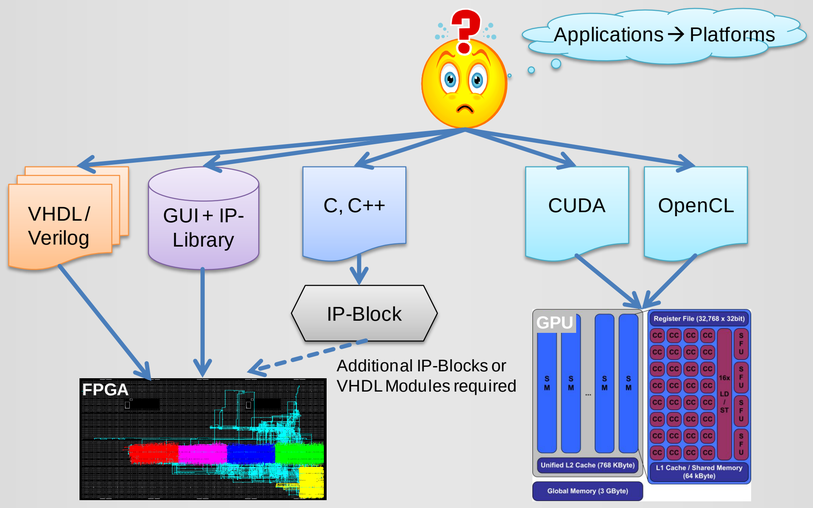
High-Level Design for FPGA-based Multiprocessor Accelerators

Diana Göhringer[1], Matthias Birk[2], Michael Hübner[2] , Jürgen Becker[2]

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

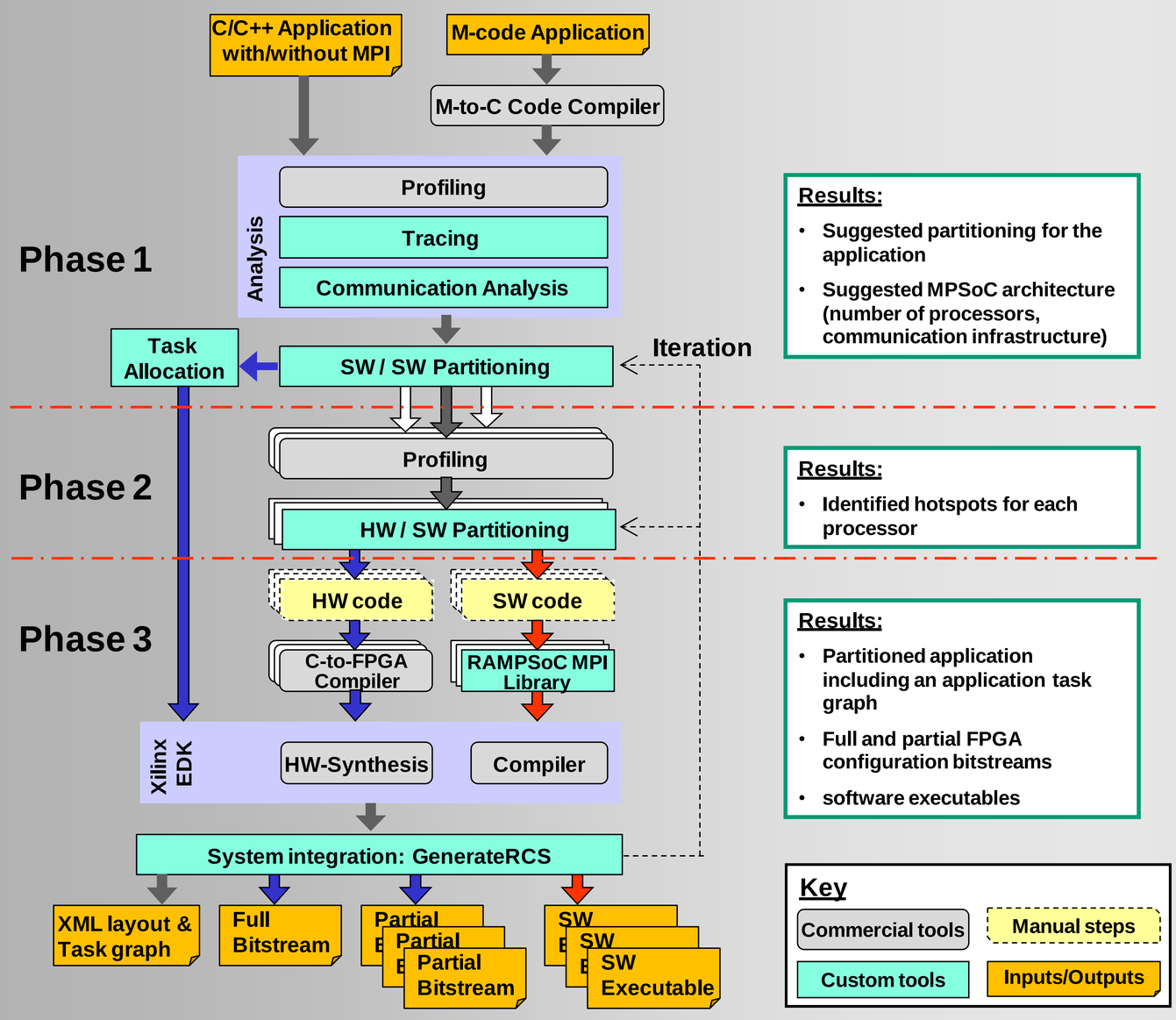
1. Introduction: FPGA versus GPU Programming

