

# Hardware-Synthesis from UML-Models

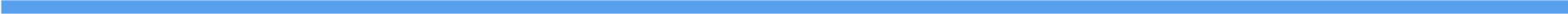
Thomas Beierlein<sup>1</sup>, **Dominik Fröhlich**<sup>1,2</sup>, Bernd Steinbach<sup>2</sup>

<sup>1</sup> University of Applied Sciences Mittweida

<sup>2</sup> Freiberg University of Mining and Technology

6st International Workshop on Boolean Problems

23.09.2004 Freiberg



# Agenda

- Introduction
  - System-Level Design with UML
  - MOCCA – Environment
  - Example: TVL-Processor
  - Experimental Results
  - Conclusions
-

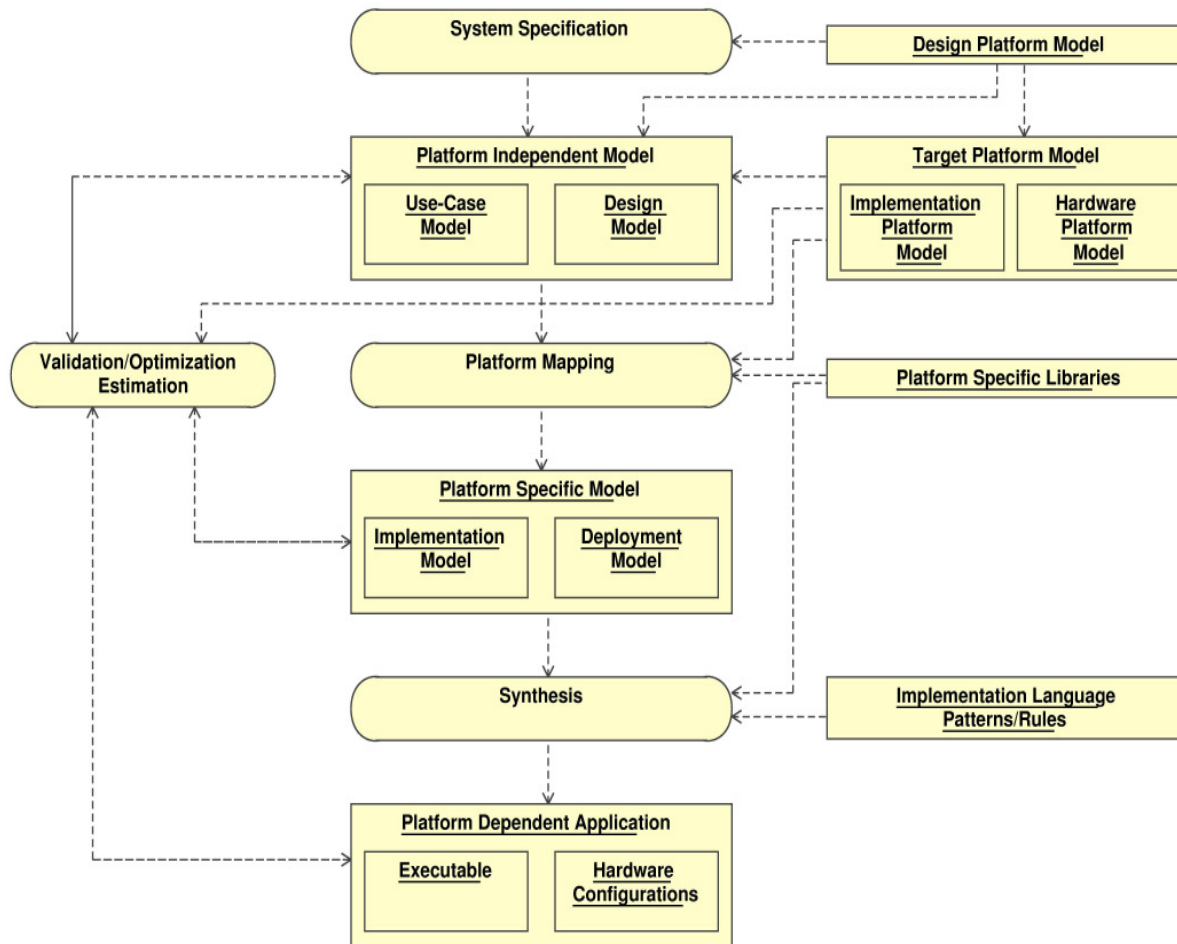
# Introduction

- Goal: Approach for the development of computationally intensive applications
- Current solutions often with unsuitable resources
  - Typical Boolean problems allow for highly parallel solutions
  - Typical solutions use highly sequential resources
  - Standard platform available
- Suitable resources are currently hard to use
  - Development of ASICs is complex and time-consuming
  - Unsuitable development methodologies and languages
  - Huge variety of different platforms

