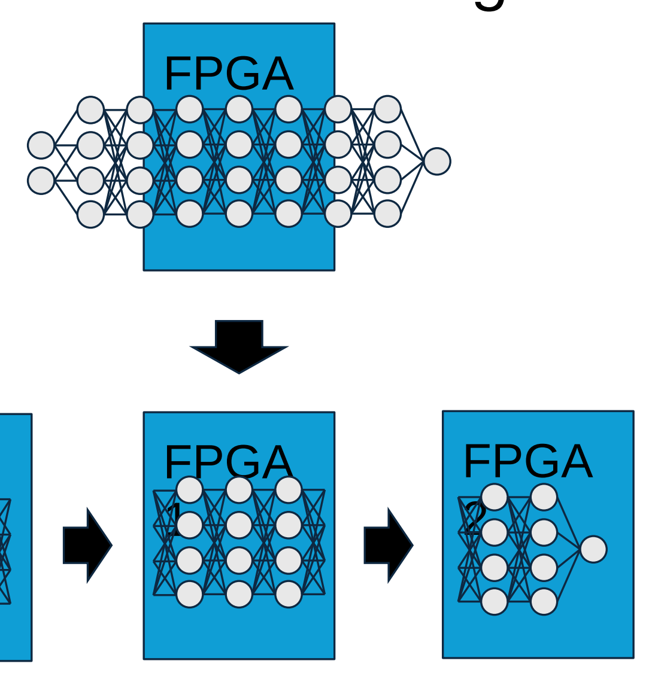
FINN ACCL

Author Streich Georg, De Gendt Antoine

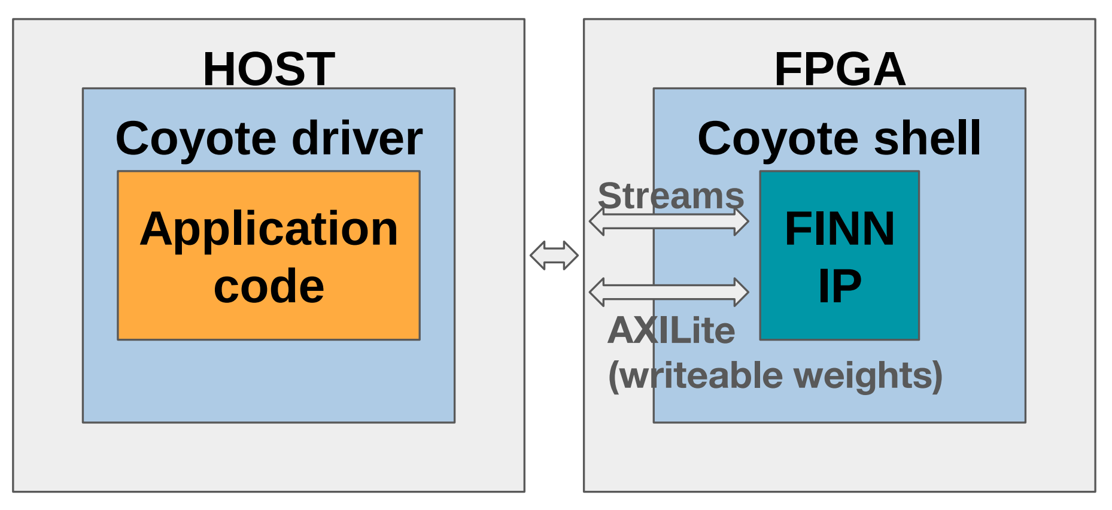
Motivation

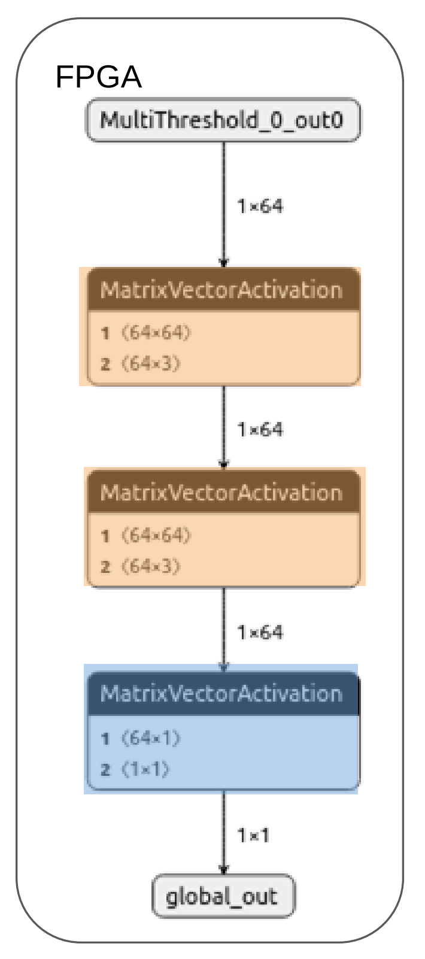
- Neural Networks are too large to fit on a single FPGA
  - U55C has 16GB of HBM memory. Modern LLMs are many times larger even at low quantization
  - Need to be split across multiple boards for efficient execution
