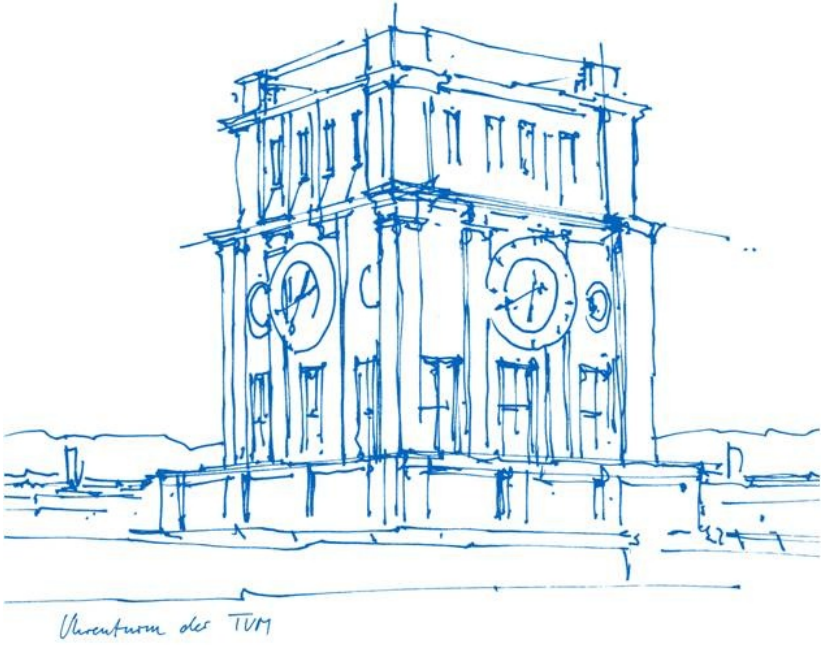
Fast architecture prototyping on FPGAs: frameworks, tools, and challenges

