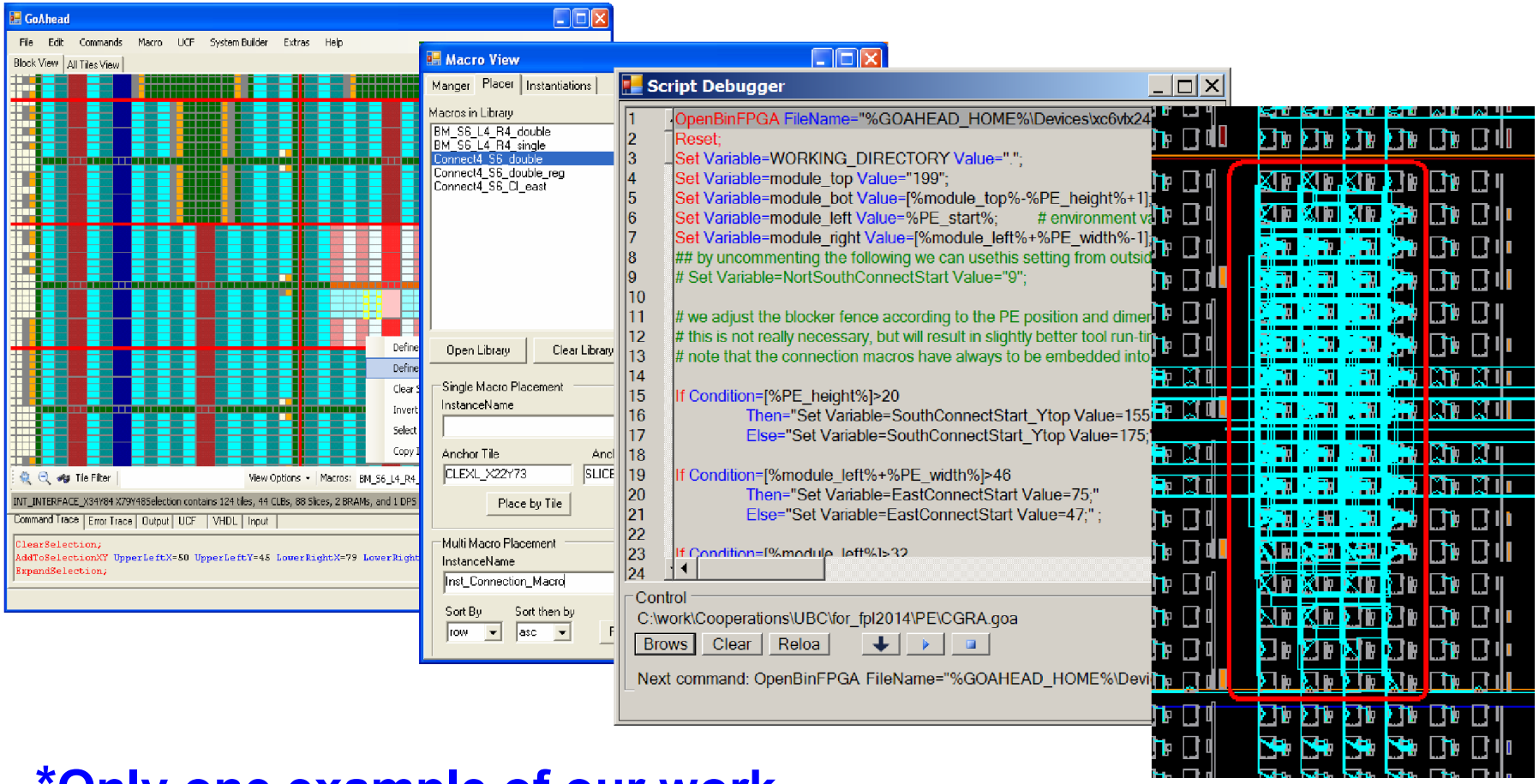
# FPGA Virtualization for Enabling General Purpose Reconfigurable Computing


Dirk Koch, University of Manchester, dirk.koch@manchester.ac.uk

# FPGA Research in Manchester*

**GoAhead design tool for partial reconfiguration (Xilinx FPGAs)**

- **Probably most advanced PR tool available! (new release for Vivado coming soon)**




*Only one example of our work

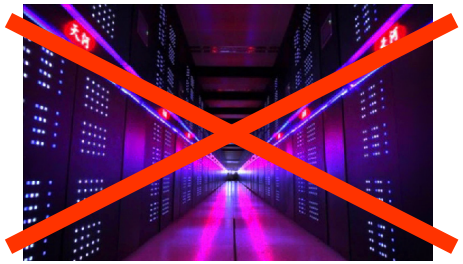
# FPGA Research in Manchester

- **Module relocation* & multi module instantiation***
- **Multiple modules in a combined region***
- **Direct communication between partial modules***
- **Real design preservation***
- **Component-based design using physically implemented netlists* (already placed and routed)**
  **→ provides portability across systems***
  **→ permits rapid recompile***
- **Module encapsulation (also available by Xilinx, but we can do it smarter)**
- **High performance/low latency***

*** not available by the Xilinx or Altera design tools**

# FPGAs in Datacenters: Where are we?

**No recent TOP500 Supercomputer uses FPGAs!**



- **But Industry is catching up:**

**Intel took over the second biggest FPGA vendor (Altera) one year ago for US$ 16.7B**

**Microsoft: 1632 servers with FPGAs for Bing Search**
**→2x throughput, 29% less latency, ~2x energy efficiency**

**Baidu: search algorithm acceleration using FPGAs**
**→3.5x more throughput than a GPU at 10% power**

- **Interest is fueled by C-to-gates compilation for FPGAs**
- **Does not address logic explosion → Virtualization**

# FPGAs Virtualization: Where are we?

**The most universal definition:**

**FPGA virtualization is a technique that provides an abstraction to the used FPGA hardware. This is commonly used for hiding low level details (such as the physical resources) from the user.**

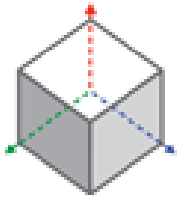
**Two main directions:**

- **Overlays: a programmable architecture on top of an FPGA**
- **Dynamic modules: temporal partitioning**

- **FPGA platform virtualization: virtual memory, virtual PCIe, etc.**
- **Configuration protection**

# FPGA Overlays

**Reasons:**

- **Programmability** (software like instead of RTL coding)
- **Portability** across different FPGAs
- **Limits logic explosion** (by reusing functional blocks)
-  VectorBlox embedded supercomputing — **FPGA vector processor**
- 

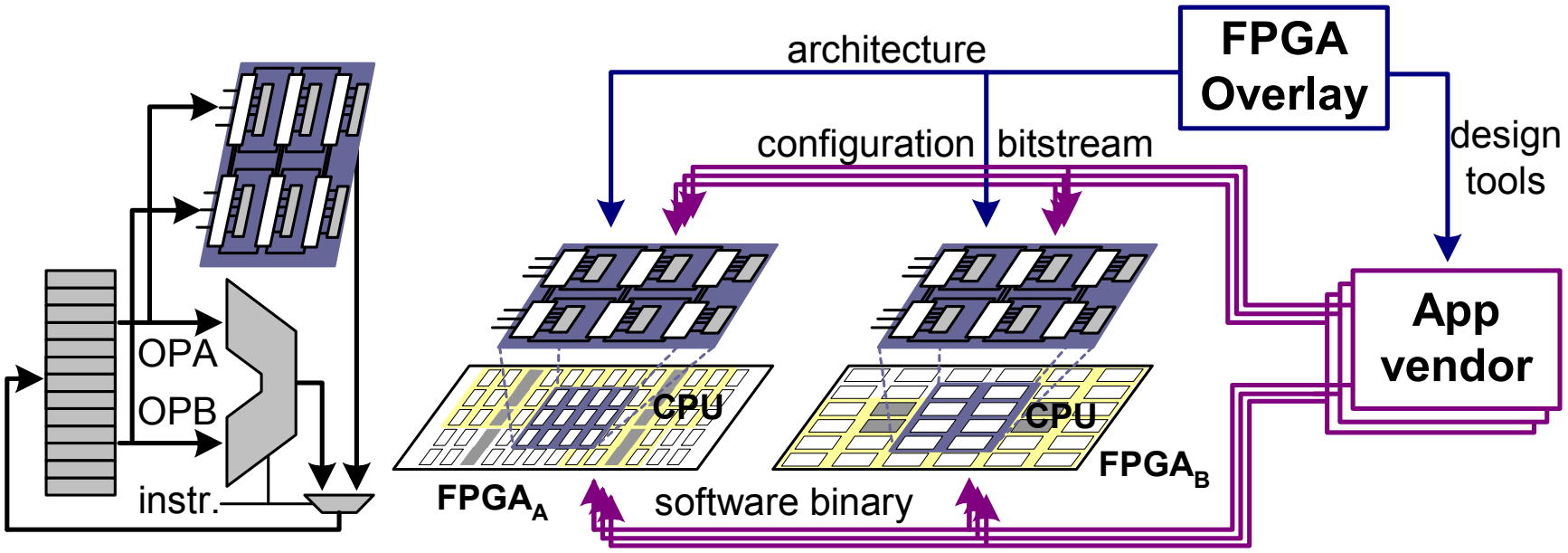
**Dragon processor (bioinformatics)**

# The Cost of Programmability

- **Software (CPU): might provide 100 possible instructions, but only one fired in each operational cycle**
- **Hardware (FPGA):**
  - **not all blocks are usable**
  - **interconnection network (for wiring logic cells / PEs)**