- Distributed also benefits smaller networks since having more logic resources allows for a higher unroll factor



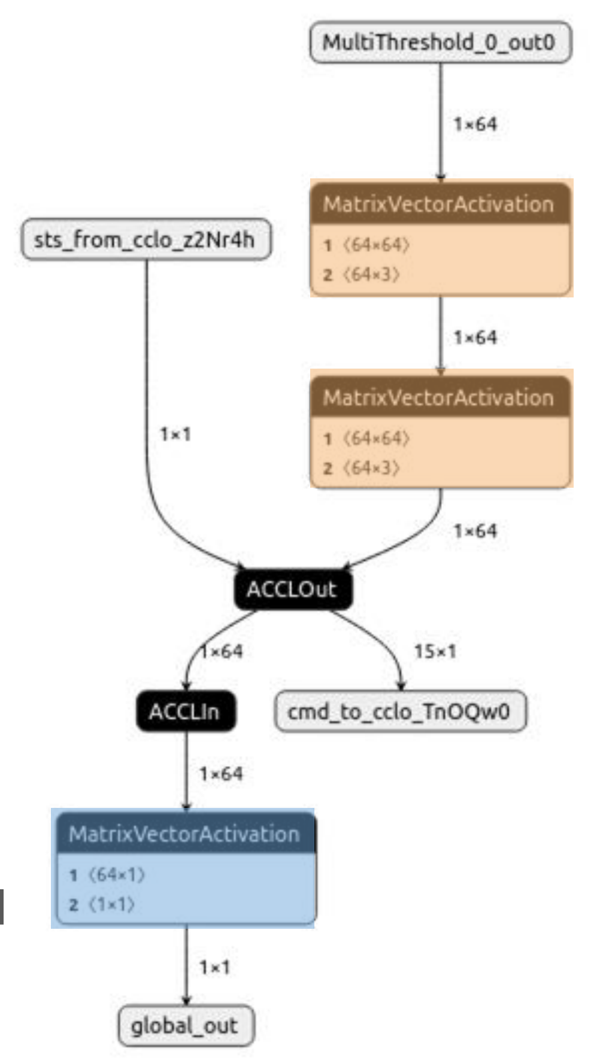
Key Ideas

- Leverage Coyote's ACCL support to easily distribute work across multiple FPGAs
- Introduce ACCL dataflow primitives as FINN nodes to leverage existing compilation infrastructure
- Validate design using a small example network

