

OpTiMSoC

Build Your Own System-on-Chip!

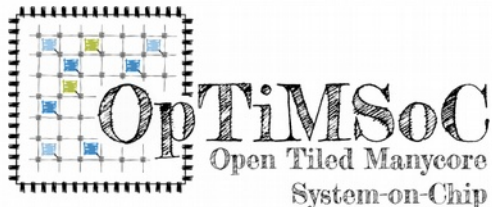
Philipp Wagner

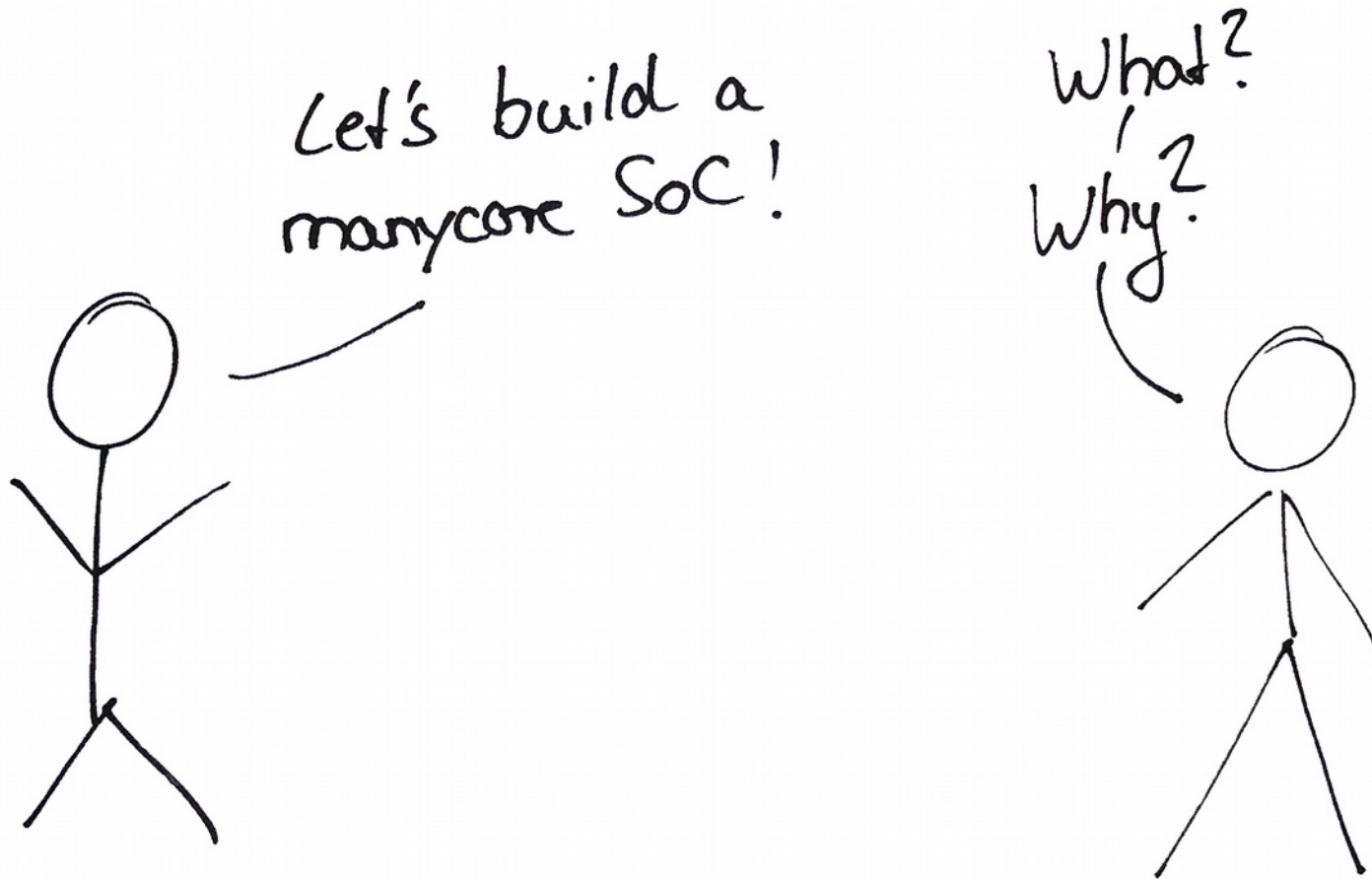
FOSDEM 2014
February 2nd 2014

Institute for Integrated Systems
Prof. Dr. Andreas Herkersdorf

Arcisstraße 21
80333 München
Germany

<http://www.lis.ei.tum.de>





It's his fault! (not really)

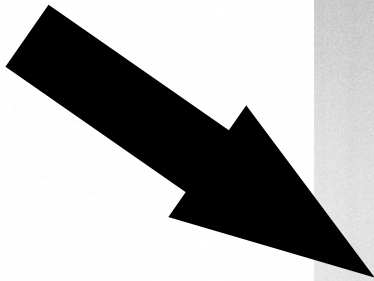


Photo by IntelFreePress, CC-BY-SA-2.0
<http://www.flickr.com/photos/intelfreepress/8429166752/sizes/o/in/photostream/>

Goals

scalable

robust

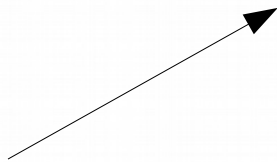
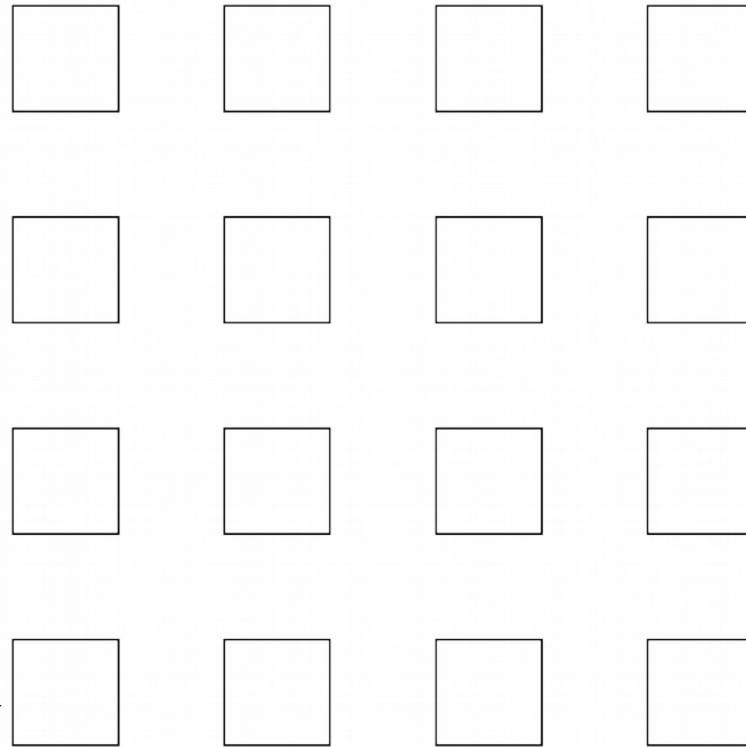
reusable