Application Description	Advantages	Disadvantages
VHDL, Verilog	Optimized Designs, 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: Drag and Drop	Restricted to a specific IP library
C, C++ (e.g. CatapultC, Autopilot)	Fast design space exploration, automatic generation of IP-blocks	Special C, C++ coding style required for good results, generates only IP blocks not full FPGA designs

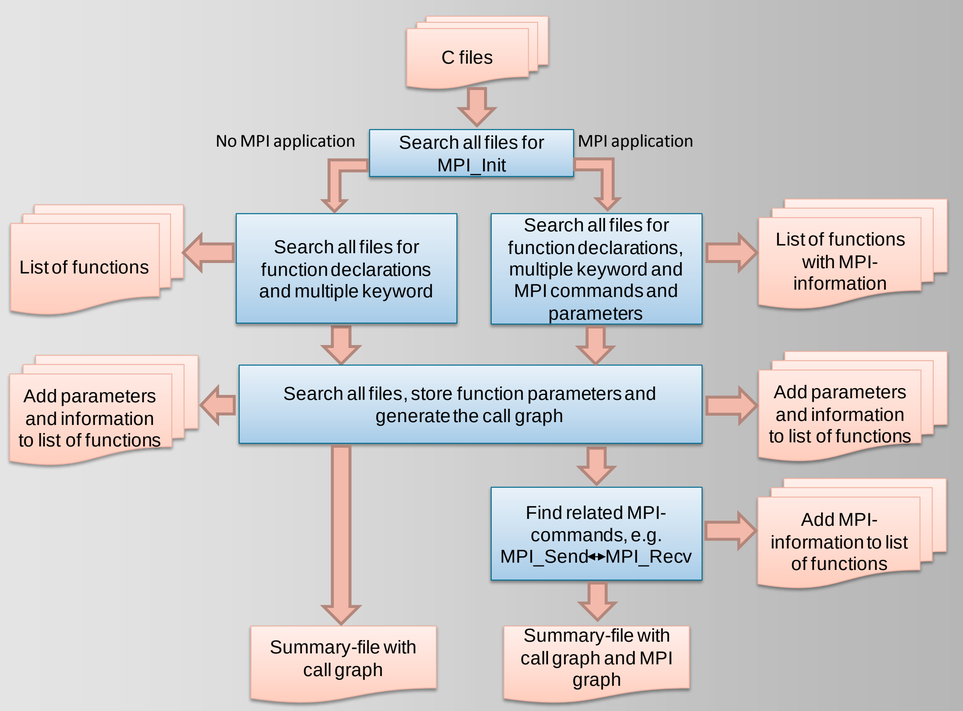


Characteristic	Description	Consequence
Programming Language	Programmed in C-like languages such as NVIDIA's CUDA or OpenCL	Easier and faster design flow than for FPGAs, because more familiar to application developers
Programming Model	Based on an implicit data-parallel programming model (single instruction, multiple threads)	Write scalar code and execute on thousands of threads / data elements, no support for task-parallelism
Communication & Synchronization	Limited and costly inter-thread communication and synchronization mechanisms	Most efficient if no global communication between threads and independent on order of processing

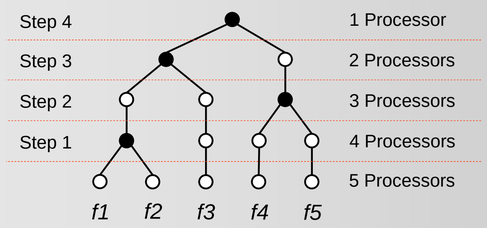
2. Design Methodology for FPGA-based Multiprocessor Accelerators



a) Communication Analysis



b) SW/SW Partitioning



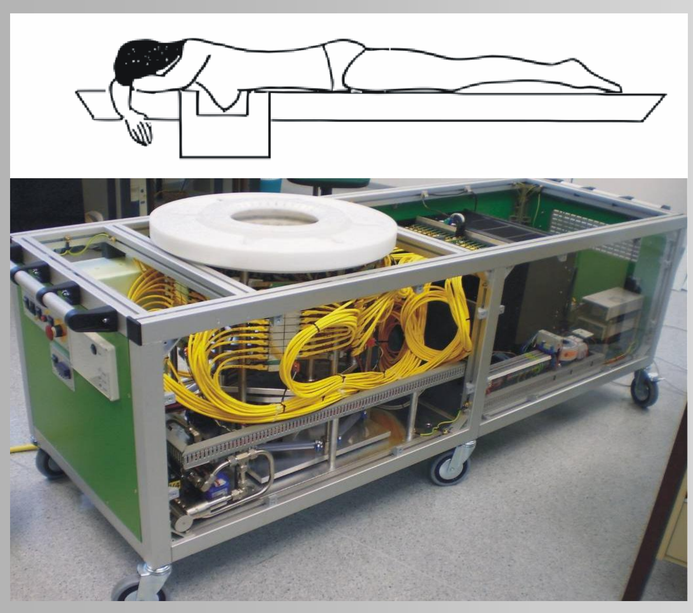
$$C(x,y) = \begin{cases} \omega_{MPI}\dfrac{MPI\_COM(x,y)}{T(x,y)} + \omega_{Call}\dfrac{Call\_COM(x,y)}{T(x,y)} \\ \dfrac{NH(x,y)}{T(x,y)}, \text{else if } MPI\_COM, Call\_COM \text{ unknown} \end{cases}$$