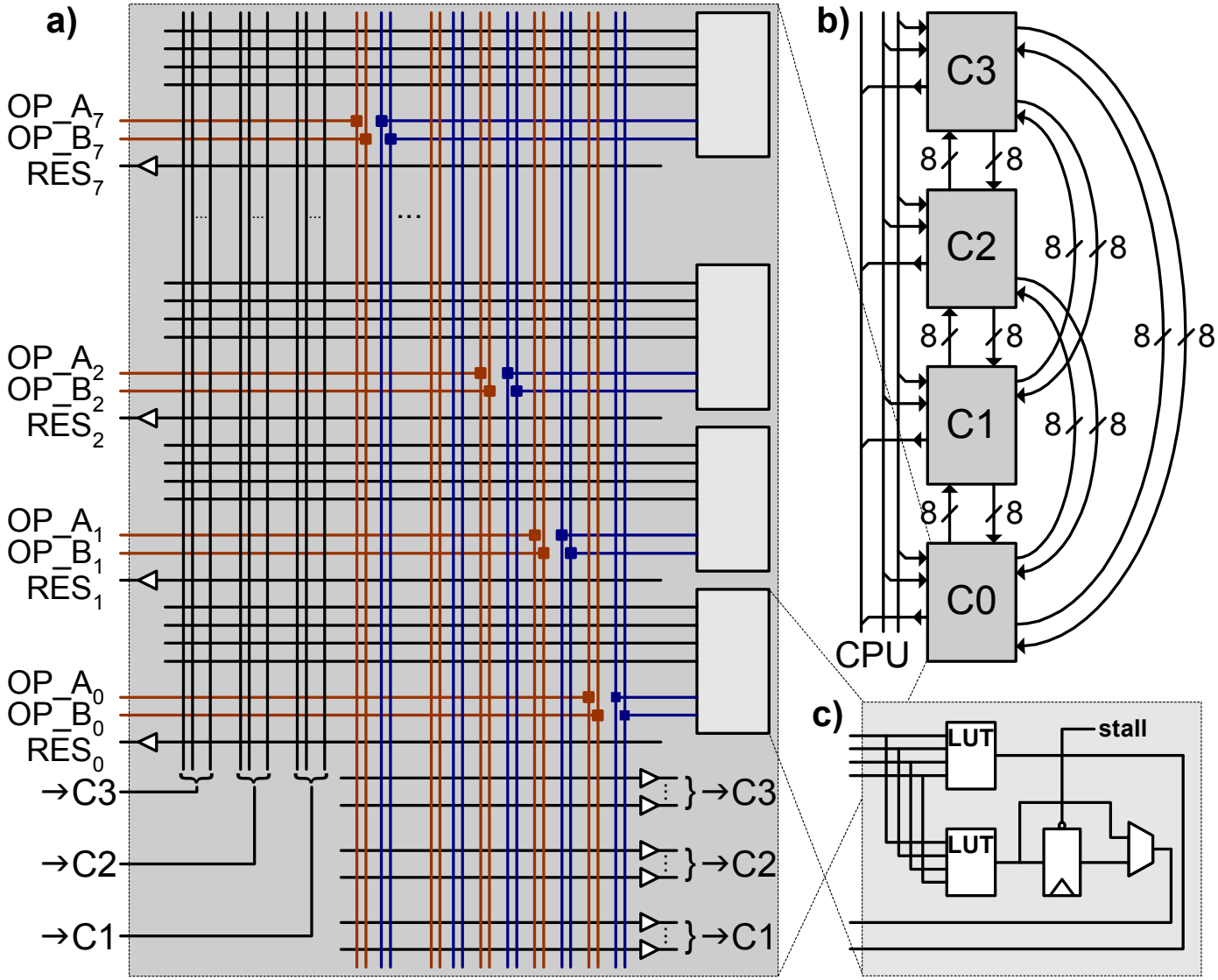
- **Overlays: software and hardware overhead!**
  - → **Needs very good reasons to justify this extra level of programmability**
  - → **Forget about "static overlays" (static dataflow machine) (Except you use FPGAs for CGRA prototyping)**
  - → **Time variant usage of resources mandatory**

# Make Overlays Efficient



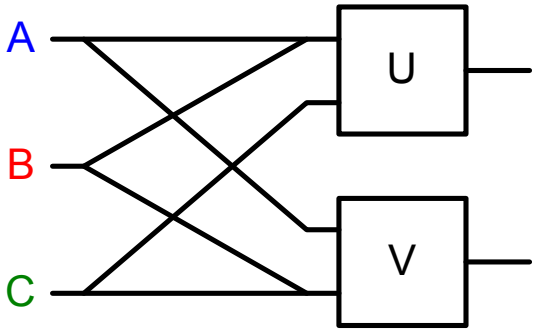

- **A CPU can be synthesized for different target FPGAs (trivial)**
- **Custom Instructions can be made portable using an overlay**
- **An application consists of**
  - **1) software binary**
  - **2) overlay configuration bitstream**