The solution is not new



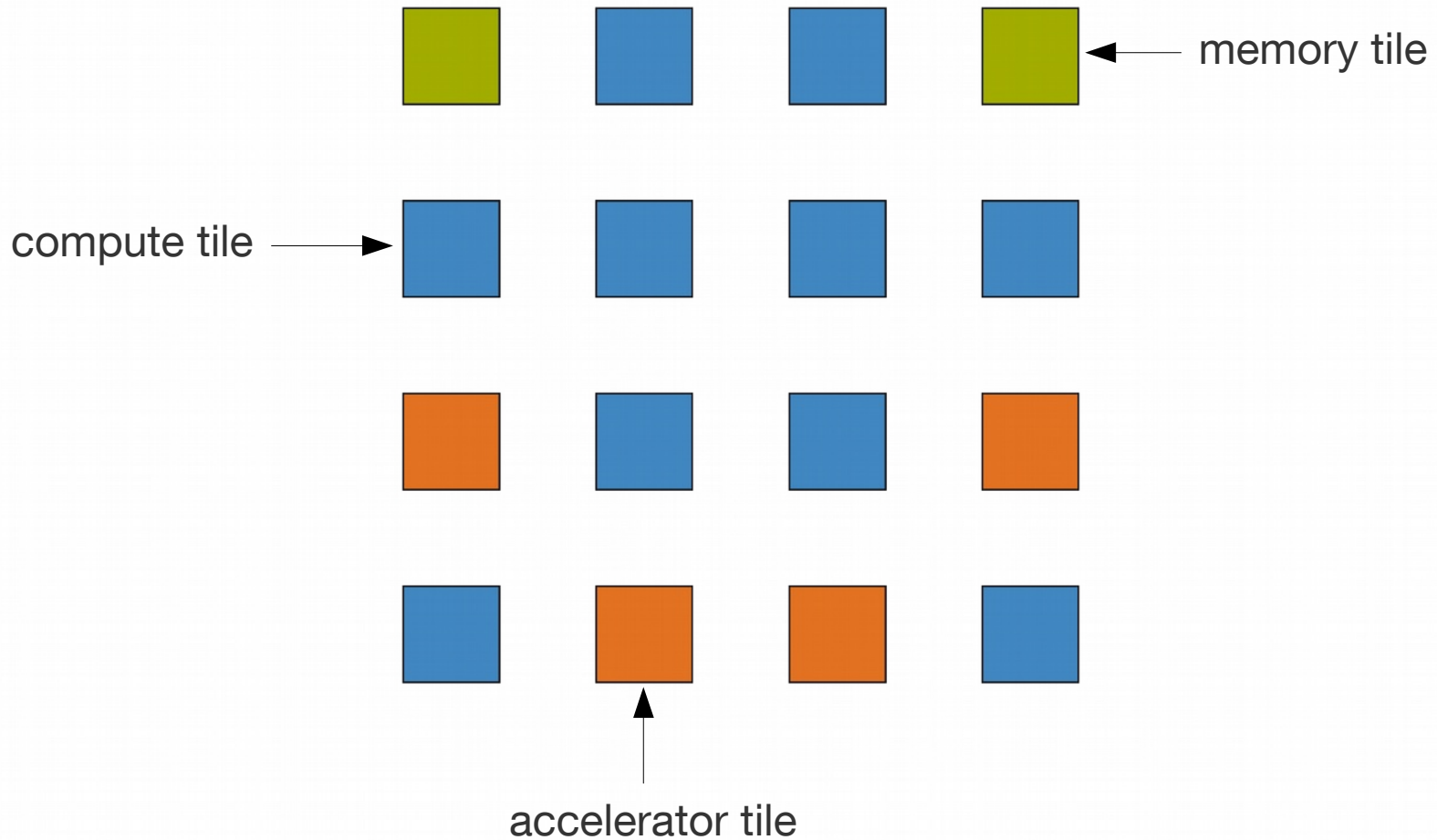
Photo by mharsch on Flickr, CC-BY-SA-2.0
<http://www.flickr.com/photos/mharsch/29319685/sizes/o/in/set-653640/>

Tiles: our basic building blocks

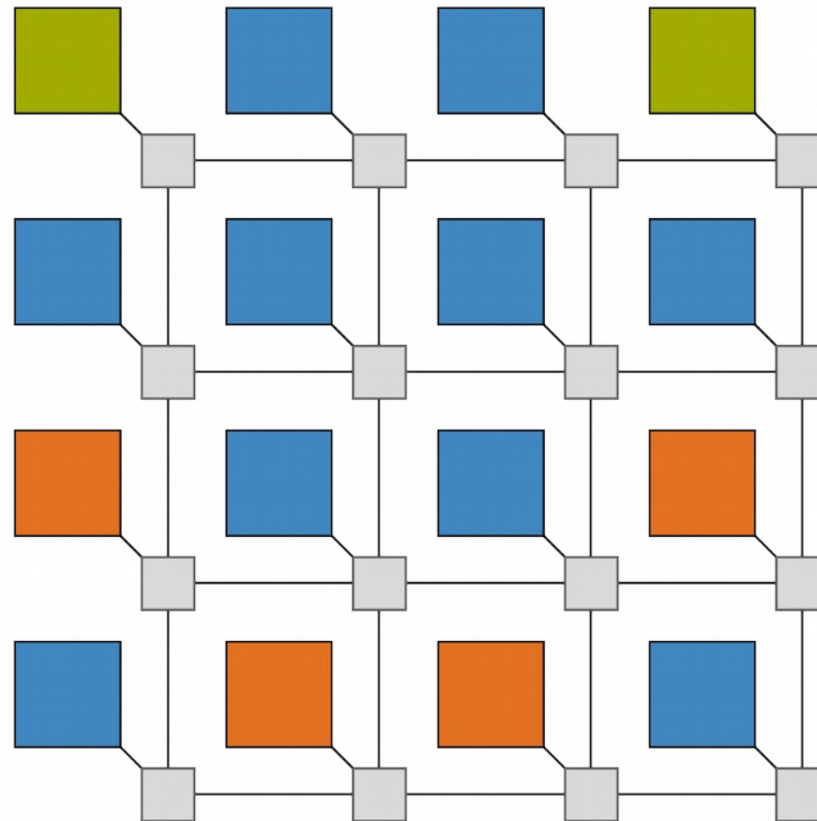


a tile

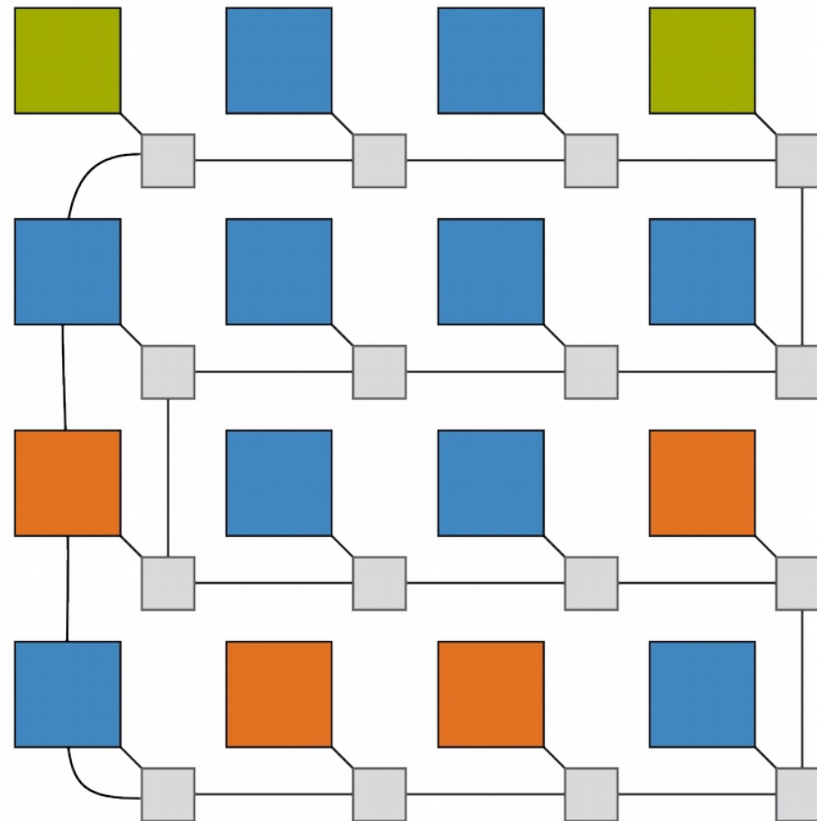
Tiles can be very different ...