Philipp Wagner

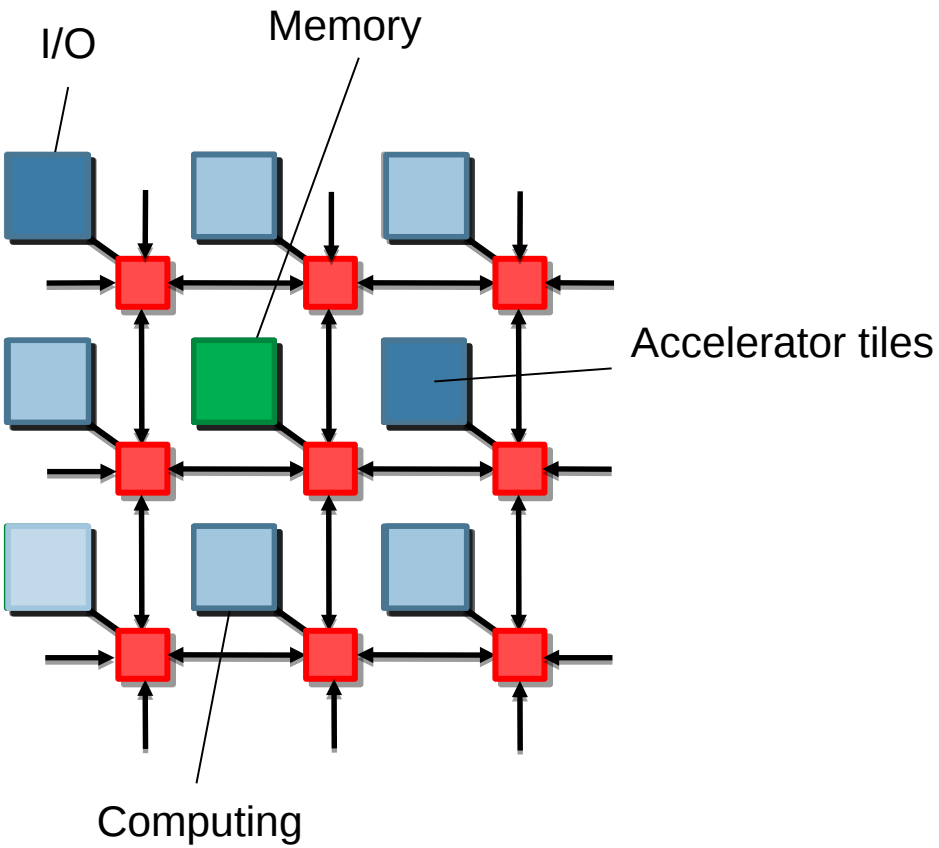
Technische Universität München

Lehrstuhl für Integrierte Systeme

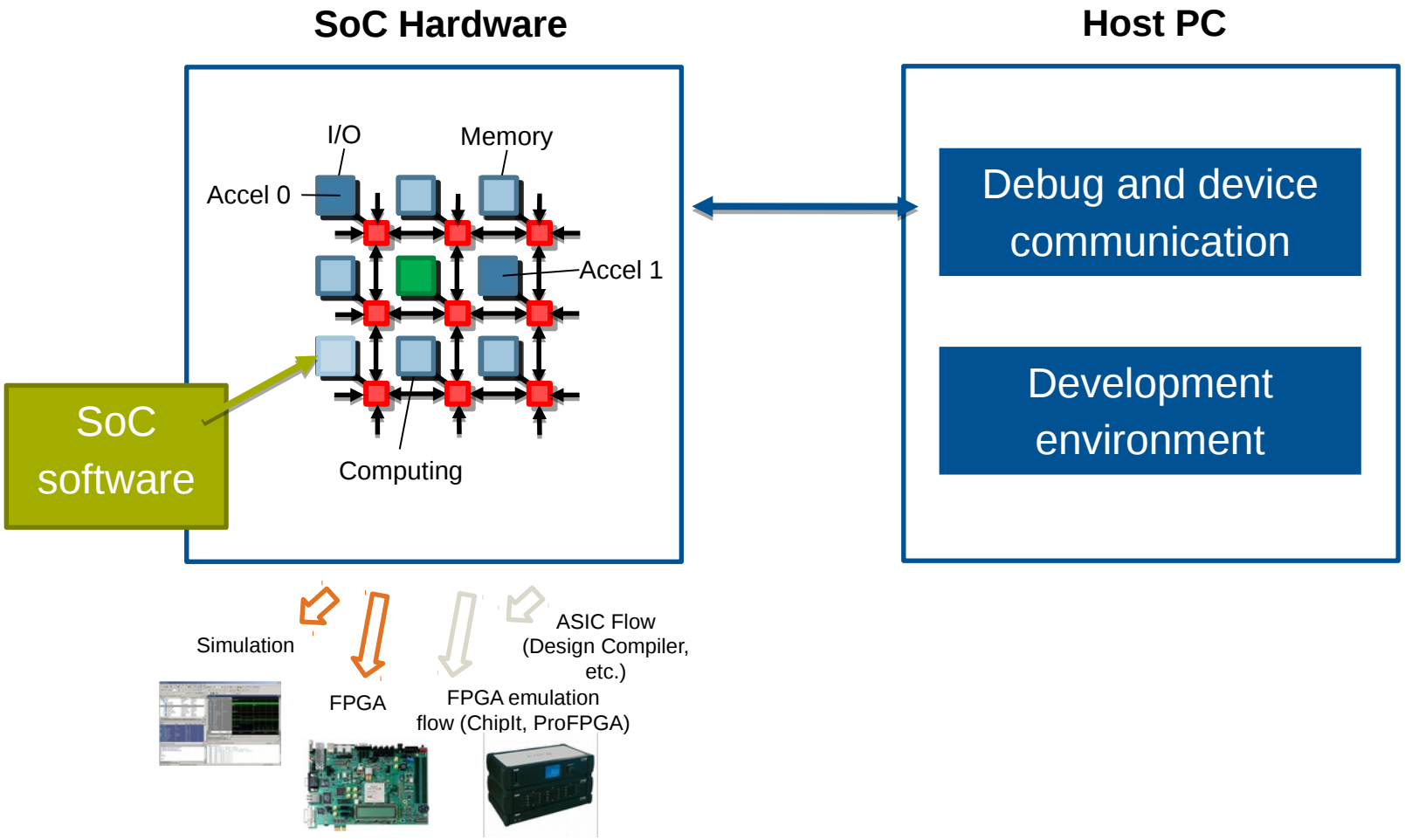
10.04.2017



Our Goal: Improving MPSoC Architectures



Much more is needed ...