# Overlay Architecture

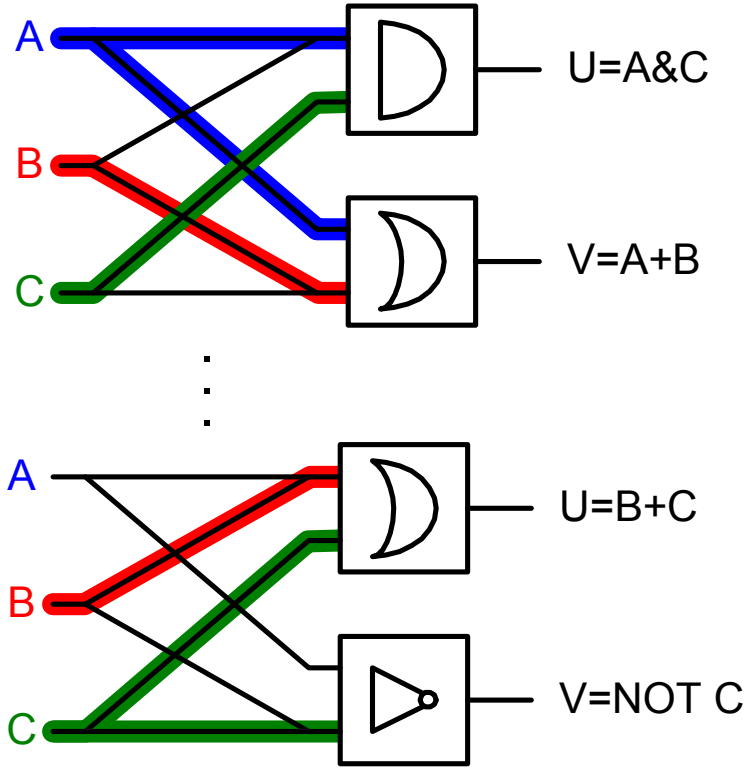


# Direct Overlay Routing: Problem Definition

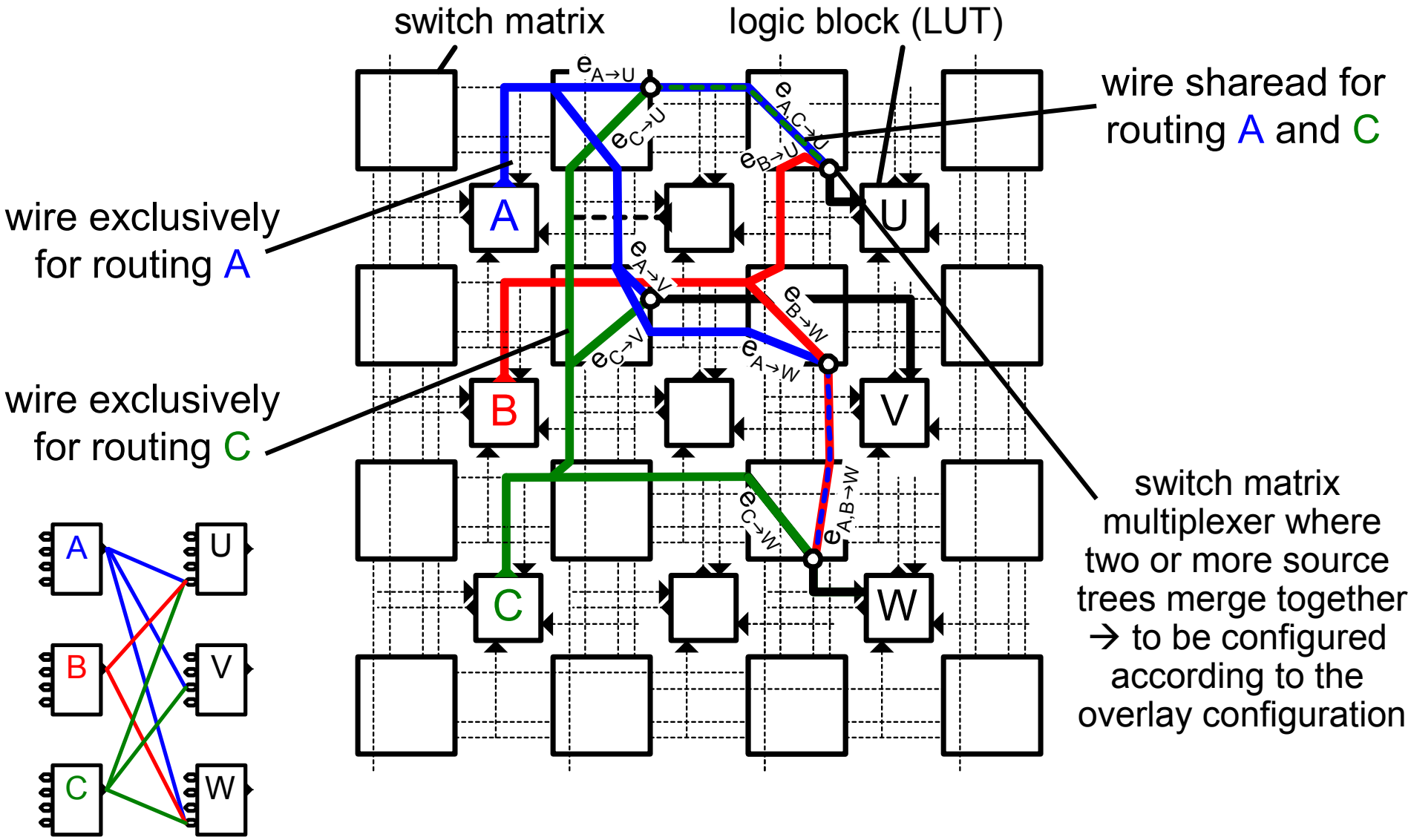
A reservation



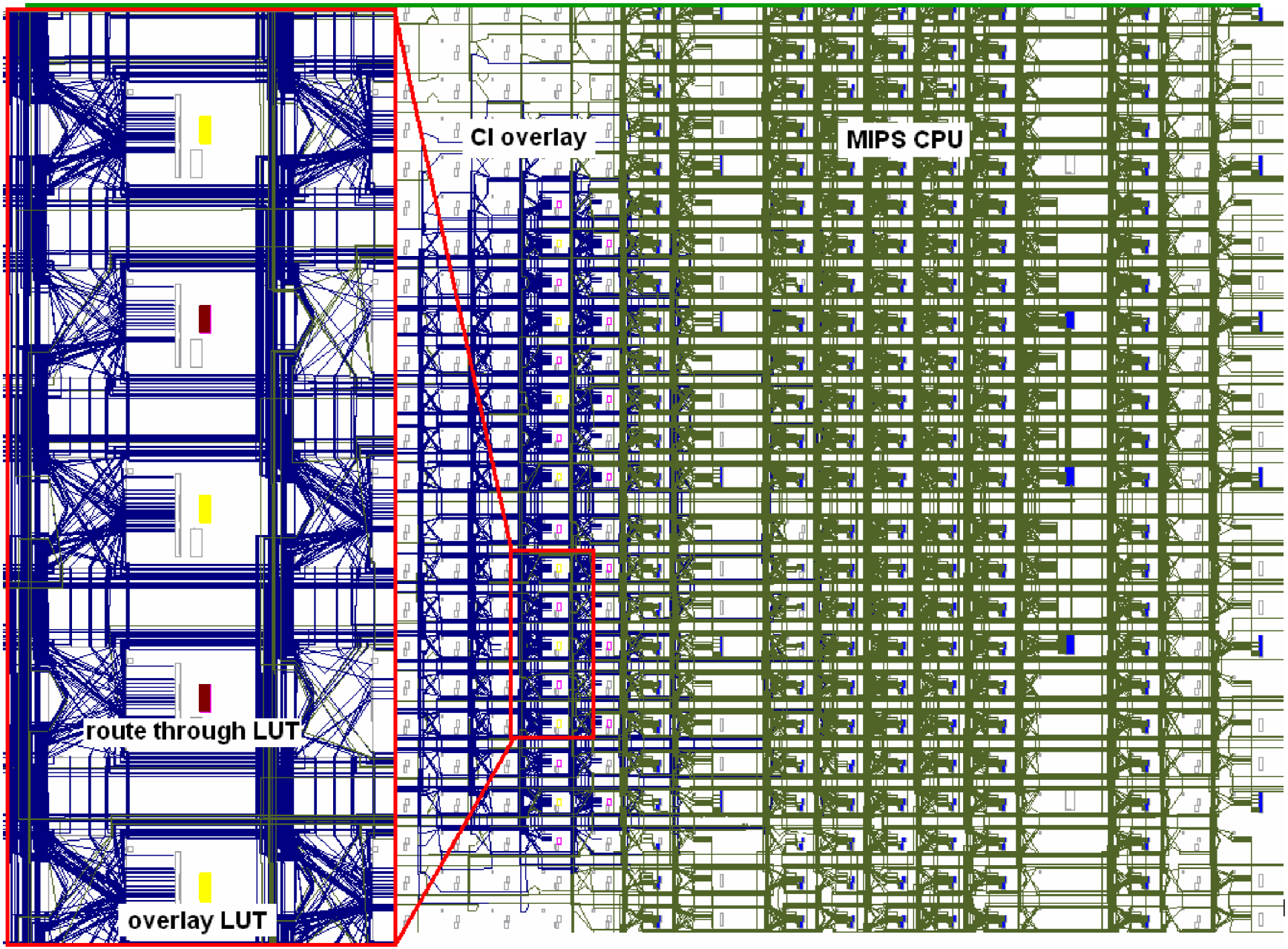

for implementing different circuits



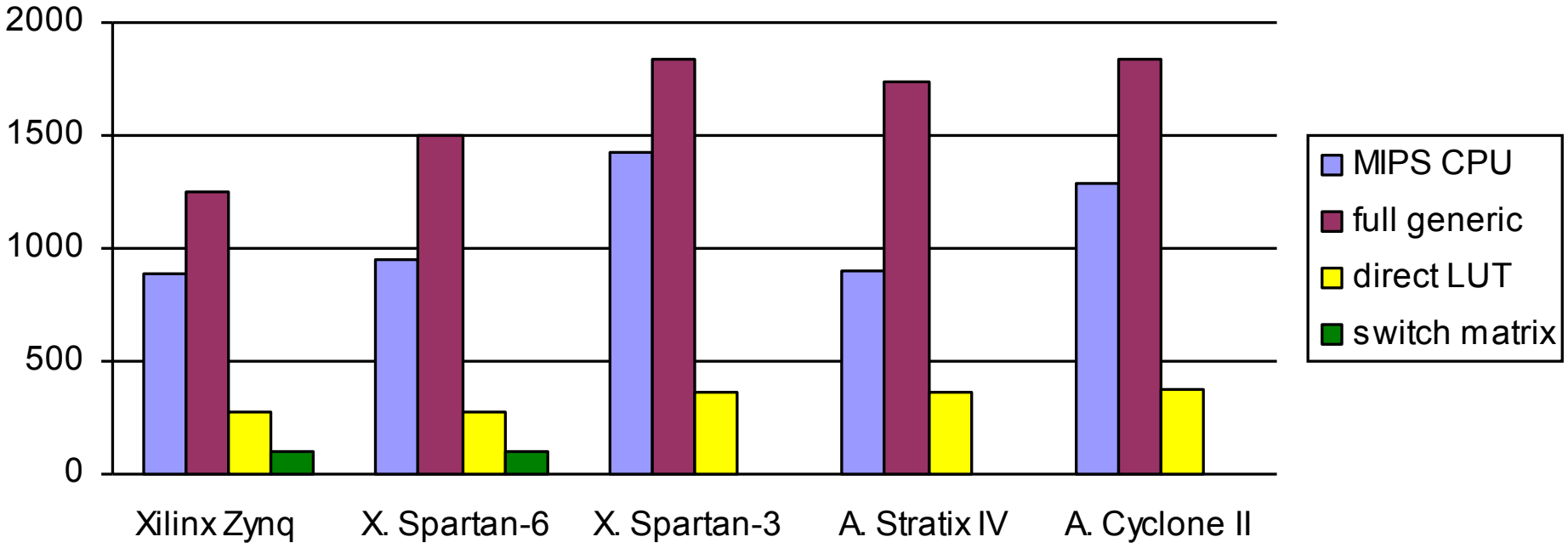
# Direct Overlay Routing: Optimization



# Direct Overlay Routing: Implementation

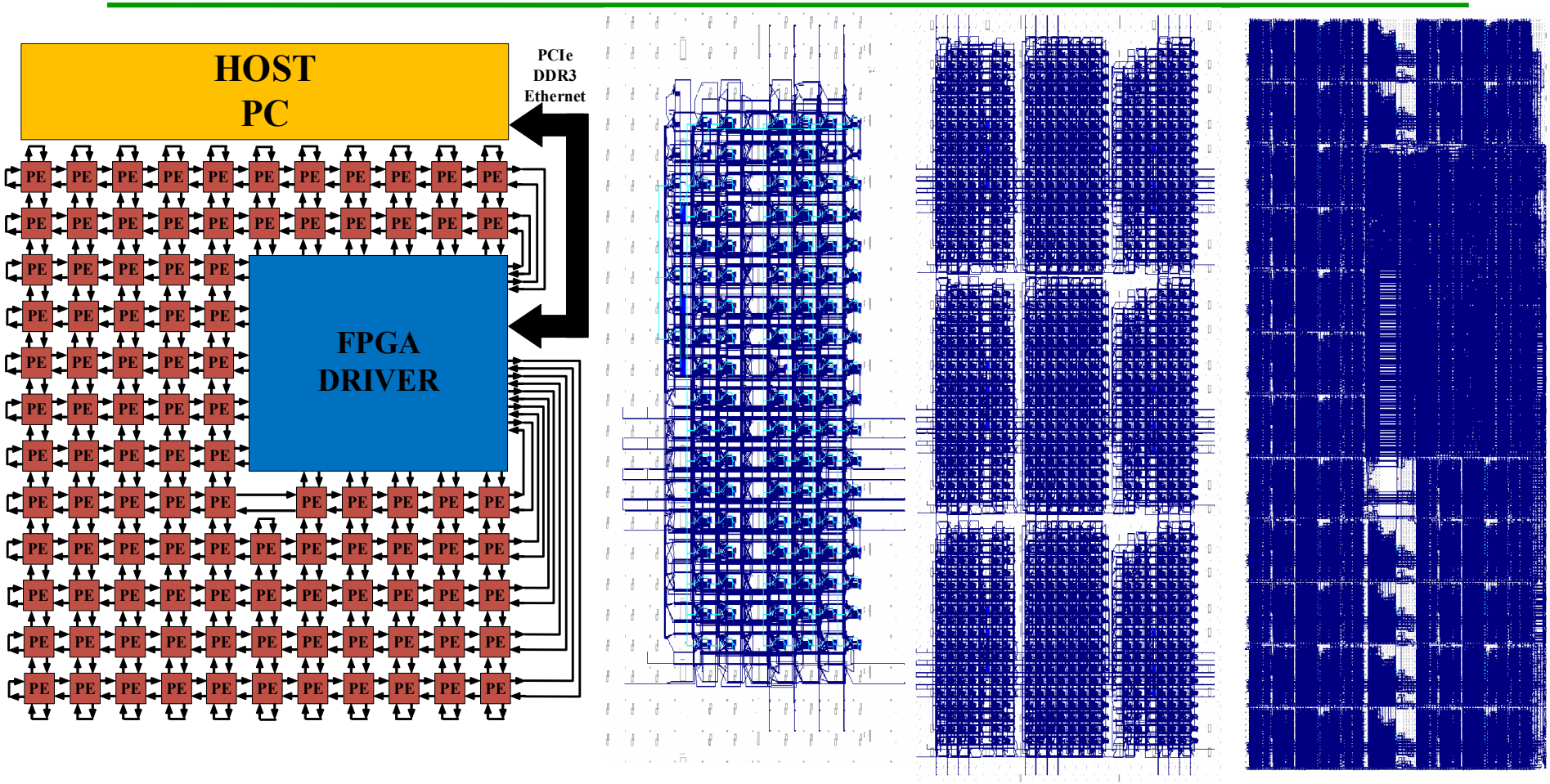