Network-on-Chip (NoC): Mesh



Network-on-Chip (NoC): Ring



Intel Single-chip Cloud Computer (SCC)

Inside the SCC

24 Tiles
24 Routers
48 IA cores

ROUTER

Dual-core SCDC Tile

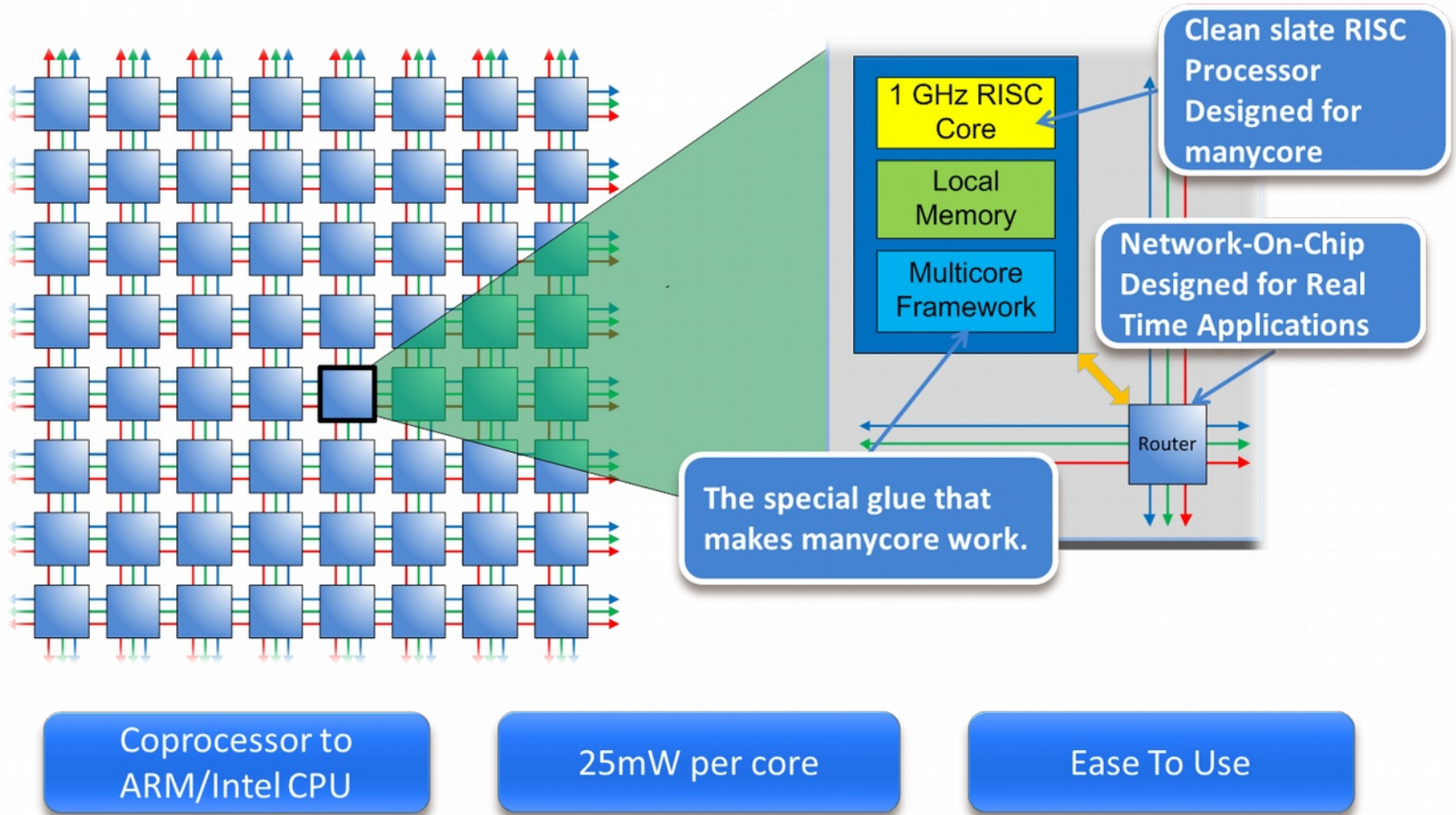
MEMORY CONTROLLER

1 TILE

- 2D mesh network with 256 GB/s bisection bandwidth
- 4 Integrated DDR3 memory controllers (64GB addressable)

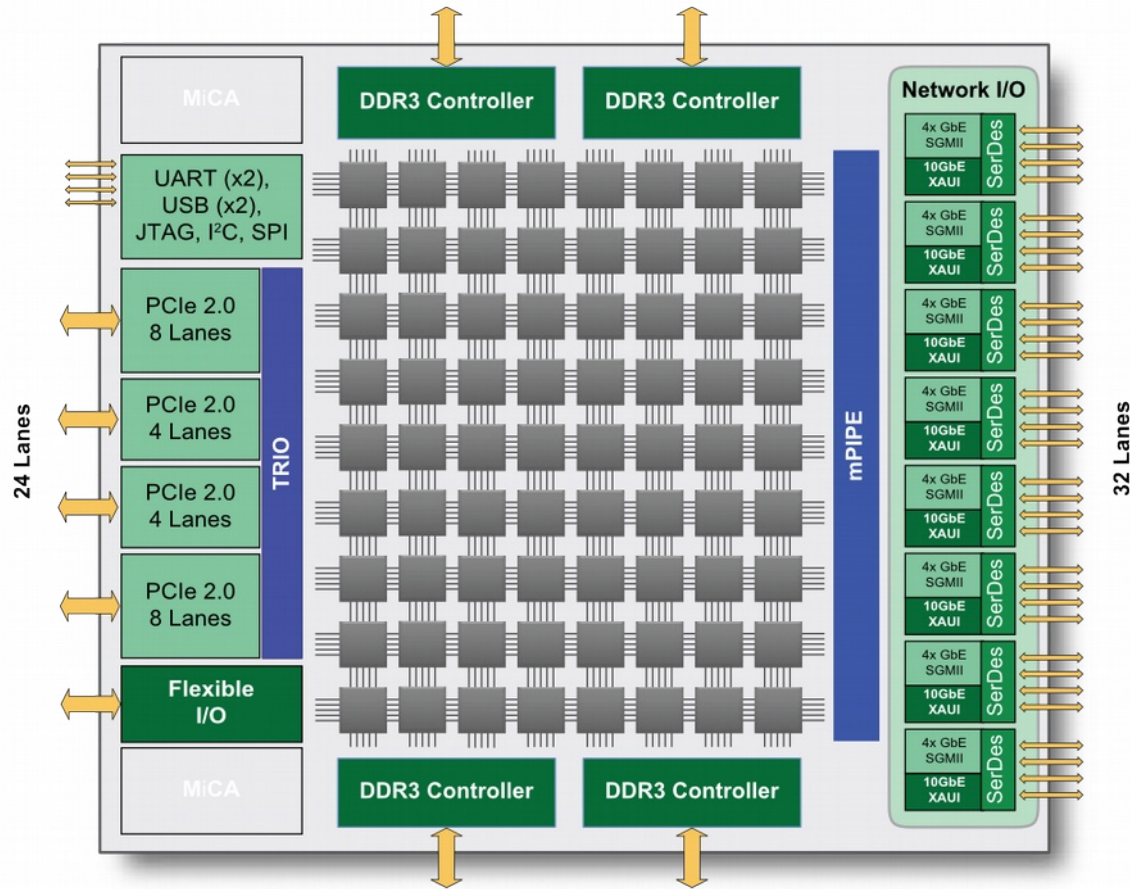
Slide 6 from SCC Announcement Presentation by Justin Rattner (CTO Intel), Dec. 2009
http://download.intel.com/pressroom/pdf/rockcreek/SCC_Announcement_JustinRattner.pdf

