Flexible Data-Flow Processing

Christian Brugger

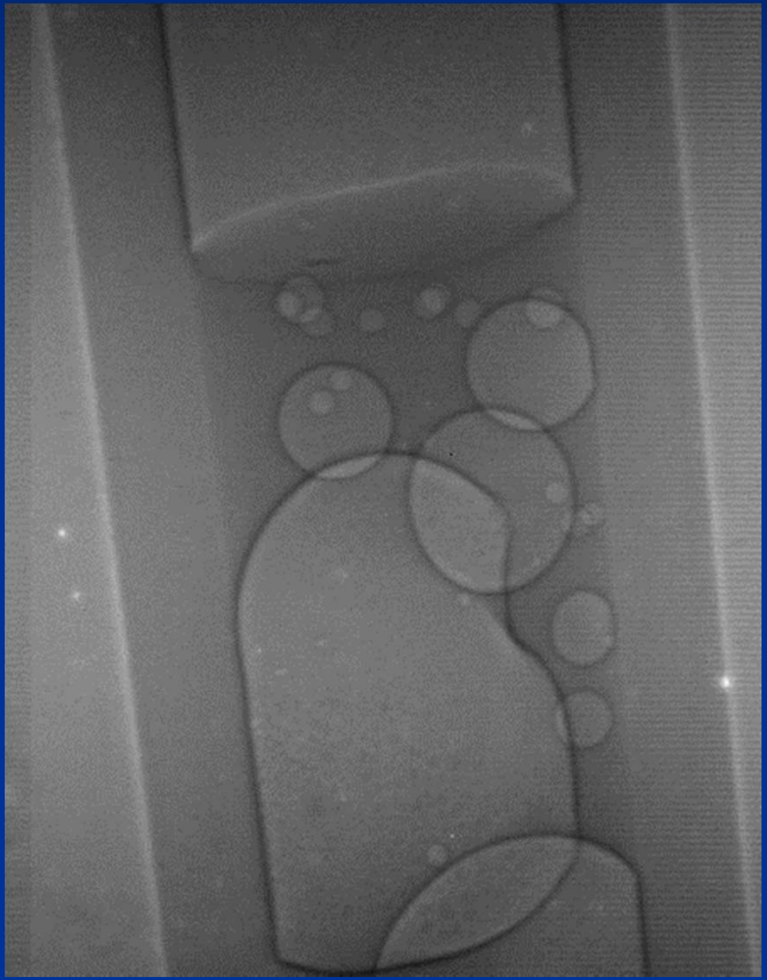
www.kit.edu

01.09.2011

Outline

- Introduction
- System Vision
- Related Work
- Methodology
- Requirements
- Architecture
- Current State