# Direct Overlay Routing: Results




- **Only 5x - 10x logic cost over direct FPGA implementation**
- **Working on Xilinx Zynq and Spartan-6 (including bitstream manipulation)**
- **Shares some ideas of the TLUT / TCON approach (Gent)**
- **FPL 2013: "*An Efficient FPGA Overlay for Portable Custom Instruction Set Extensions*"**

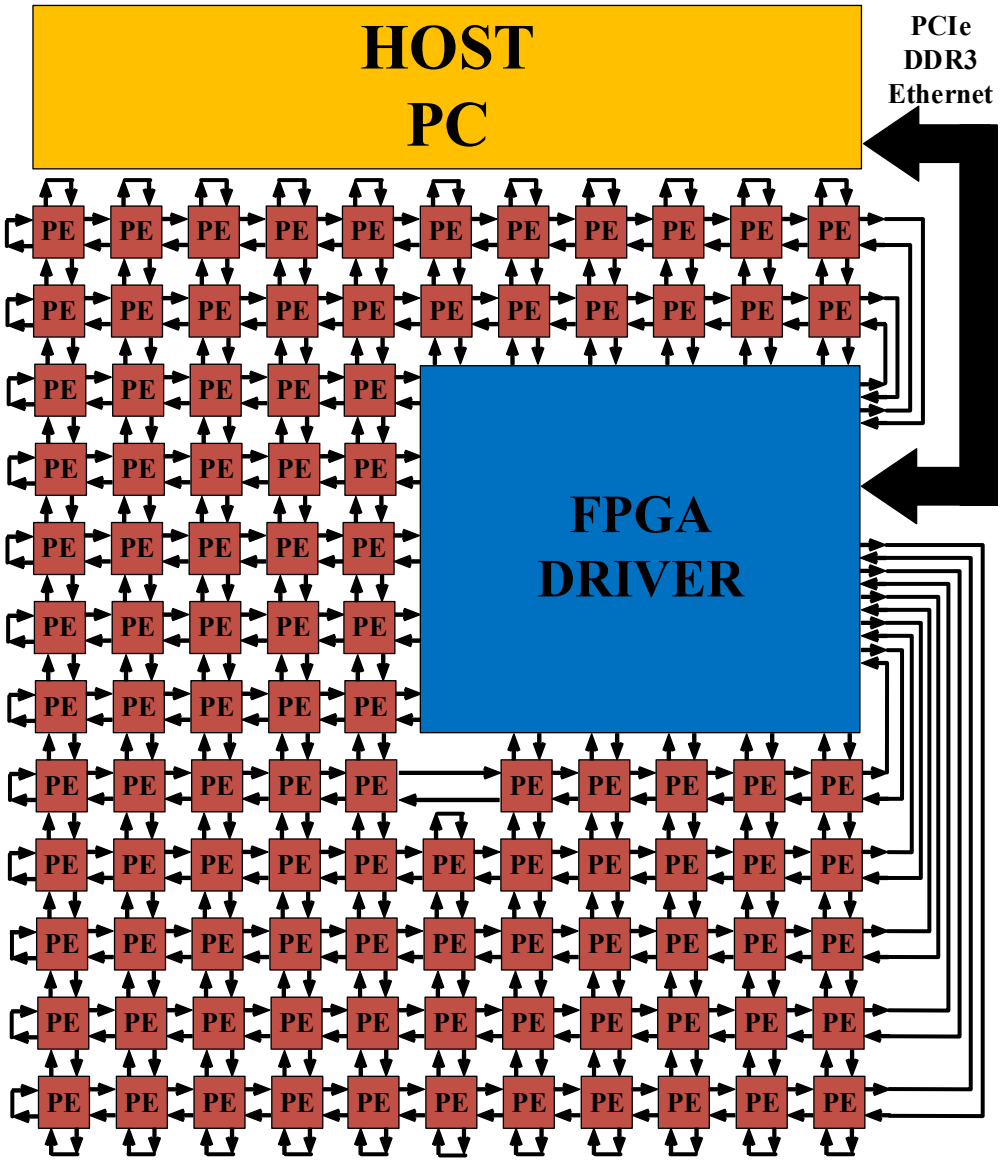
# Rapid Overlay Builder for Xilinx FPGAs