# Development Methodology (1)

- Specification, Design, and Implementation
  - MOC: Objects communicating through structured messages
  - Language: UML and Action Language
  - Method: Model-Driven Architecture, Co-Design, Platform-Based Design
  - Tool: MOCCA – Model Compiler for reConfigurable Architectures
- Advantages
  - Exploit expressiveness and generality of UML 2.0
  - Platform Independence
  - Entirely OO (Software and Hardware!)
  - Visualization and Manipulation Support

# Development Methodology (2)



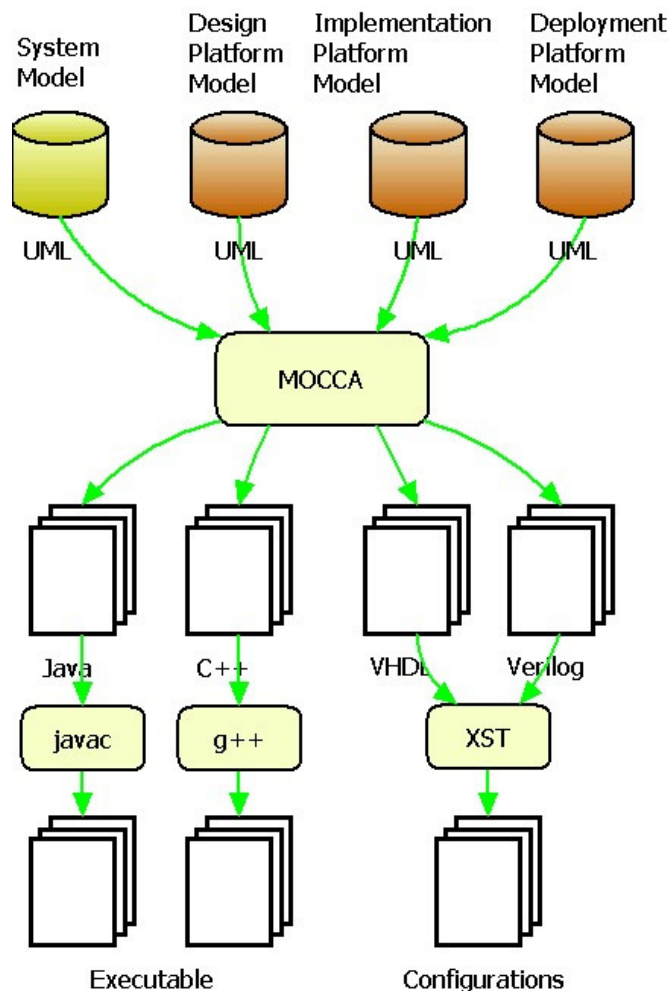
- Goals:

- Comprehensibility
- Insulate Design from Implementation
- Maximize Re-Use

- Key Concepts:

