Improving ASIC Reuse with Embedded FPGA Fabrics

John Teifel, Matt E. Land, Russel D. Miller
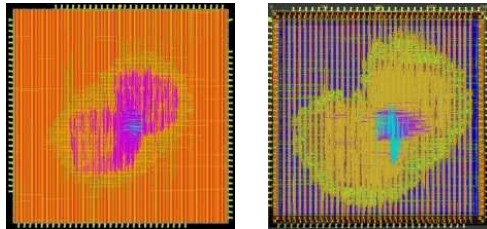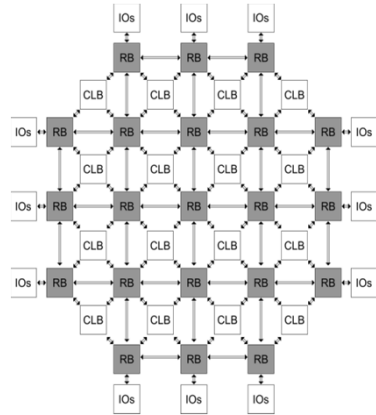
March 16, 2016

Orlando, FL

Outline

- Overview
  - Motivation
  - Moore's Law Scaling
  - Challenges
- Embedded FPGA fabric
  - Architecture
  - Software flow
- FPGA Physical Design
  - Layout
  - Performance
- Conclusions

Motivation

- ASIC development costs and schedules continue to escalate
  - New requirements drive ASIC re-spins/re-qualifications ($$$)
  - Mask-configurable ASICs can lower the cost/schedule of an ASIC re-spin by 2-5X, but they still do not allow for changes after fabrication

- Many system-on-chip designers have long desired FPGA blocks as a way to lower the risk of an ASIC re-spin
  - Enables post-fabrication design changes to be realized in the FPGA portion of the ASIC (if the ASIC is partitioned appropriately)
    - Fixing logic bugs (state machines, etc.)
    - Addressing new requirements (interface timing, etc.)

- Also allows critical IP to be protected in non-Trusted foundry flows
  - By implementing the sensitive logic in the FPGA blocks (after fabrication) instead of hard-wired into the ASIC
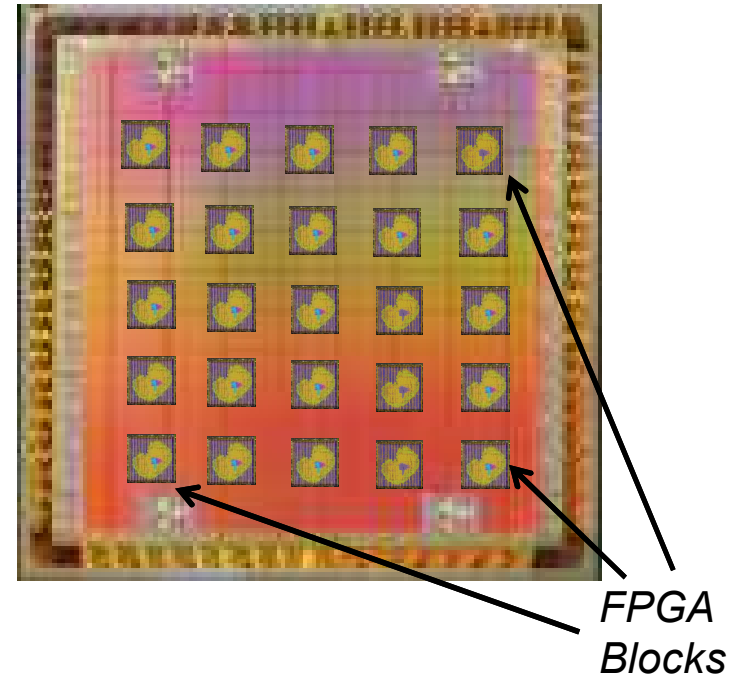
Embedded FPGA Architectures