- **Stitching of PE tiles (reuse place & route effort)**
- **Dramatical improvement in CAD tool time**
- **FCCM 2015: "*Rapid Overlay Builder for Xilinx FPGAs*"**

# ToDo: PE Customization (Instr. Subsetting)

- Idea: use customized PEs (= cheaper and faster) for arithmetic or logic or …
- Problem: only after the CGRA mapping we know the PE type
- Solution: PE stitching through bitstream linking
- Also possible at runtime
- Manchester builds a bitstream parser for recent Xilinx FPGAs



# Dynamic FPGA Modules

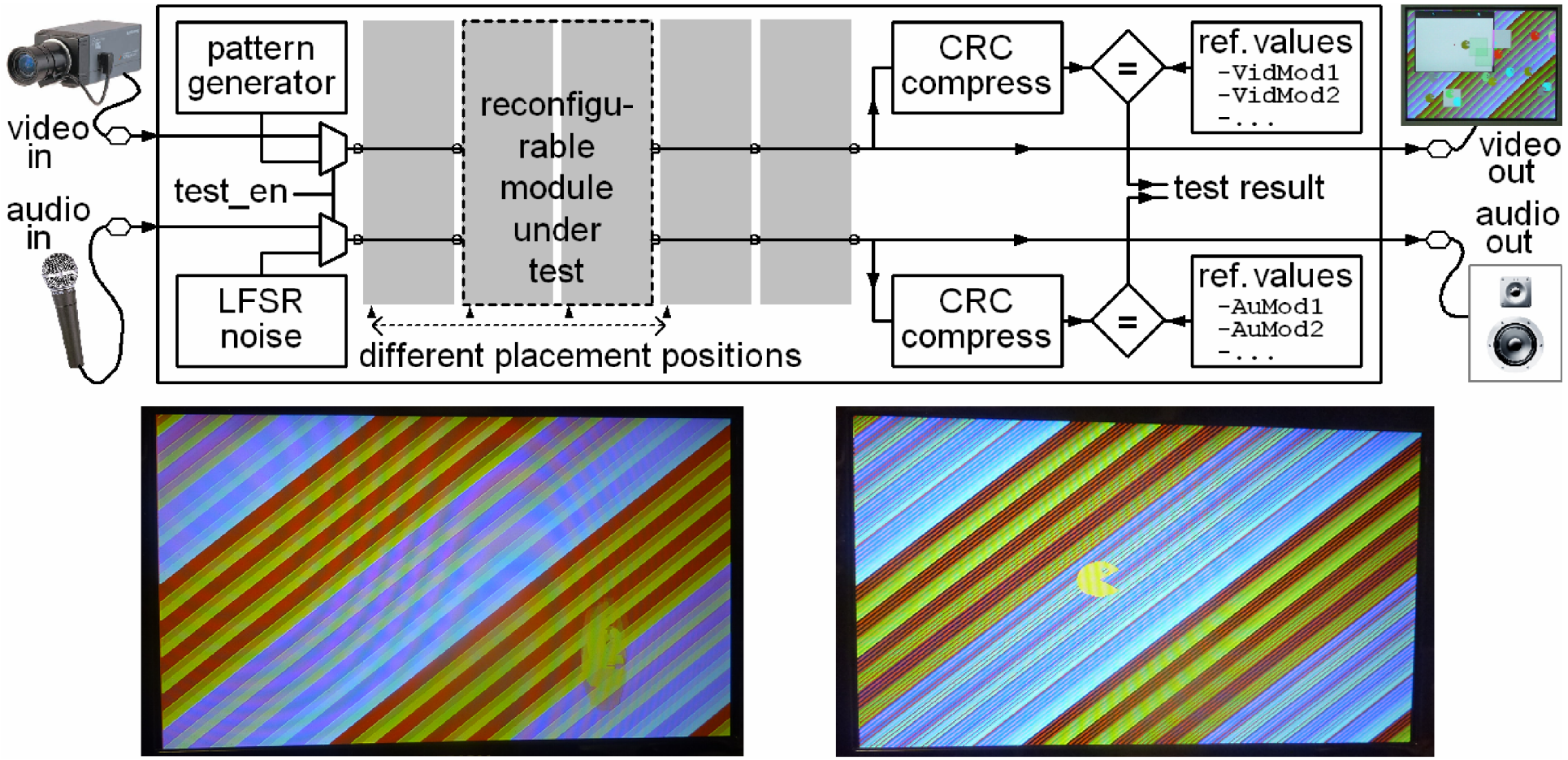
**Reasons:**

- **Maximize resource utilization** (minimize idle FPGA resources)
- **Programmability** (allows creating FPGA objects at runtime)

**FPGAs are more than just an ASIC substitute!**

- **Runtime adaptability** (deal with variant workloads and environmental changes)
- **Fault tolerance** (mask defect / imperfect resources)
- **Saving energy** (e.g., by moving computation to the data in a datacenter – one goal in the H2020 project **ECOSCALE**)

# FPGA Virtualization for Fault Tolerance

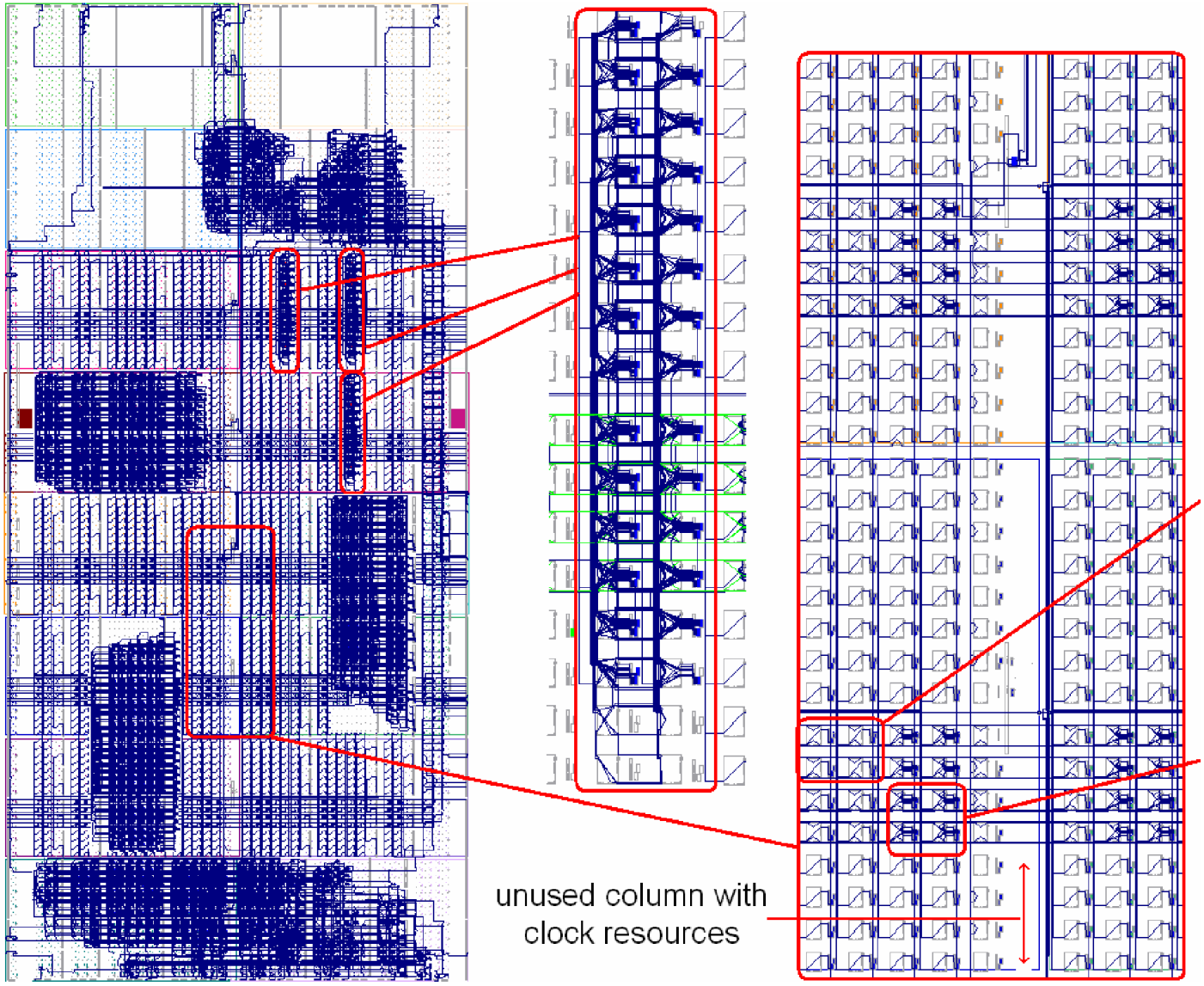



- **Use reconfiguration for detecting and masking faults**
- **TRETS 2014: "*Design Tools for Implementing Self-Aware and Fault-Tolerant Systems on FPGAs*"**