Introduction



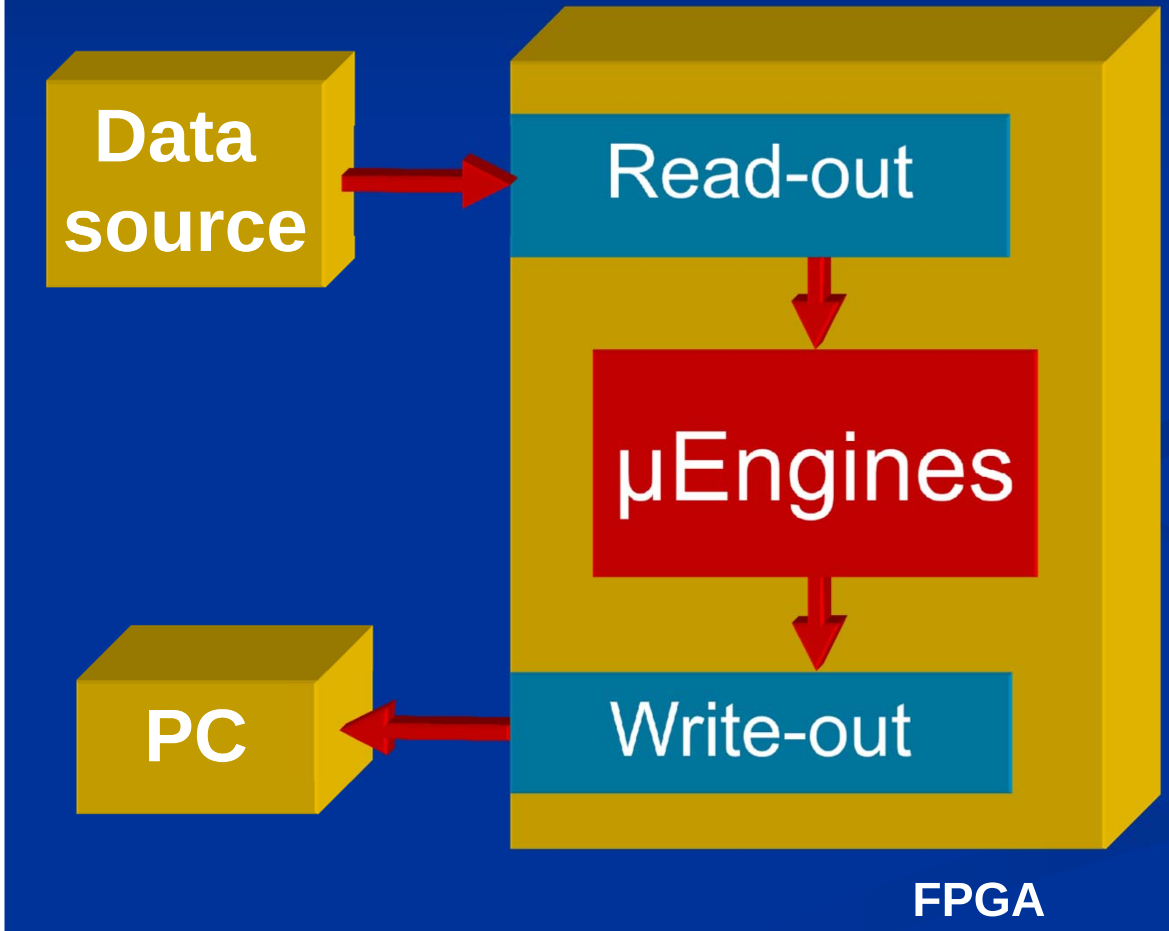
- Projects:
  - UFO – visible light
  - MEDIPIX – x-ray
  - USCT – ultra sound
- High bandwidth:
  - Streaming:
  - 1 GB/s .. 100 GB/s
  - Preprocessing
  - Analysis

