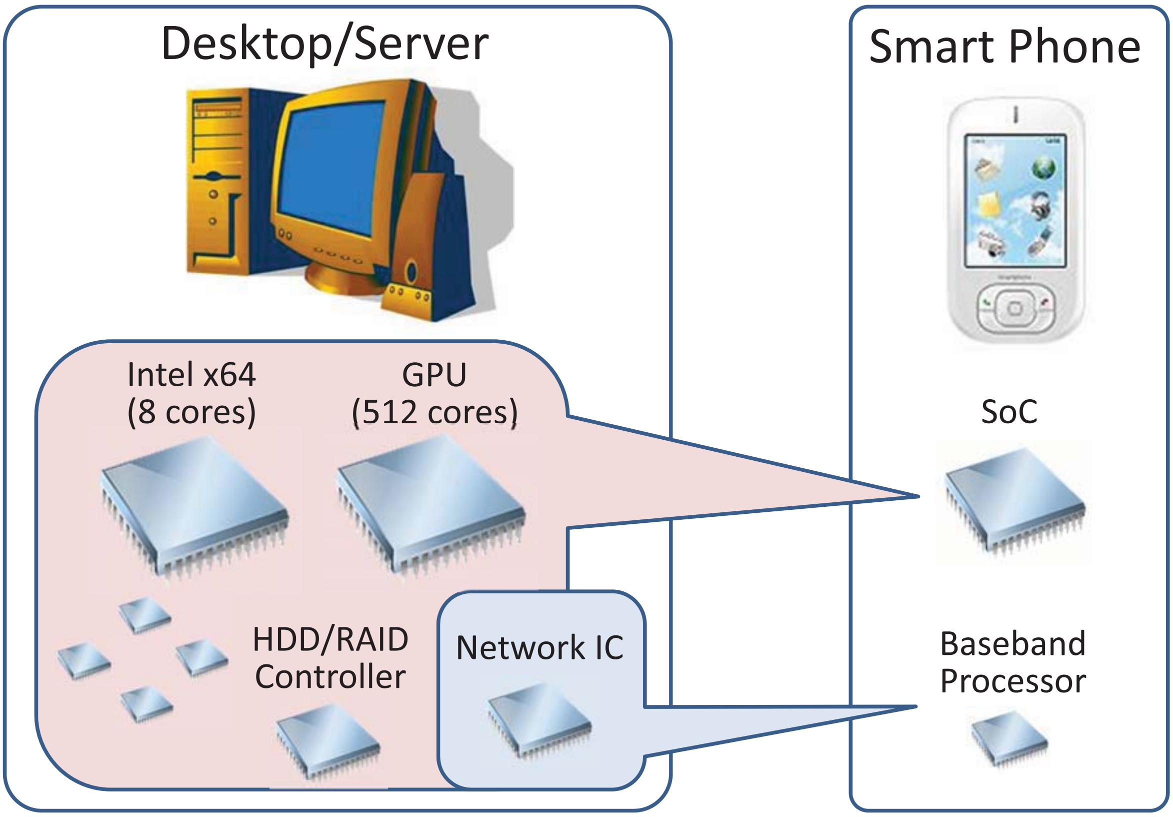
Challenges of Programming Embedded Many-Core SoCs with OpenCL

Hiroyuki Tomiyama

Ritsumeikan University
http://hiroyuki.tomiyama-lab.org/

MPSoC 2011

Increasing Cores



Desktop/Server

Intel x64
(8 cores)

GPU
(512 cores)