Traditional

ASIC

12mm

5mm

FPGA Block

Distributed
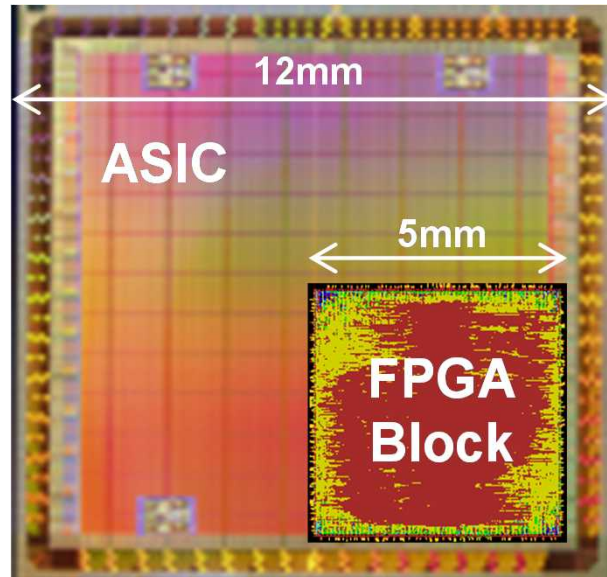
FPGA Blocks
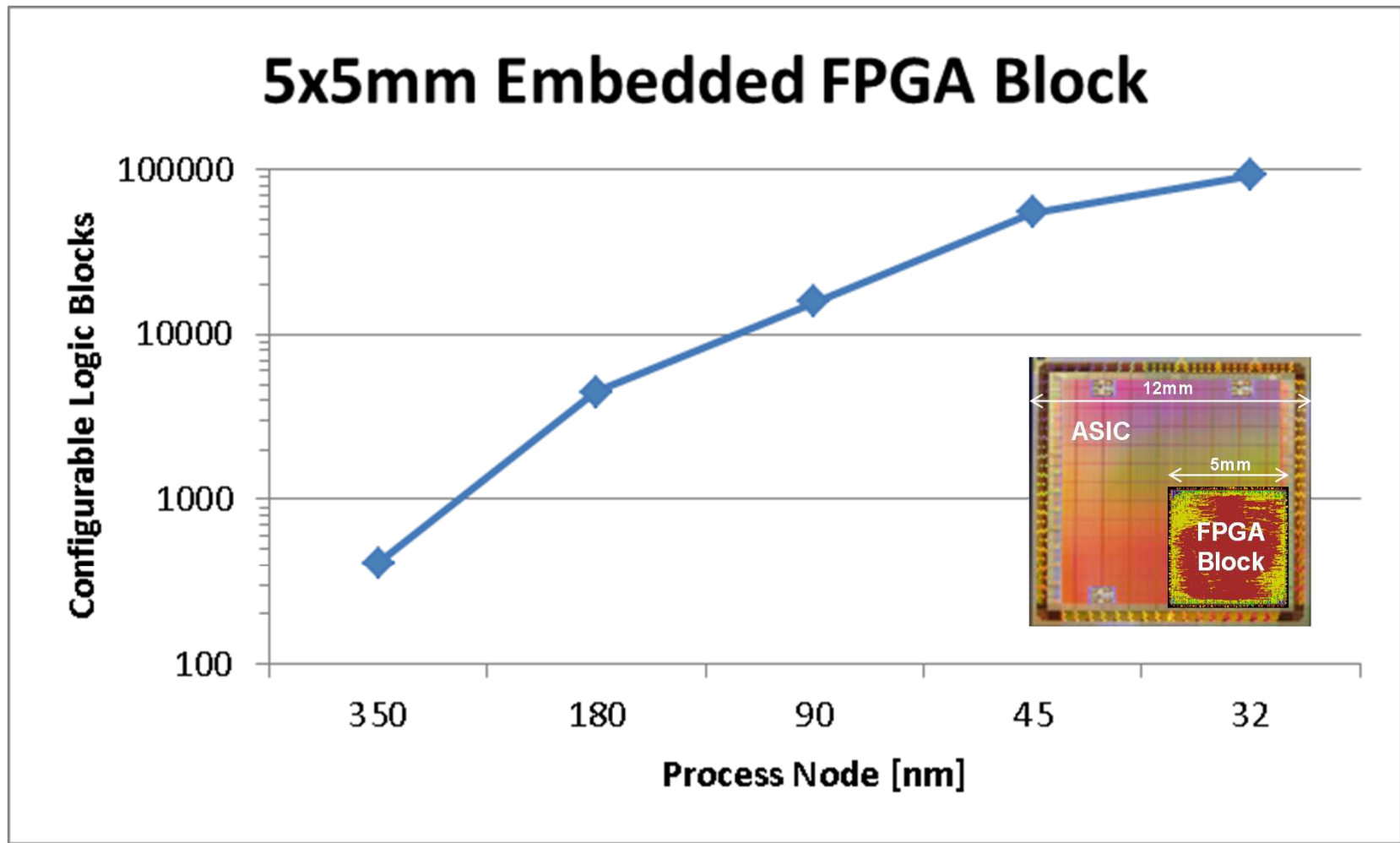
- A 5x5mm <u>embedded</u> FPGA block within a 12x12mm ASIC