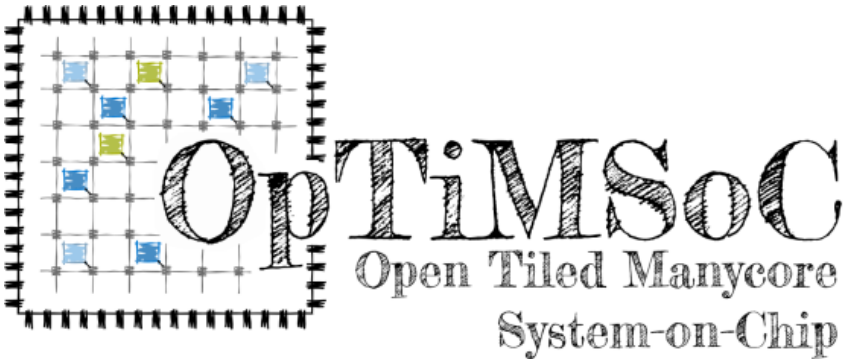


Introducing OpTiMSoC

- An open source, “batteries included” **framework** to build tiled Many-Core System-on-Chip

- easy to use
- clear extension vectors
- reproducible results

- design philosophy
  - don’t reinvent the wheel
  - clean trumps clever

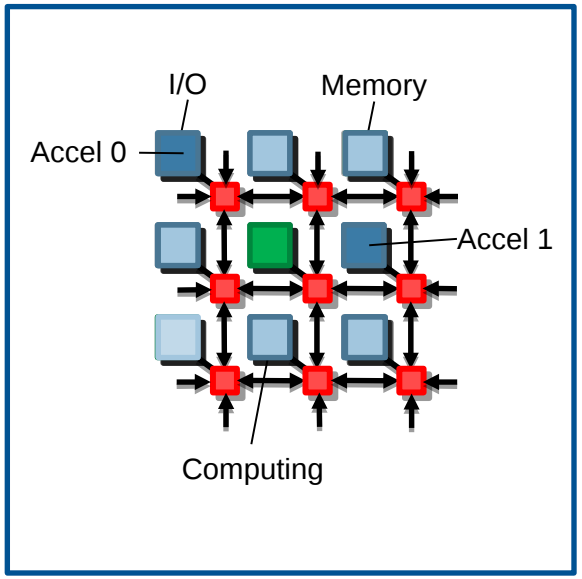


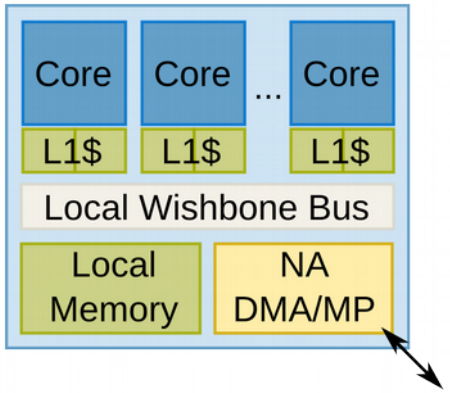
SoC Hardware

- Collection of **tiles**
  - mor1kx CPU tile (1-4 CPU cores per tile) with multi-core extensions (CAS; LL/SC; TSL)
  - memory tile
  - I/O tiles [most in internal development]
    - camera
    - VGA
    - GPIO

- **NoC**: LISNoC
  - packet-based, wormhole routed
  - VC support

- Building blocks
  - large library of standard blocks (FIFOs, Wishbone/AXI bus, clocking, …)



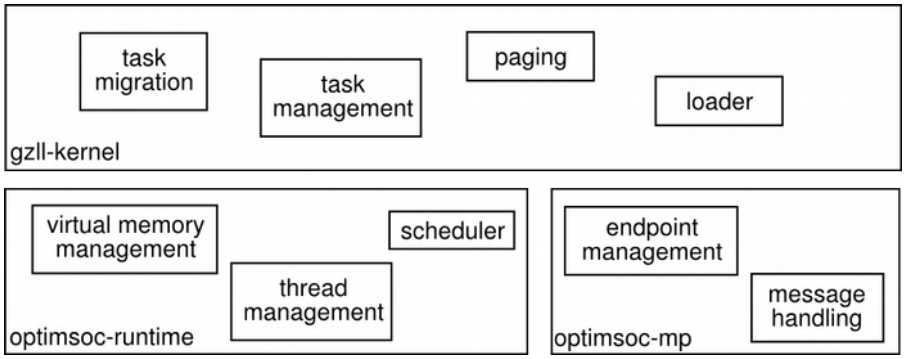


SoC Software

- full C/C++ programming support

- baremetal programming
  - “minimal OS” (like a microcontroller)
  - message passing
  - DMA

