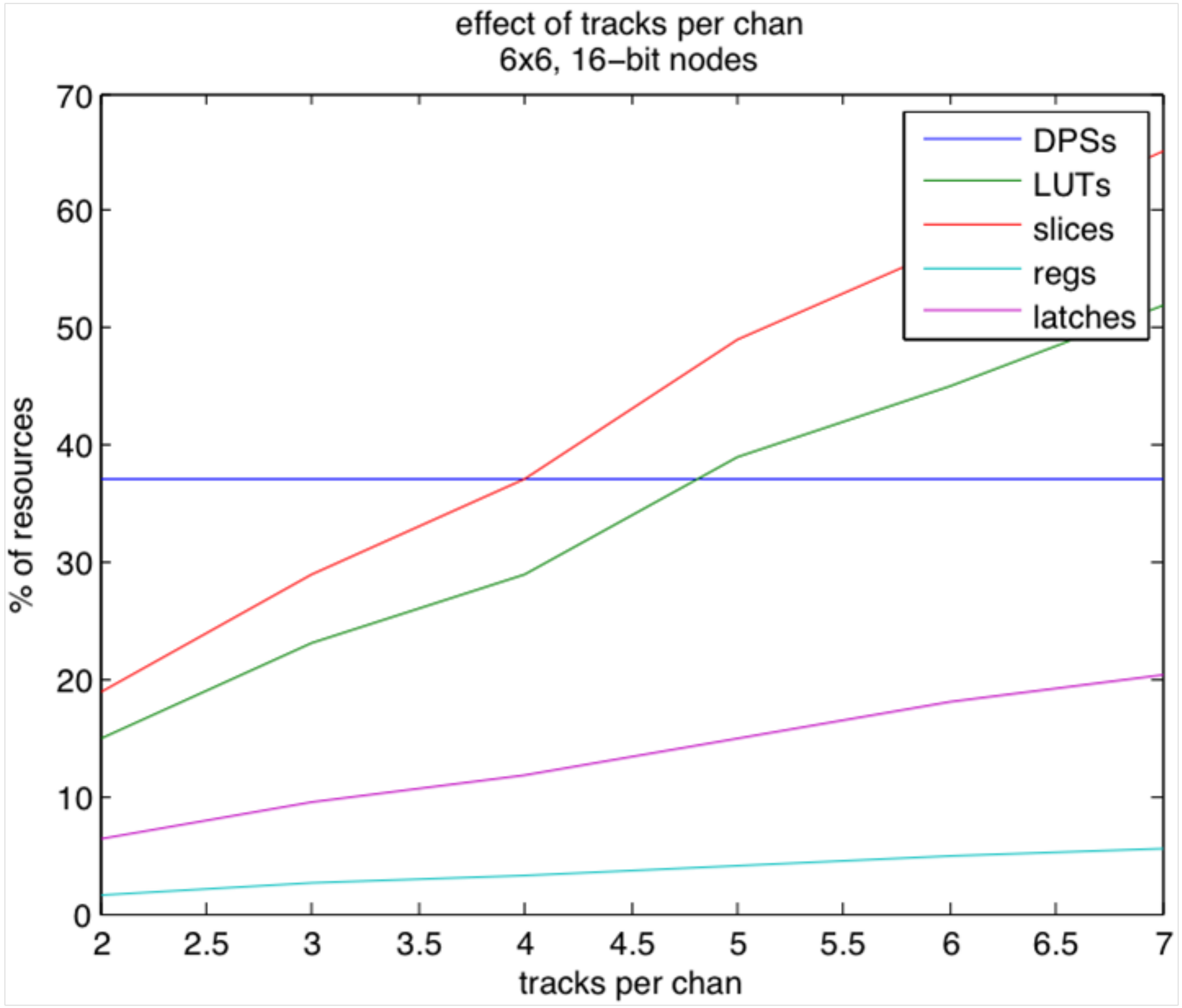
```
Logic Utilization:
  Total Number Slice Registers:    35,798 out of  98,304   36%
    Number used as Flip Flops:      7,622
    Number used as Latches:        28,176
  Number of 4 input LUTs:          68,307 out of  98,304   69%
Logic Distribution:
  Number of occupied Slices:                        42,894 out of  49,152   87%
    Number of Slices containing only related logic: 42,894 out of  42,894  100%
    Number of Slices containing unrelated logic:         0 out of  42,894    0%
      *See NOTES below for an explanation of the effects of unrelated logic
Total Number of 4 input LUTs:      68,307 out of  98,304   69%
  Number of bonded IOBs:              300 out of     768   39%
  Number of BUFG/BUFGCTRLs:             1 out of      32    3%
    Number used as BUFGs:               1
    Number used as BUFGCTRLs:           0
  Number of DSP48s:                    81 out of      96   84%

Total equivalent gate count for design:  693,994
Additional JTAG gate count for IOBs:  14,400
Peak Memory Usage:  740 MB
Total REAL time to MAP completion:  2 mins 27 secs
Total CPU time to MAP completion:   2 mins 27 secs
```