Motivation

- UFO
  - Short setup time & one shot experiments

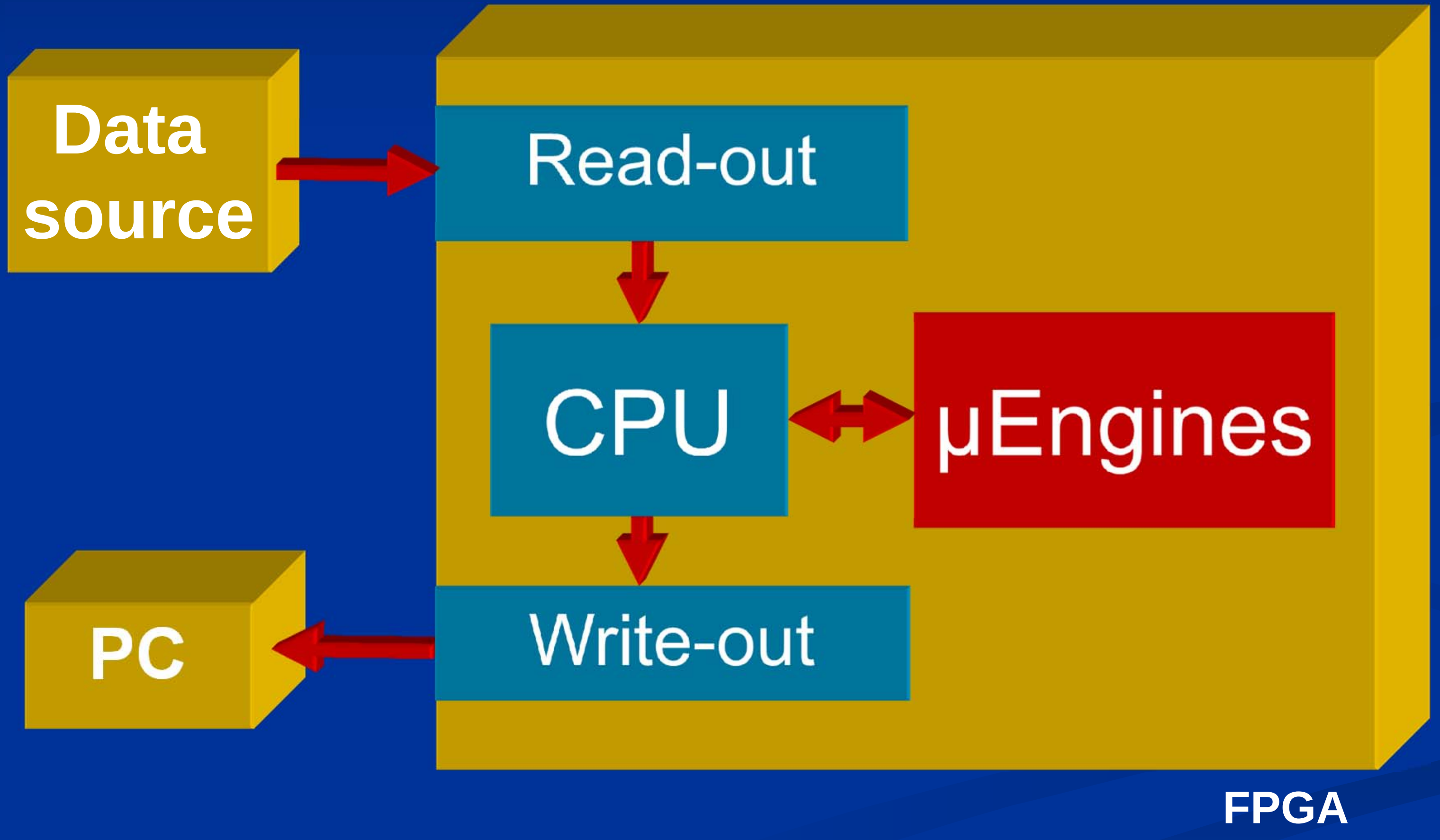
    “Beam-time is expensive”
- Goal: “Quick & easy” framework for stream processing (> 1 GB/s)
  - Focus on algorithm (not HDL-design)
  - Reusable across projects
  - Customizable / Extensible
    - “Easy things simple, difficult things possible”

System Vision



- Stream processing:
  - “µEngines”
- Software:
  - Compiler, Assembler, Simulator

System Vision



Related Work

- FPGA – IP Core
  - Matlab/Simulink – Image/Video Processing toolbox [1]
  - Inhouse core
- CPU – Processing – 12 cores
  - OpenCV library C++ [2]
- GPU – Processing
  - NVIDIA Performance Primitives [3]

Methodology

- Requirements
- Infrastructure
  - Risk assessment, prototyping
  - Design space exploration / automation
- Application driven design
- Design iterations: simple → complex

Methodology

- Iteration 1: Simple Filter
- Iteration 2: Image Analysis
- Iteration 3: Compression