Adapteva Epiphany (→ Parallela)



Graphic from the "Epiphany Introduction" by Adapteva
<http://www.adapteva.com/introduction/>

Tilera TILE-Gx64



Graphic taken from "Tilera TILE-Gx8072Processor Product Brief"
http://www.tilera.com/sites/default/files/images/products/TILE-Gx8072_PB041-03_WEB.pdf

Let's talk about

Op

Open

Ti

Tiled

M

Many-
Core

SoC

System-on-Chip

Building Blocks for *your* Tiled Manycore System-on-Chip

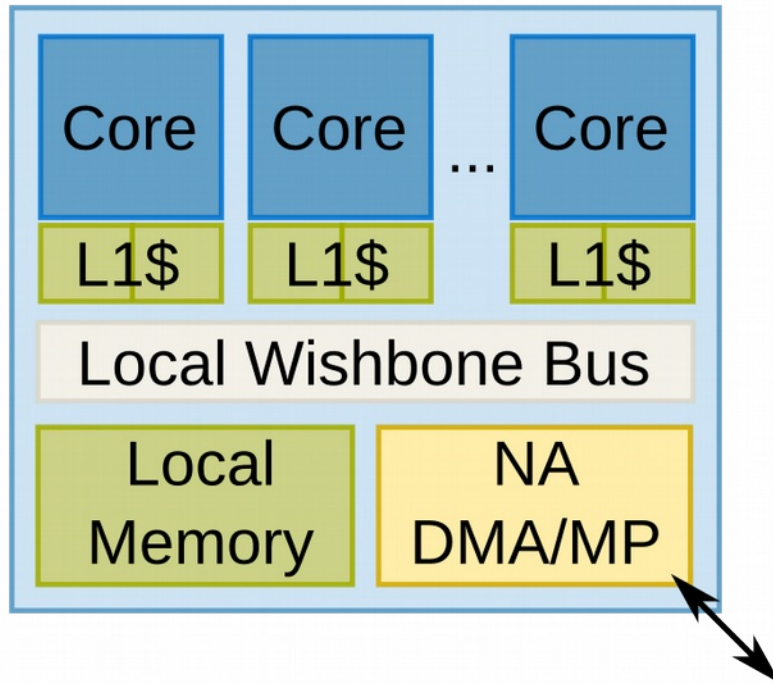
Not a production ready solution!

The OpTiMSoC “Bag of Components”



Jeffrey Beall via Flickr, CC-BY-SA-2.0 (edited)
www.flickr.com/photos/denverjeffrey/4392418334

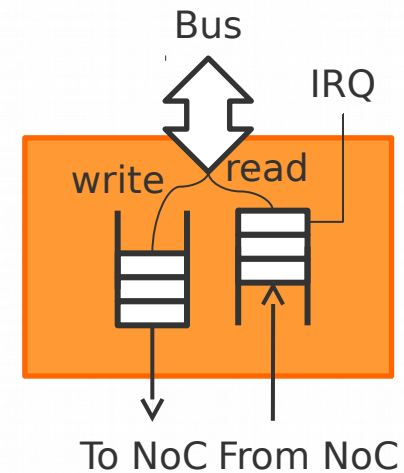
Compute Tile: Distributed Memory



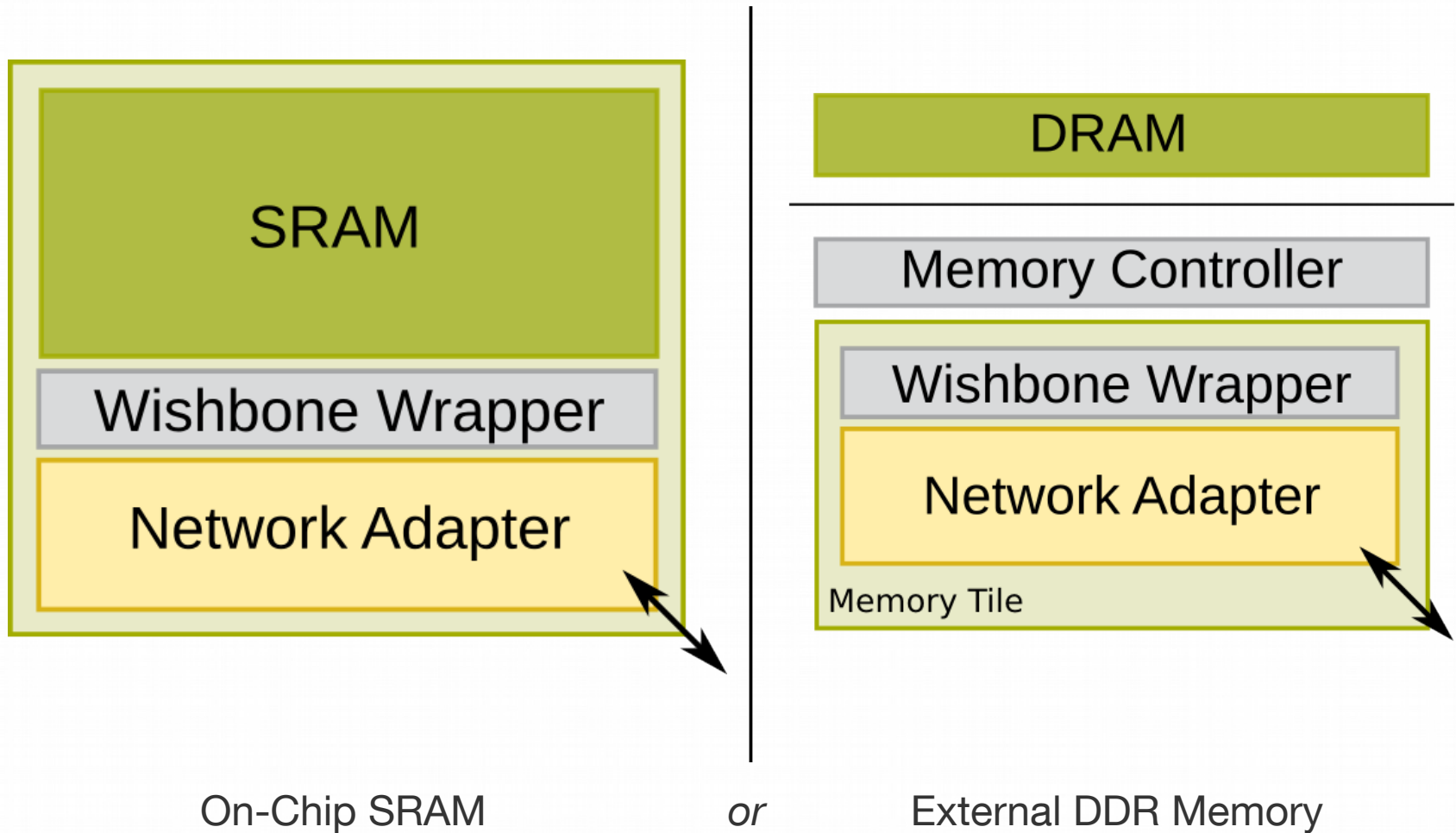
- OpenRISC Core
 - OR1200 (old)
 - mor1kx (new)
- Extended for multicore
 - core identifier SPR
 - Compare-and-Swap (CAS) unit (memory mapped)
 - scratch memory for each core
- Shared memory (with L2\$) version currently under development