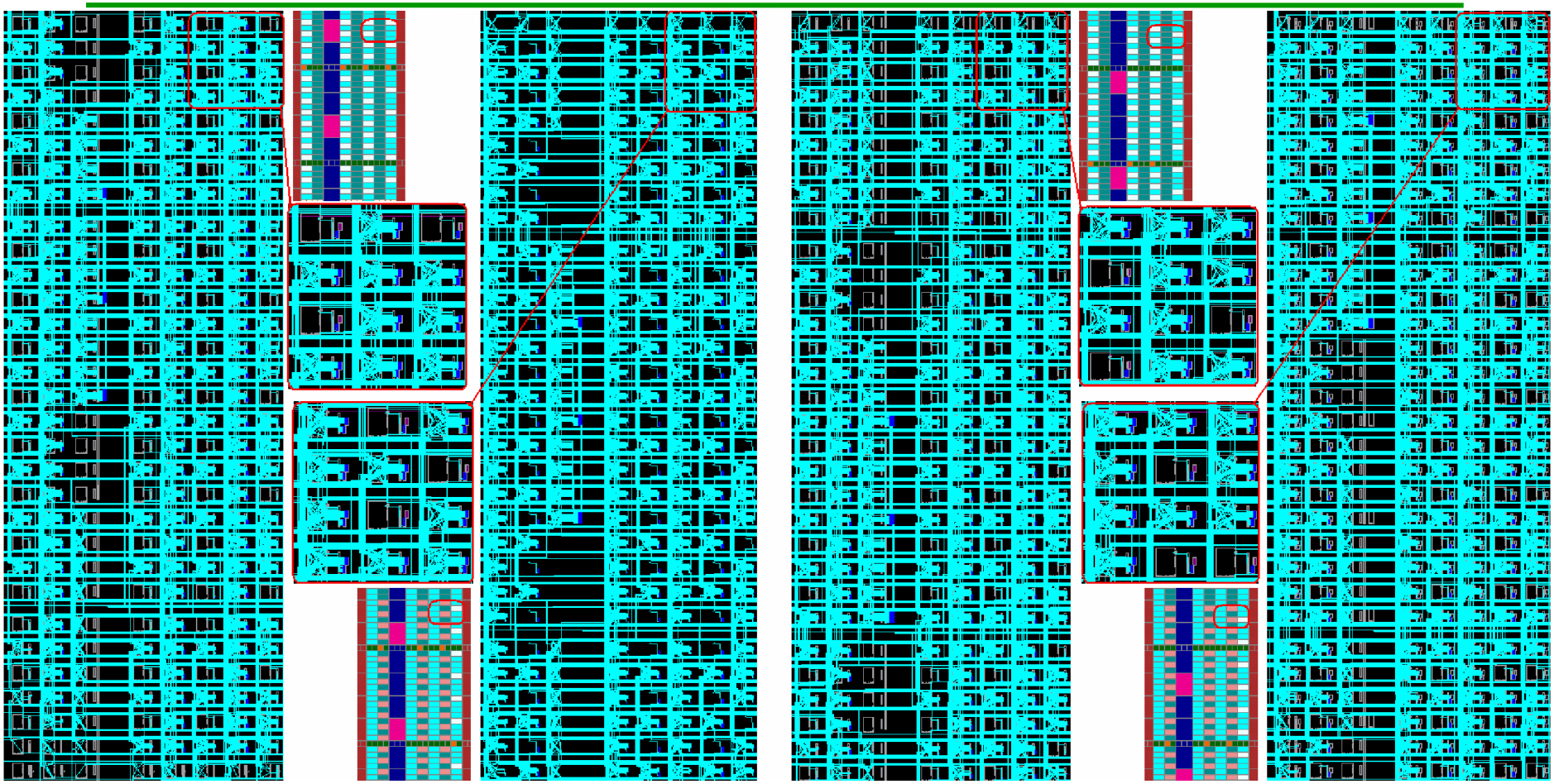
# FPGA Virtualization for Fault Tolerance

**Move modules around for masking faults**

**Implemented for video processing system (Spartan-6)**



# FPGA Virtualization for Fault Tolerance

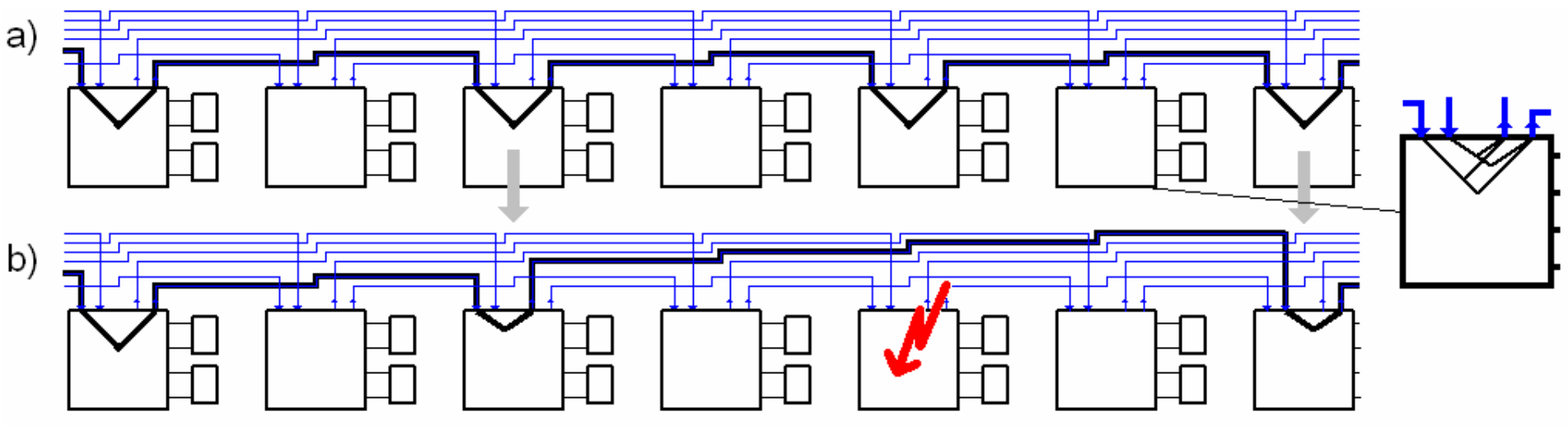


- **Can deal with defects in the static system**

**(memory controllers, control CPUs, reconfiguration controller, …)**

- **Idea: use different configurations that each leave different sets of the resources unused (LUTs, BRAMs, DSPs, DCMs, wires, …)**

