High-Level Design for FPGA-based Multiprocessor Accelerators

Diana Göhringer¹, Matthias Birk², Michael Hübner¹, Jürgen Becker²

¹Fraunhofer-Institute for Optronics, System Technologies and Image Exploitation IOSB, Germany
²Karlsruhe Institute of Technology (KIT), Germany,
diana.goehringer@iosb.fraunhofer.de, (matthias.birk, michael.huebner, becker)@kit.edu

1. Introduction: FPGA versus GPU Programming

- **Application Description**
  - VHDL/Verilog
    - Optimized design, full control over the design
    - Low-level, difficult to debug, seldom used by application engineers
  - Graphical Tools (e.g., MATLAB HDL-Coder, System Generator)
    - High-level, easy to use, can use Drag and Drop
    - Restricted to a specific IP library
  - C/C++ (e.g., Cutoff, Autoptil)
    - Fast design space exploration, automated generation of IP blocks
    - Special C/C++ coding style required for good results
      - Generates only IP blocks not full FPGA designs

- **Advantages**
  - VHDL/Verilog
    - Optimized design, full control over the design
  - Graphical Tools
    - High-level, easy to use
  - C/C++
    - Fast design space exploration

- **Disadvantages**
  - VHDL/Verilog
    - Low-level, difficult to debug
  - Graphical Tools
    - Restricted to a specific IP library
  - C/C++
    - Requires special coding style


2. Design Methodology for FPGA-based Multiprocessor Accelerators

- **Phase 1: Task Allocation**
  - **Analysis**
    - Profiling
    - Tracing
    - Communication Analysis
  - **Iteration**
    - SW/SW Partitioning
      - HW/SW Partitioning
      - HW/SW Synthesis
      - Compiler

- **Phase 2: SW/SW Partitioning**
  - **Analysis**
    - SW code
    - HW code
    - C/FPGA Compiler
  - **Compilation**
    - RAMPS/CMP compiler

- **Phase 3: HW/SW Integration**
  - **Integration**
    - XML layout & Task graph
    - Full Bitstream
    - Partial Bitstream
    - SW SW Executable

- **Key**
  - Commercial tools
  - Manual tools
  - Custom tools
  - Input/Output tools

3. Application Exploration

- **a) 3D Ultrasound computer tomography**
- **b) Object Recognition**
  - Input Image
  - Histogram Equalization
  - Squared Normalized Cross Correlation
  - Hotspot Detector
  - Output Image

4. Results

- **a) Performance 3D USCT**
  - Execution time per A-Scan
  - CPU: 315us, GPU: 315us, RAMPS/C: 315us

- **b) Performance Object Recognition**
  - Execution time per frame
  - CPU: 315us, GPU: 315us, RAMPS/C: 315us

- **c) Energy Efficiency 3D USCT**
  - Energy efficiency
  - CPU: 315us, GPU: 315us, RAMPS/C: 315us

- **d) Energy Efficiency Object Recognition**
  - Energy efficiency
  - CPU: 315us, GPU: 315us, RAMPS/C: 315us

5. References