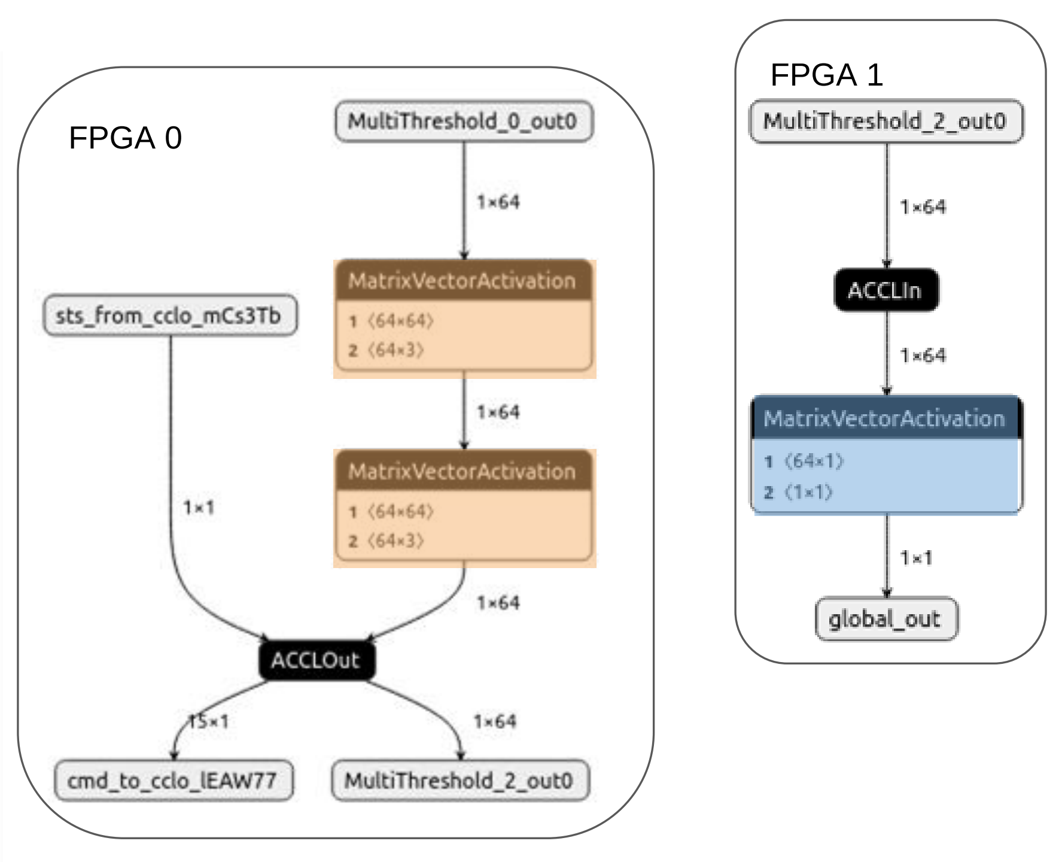




Single FPGA model



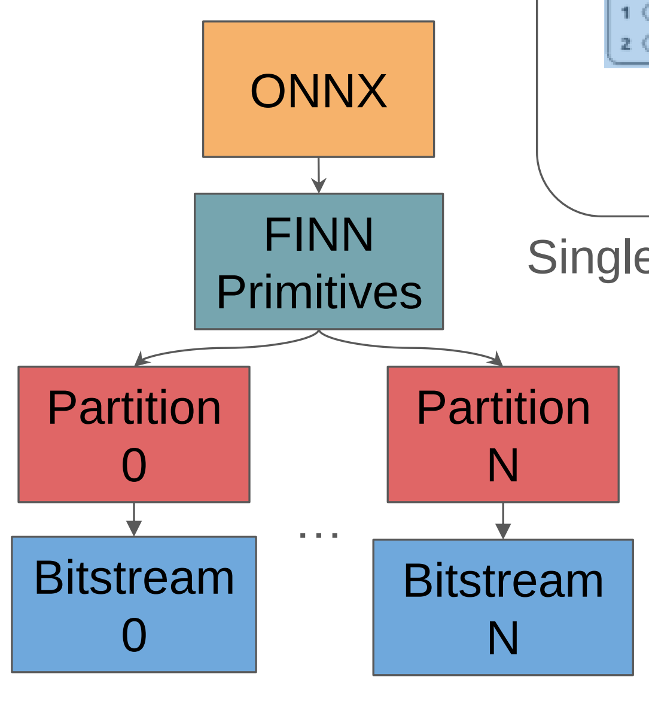
`insert_accl` transformation



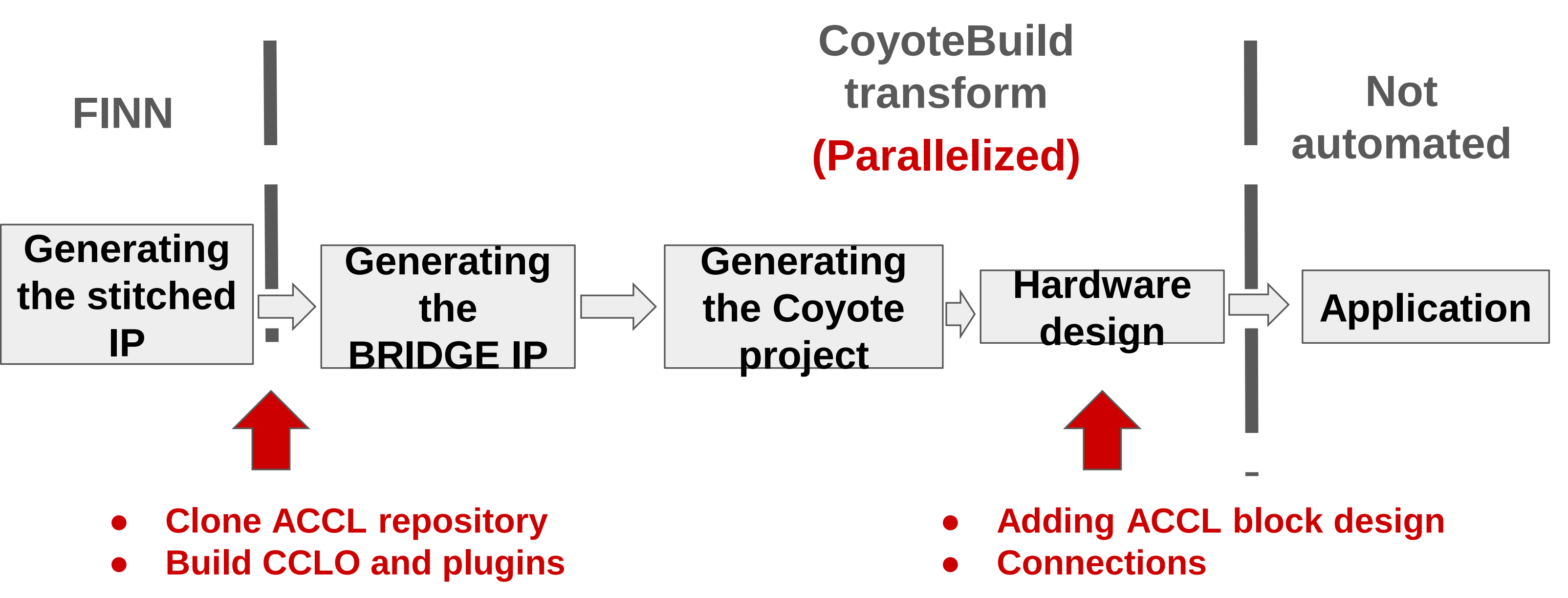
Partitioned model

