Towards a Rocket Chip Based Implementation of the RISC-V GPC Architecture

Paderborn University / Heinz Nixdorf Institute

Lars Luchterhandt[a], Tom Nellius[a], Robert Beck[a],
Rainer Dömer[b], Pascal Kneuper[a], Wolfgang Müller[a], and Babak Sadiye[a]

[a]Paderborn University / Heinz Nixdorf Institute, Paderborn, Germany
[b]University of California, Irvine, USA

Towards a Rocket Chip Based Implementation of the RISC-V GPC Architecture

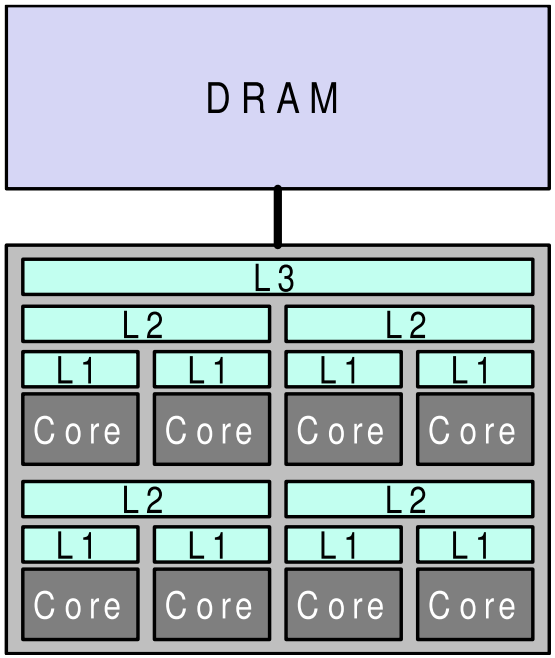
- Grid of Processing Cells (GPC) Platform [1]
  - Prototype implementation at RTL

  - Design a RISC-V SoC of the GPC Platform
  - RTL simulation of SoC
  - Functional test using a demo program

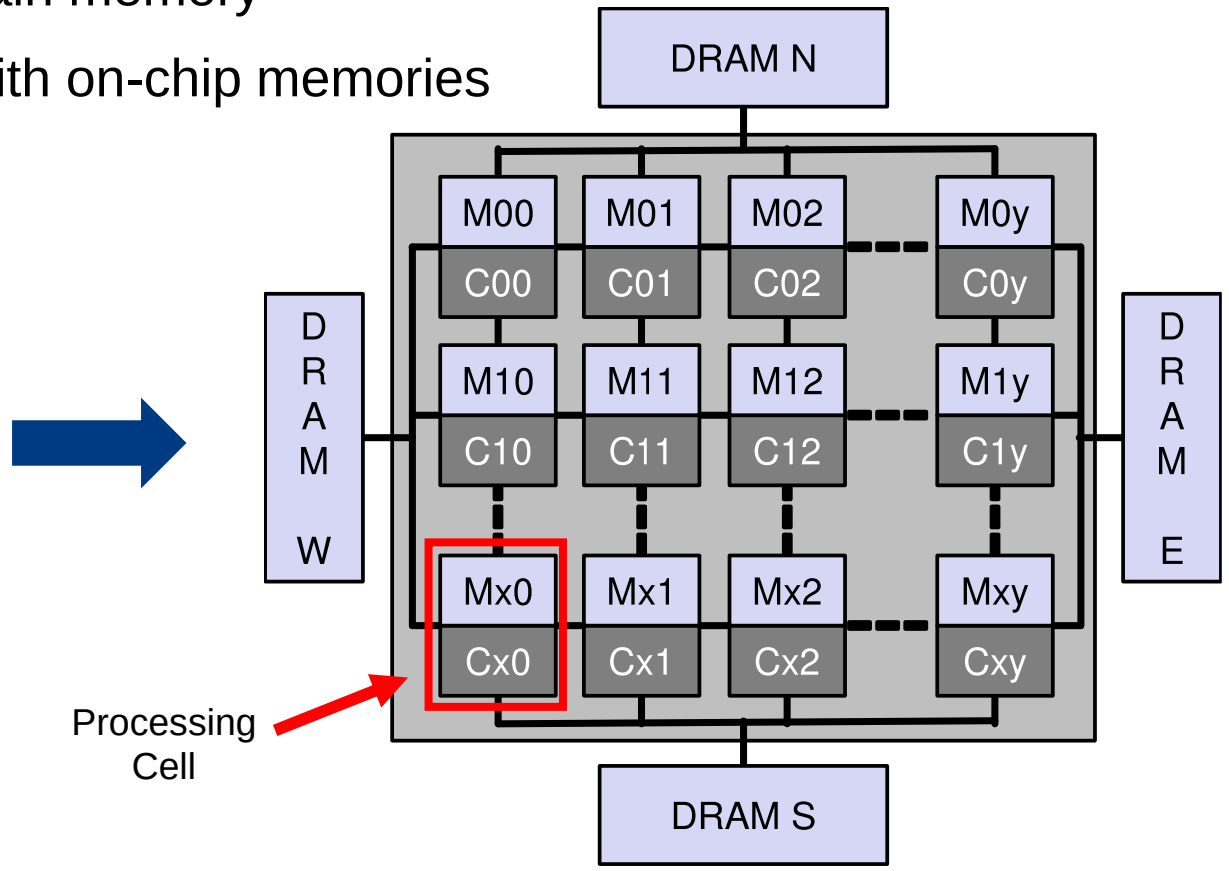
[1] Rainer Dömer. A Grid of Processing Cells (GPC) with Local Memories.
Tech. rep. CECS-TR-22-01. UCI: Center for Embedded and Cyber-physical Systems, Apr. 2022.