HDD/RAID
Controller

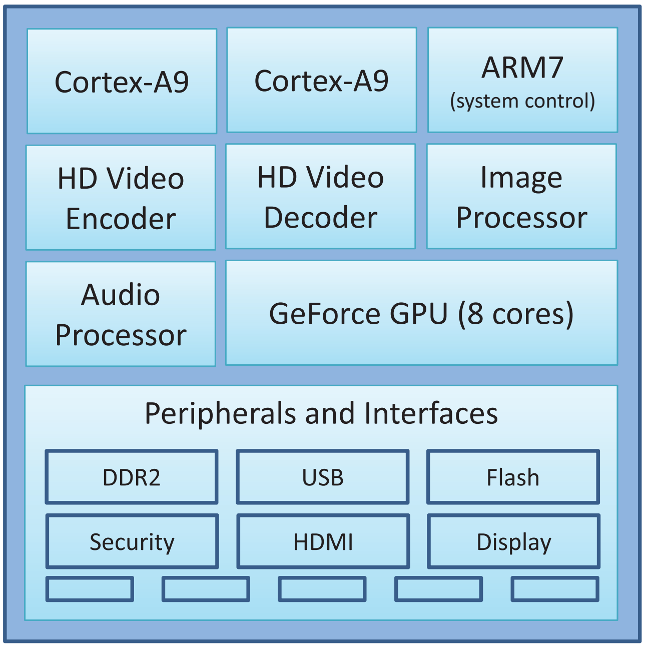
Network IC

Smart Phone

SoC

Baseband
Processor

Nvidia Tegra 2



Cortex-A9 | Cortex-A9 | ARM7 (system control)

HD Video Encoder | HD Video Decoder | Image Processor

Audio Processor | GeForce GPU (8 cores)

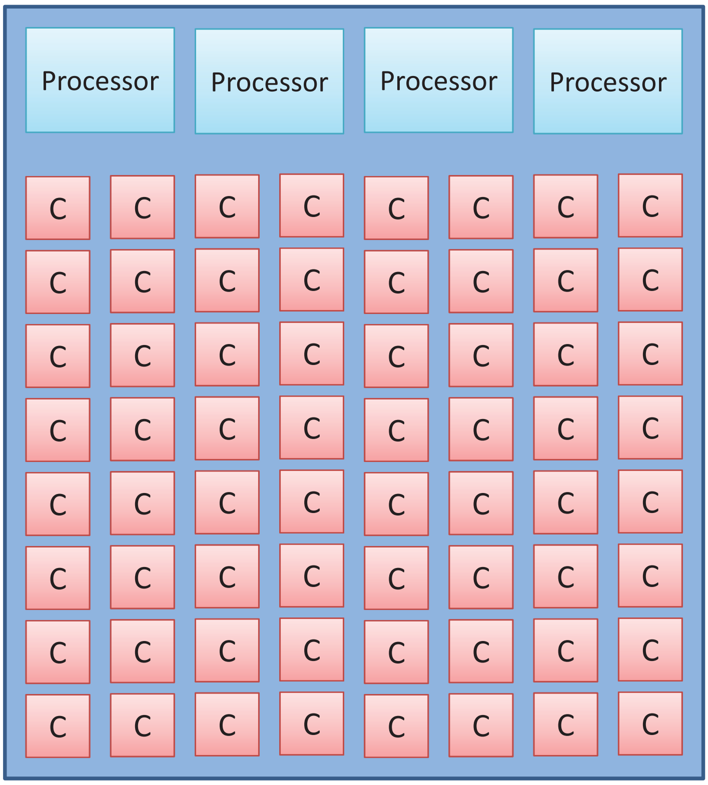
Peripherals and Interfaces

DDR2 | USB | Flash

Security | HDMI | Display

- Nvidia’s SoC for mobile terminals
  - 2 main processors
  - 1 system controller
  - 4 media co-processors
  - 8-core GPU
- Latest SoC, named Kal-El, has 4 main processors and 12-core GPU
- Good for mobile terminals, but not so flexible for other applications