Network Adapter

- **Computation ↔ Communication**
- **DMA Engine**
 - Transfer bulk data between tiles
- **Message Passing (MP)**
 - Explicit communication between tiles
 - Asynchronous



Memory Tile



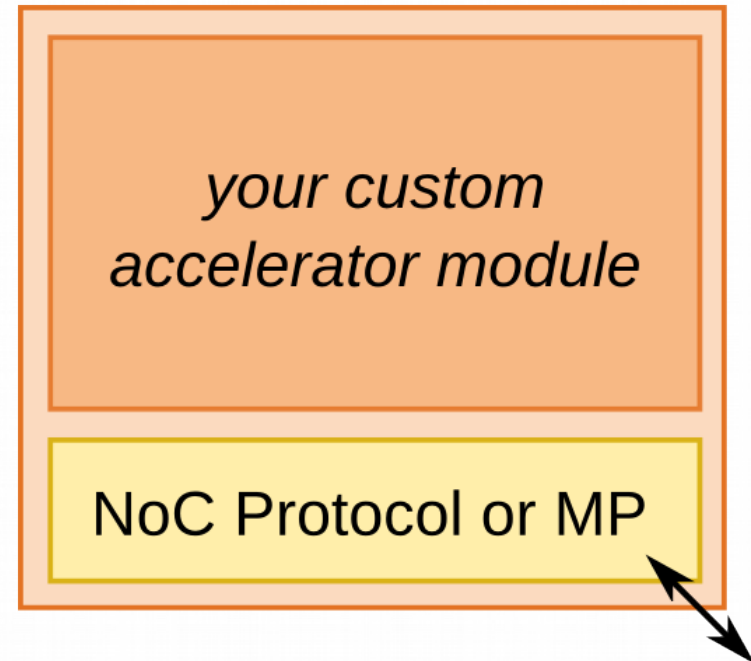
I/O Tiles

- UART
- SPI
- LCD dotmatrix display



Accelerator Tiles

- For CPU offload
- Needs to handle NoC protocol and possibly Message Passing protocol
- currently working on AES (crypto) tile



Network-on-Chip: LISNoC

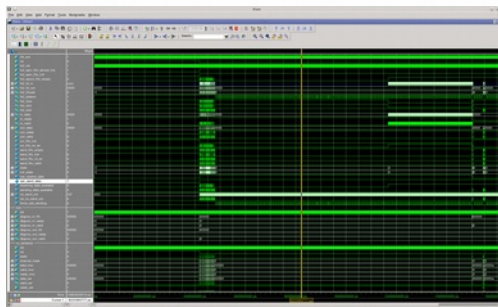
- Basic NoC implementation
 - mesh or ring
 - wormhole
 - packet switched
 - dimension routing
 - configurable buffers
 - virtual channel support
- Other features “available on request” (bufferless, multicast, ...)



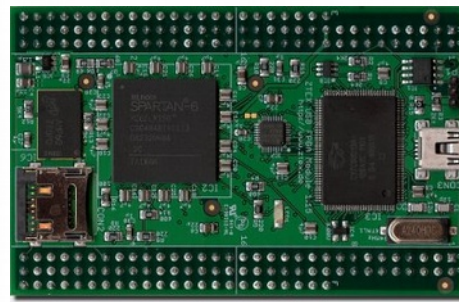


Photo by cliff1066TM, via Flickr: CC-BY-2.0
<http://www.flickr.com/photos/nostri-imagor/3137422976/>

Target Platforms



RTL Simulation



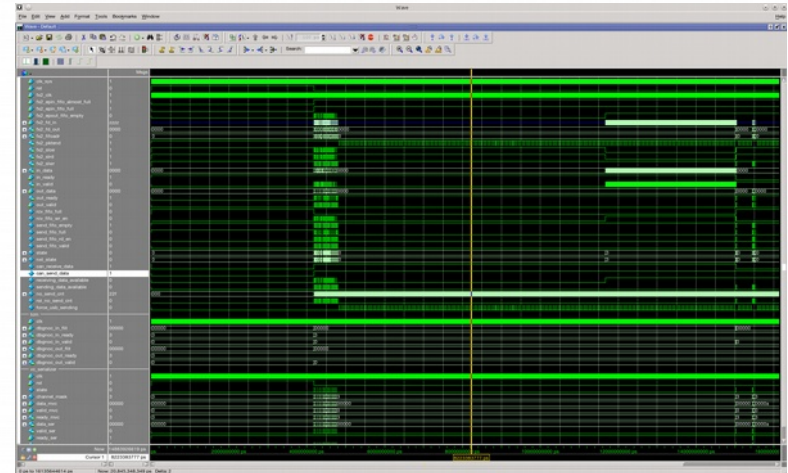
FPGA



Emulation

Target: RTL Simulation

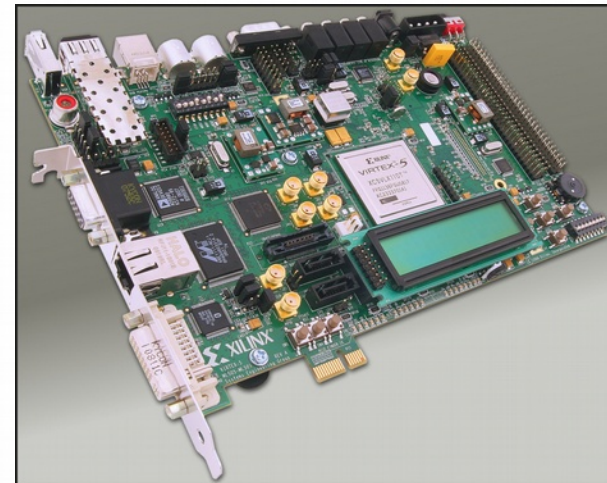
- ModelSim
 - signal-level insight into HW
 - short turnaround times
 - rather slow
- Verilator
 - great for system-level and software development
 - much faster than ModelSim
 - free software!
- Icarus Verilog
 - *free software*
 - *never tested*



Target: FPGA



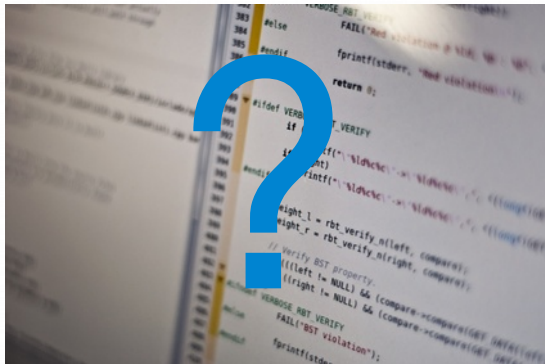
ZTEX USB-FPGA 1.15b/d
Xilinx Spartan 6 FPGA and
Cypress FX2 USB 2.0



Xilinx University Program Virtex 5
(XUPV5; almost ML505)

pro: fast, real hardware
con: hard-ware = hard bugs

Programming?