GPC Platform – Grid of Processing Cells (GPC)

- Introduced by R. Dömer at UC Irvine
- Addresses bottleneck to main memory
- Replace cache hierarchy with on-chip memories



Multi-core architecture
with cache hierarchy



Grid of Processing Cells

Chipyard Framework

- Open-source **framework** for RISC-V **SoC development** by UC Berkeley
- Powered by **Chisel** (Hardware Construction Language)

- Chipyard Features…
  - SoC generators
    - Rocket Chip SoC generator
  - RTL simulation
    - Verilator simulator
  - SoC software development
  - Support for VLSI flow & FPGA prototyping flow

Chisel HCL

- Open-source Hardware Construction Language by UC Berkeley
- “**C**onstructing **H**ardware **I**n a **S**cala **E**mbedded **L**anguage”
- Embedded in programming language Scala

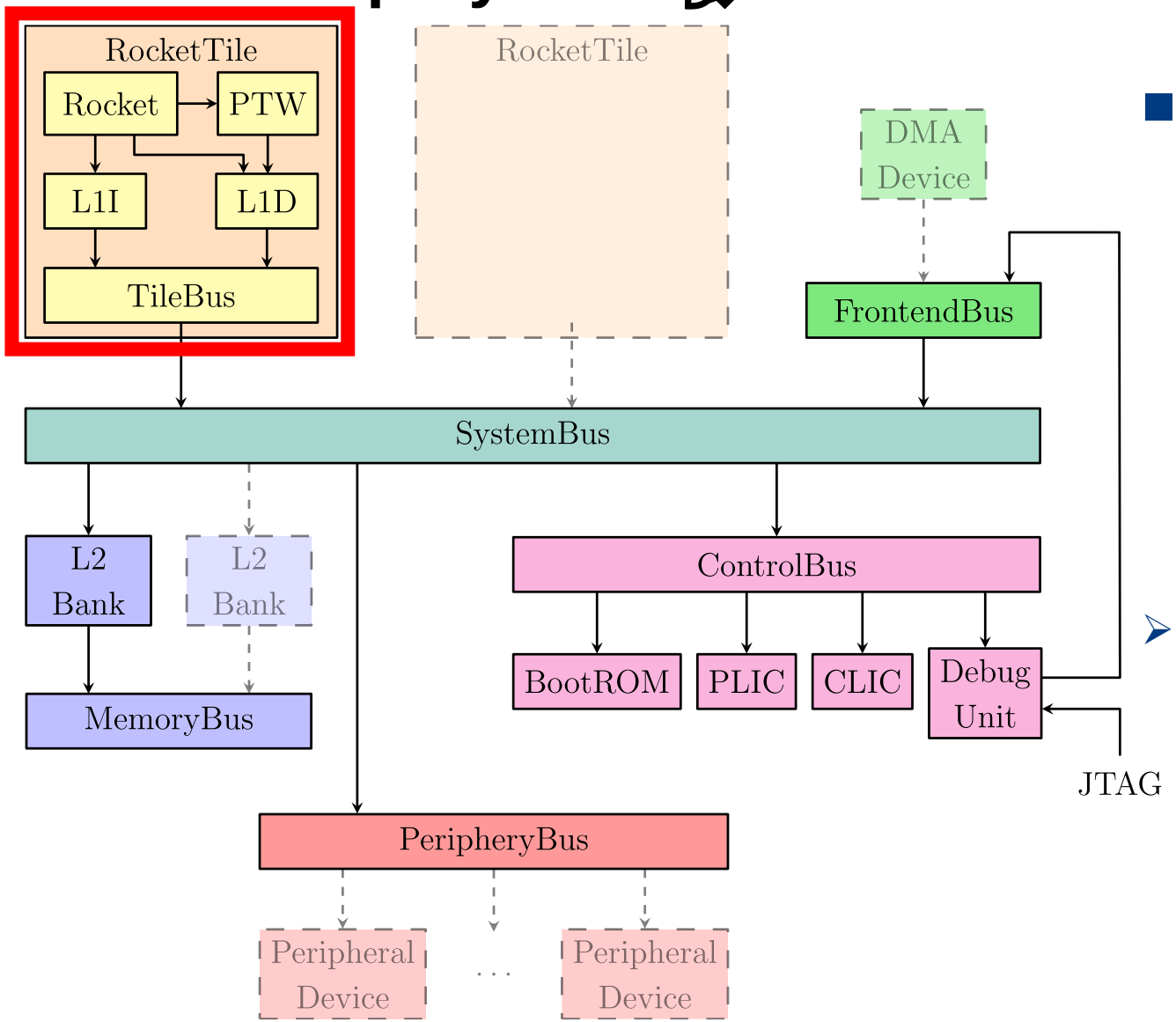
- Benefits from Scala
  - Object-orientated
  - Functional-programming
  - ➢ **Higher** level of **abstraction** in hardware design
- Allows implementation of parameterizable **generators**
- Generators **generate specific circuits**
- **Verilog** HDL can be **emitted**