Many-Core SoC in Near Future

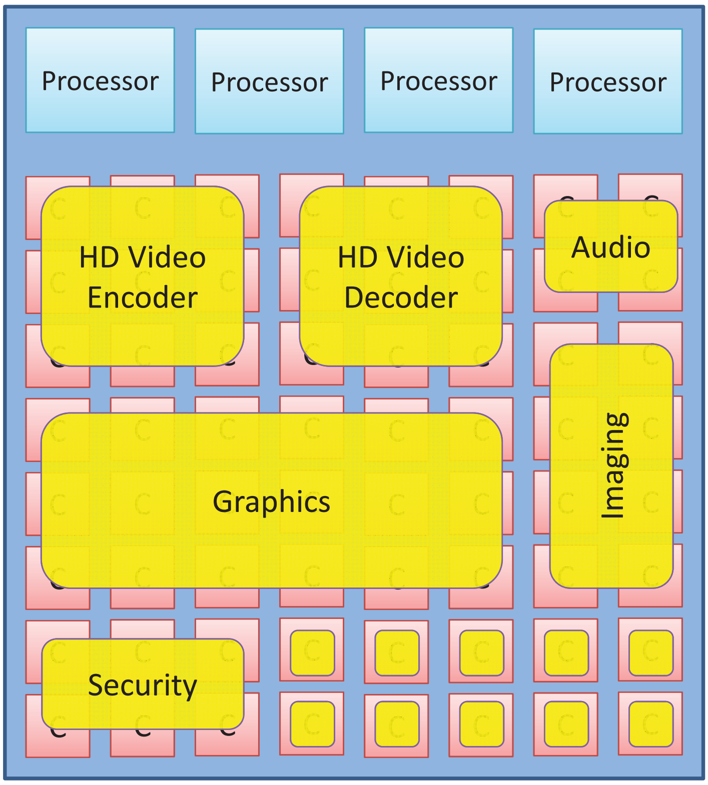


Processor | Processor | Processor | Processor

C	C	C	C	C	C	C	C
C	C	C	C	C	C	C	C
C	C	C	C	C	C	C	C
C	C	C	C	C	C	C	C
C	C	C	C	C	C	C	C
C	C	C	C	C	C	C	C
C	C	C	C	C	C	C	C
C	C	C	C	C	C	C	C

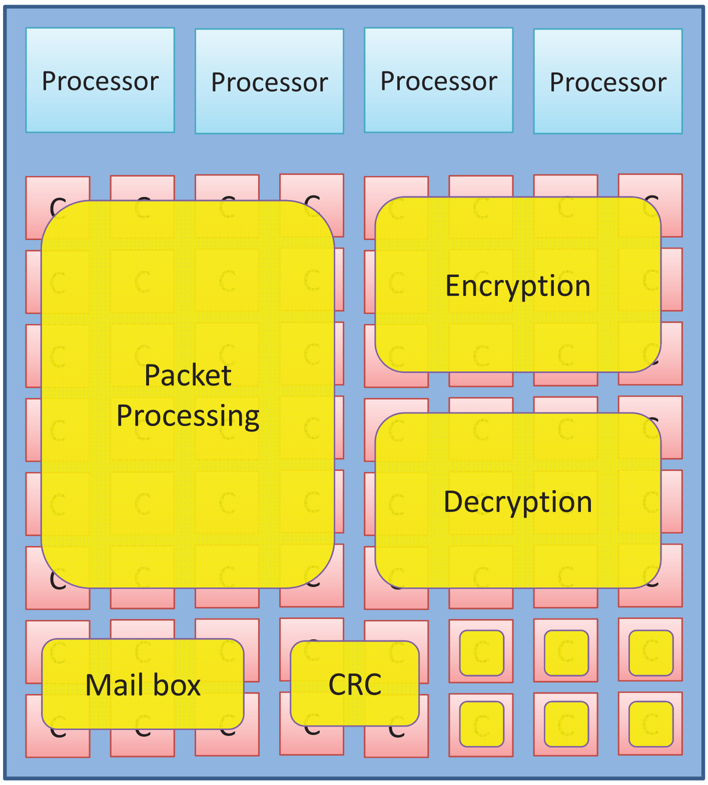
- Multiple main processors
  - Symmetric MP
  - Cache coherency
- Sea of cores
  - Homogeneous many-cores
  - Allow optimized mapping for applications

Mapping for Mobile Terminal



- Multiple main processors
  - Symmetric MP
  - Cache coherency
- Sea of cores
  - Homogeneous many-cores
  - Allow optimized mapping for applications

Mapping for Network Application



- Multiple main processors
  - Symmetric MP
  - Cache coherency
- Sea of cores
  - Homogeneous many-cores
  - Allow optimized mapping for applications