- Array of 1x1mm <u>distributed</u> FPGA blocks within a 12x12mm ASIC

Moore's Law FPGA Scaling

FPGA logic densities have dramatically increased

Challenges

1. **IP Availability**
   - Leading FPGA vendors do not offer IP blocks that can be used in ASICs
     - Custom layout makes it difficult to support multiple fabrication processes
     - Use in high-volume commercial ASICs is difficult (due to increased silicon cost)
   - Many academic papers in the mid-2000's on "synthesizable" FPGA IP
     - Able to easily re-target the FPGA layout to any fabrication process
     - Two recent startups attempting to commercialize the concepts (Adicsys & Menta)

2. **Design Tool Support**
   - How is the FPGA IP specified (size, resources, I/O, etc.)?
   - How is the FPGA IP programmed?
   - How is the FPGA IP handled during ASIC synthesis, layout, etc.?
   - How is the timing/logic verified between the ASIC & FPGA domains?
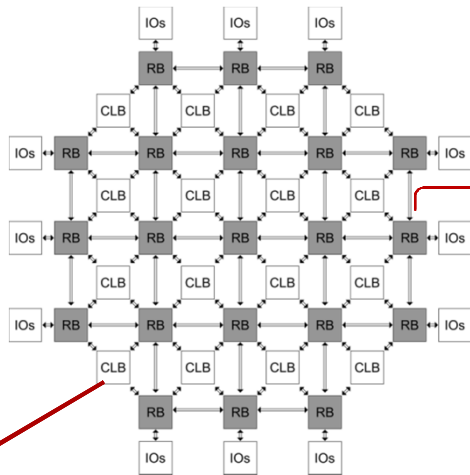
3. **Design Partitioning**
   - Not trivial to divide large designs between the ASIC & FPGA domains
   - The FPGA's overheads need to be weighed against the benefit of re-configurability

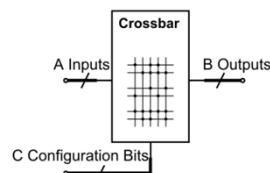## This work investigates the 1st two issues

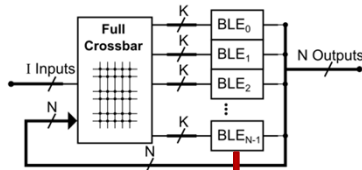# Outline

Embedded FPGA Fabric

IOs · RB · CLB array diagram

RB (Routing Block)
A: Number of inputs to the routing block
B: Number of outputs of the routing block
C: Number of configuration bits

Crossbar
A Inputs — B Outputs
C Configuration Bits
Generic view of routing interconnect

CLB (Configurable Logic Block)
K: Number of inputs to the LUT
N: Number of BLEs
I: Number of inputs to CLB

Full Crossbar
I Inputs
N
BLE_0
BLE_1
BLE_2
BLE_{N-1}
K
N Outputs
Representative CLB

BLE (Basic Logic Element)
K: Number of inputs to the LUT
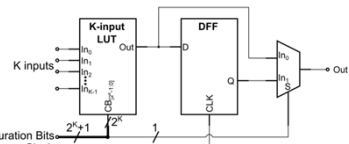
K-input LUT
Out
In_0
In_1
In_2
In_{N-1}
K inputs
DFF
D
Q
CLK
Out
2^K+1 Configuration Bits
2^K
Clock
Representative BLE

- Standard "Island" style FPGA
  - Array of configurable logic blocks connected together with crossbar routing blocks
  - Lookup-table (LUT) logic elements
- Almost all architecture features are configurable, including
  - Size of the FPGA array
  - Number of I/Os into FPGA
  - Size & number of LUTs
  - Number of I/Os to CLBs
  - Number of routing tracks
  - Connectivity of routing tracks
  - Long vs short routing tracks
  - …
- An XML file is used to concisely specify the FPGA's high-level architecture parameters
  - Compatible with VTR FPGA tools from the University of Toronto*
  - A Sandia script is used to generate an "architecture-driven RTL model" from the XML specification