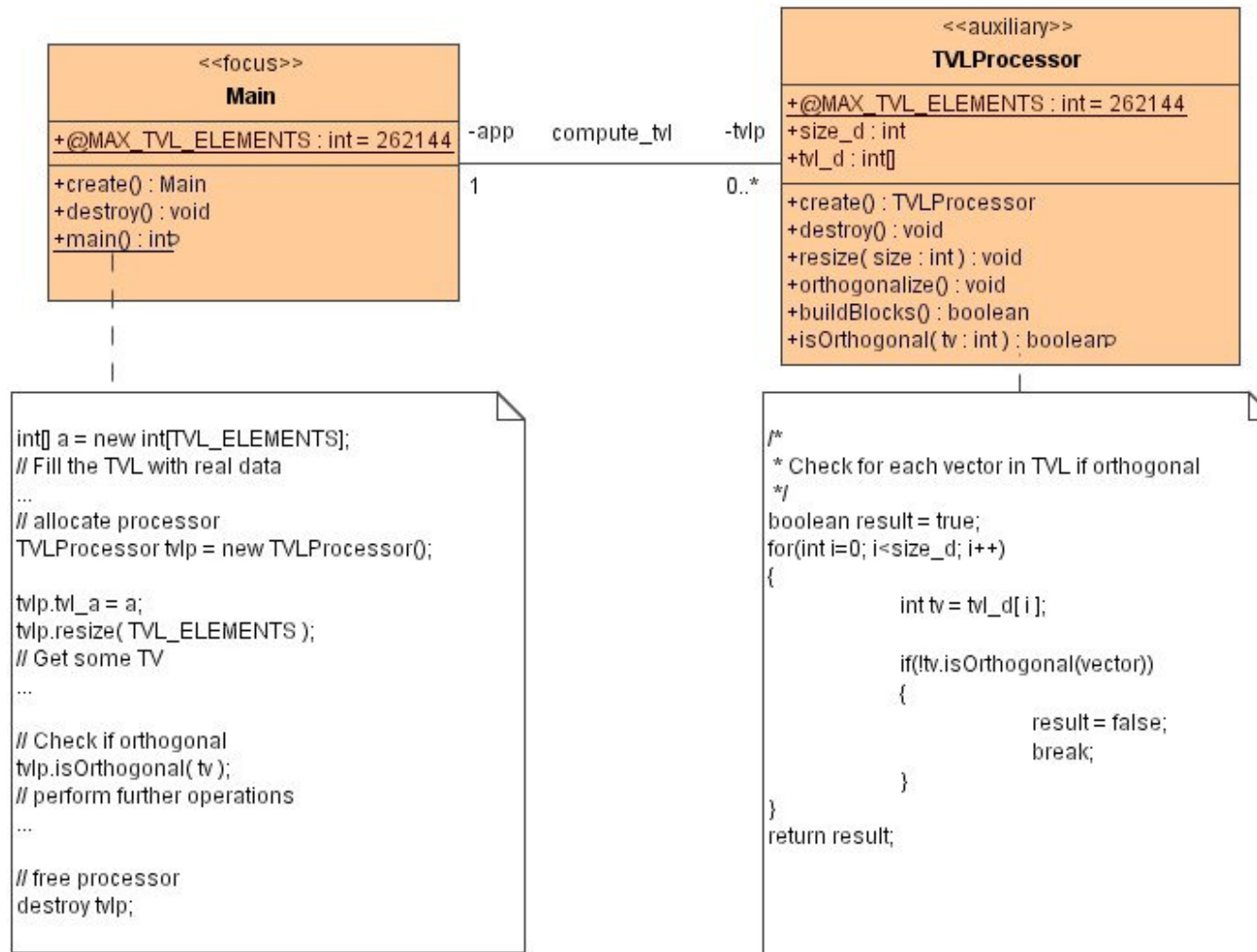
- Platform Based
- Platform Models
- Application Dependent Models
- All Models specified with UML + AL
- OO-Paradigm is preserved down to hardware
- Automation: Platform Mapping → Synthesis