Rocket Chip Generator

- RISC-V SoC generator written in Chisel
- **Generates** scalable **Rocket Chip system**
- Basis of Chipyard framework
- Highly customizable

- Rocket Chip generator features…
  - **Rocket Core**
    - In-order RISC-V core with five-stage pipeline
  - Memory system
  - On-chip network (TileLink)
  - Debug Unit, Interrupt Controller, …

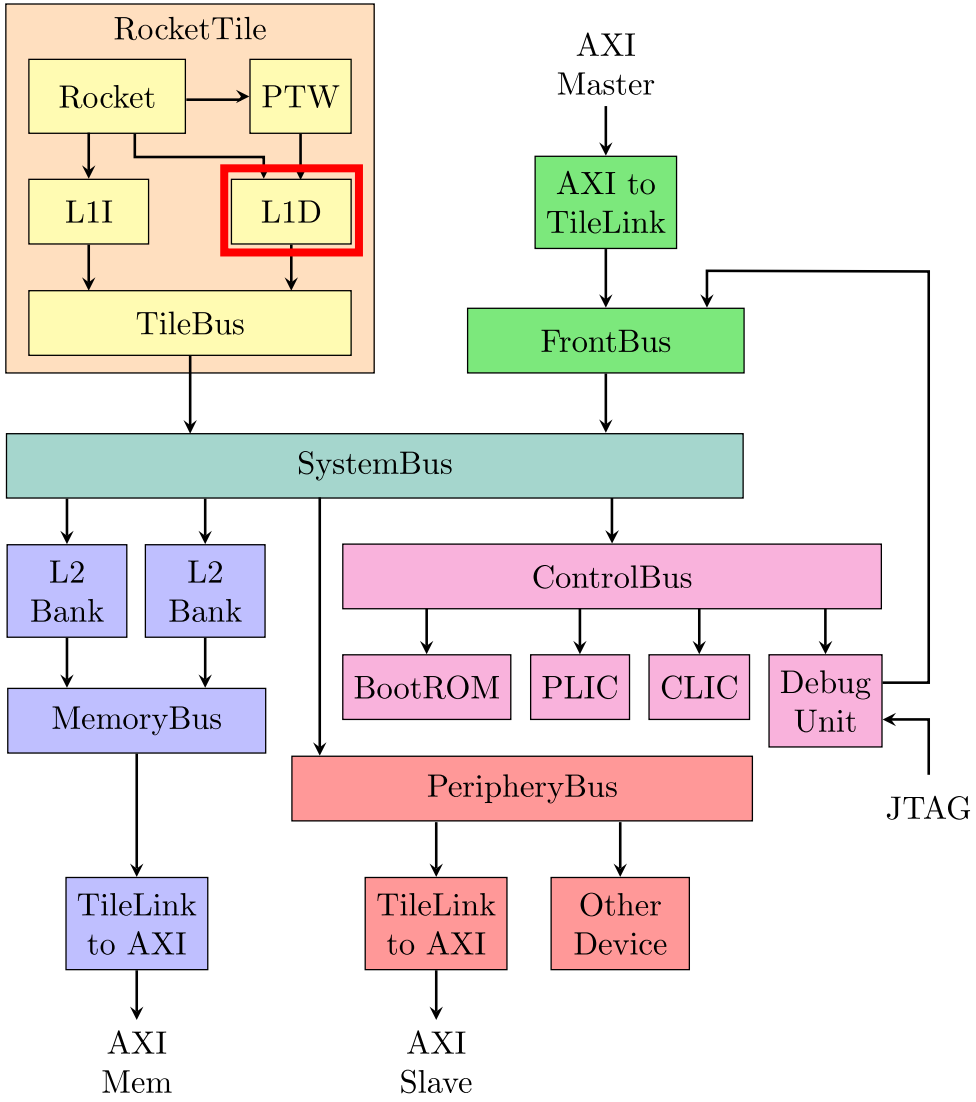
Rocket Chip System



- RocketTile
  - Rocket Core
  - L1 Data Cache
  - L1 Instruction Cache
  - Page-Table Walker
  - SystemBus connection

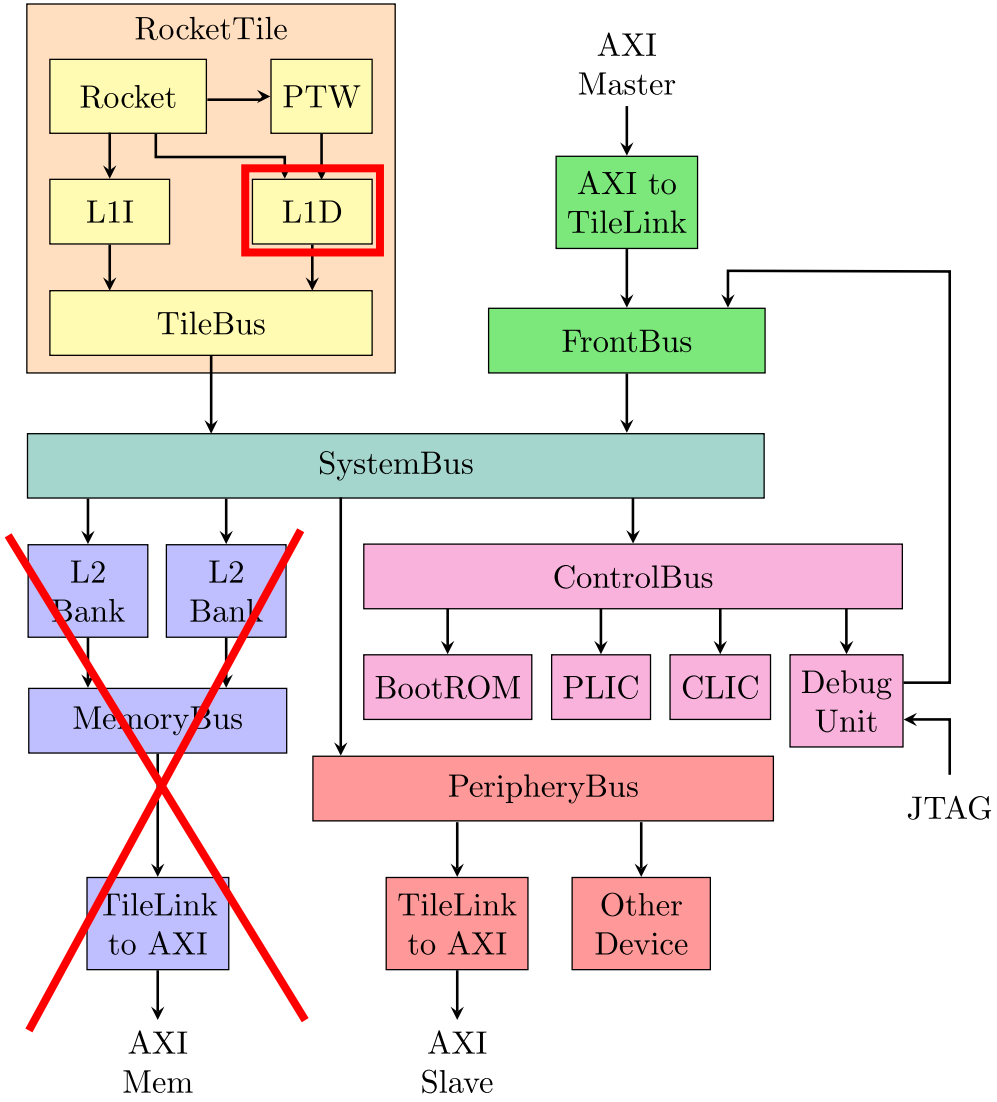
➢ **Starting point** for GPC prototype implementation