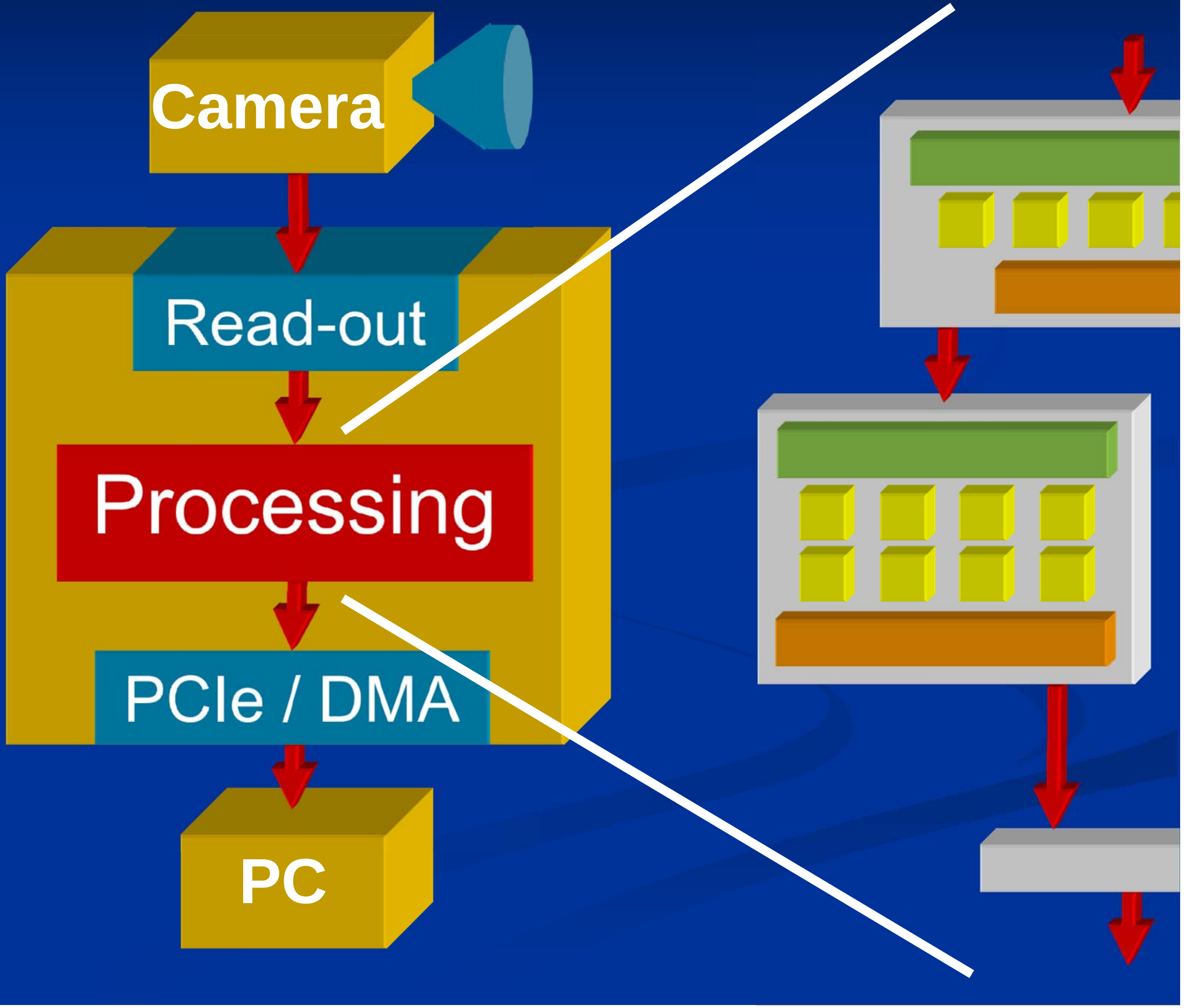
Each iteration:

- Design Space Exploration
- Evaluation
  - IP core vs. our architecture
  - Suitablability

High-Level Requirements

- Real-time image filtering
- Support: “Quick turnaround & prototyping”
- Reusable across various projects
  - Extensible
  - Configurable

Design: Architecture



HW Components:

- Data Stream Mixer
  - Segmentation, differential pictures
- Nano-Engine - Types
  - Few specialized instructions e.g. MAC, SAD
  - VLIW – very long instruction word
- Accumulator
  - Reassemble data stream

Design: Flow

- Algorithmic Description
  - Assembler or C-like
  - Graph (Simulink)
- Compiler
  - Data flow → mixer / routing
  - Computations → Nano-Engines
- Bitstream
  - Pre-synthesized + dynamic reconfiguration

Current status

- Binomial Filter
  - VHDL
  - Bluespec
- Design space exploration: Cluster Job System
  - Automation
  - Parallelization

