TIGRA: A Tightly Integrated Generic RISC-V Accelerator Interface

Brad Green, Dr. Melissa C. Smith {wbgreen, mcsmith}@clemson.edu

CLEMSON UNIVERSITY

FCTL FUTURE COMPUTING TECHNOLOGIES LABORATORY

Introduction

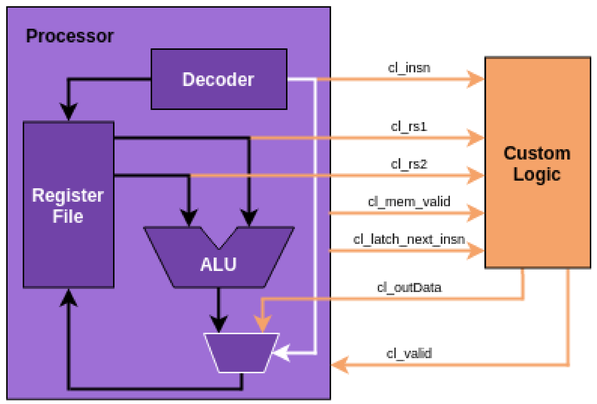
FPGAs can provide an ultra-low latency, low power solution as an accelerator in many HPC applications. TIGRA aims to broaden FPGA usage beyond what is capable today.

Problem: Current FPGA communication occurs via a CPU bus, creating overhead.

Motivation: Hardware exists with CPUs and FPGAs on the same silicon.

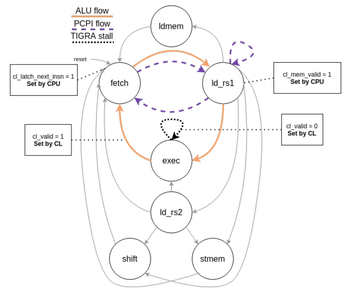
Solution: Create an interface that leverages this hardware to reduce the communication overhead and integrate an FPGA into the CPU pipeline.

TIGRA

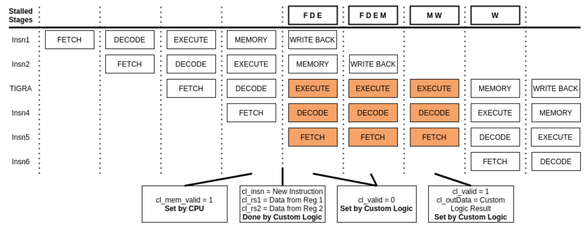


TIGRA: A generic interface between a CPU and an FPGA that introduces 0 latency and provides more flexibility in FPGA designs by integrating into the CPU pipeline directly. The minimal interface, shown by the signals between the processor and custom logic, reduces end-user programming effort.

CPU Design

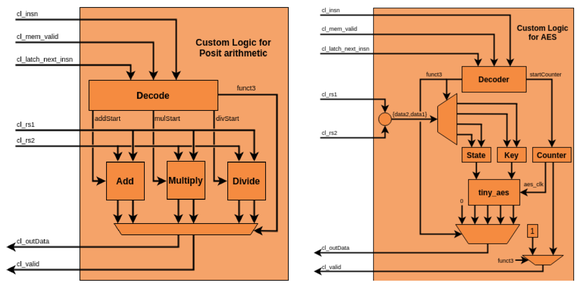


PicoRV32 State Diagram: A simple RISC-V based CPU with the TIGRA interface, used as the initial testing platform. This shows the modifications made to PicoRV32 core to incorporate TIGRA.



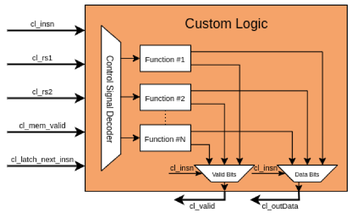
Rocket Pipeline Example: A more feature robust RISC-V CPU chip generator, resembling realistic HPC architecture to test the interface. This shows the modifications made to Rocket to integrate TIGRA and demonstrates the effect of a pipeline stall initiated by the FPGA.