Rocket Chip Based GPC Prototype



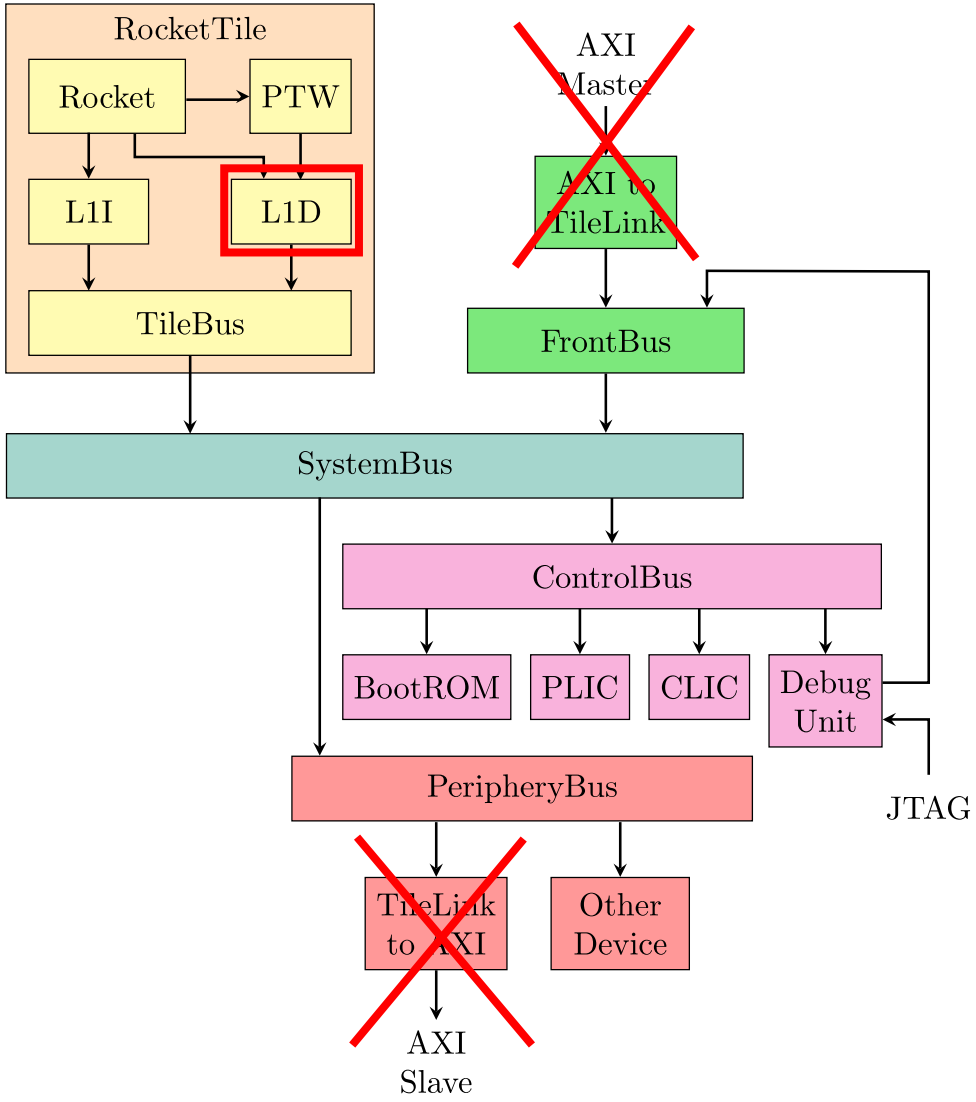
- RocketTile → Processing Cell
- Modifications
  - Scratchpad memory instead of L1-Data cache