Binomial Filter

- Bluespec vs. HDL: both 300 MHz

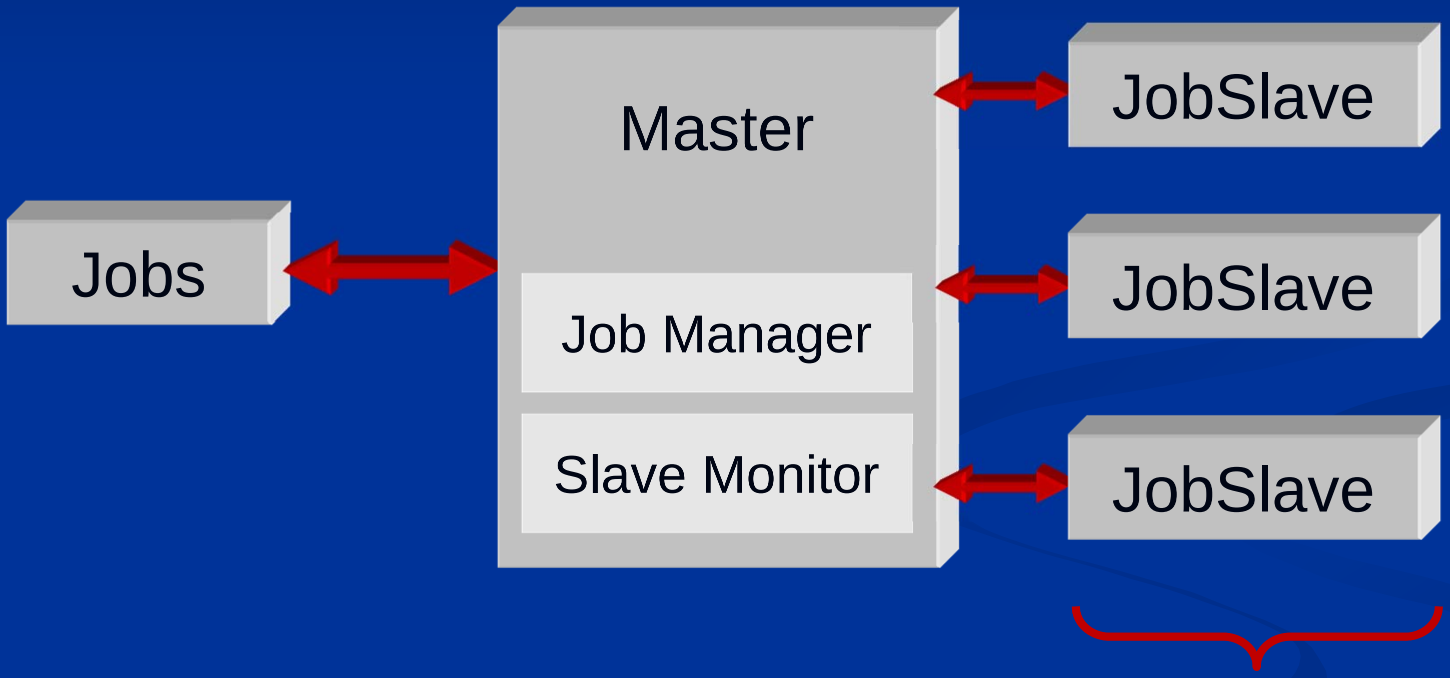
```
typedef UInt#(10) Pixel;

interface Filter_ifc;
    interface Put#(Pixel) din;
    interface Get#(Pixel) dout;
Endinterface
```

```
let row_filter <- mkFilterByRow;
let col_filter <- mkFilterByCol;

mkConnection(row_filter.dout, col_filter.din);
```