FPGA Design



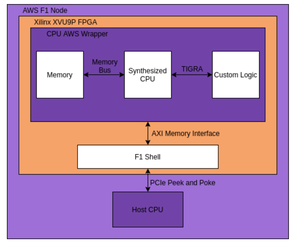
Two example FPGA designs leveraging existing implementations, each showing the required hardware to incorporate these with TIGRA: posit numbers (left), a rising competitor to floating point, and AES encryption (right), a common algorithm used in many HPC designs. Both were tested and verified on both PicoRV32 and the Rocket chip generator.

Expanding the Logic



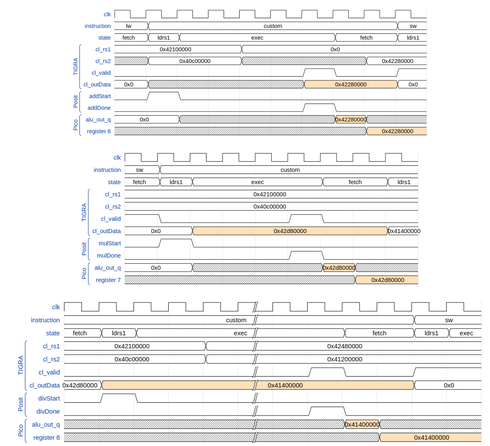
Example FPGA design with multiple incorporated libraries. TIGRA in RISC-V can support 1024 unique instructions, and functions can be combined as shown above with very little additional required hardware.

Incorporation with AWS



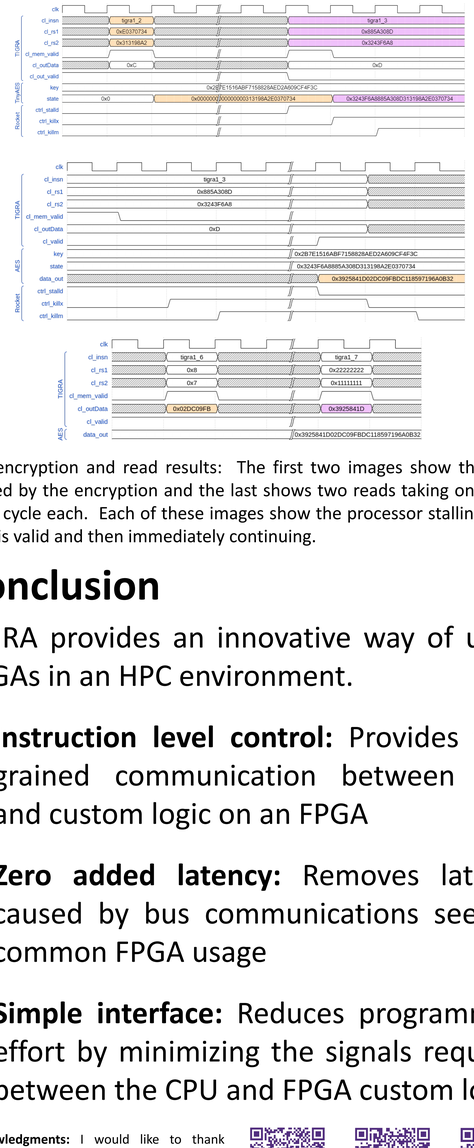
Design in AWS that enables real FPGA testing of the TIGRA interface for verification. This shows the layout of an F1 instance and the logic created to enable testing within the AWS Cloud environment.

PicoRV32 Results with Posits



Posit arithmetic results: add, multiply, and divide from top to bottom, respectively. This demonstrates the stalls required by each function and shows the processor continues as soon as data is valid on the output.

Rocket Chip Results with AES



AES encryption and read results: The first two images show the stall caused by the encryption and the last shows two reads taking only one clock cycle each. Each of these images show the processor stalling until data is valid and then immediately continuing.

Conclusion

TIGRA provides an innovative way of using FPGAs in an HPC environment.

- **Instruction level control:** Provides fine-grained communication between CPU and custom logic on an FPGA
- **Zero added latency:** Removes latency caused by bus communications seen in common FPGA usage
- **Simple interface:** Reduces programming effort by minimizing the signals required between the CPU and FPGA custom logic

Acknowledgments: I would like to thank Amazon Web Services for their generous credit donation which enabled the use of F1 instances for this and future research.

Publications Resume LinkedIn