Very flexible FPGA architecture

*Luu, et.al., VTR 7.0, ACM TRTS, Jun 2014
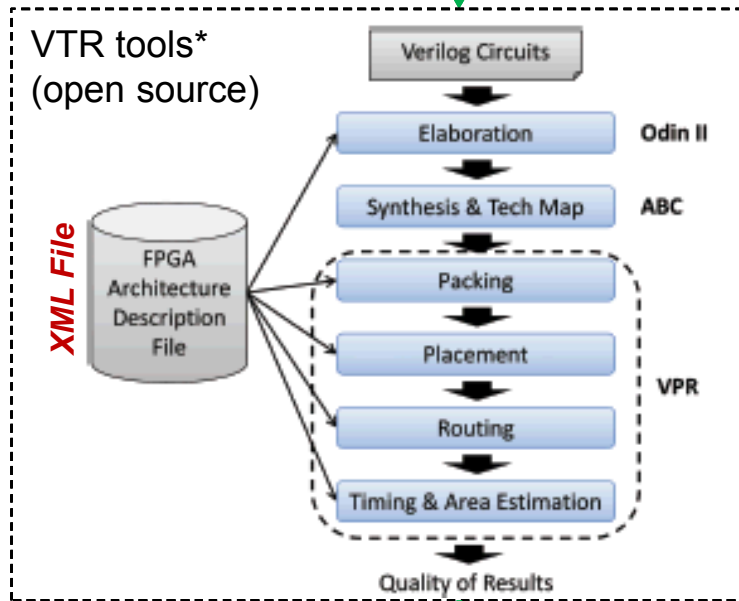
FPGA Architecture Studies
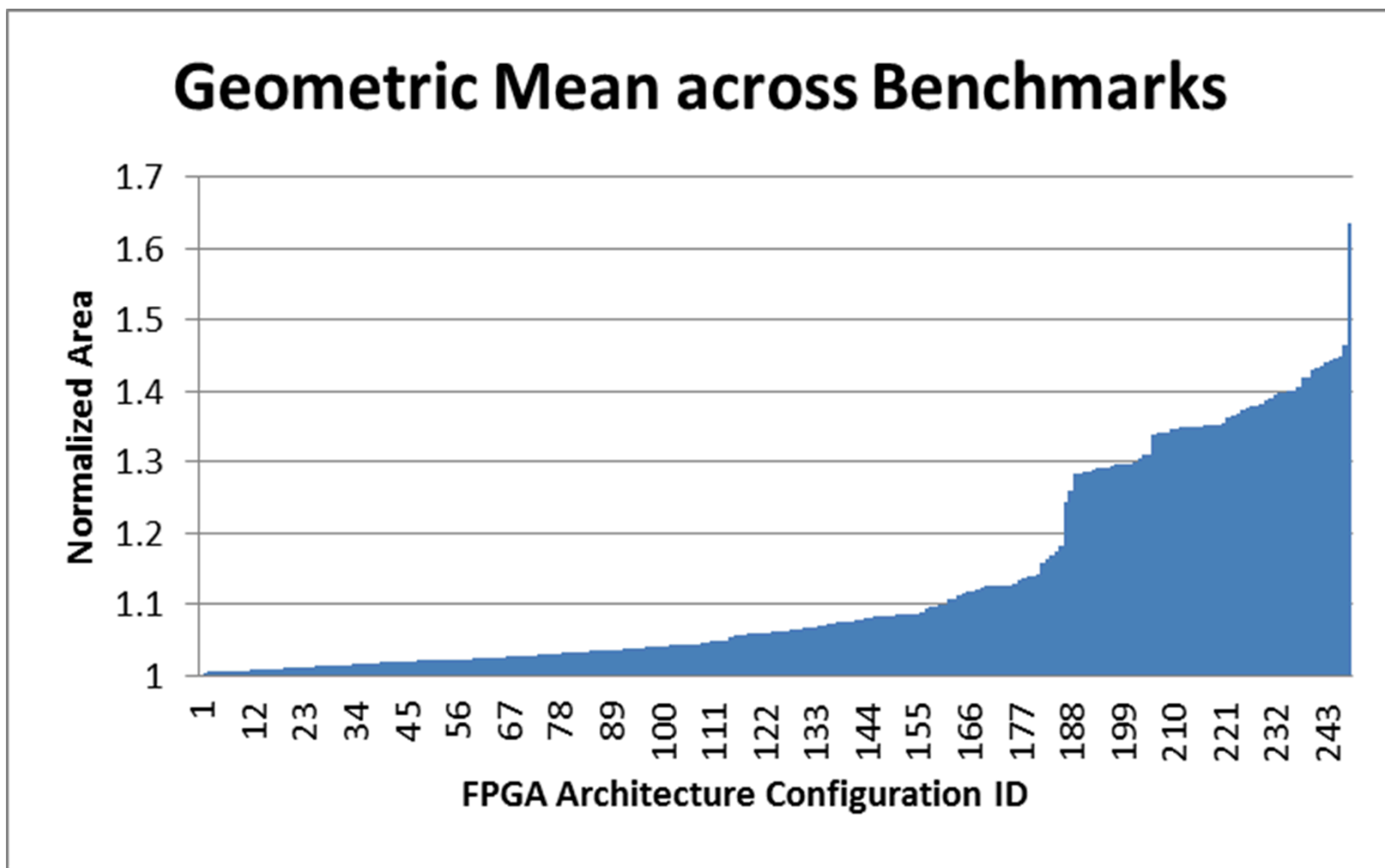
3-input Look-up-Table architectures were most efficient