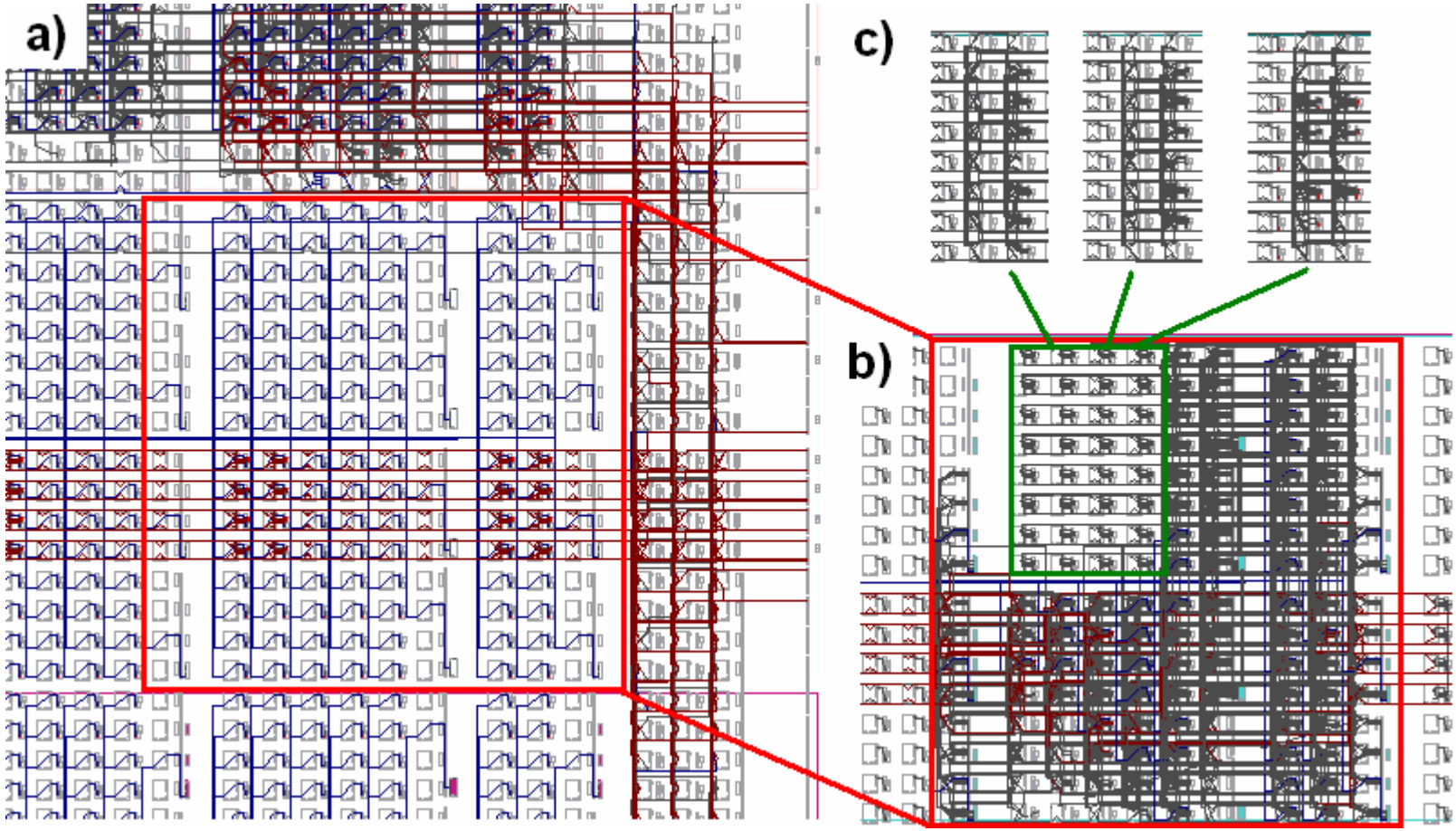
# FPGA Virtualization for Fault Tolerance




- **Can deal with defects in the communication infrastructure (the wires that connect the different reconfigurable modules)**
- **Idea: precomputed bypass paths (rerouted in the runtime system)**

- **Very holistic solution (probably the most complete prototype)**

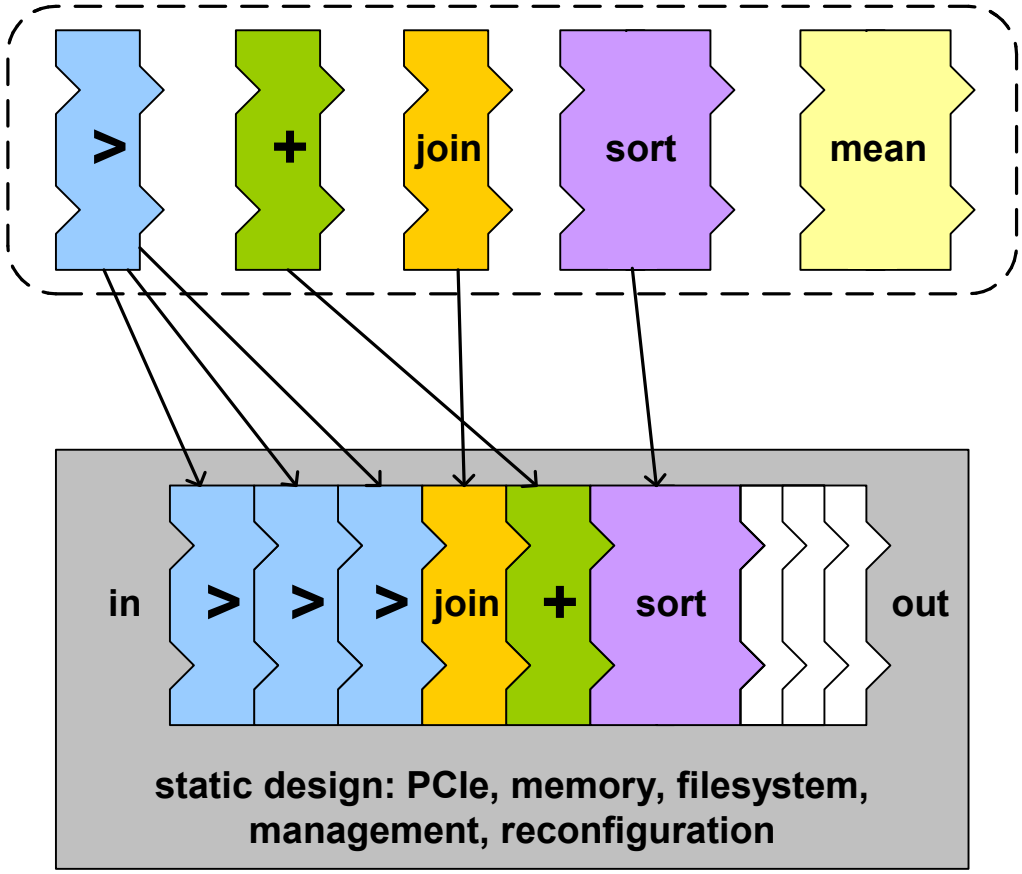
# Hierarchical Reconfiguration of FPGAs




- **Partial modules inside partial modules**
- **Demonstrated for custom instruction set extensions**
- **FPL 2014: "*Hierarchical Reconfiguration of FPGAs*"**