# MOCCA – MModel Compiler for reConfigurable Architectures

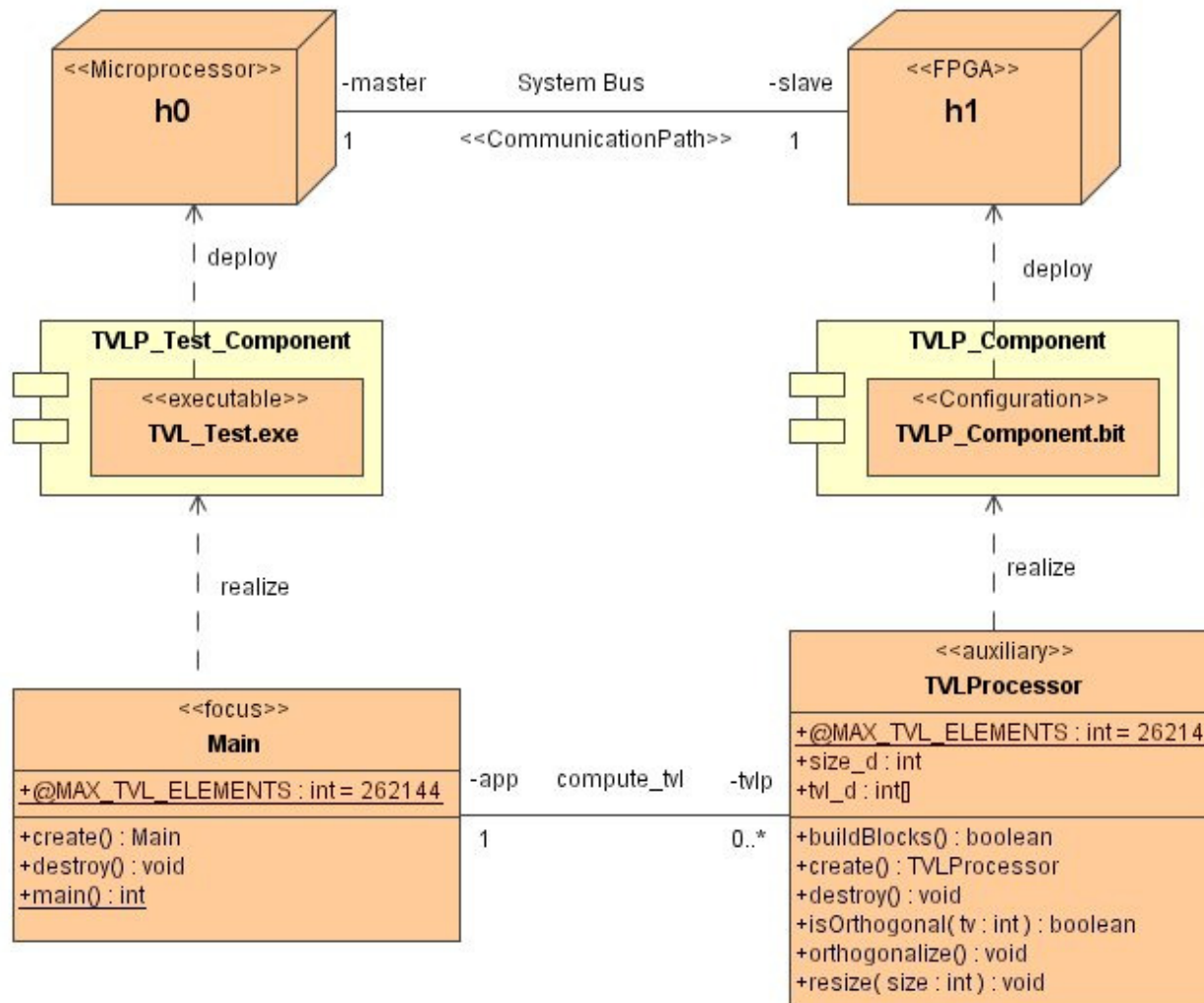


- UML 2.0 Model Compiler
- Automated transition from platform independent application design to platform specific implementation
- Automated Design Partitioning
  - Support for partially or completely manually specified partitioning
  - Estimation of Execution and Implementation characteristics
    - Type and Message Analysis
    - Scheduling, Allocation, Binding
    - Branch Prediction
    - Automated Synthesis of Platform Dependent Implementation
- RTR-Manager and Broker