CAD Flow: RTL to bit-stream

User design/benchmark

Front-end Logic Synthesis (commercial)

VTR tools*
(open source)

Verilog Circuits

Elaboration — Odin II

Synthesis & Tech Map — ABC

FPGA Architecture Description File

XML File

Packing

Placement — VPR

Routing

Timing & Area Estimation

Quality of Results

FPGA Fabric

XML File

Architecture-driven RTL model of FPGA (Sandia)

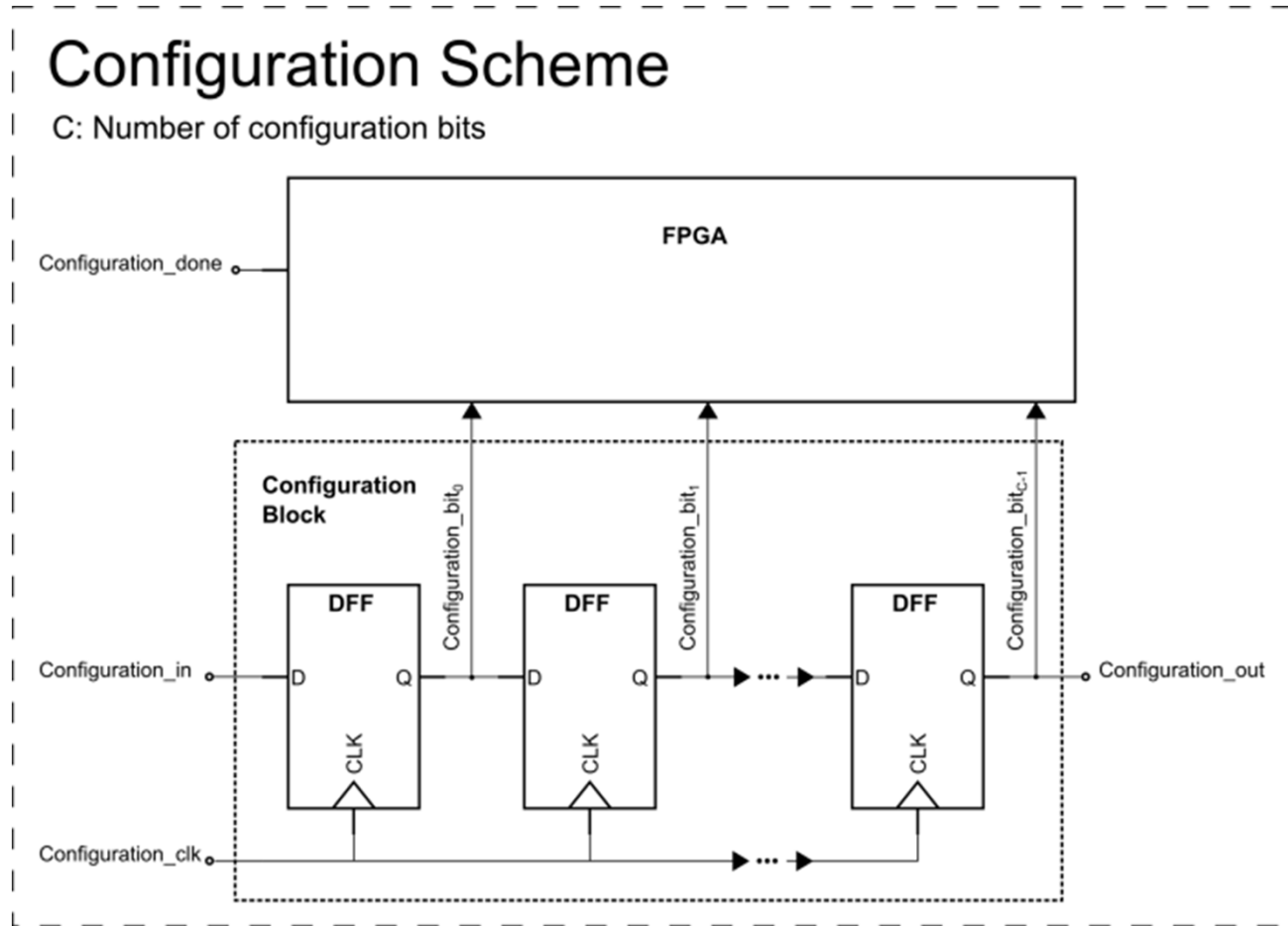**FPGA fabric synthesis (commercial)**

Programming bit stream generation (Sandia)

FPGA fabric layout (commercial)
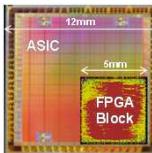
Full software flow developed (details are in the paper)
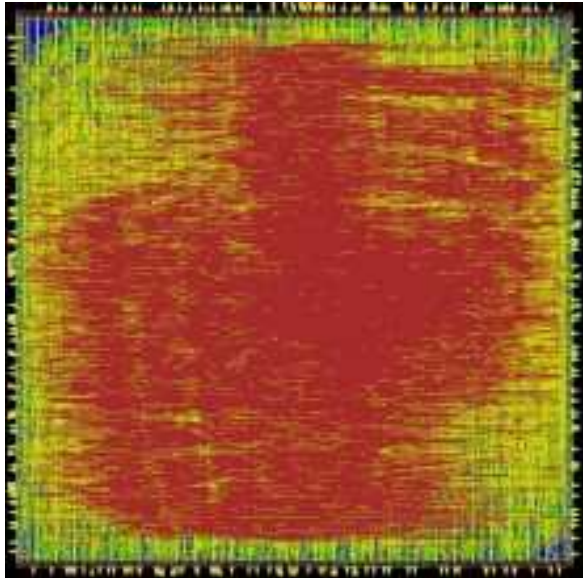
Embedded FPGA Configuration

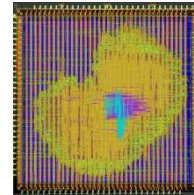**Programming bits are stored in shift registers**

Outline

- Overview
  - Motivation
  - Moore's Law Scaling
  - Challenges
- Embedded FPGA fabric
  - Architecture
  - Software flow
- FPGA Physical Design
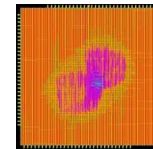 - Layout
  - Performance
- Conclusions

Physical Design

*350-nm SOI, 6x6mm*

90-nm bulk (2x2mm)

45-nm SOI (1.5x1.5mm)

- Laid out FPGA fabrics in 350nm, 90nm, and 45nm processes
  - 400 logic elements (with 3-input Lookup Tables)
  - Synopsys logic synthesis & Cadence auto-place-and-route
- Reasonable results with 1st pass layout attempts
  - Desirable to keep most of the layout focus on the ASIC (not the FPGA)

Laid out in days (instead of months)