# Database Accelerator

- **Build library with SQL operators**
- **Compose graph at run-time**









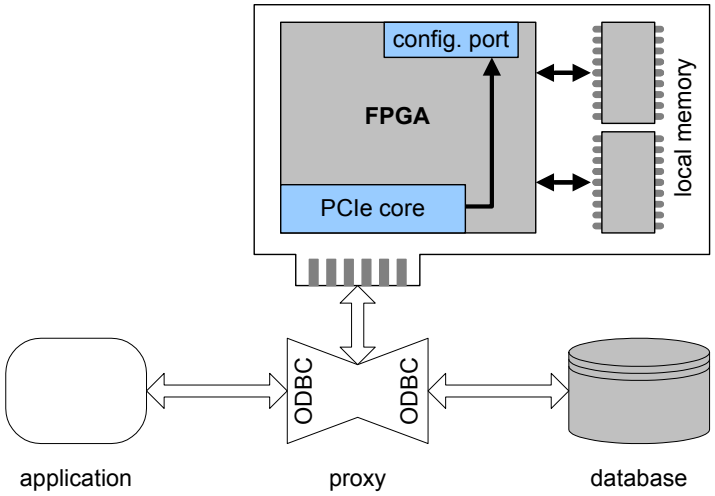








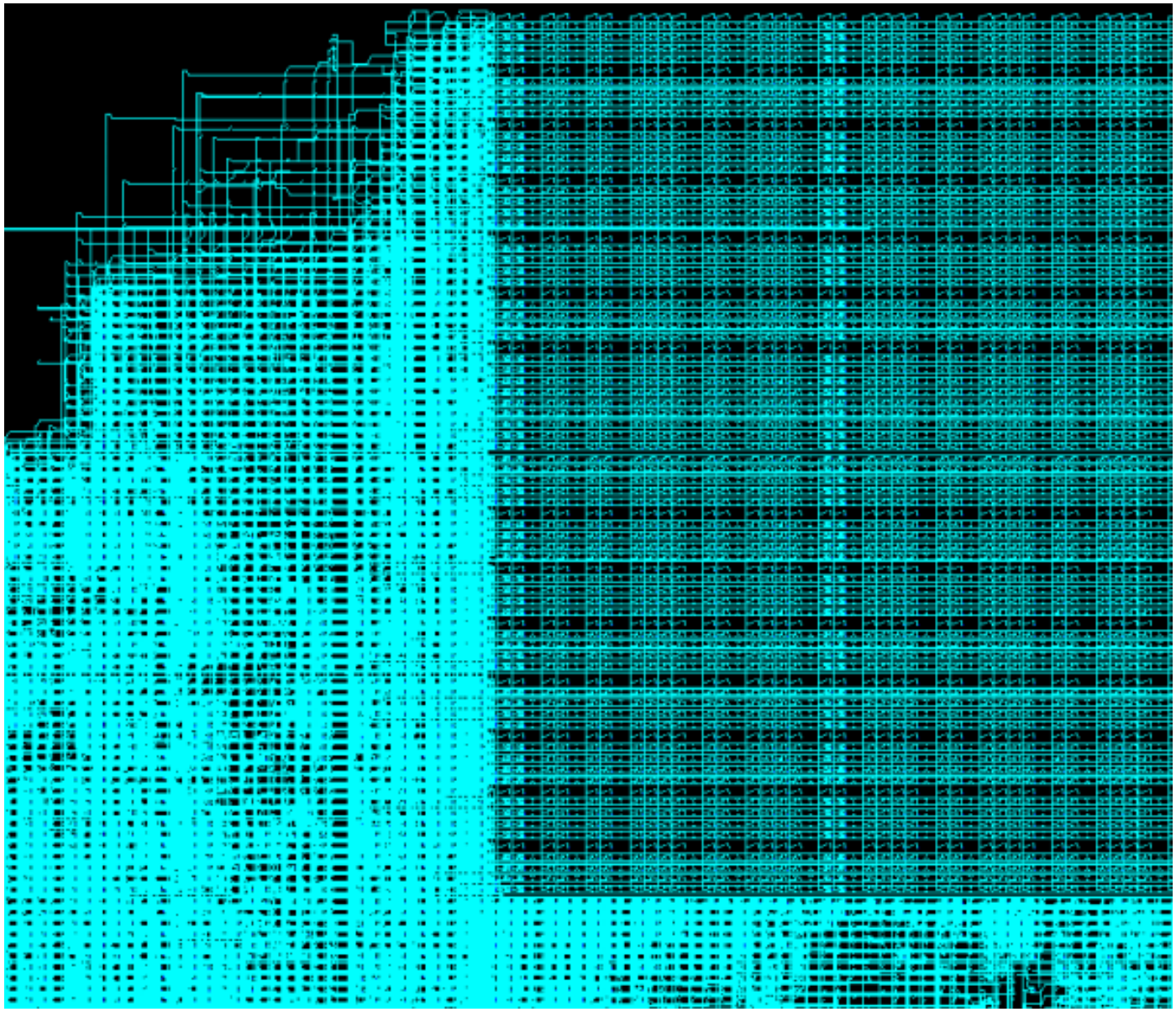



**Design goals:**

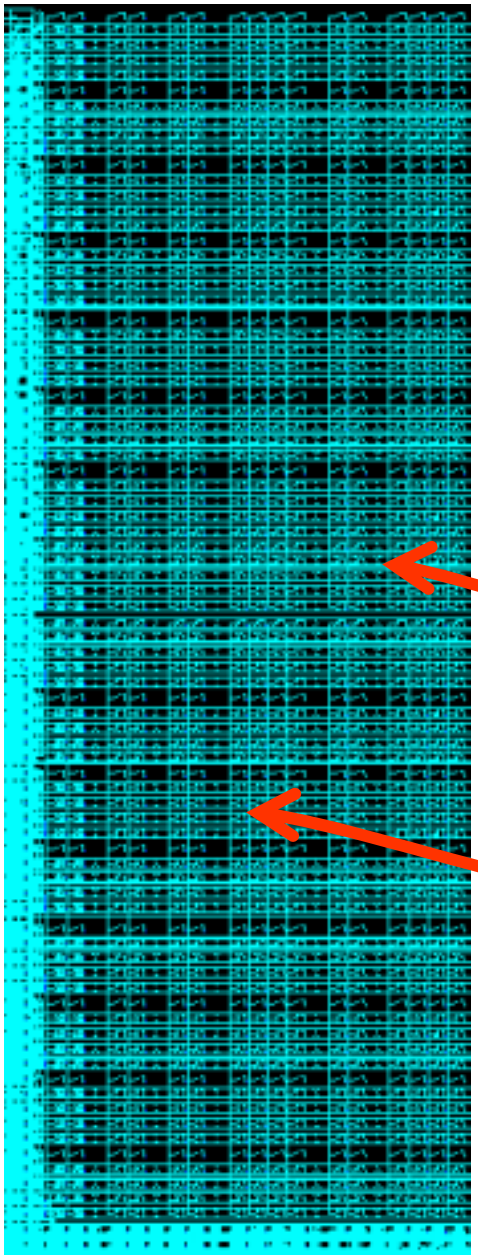
- **512 bit datapath**
- **300 MHz (Virtex-6)**
- **Tables are stored in 24 GB on board RAM**
- **Prototype on MAX3**

# FPGA Database Accelerator



- Reconfigurable region
- Regularly routed
- 16 x 32-bit (512-bit total)
- @ 300 MHz
- ~20 GB/s throughput
- Demosystem on Maxeler MAX3

# FPGA Database Accelerator



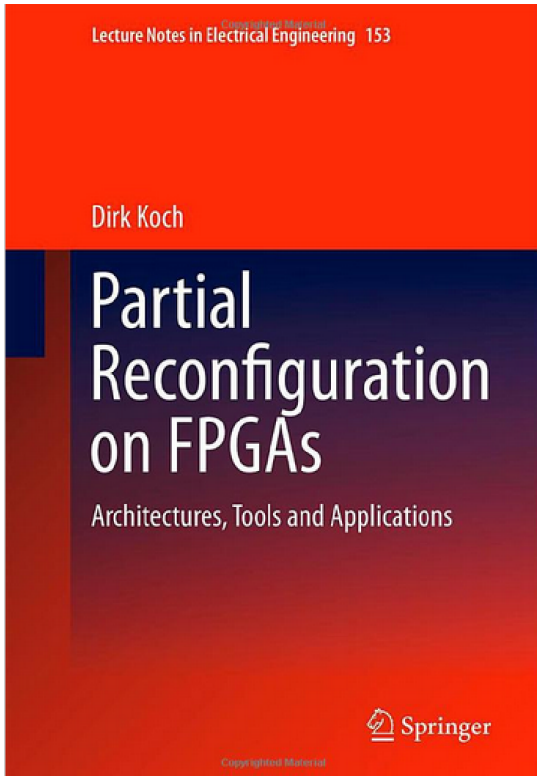
- Build library with SQL operators
- Compose steam processing machine at runtime

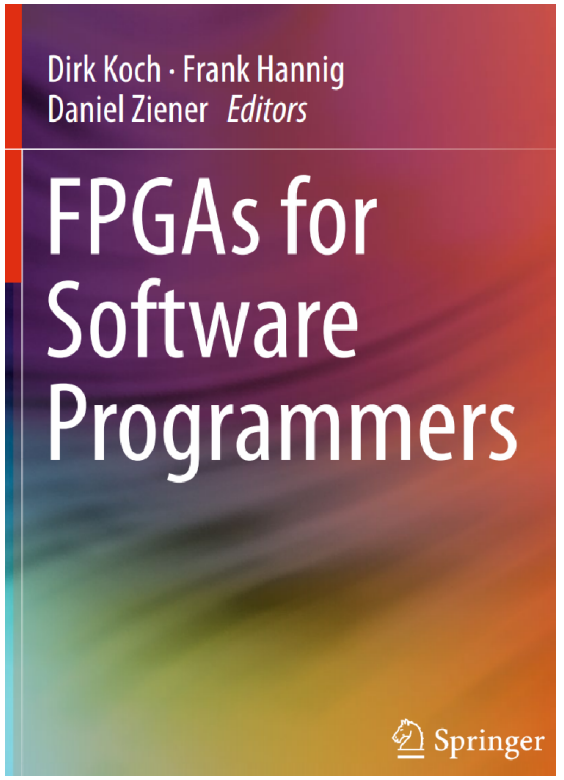
# Contributors

- **Grigore Nicolae Bogdan**
  Query optimization, resource management, virtualization
- **Athanasios Stratikopoulos**
  FPGA virtualization
- **Malte Vesper**
  PCIe/SSD stream processing infrastructure, applications
- **Raul Garcia**
  Reconfigurable instruction set extensions
- **Christian Beckhoff** (hobbyist)
  GoAhead support (tool for building reconfigurable systems)
- **Edson Horta**
  HLS to PR-module tooling
- **Khoa Pham**
  Bitstream generation and runtime reconfiguration drivers

# Advertisement

- **Books**




- **Open positions** (for PhD contact me ASAP! dirk.koch@manchester.ac.uk)
  **Reader / Senior Lecturer / Lecturer in Computer Science:**
  `https://www.jobs.manchester.ac.uk/displayjob.aspx?jobid=11610`
- **Collaborations** (particularly welcome is everything related to runtime reconfiguration)