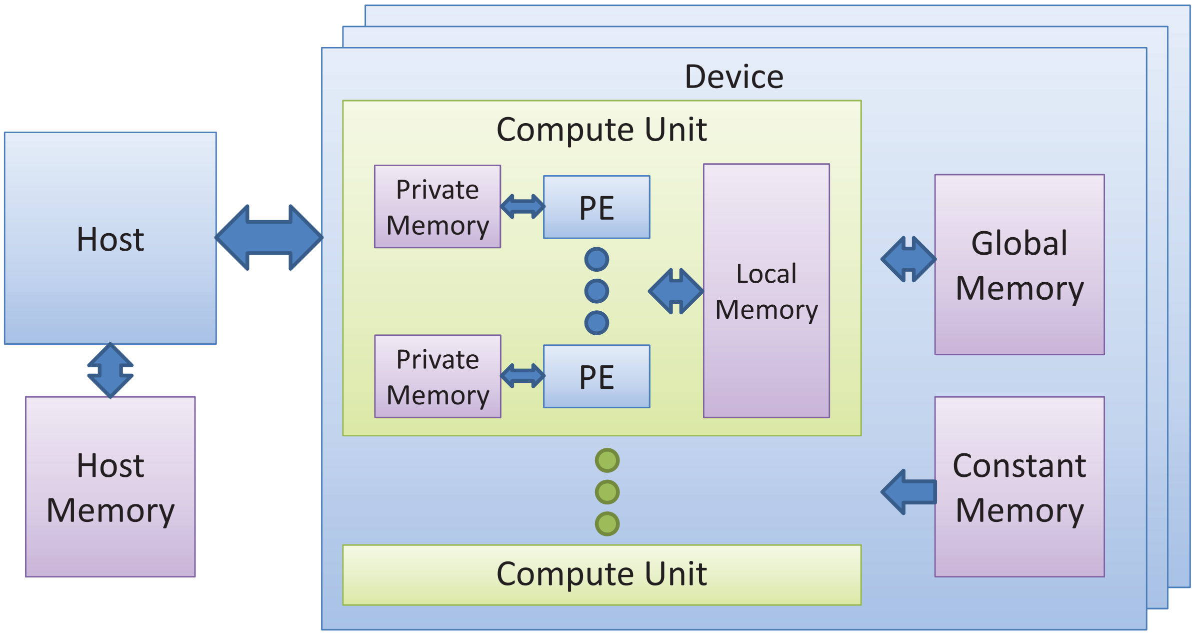
How to Program?

- A number of parallel programming models, languages, and frameworks
  - OpenMP, MPI, OpenCL, Intel Threading Building Blocks, Nvidia CUDA, etc
- **OpenCL** is worth trying
  - Support of heterogeneous architectural platforms
  - Platform independent