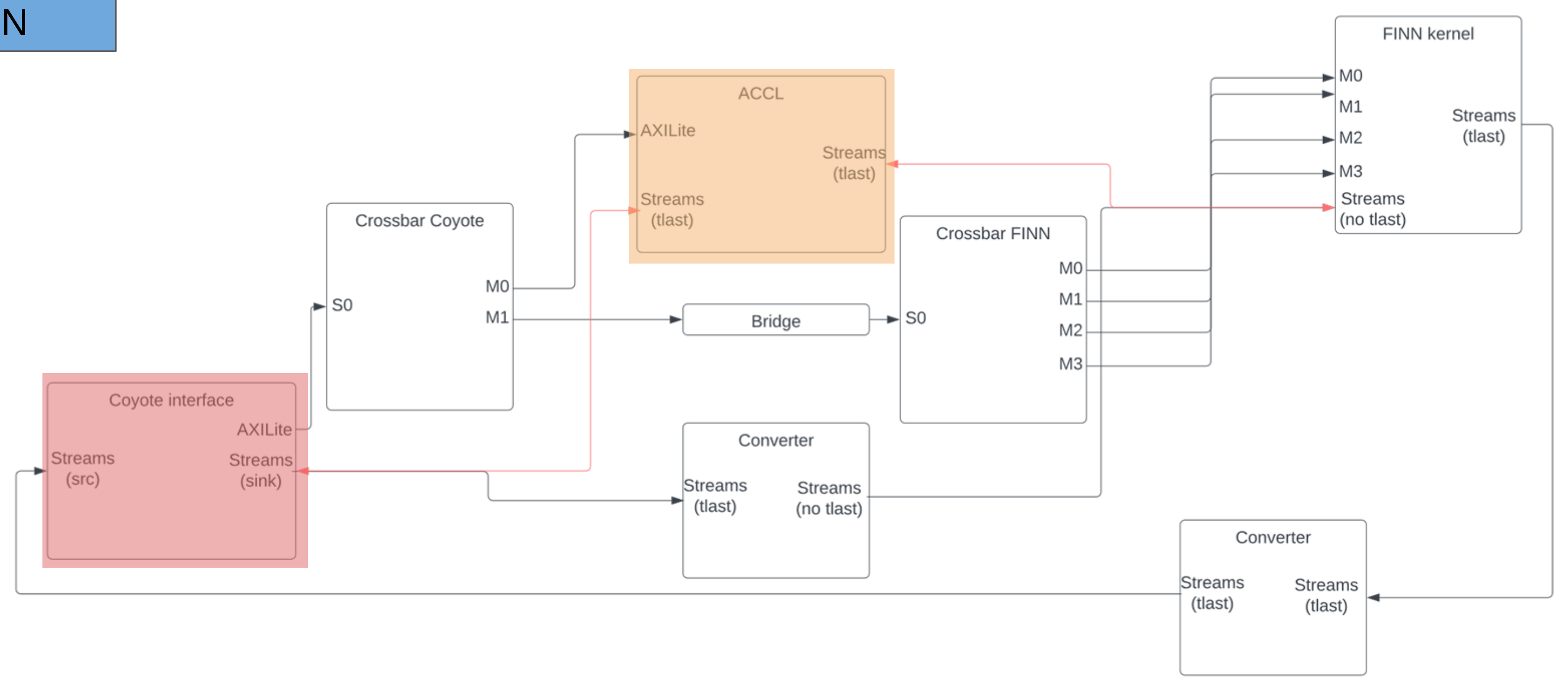
Contribution

- Designed and implemented specialized nodes for integration with ACCL
  - ACCL Out → Sending data to an FPGA
  - ACCL IN → Receiving data
- Coyote compatible design for FINN produced IPs, that supports single and distributed FPGA configurations
- Added passes in the FINN compiler to automate the generation of the design
- Extended build flow infrastructure to enable parallel multi FPGA builds