9x9, 4 tracks per channel, 16-bit granularity

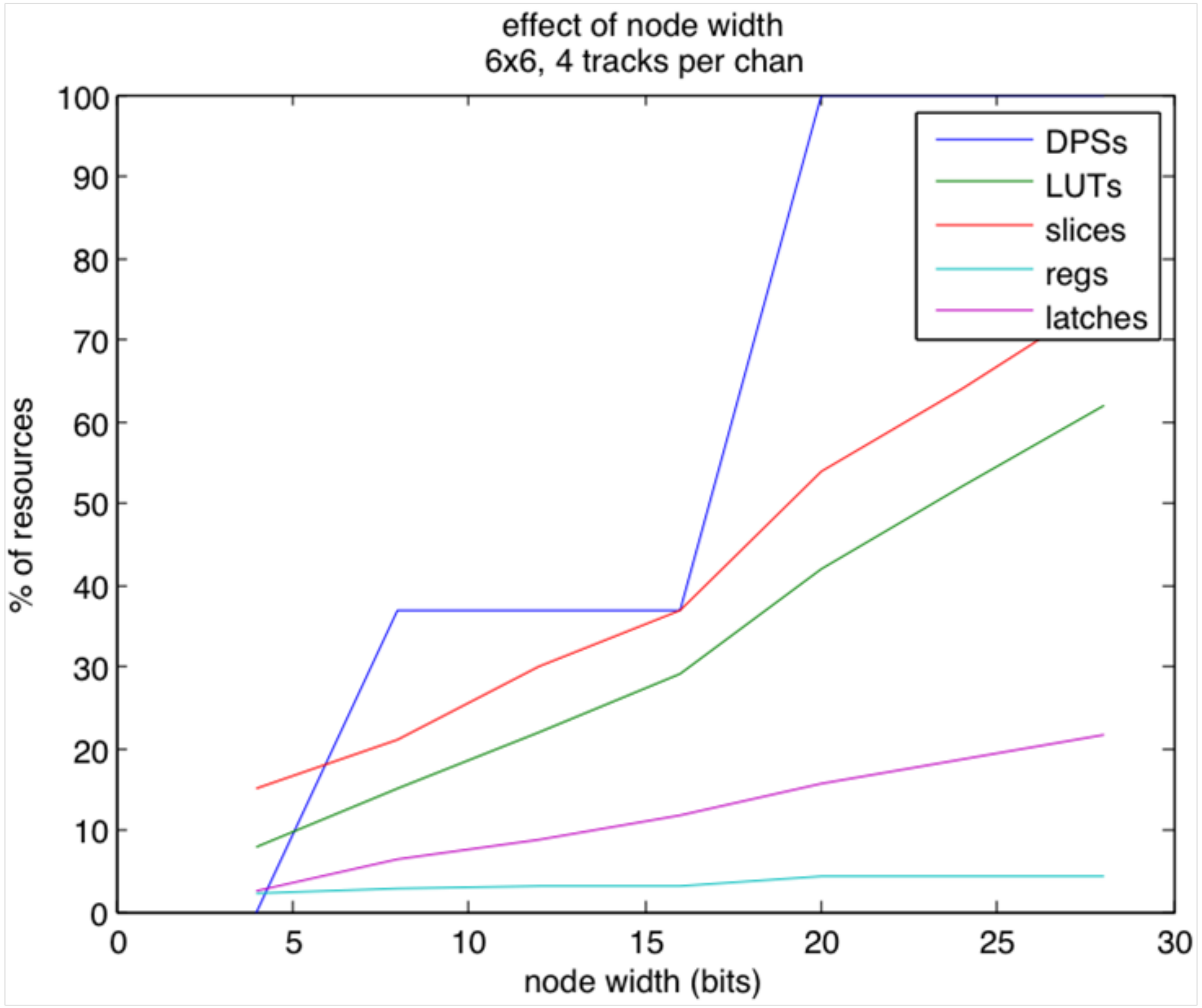
Area Results: Fabric Size



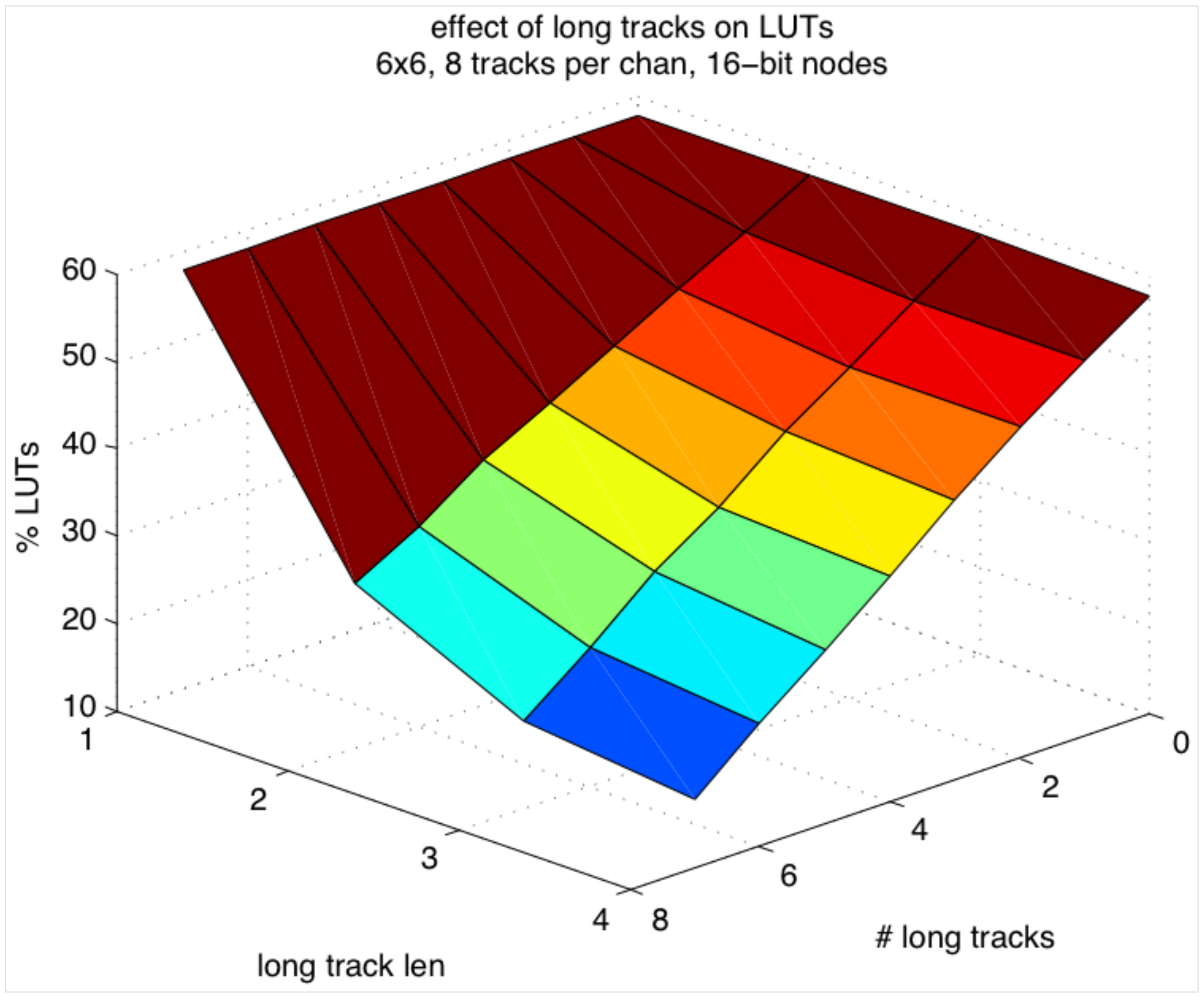
Area Results: Track Density