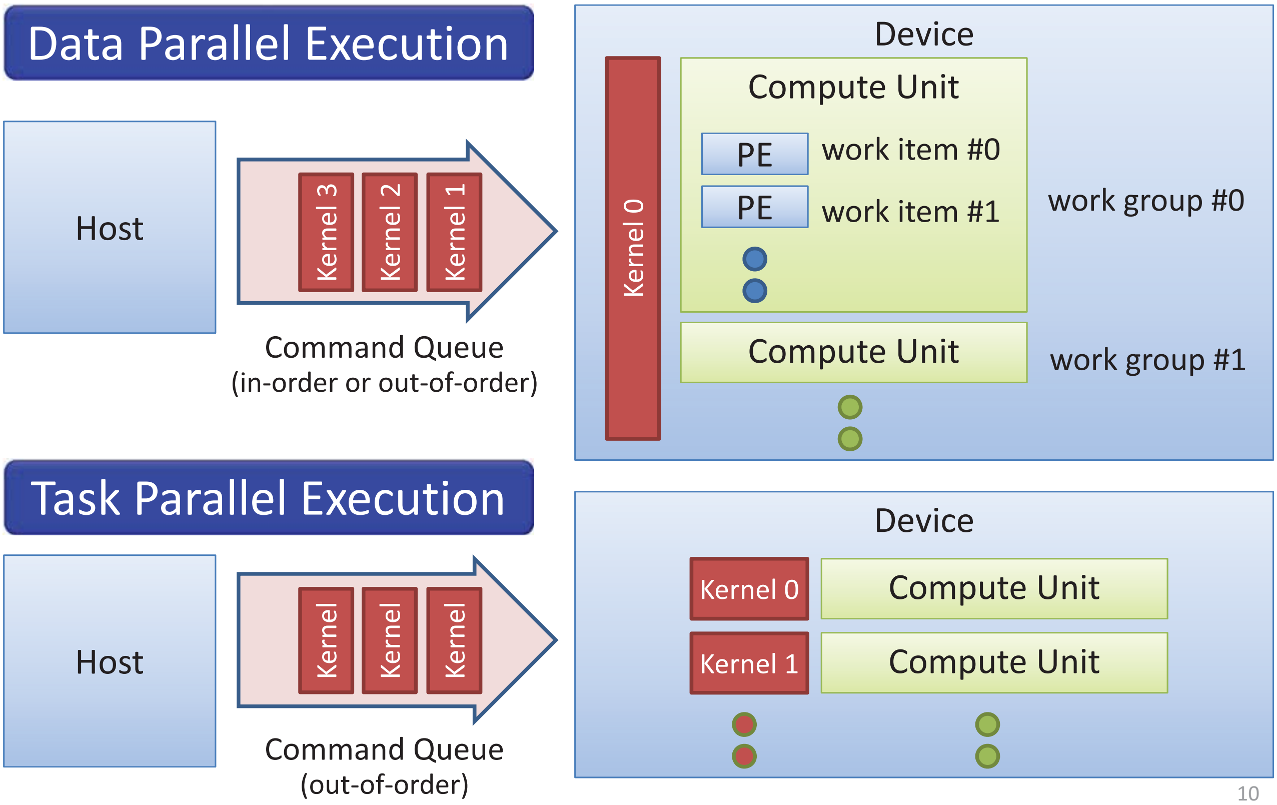
OpenCL

- Open Computing Language
- Programming framework for parallel computing
- Open, royalty-free standard by Khronos Group
  - Specification 1.0 released in 2008
- Platform independent
  - Intel's multi-core CPUs
  - Nvidia's GPUs
  - AMD's GPUs
  - SONY/IBM/Toshiba's Cell B./E.
- Based on C Language
- Supports both data- and task-parallelisms

OpenCL Architecture/Memory Model

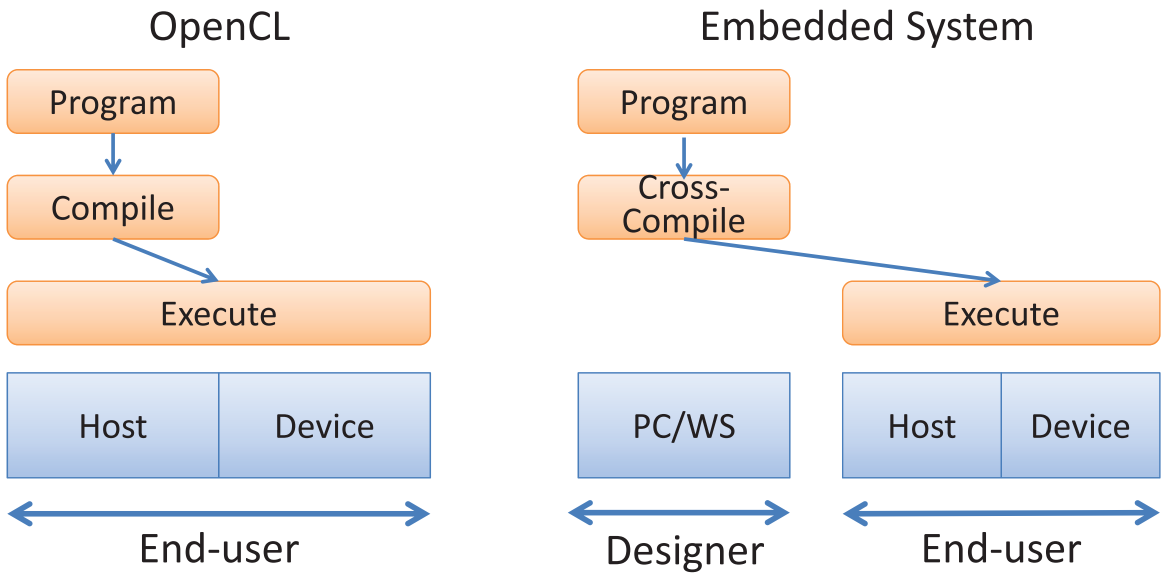


OpenCL Programming/Execution Model



Difference between OpenCL and ES

- Compilation-execution scenario is completely different between OpenCL and embedded systems