T(x, y) : Sum of the profiled runtimes of the two tasks to be clustered
MPI_COM(x, y) : Communication costs between two tasks communicating via MPI
Call_COM(x, y) : Communication costs between two tasks in the callgraph
NH(x, y) : Proximity of two tasks based on the call graph
ω_{MPI}: Weighting factor for MPI communication
ω_{Call}: Weighting factor for call graph communication

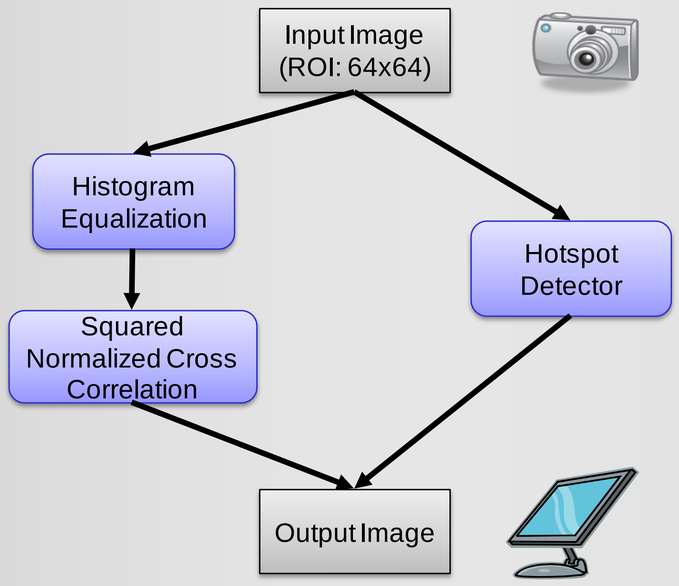
fx: Function x, (Task Granularity used for the Hierarchical Clustering)

3. Application Exploration

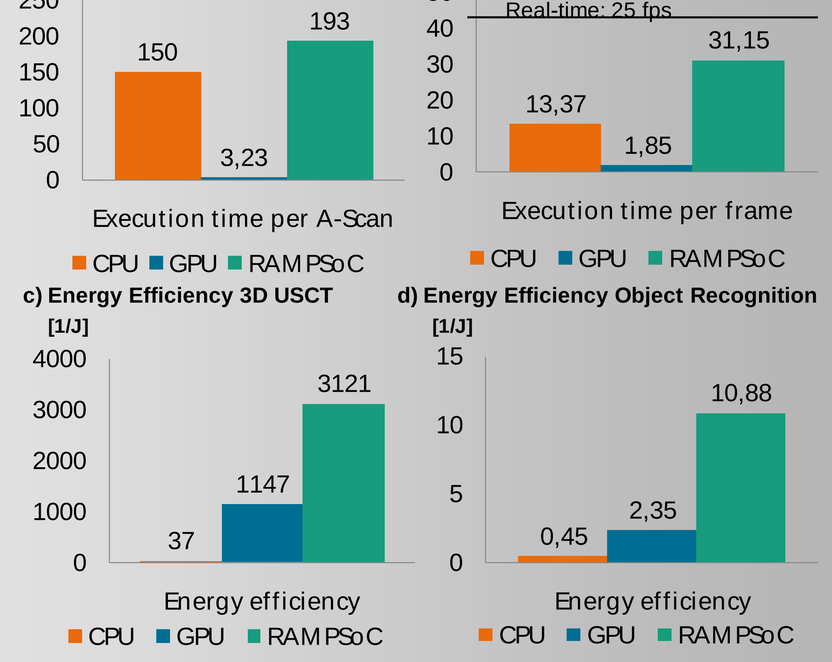
a) 3D Ultrasound computer Tomography



b) Object Recognition



4. Results



5. References

(1) D. Göhringer, M. Hübner, J. Becker, "Adaptive Multiprocessor System-on-Chip Architecture: New Degrees of Freedom in System Design and Runtime Support," in Multiprocessor System-on-Chip: Hardware Design and Tool Integration, M. Hübner and J. Becker, Ed. Springer, 2010, pp.125-149.

(2) H. Gemmeke, N.V. Ruiter, "3D ultrasound computer tomography for medical imaging," Nuclear Instruments and Methods in Physics Research Section A, vol. 580, no. 2, pp.1057-1065, 2007.