Programming: the basics

- **newlib** C library port
- **OpenRISC GCC**

Programming: “Operating System”

- **libbaremetal**: drivers, utility functions
 - Mutex, CAS, interrupt handling, DMA, timer
- **libruntime**: simple runtime system
 - scheduler, thread handling
- Multicore Association Communication (MCAPI) and Task Management API (MTAPI)
 - message passing (MCAPI) and heterogeneous task management (MTAPI) APIs
- *Linux support is being worked on, ETA end of 2014*

Programming: Hello World

```
$> cat hello_simple.c
#include <stdio.h>

void main() {
    printf("Hello World!\n");
}

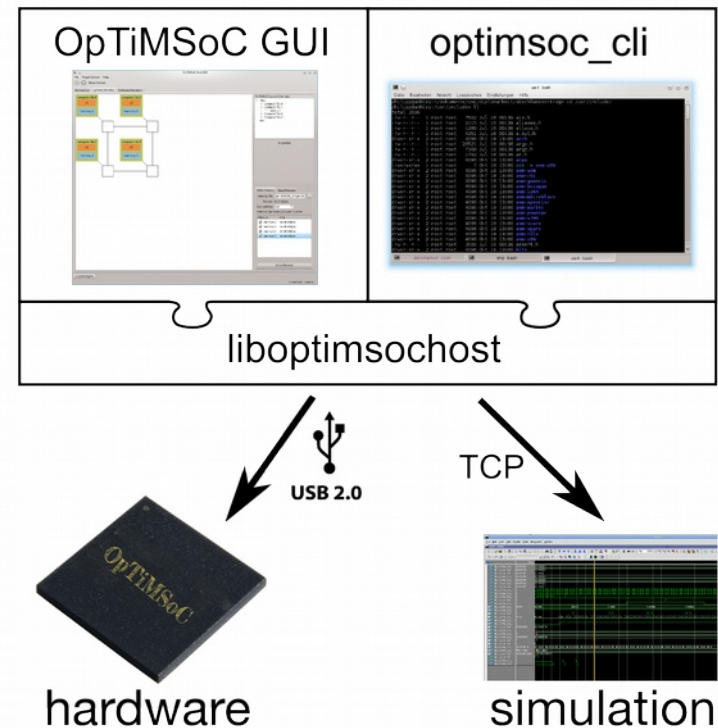
$> cat Makefile
PROGRAM=hello_simple

BUILDSRIPTS=$(shell pkg-config --variable=buildscriptdir optimsoc-baremetal)
include $(BUILDSRIPTS)/Makefile.inc
```

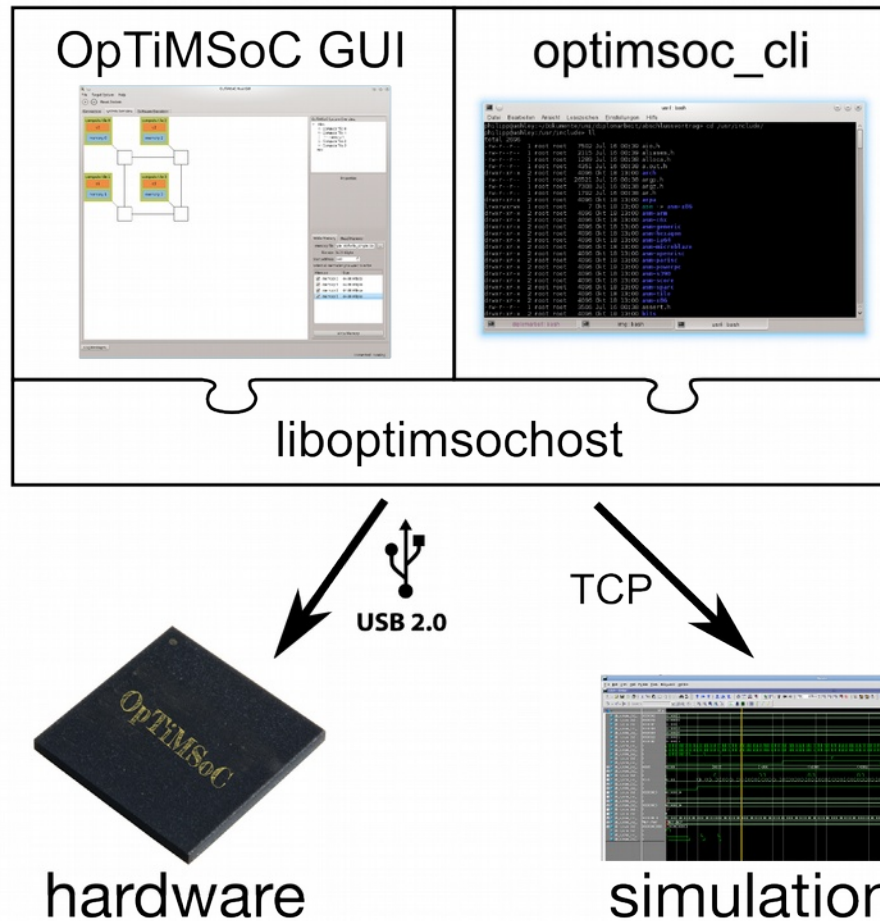
All build infrastructure is included!

Debugging: Concept

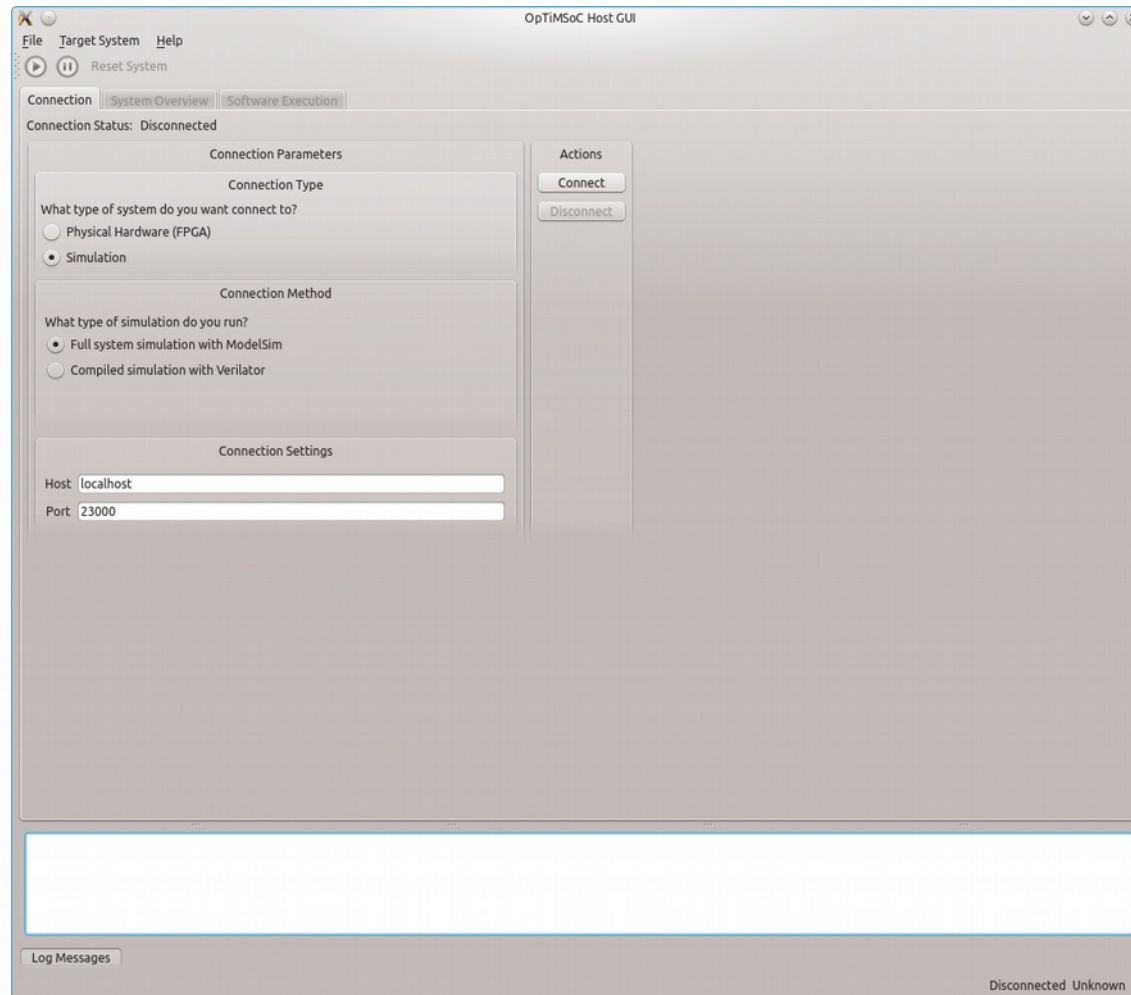
- **tracing** only
- hardware support
 - filtering, compression
 - no interference with system
- debug modules in hardware
 - instruction traces
 - software traces
 - `printf()`
 - send events from software
 - NoC status
- initialization of memories