OpenCL

Program

Compile

Execute

Host

Device

End-user

Embedded System

Program

Cross-Compile

Execute

PC/WS

Host

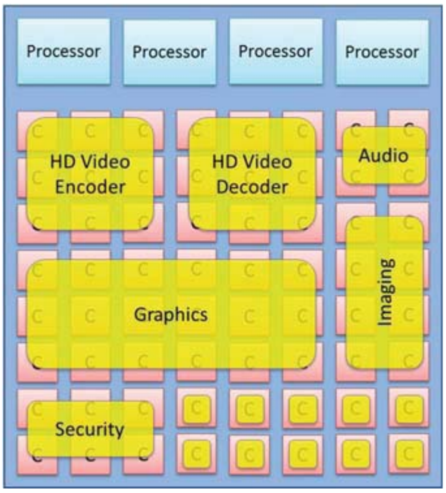
Device

Designer

End-user

Limitations and Problems

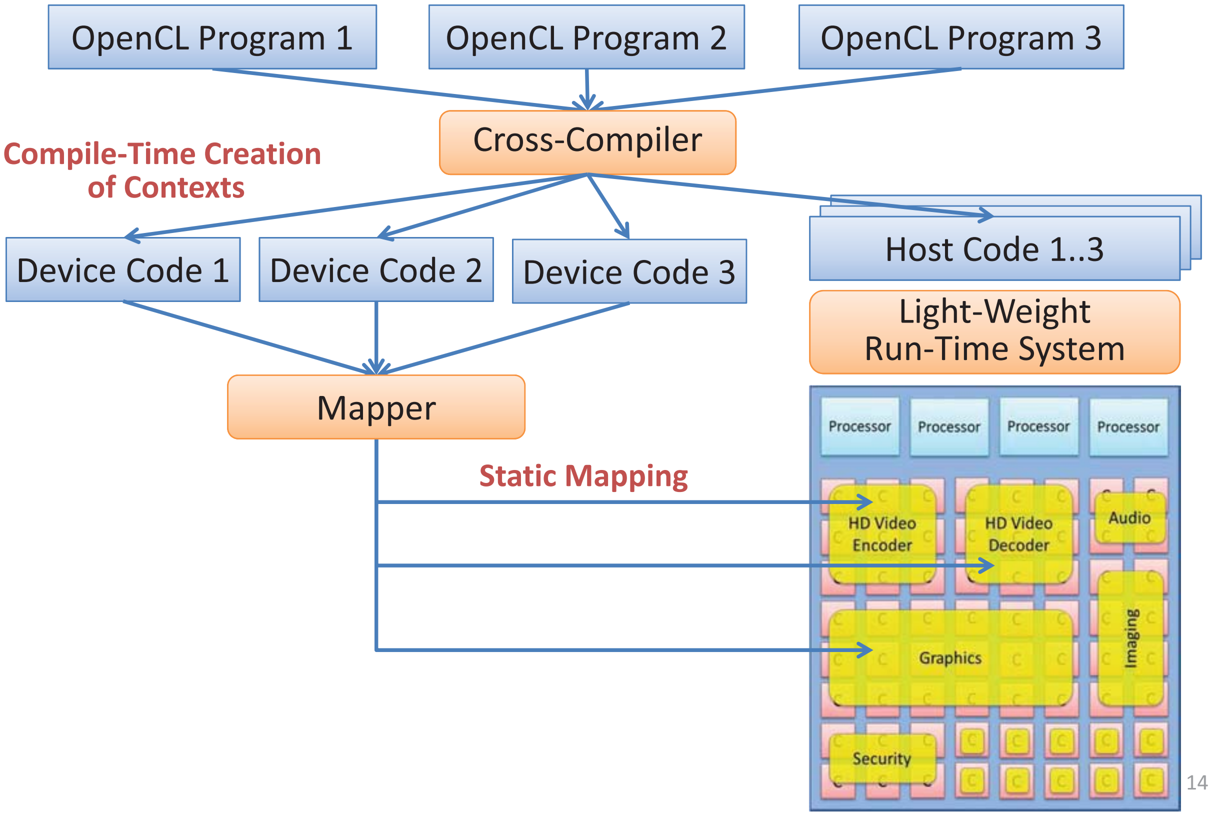
- Parallel execution of multiple applications is impossible
  - A single application occupies the entire device (all cores) at a time
- Hard to guarantee real-time constraints
  - Large performance overhead for context creation and dispatch
  - What is worse, such overhead is hardly predictable



Our Approach

- The most critical problem lies in the implementation of compiler and run-time system, rather than in the OpenCL language
  - The language has several problems, though…
- Develop a new OpenCL toolkit ***for embedded system designers***
  - Compiler
  - Mapper
  - Run-time system
- **As static as possible** for improved predictability

Our Toolkit under Development



OpenCL Program 1

OpenCL Program 2

OpenCL Program 3

Cross-Compiler

Compile-Time Creation of Contexts

Device Code 1

Device Code 2

Device Code 3

Host Code 1..3

Light-Weight Run-Time System

Mapper

Static Mapping

Concluding Remarks

- OpenCL is a candidate for programming many-core SoCs, but existing toolkits are not suitable to embedded system design
- We are developing a new OpenCL toolkit for embedded system designers
- Acknowledgments
  - This work is in part supported by New Energy and Industrial Technology Development Organization (NEDO), Japan
  - In collaboration with
    - Kyushu University
    - The University of Electro-Communications
    - Fixstars Corp.
    - TOPS Systems Corp.