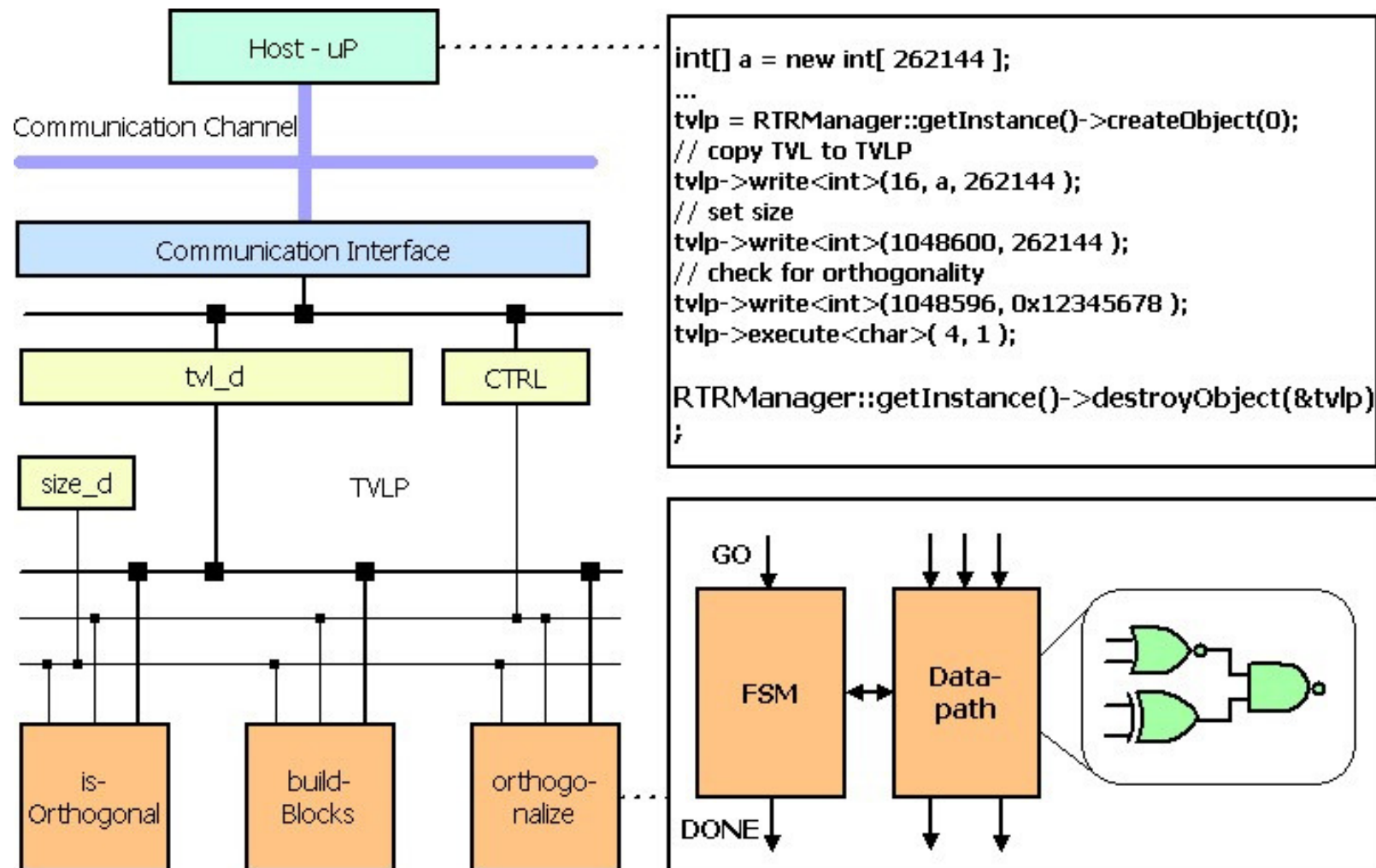
# Example: TVL-Processor (1)



# Example: TVL-Processor (2)



# Example: TVL-Processor (3) - Implementation



# Experimental Results (1)

- Implementation Characteristics:
  - Block-Building + Orthogonality-Test + Orthogonalization + CI
  - Area:
    - Logic: ~120K Gates (Xilinx Virtex-II)
    - RAM: 2MByte
  - Clock-Rates:
    - Logic: 100 MHz (limited by PCB and local RAM)
    - PCI: 33 MHz
  - Power:
    - ~30W (Xilinx Virtex-II 3000)
  - UML-Modeling-Time: ~4h
  - Implementation-Time (Compilation+Synthesis): <15min
- Execution Characteristics:
  - Orthogonality-Test
  - Block-Building
  - (Orthogonalization)