Rocket Chip Based GPC Prototype



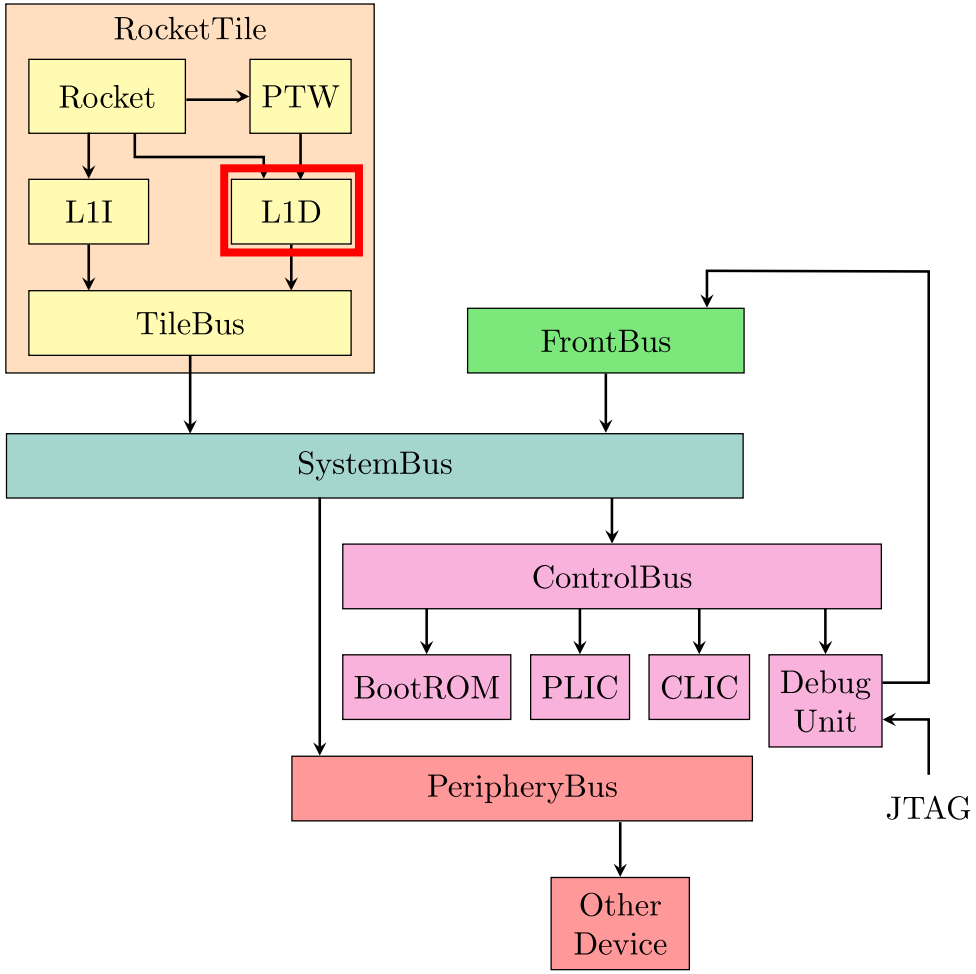
- RocketTile → Processing Cell
- Modifications
  - Scratchpad memory instead of L1-Data cache
  - No L2 cache