- gzll: compute-node OS
  - task management
  - more abstracted communication primitives



Currently work in progress:
- Linux
- LittleKernel (LK)

Host Software

- toolchain
  - for mor1kx
    - GCC or1k toolchain with multi-core extensions
    - newlib C library

- build system

- many scripts for automation of all common tasks

Implementation Targets

- Compiled Simulation
  - Verilator
  - Easy integration with SystemC and DPI (C++) modules
  - cycle-accurate, reasonably fast

- Behavioral RTL-Simulation
  - Vivado XSIM or Modelsim/QuestaSim
  - full simulation, including DRAM and off-chip interface

- FPGA Synthesis
  - Xilinx Vivado

Debug and Trace Support

- we integrate Open SoC Debug
  - shared components with Cambridge (lowrisc) and ETH (PULP)
- Run-Control Debug
  - Currently not upstream, internal prototype available
- memory access
  - read and write of all memories from host
- instruction traces
- other diagnosis modules: event generators [working on integration]
- system trace

System Trace Example

- Key-Value trace messages

```
…
// Send a trace message in
// the SoC software
OPTIMSOC_TRACE(0x200, 0xdeadbeef);
…
```

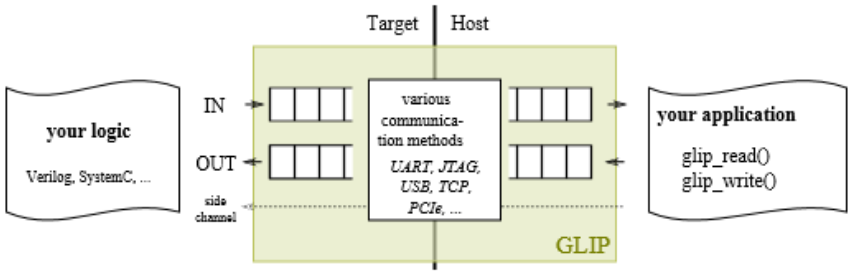
```
$ osd-cli
> stm log stm.log 2
> start
$ cat stm.log
4336d4fc 0200 deadbeef
```

- printf() through the debug interface

```
…
printf("Hello World!");
…
```

```
$ osd-cli
> stm log stm.log 2
> start
$ cat stm.log
86df758b Hello World!
```

Communication



- simple FIFO interface on host and target
- same interface for simulation and FPGA
- actual interface is hidden from user in backend
- currently available backends
  - UART with ~ 10 MBit/s
  - JTAG with ~ 5 MBit/s
  - USB 2.0 with ~ 20 MByte/s
  - USB 3.0 with ~ 90 MByte/s [currently WIP]
  - For simulations on a PC: TCP
- http://www.glip.io/

all speeds for bidirectional (full-duplex) transfers, net data rate

Communication: GLIP API example

```c
/*
 * open a connection to the target (in this case a TCP connection to
 * localhost:12345 with 1 channel.
 */
glip_open(gctx, 1);

/*
 * Send a reset signal to the attached logic. This does *not* reset the
 * communication interface, the signal is only passed through to the user
 * logic.
 */
glip_logic_reset(gctx);

/*
 * Write 4 bytes of data to channel 0 and wait for it up to 1 second
 * (1000 ms).
 */
uint8_t some_data[] = { 0x01, 0x02, 0x03, 0x04 };
size_t size_written;
glip_write_b(gctx, 0, sizeof(some_data), some_data, &size_written, 1*1000);
printf("%zu bytes written to target.\n", size_written);

/*
 * Read up to 4 bytes from the target on channel 0 and wait up to 1 second
 * for it.
 */
uint8_t read_buf[4];
size_t size_read;
glip_read_b(gctx, 0, sizeof(read_buf), read_buf, &size_read, 1*1000);
printf("%zu read from target.\n", size_read);

/* close the connection to the target */
glip_close(gctx);

/* free all resources */
glip_free(gctx);
```