Cluster Job System



Jobs

Master

Job Manager

Slave Monitor

JobSlave

JobSlave

JobSlave

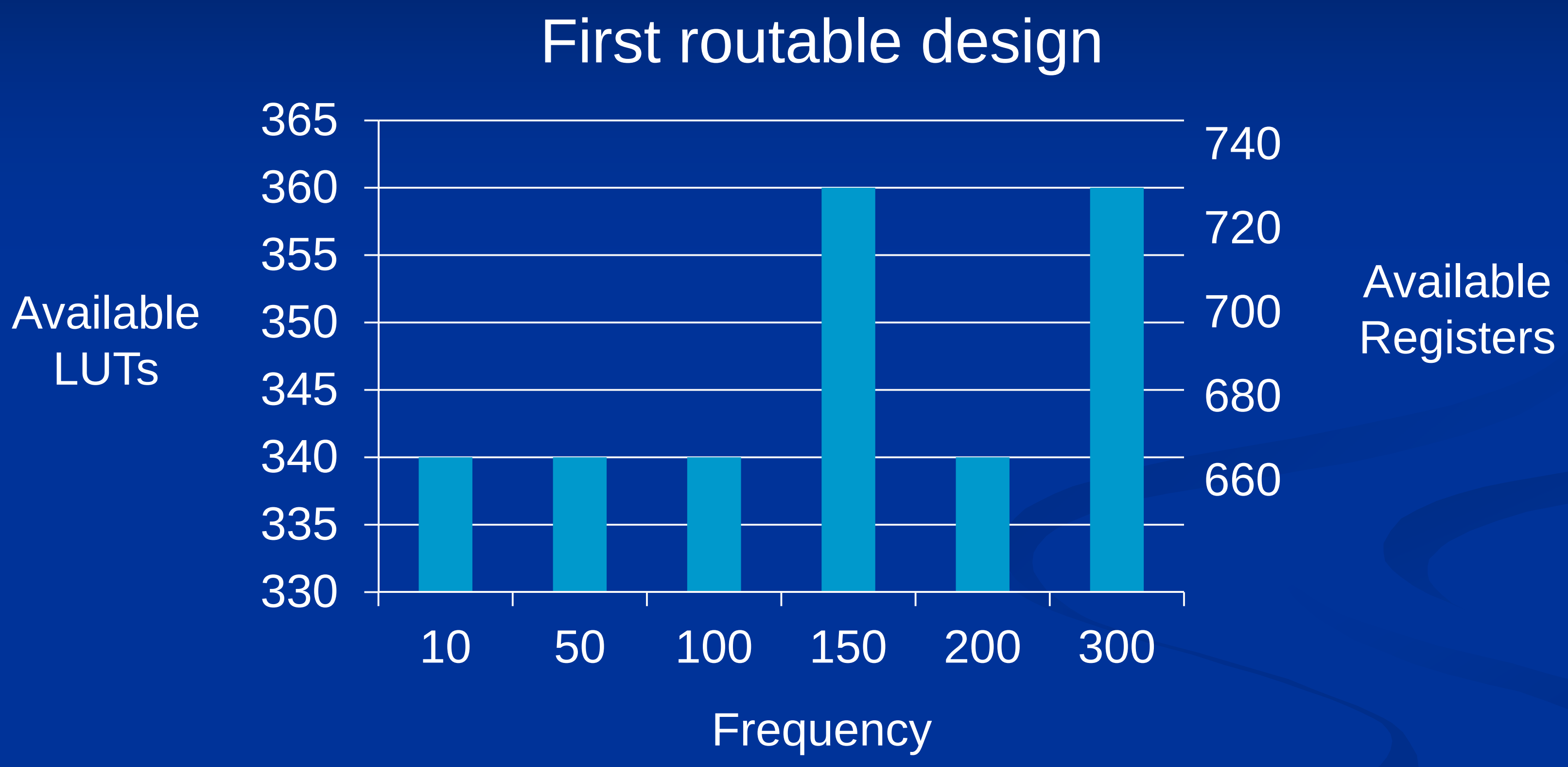
40 slaves
24 cores each

Demo

First results

- Design space exploration:
  - Maximum frequency / number of Slices
  - Power
  - Virtex 6 Quiescent Power:
    - 2.2 W
    - +1.2 W: GTX Transceiver [4]