Rocket Chip Based GPC Prototype



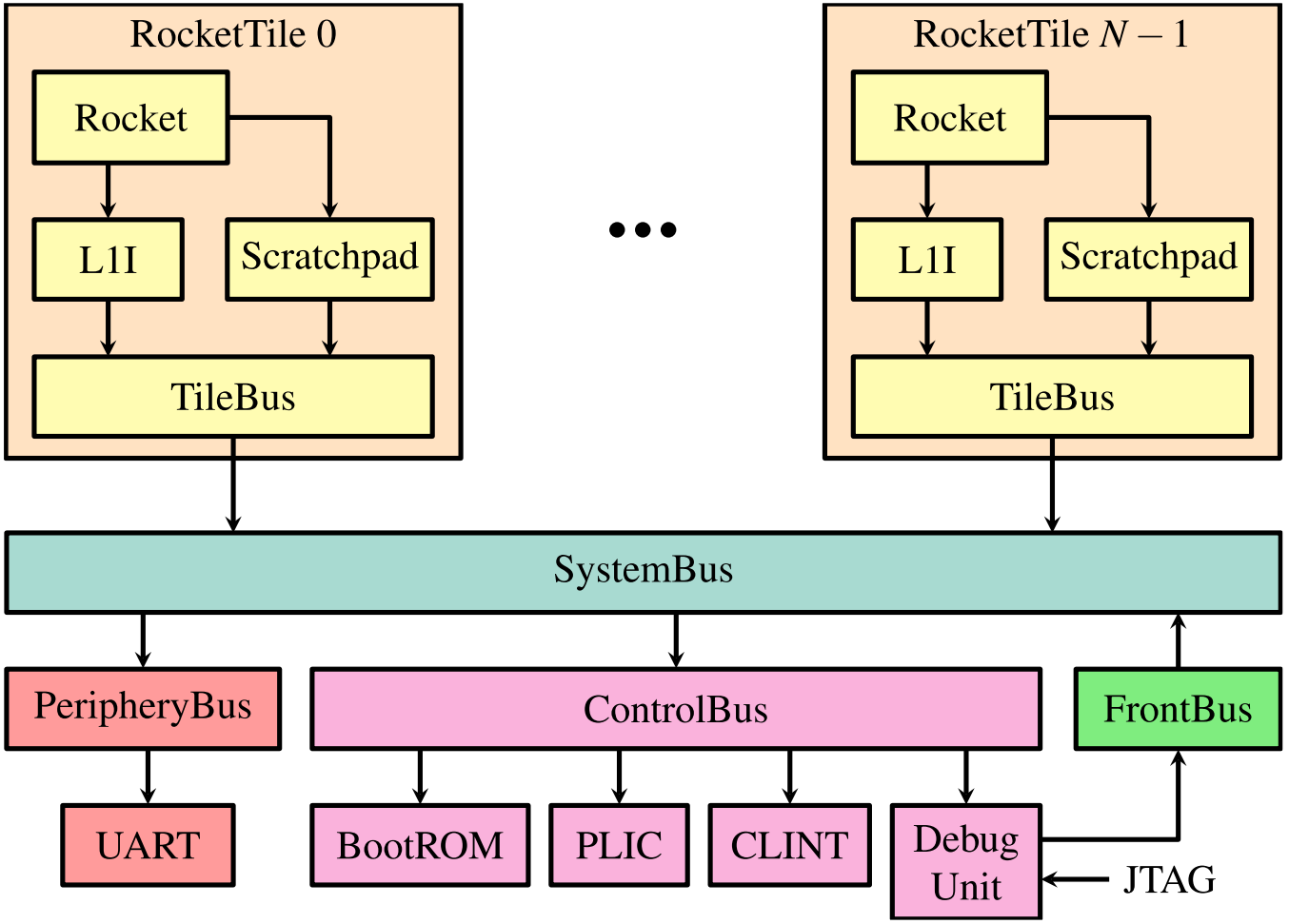
- RocketTile → Processing Cell
- Modifications
  - Scratchpad memory instead of L1-Data cache
  - No L2 cache
  - No AXI interfaces

Rocket Chip Based GPC Prototype



- RocketTile → Processing Cell
- Modifications
  - Scratchpad memory instead of L1-Data cache
  - No L2 cache
  - No AXI interfaces
  - Replicate RocketTiles

Rocket Chip Based GPC Prototype



GPC Prototype

- N RocketTiles
- Each with own scratchpad memory
- Access to neighboring scratchpads via SystemBus

GPC Prototype – Memory Map

- Each core has its own scratchpad memory
- Scratchpads have non-overlapping memory addresses ranges

Base	Top	Device	Size
...	...	...	
0x 8000 0000	0x 8000 8000	Scratchpad 0	32 KiB
0x 8000 8000	0x 8001 0000	Scratchpad 1	32 KiB
0x 8001 0000	0x 8001 8000	Scratchpad 2	32 KiB
0x 8001 8000	0x 8002 0000	Scratchpad 3	32 KiB

Bare-Metal Software Compilation

- Cross-compile C programs for GPC prototype
- Configure RISC-V cross compiler toolchain

- Simple Hello-World demo C program

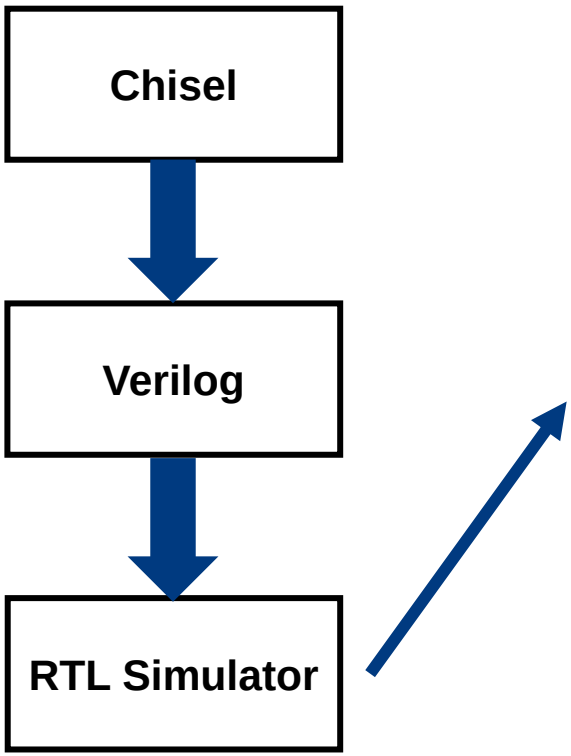
```
1  void thread_entry(int cid, int nc)
2  {
3    printf("Hello from core %d of %d!\n",
4           cid, nc);
5    exit(0);
6  }
```