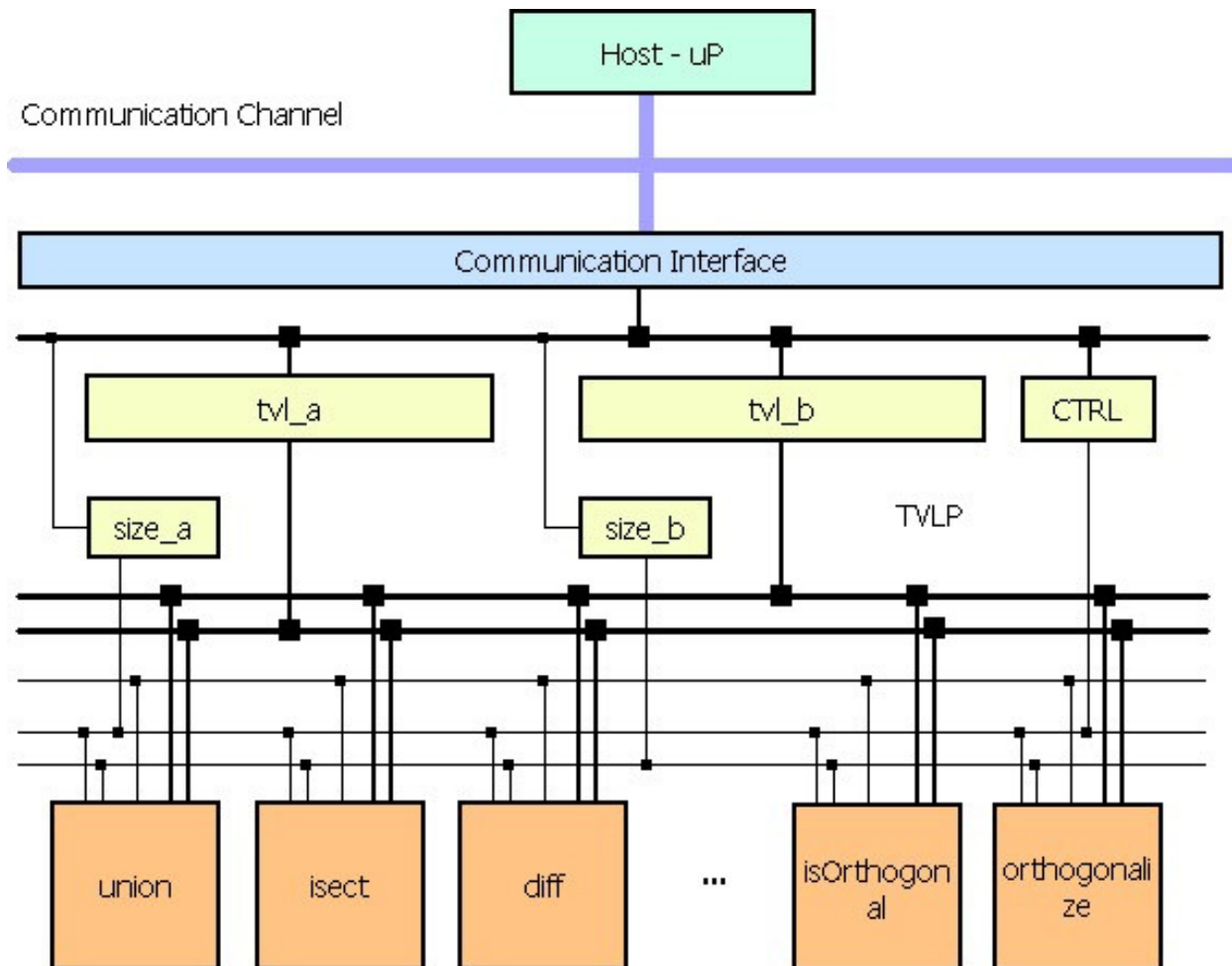
# Experimental Results (2)

- Experimental-Setup:
  - A: Xilinx V-II 3000 (-6), 6 MByte local RAM, 100 MHz
  - B: Intel Pentium-4 2.4 GHz, 1 GB RAM
- Communication Host → FPGA:
  - 25ms/MByte
- Orthogonality-Test:
  - ~200k TV, worst-case
  - TV-Encoding: TV-Sub-Atoms in neighbored bits
    - A: ~0.4s
    - B: ~0.7s
  - TV-Encoding (XBoole): Distributed TV-Sub-Atoms
    - A: ~0.4s
    - B: ~0.04s

# Experimental Results (3)

- Block-Building
  - 10k TV, average case
  - A: ~8s
  - B: ~1..8s (13s w/o fail-fast)
- Problems:
  - Lower Clock-Rate 100MHz vs. 2.4 GHz
  - Slow Memory Interface: 4 Wait-States!
  - Unoptimized FSM + Data-Path + Operators
- Advantages:
  - Less Power-Consumption
  - Scalable (only 4% of logic resources and 16% of memory are used)

# Example: TVL-Processor (4) - Extensions



- ↑Atoms/TV
- ↑Operations
- ↑TVLs/Processor
- ↑Processors/Device
  - Space
  - Time (RTR)
- Supplement with uP
- Host-uP + TVLP

# Conclusions

- We presented an approach to the system-level development of ASICs
  - Based on UML 2.0
  - Development Methodology
  - Development Environment
- Example of an ASIC for the processing of TVLs
  - Scalable (#/Width of TVs, TV-Operations, #Processors)
  - Relatively Efficient

# Thank you!