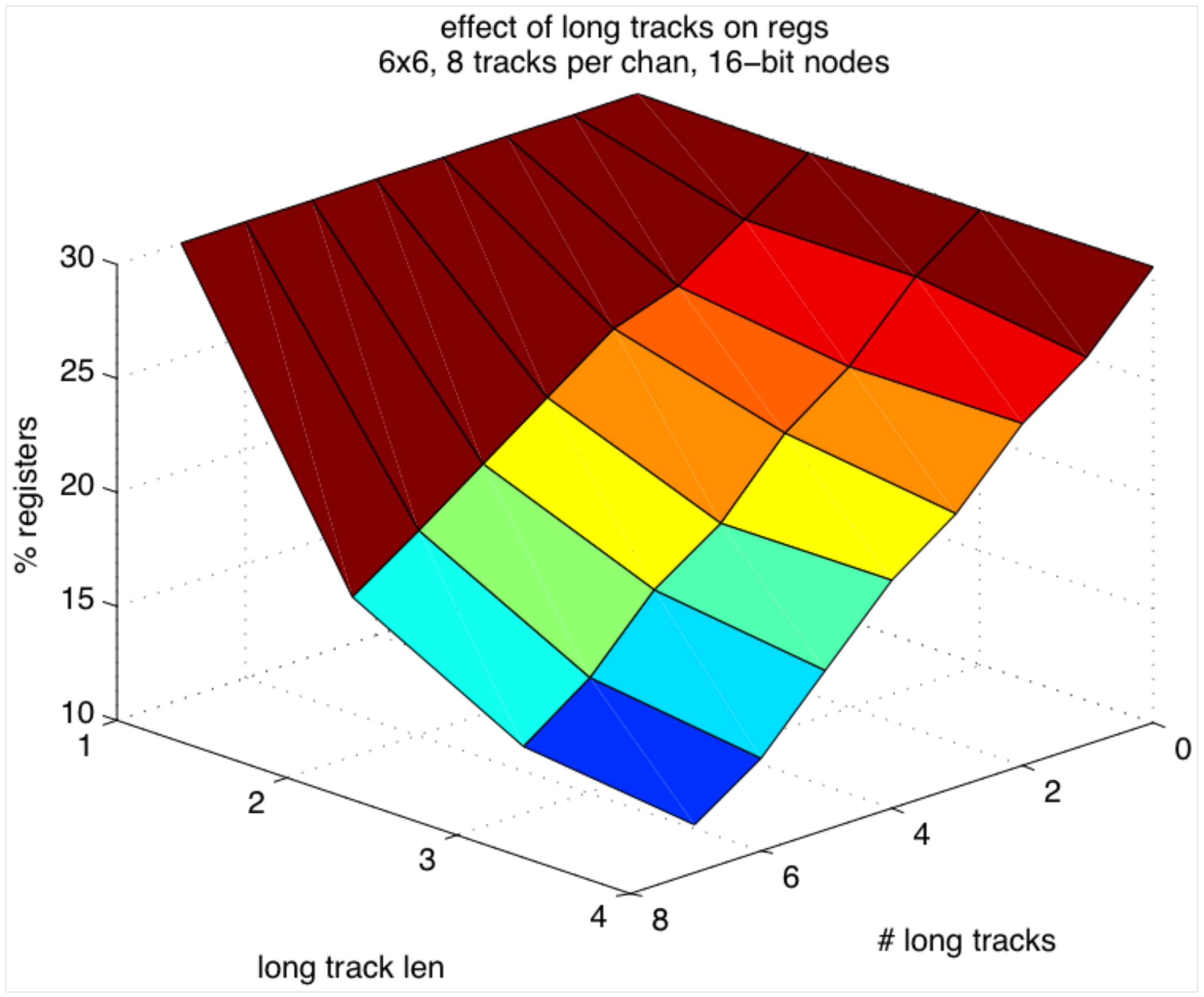


Area Results: Granularity



Area Results: Long Tracks





Area Results: Pending

Other parameters

- variable track density: different tracks count per channel in different locations in the fabric
- connection box flexibility
  - # and location of connection boxes
  - # of adjacent nodes connected

Tool Demo