Further Modifications and Enhancements

- Bootloader modification
  - Rocket cores should boot from own scratchpad memory
- RISC-V Frontend Server (FESVR) modification
  - Load **multiple** binaries to the scratchpad memories
  - Used in RTL simulation

Simulation of the GPC Prototype

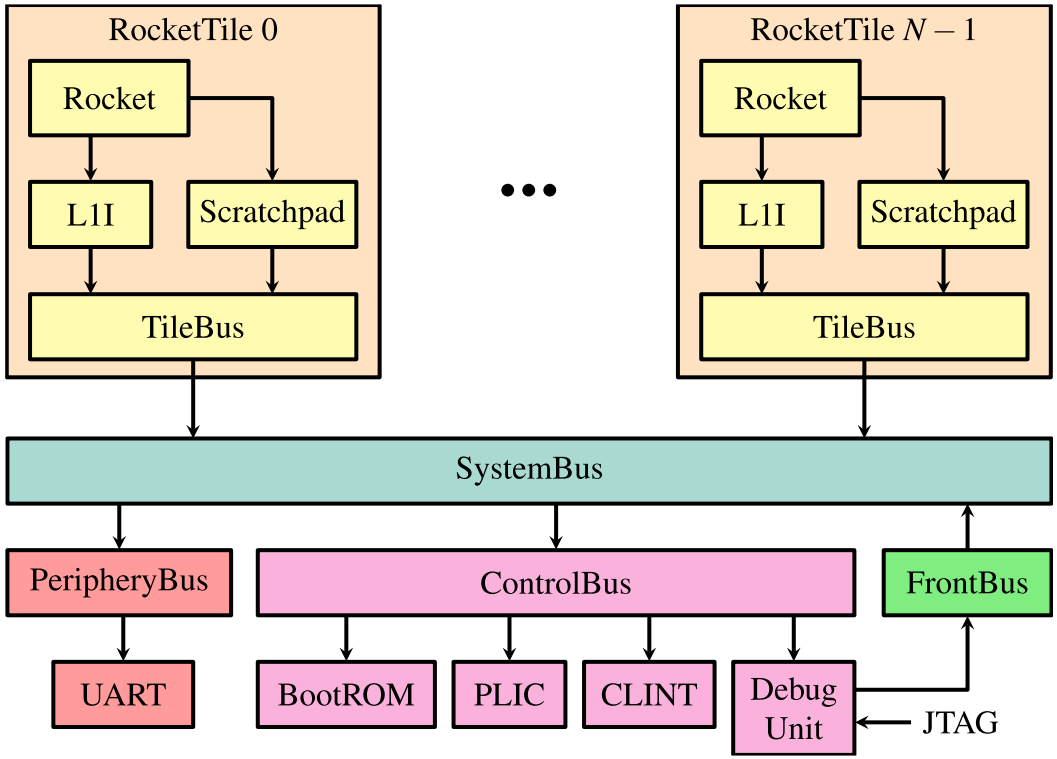
- Open-source Verilog and SystemVerilog simulator
- Generates C++ Code to simulate given Verilog RTL model




```
$ ./simulator-QuadGPCTinyRocketConfig
  ↪  multiload=4,0,0x8000,./hello
Listening on port 33501
[UART] UART0 is here (stdin/stdout).
Loading hello with offset     0x0 done
Loading hello with offset  0x8000 done
Loading hello with offset 0x10000 done
Loading hello with offset 0x18000 done
Hello from core 1 of 4!
Hello from core 2 of 4!
Hello from core 3 of 4!
Hello from core 0 of 4!
```

Conclusion

- Early Rocket Chip Based GPC prototype at RTL
- Functional test by RTL simulation of GPC prototype
- ➢ Promising basis for further implementation



Future Work

- Implement entire **communication architecture** of GPC
  - Avoid SystemBus access
  - Communication between processing cells
  - Virtual address space per processing cells
- FPGA synthesis of GPC Prototype