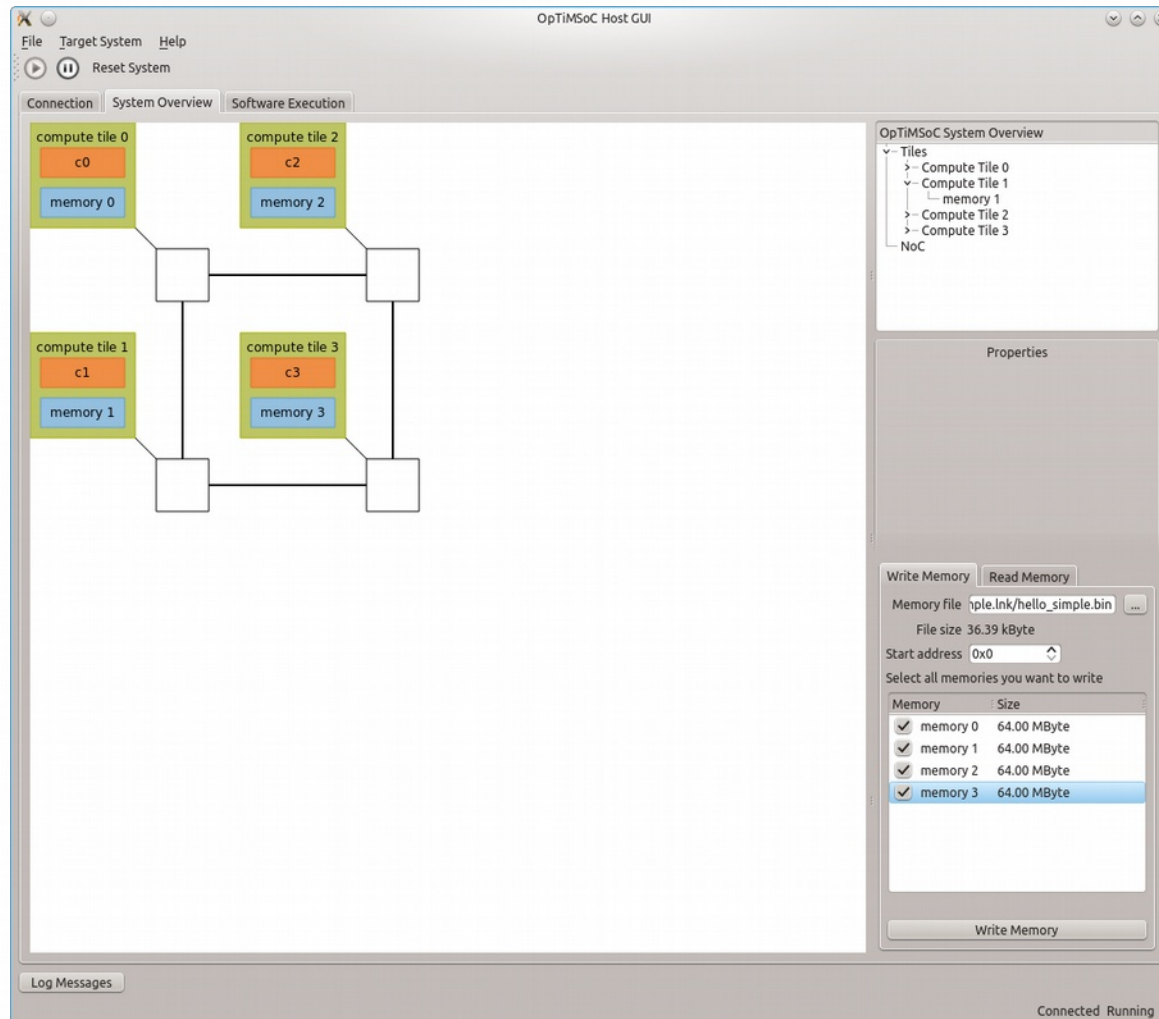
Debugging: Implementation



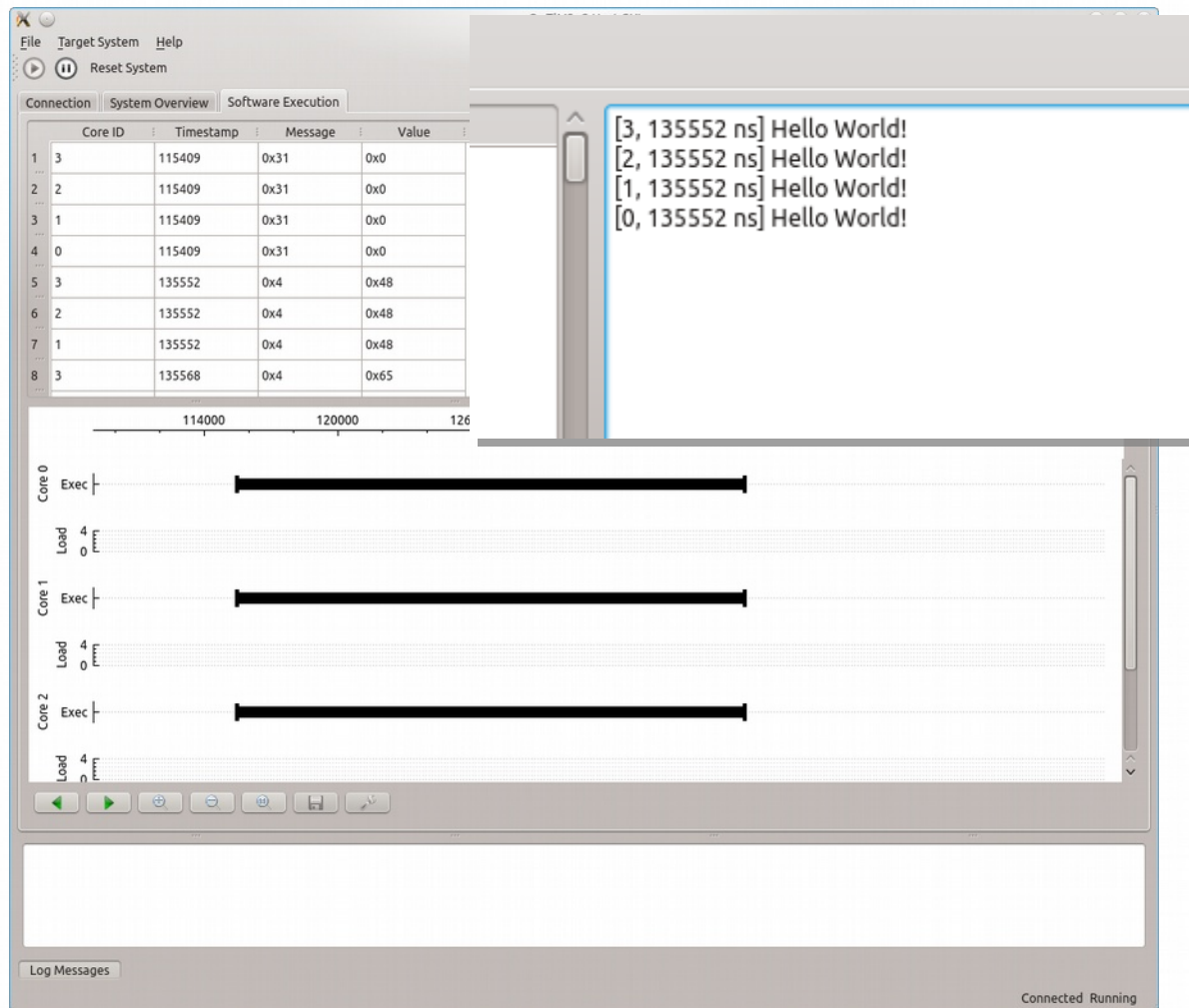
OpTiMSoC Host GUI



OpTiMSoC Host GUI



OpTiMSoC Host GUI

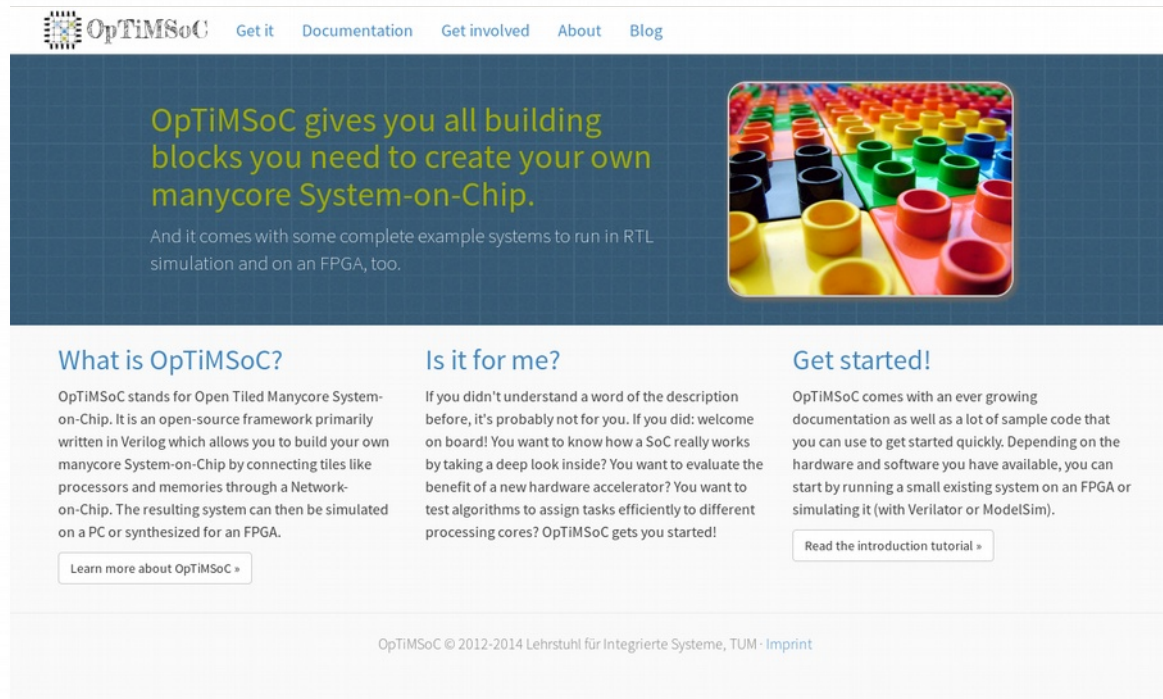


What can I use OpTiMSoC for?

- Use your FPGA: Build your own SoC
 - often 1 or 2 CPU cores are enough
- Develop/evaluate hardware architectures/extensions
- Look inside a SoC
- *<insert your ideas here>*

Get Started!

Code & Documentation @ www.optimsoc.org



The screenshot shows the OpTiMSoC website homepage. At the top, there is a navigation menu with links for 'Get it', 'Documentation', 'Get involved', 'About', and 'Blog'. The main header features the OpTiMSoC logo and a large blue banner with the text: 'OpTiMSoC gives you all building blocks you need to create your own manycore System-on-Chip. And it comes with some complete example systems to run in RTL simulation and on an FPGA, too.' To the right of this text is an image of colorful interlocking plastic blocks. Below the banner, there are three columns of content: 'What is OpTiMSoC?', 'Is it for me?', and 'Get started!'. Each column contains a short paragraph and a button with a right-pointing arrow. At the bottom of the page, there is a small copyright notice: 'OpTiMSoC © 2012-2014 Lehrstuhl für Integrierte Systeme, TUM - Imprint'.

OpTiMSoC [Get it](#) [Documentation](#) [Get involved](#) [About](#) [Blog](#)

OpTiMSoC gives you all building blocks you need to create your own manycore System-on-Chip.

And it comes with some complete example systems to run in RTL simulation and on an FPGA, too.

What is OpTiMSoC?