Benchmark Performance
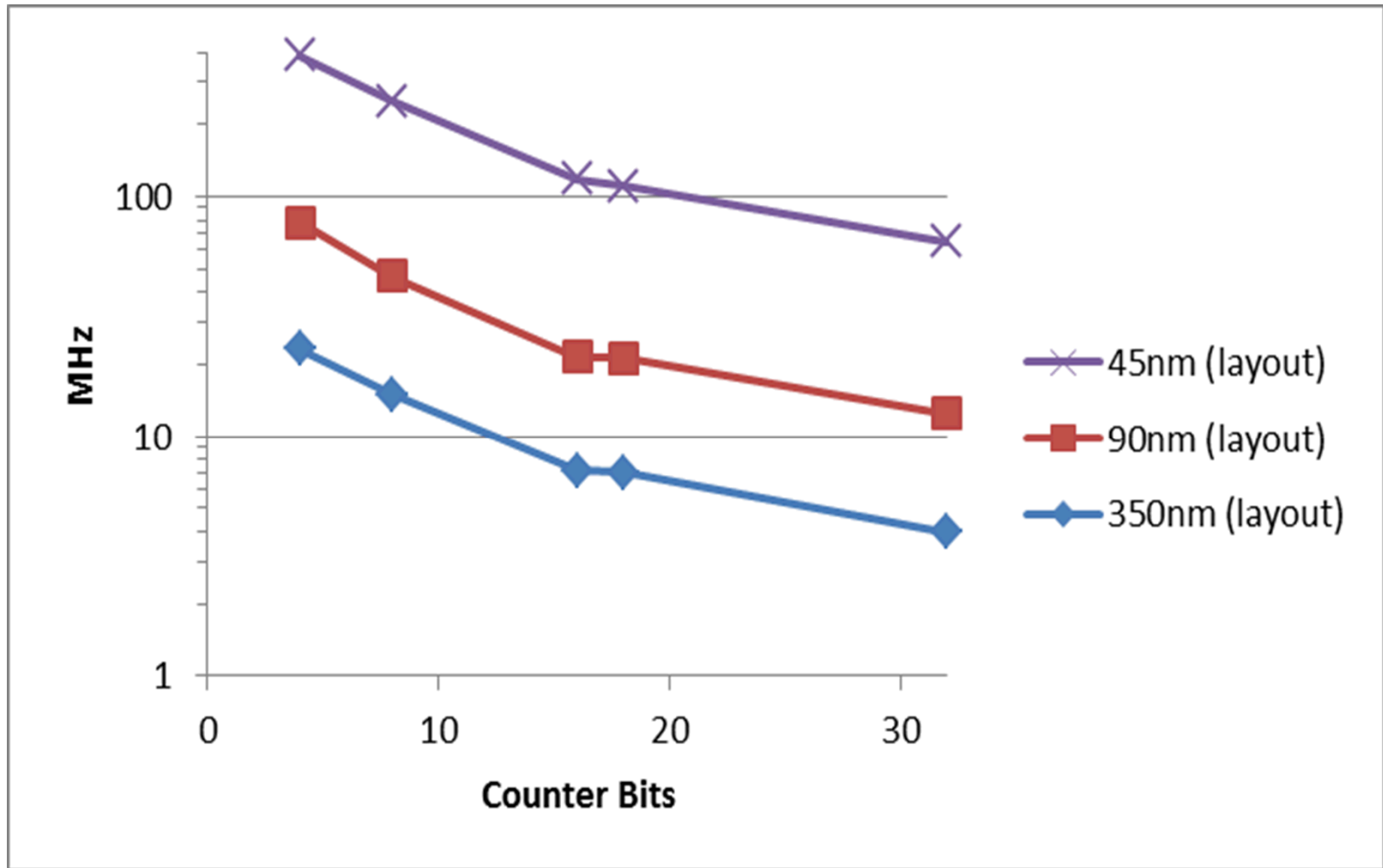
Speed is modest, but improves with Moore's Law

Conclusions

- We investigated two of the biggest challenges associated with embedded FPGA fabrics – IP availability and CAD tool support
  - IP availability can be overcome by leveraging FPGA fabric architectures that are compatible with open-source FPGA tools
  - CAD tool support can be overcome by integrating together open-source, commercial, and custom software tools
  - Required only several months of work to develop tool flow & tapeout prototype embedded FPGA fabric

- Moore's Law drives the feasibility of embedded FPGA fabrics
  - Best performance and density in advanced process nodes (≤45nm)

- Design partitioning remains a key challenge
  - How to decide which logic goes into the ASIC domain, and which logic goes into the FPGA domain
  - Make the wrong decision, and the ASIC is no longer reusable!

Questions?