Build System

- dependency tracking (module X needs module Y)
- dependencies can be fetched from external repositories
- full automation of all involved tools without writing scripts or using the GUI
- based on FuseSoC

workflow

1. write description file
2. run fusesoc
3. done

Build System: fusesoc example

```
CAPI=1
[main]
name = optimsoc:examples:compute_tile_nexys4ddr
description = "Xilinx/Digilent Nexys4 with ct"
depend =
  wallento:boards:nexys4ddr
  wallento:svchannels:nasti
  wallento:svchannels:wishbone
  wallento:wb2axi:wb2axi
  optimsoc:tile:compute_tile_dm
  optimsoc:debug:debug_interface
  opensocdebug:interconnect:debug_ring
  glip:backend:uart

simulators = xsim
```

```
[fileset rtl_files]
file_type = systemVerilogSource
usage = sim synth
files =
  rtl/verilog/compute_tile_dm_nexys4.sv

[fileset include_files]
file_type = verilogSource
is_include_file = true
usage = sim synth
files =
  optimsoc_def.vh

[fileset testbench]
file_type = systemVerilogSource
usage = sim
files =

tbench/verilog/tb_compute_tile_nexys4ddr.sv

[xsim]
top_module = tb_compute_tile_nexys4ddr
part = xc7a100tcsg324-1
```

FPGA Implementation Support

- Board support packages abstract I/O as far as possible
  - memory interface (DRAM)
  - clock generation and reset logic
  - pin descriptions

- Currently available
  - Xilinx Arty
  - Digilent/Xilinx Nexys4 DDR
  - Xilinx VCU108

Documentation

http://www.optimsoc.org/docs/index.html

- Installation guide
- Tutorials
- Reference guide

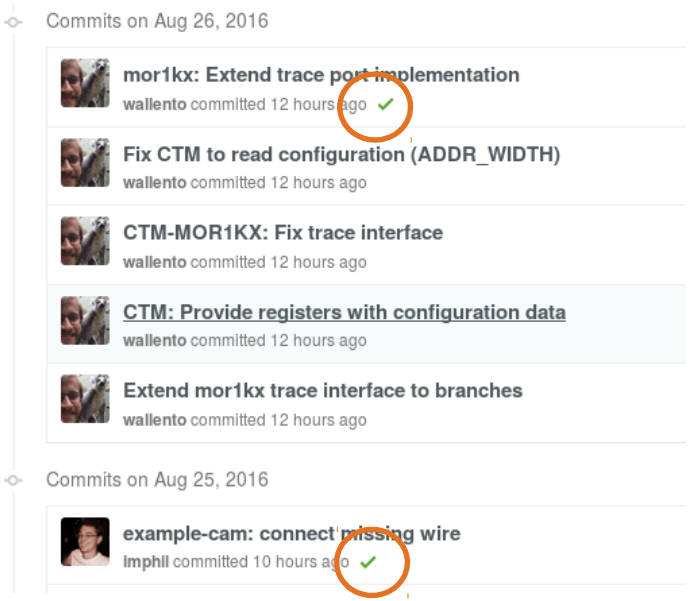
- **Quickstart**
  - less than 10 minutes to setup all required components
  - http://www.optimsoc.org/docs/master/user-guide/chap_installation.html#S2

- More documentation in source code which can be converted to API documentation
  - like http://openrisc.io/newlib/docs/html/index.html for newlib/libgloss
  - or http://www.glip.io/modules.html for glip
  - or https://optimsoc.org/docs/master/api/index.html for the OpTiMSoC HW API

Automated Testing

- Every commit to git is built and tested online
  - build full system
  - system test
    - build compiled simulations
    - run software on them
    - compare output with golden reference

- Synthesis and FPGA testing
  - Build script support
  - In-house automation WIP



Use OpTiMSoC, Extend OpTiMSoC, And More

- Download at www.optimsoc.org
  - Freely usable, MIT licensed
- Questions? Ask me directly or see the homepage for more contact information (including mailing list)
- Contribute
  - Pull requests
  - Issues

Free and Open Source Silicon Foundation

- LibreCores.org
  - Project repository for open source IP cores and associated projects
  - Main focus on quality metrics and trust
  - similar to OpenCores.org
  - LibreCores Continuous Integration
- Licensing: GPL, LGPL, MIT, BSD, ???
- Industry contacts
- Share knowledge, best practices, ...

Save the date for **the** open source digital design conference

ORConf 2017

Hebden Bridge, UK

September 8 – 10, 2017

www.orconf.org

Questions?