OpTiMSoC stands for Open Tiled Manycore System-on-Chip. It is an open-source framework primarily written in Verilog which allows you to build your own manycore System-on-Chip by connecting tiles like processors and memories through a Network-on-Chip. The resulting system can then be simulated on a PC or synthesized for an FPGA.

[Learn more about OpTiMSoC »](#)

Is it for me?

If you didn't understand a word of the description before, it's probably not for you. If you did: welcome on board! You want to know how a SoC really works by taking a deep look inside? You want to evaluate the benefit of a new hardware accelerator? You want to test algorithms to assign tasks efficiently to different processing cores? OpTiMSoC gets you started!

Get started!

OpTiMSoC comes with an ever growing documentation as well as a lot of sample code that you can use to get started quickly. Depending on the hardware and software you have available, you can start by running a small existing system on an FPGA or simulating it (with Verilator or ModelSim).

[Read the introduction tutorial »](#)

OpTiMSoC © 2012-2014 Lehrstuhl für Integrierte Systeme, TUM - Imprint

Questions?



The “Army of Awesome”: Andreas Lankes, Philipp Wagner, Stefan Wallentowitz
Alexandra Weber, Andreas Oeldemann, Andreas Wilhelm, Christian Morgenstern, Daniel Haug, Daniel Thaler, Dexin Chen,
Eugen Egel, Falco Cescolini, Hans-Christian Wild, Johannes Ehm, Julia Müller, Markus Göhrle, Martin Lowinski, Manuel
Krause, Michael Faath, Michael Tempelmeier, Preethi Parayil, Robert Specht, Sebastian Herzog, Simon Schulze, Stefan
Rösch, Umbreen Mian