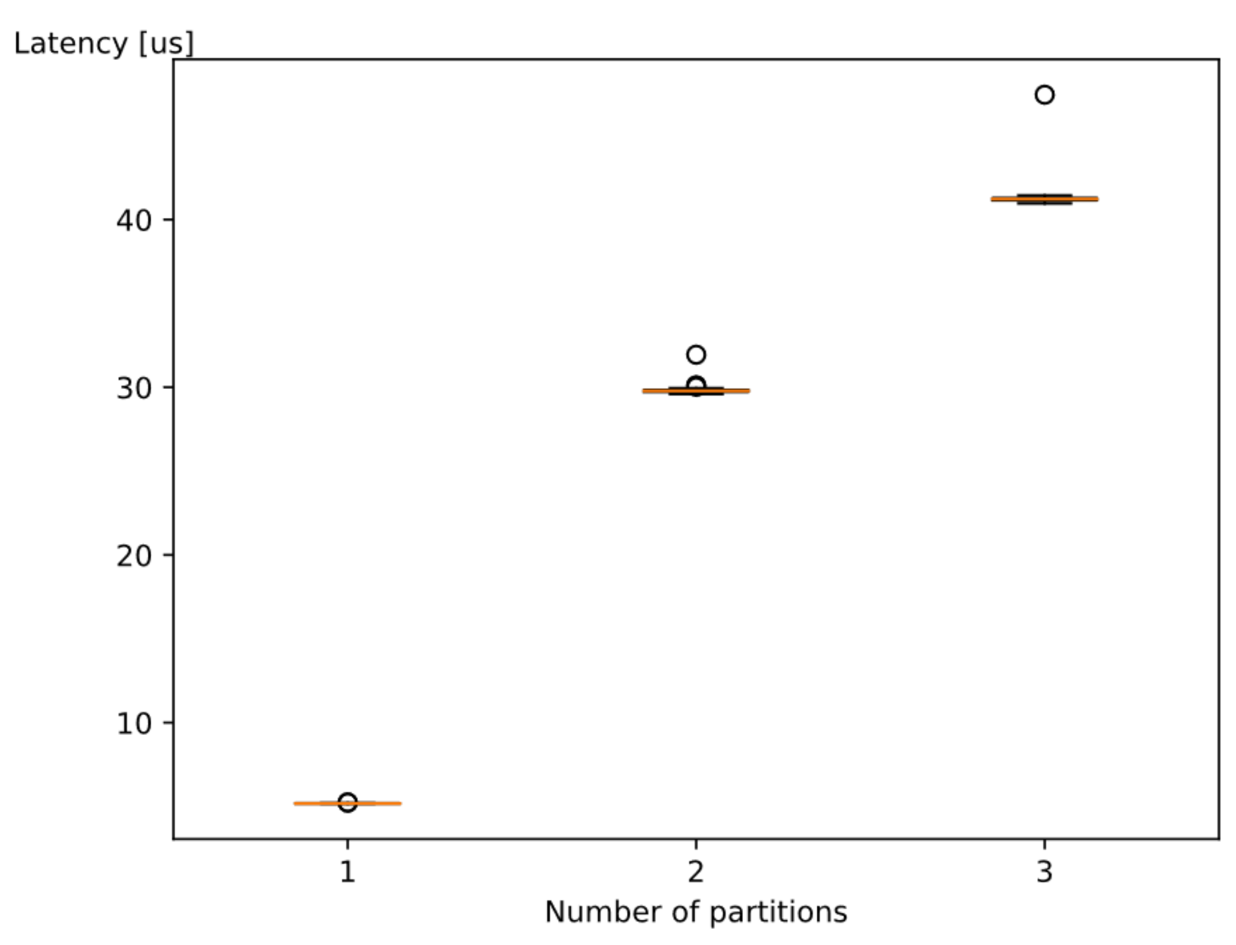
Distributed Build Flow





Future Work

- Additional testing
  - More models, larger and higher complexity
  - Validate on more boards, for now at most 3
- Benchmarking throughput and finer grained latency. For now, only measured end to end latency:



Discussion

- There is an issue with the ACCL out/in design/HLS generation preventing us from using the version of the logic that first sends the op to ACCL and then accepts data. This worked fine in simulation but not in hardware. Therefore, a version where the operations are reversed is used. This has the downside of overflowing the buffer capacity (as input has to be fully buffered before sending to ACCL) for large data transfers.
- External weights are fully supported and are handled through a recursive instantiation of AXI crossbars and a bridge. The level of indirection is necessary due to the limited AXILite address space offered by Coyote. Each host is responsible to write the weights for the partition it has access to. The automation stage outputs a memory map that can be used to write to the appropriate addresses.
- Partitioning increases latency but maintains throughput through pipelining.
- One of the downsides of partitioning is that it makes certain architectures, such as residual networks, harder to implement.

Contact: streichg@student.ethz.ch | adegendt@student.ethz.ch