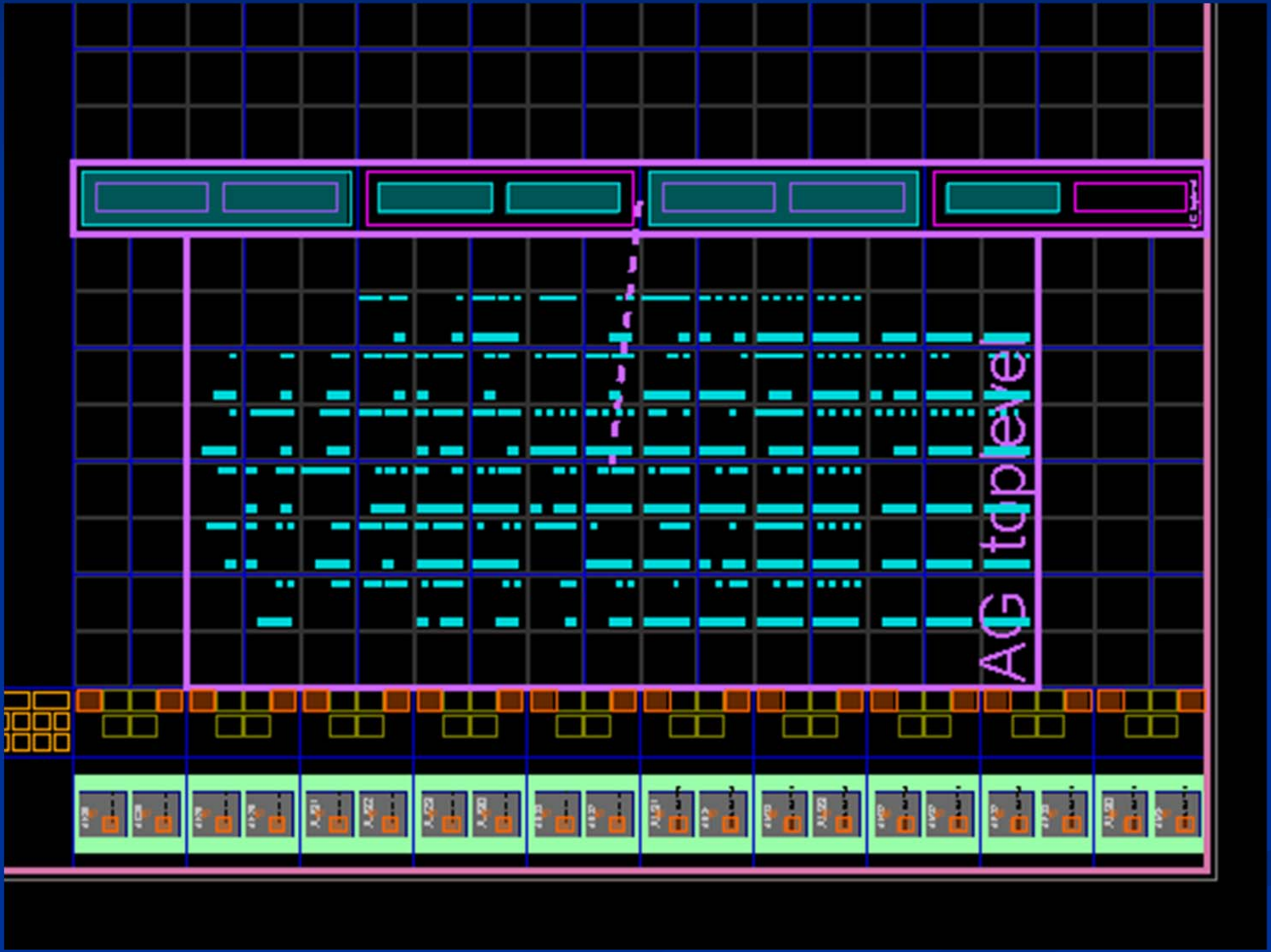
Design Space Exploration



PAR Report

Number of Slice Registers: 362
Number of Slice LUTs: 291

Utilization



References

- [1] Matlab Simulink, http://www.mathworks.de/products/simulink/index.html
- [2] http://opencv.willowgarage.com/wiki/
- [3] NVIDIA Performance Primitives, http://developer.nvidia.com/npp
- [4] http://www.xilinx.com/support/answers/35055.htm