RIDIM: Reconfigurable stream dataflow computing near memory

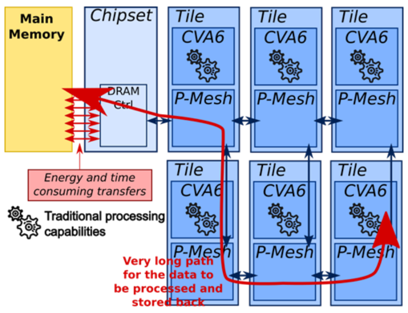
Kevin Martin, Philippe Coussy, Univ. Bretagne-Sud, Lab-STICC
Jean-François Nezan, Maxime Pelcat, INSA Rennes, IETR
Steven Derrien, Univ. RennesI, Irisa/INRIA
Shuvra S. Bhattacharyya, Univ. of Maryland, College Park, USA
and INSA Rennes

Presentation

Holistic software/hardware model by integrating processing capabilities all along the path from the main memory to the processor.
Model of computation "Passive-Active Flow Graph" (PAFG)

Passivization of an actor:

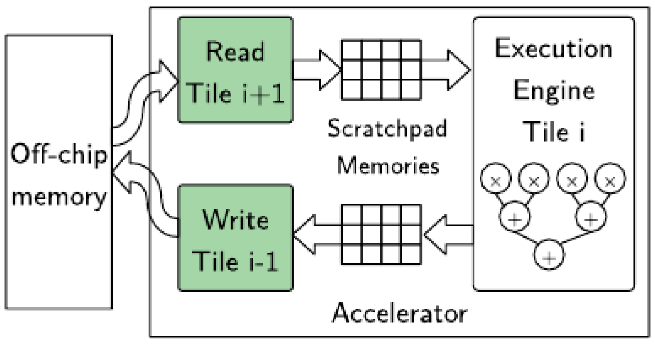
- ✓ No explicit data movement
- ✓ No firing


Main Memory
Chipset
DRAM Ctrl
Tile
CVA6
P-Mesh
Energy and time consuming transfers
Traditional processing capabilities
Very long path for the data to be processed and stored back

Optimization of data movements between chips

Memory bandwidth is known to be a performance bottleneck for FPGA accelerators, especially when they deal with large multi-dimensional data-sets. A large body of work focuses on reducing of off-chip transfers, but few authors try to improve the efficiency of transfers. The later issue is addressed by proposing (i) a compiler-based approach to accelerator's data layout to maximize contiguous access to off-chip memory, and (ii) data packing and runtime compression techniques that take advantage of this layout to further improve memory performance. We show that our approach can decrease the I/O cycles up to 7× compared to un-optimized memory accesses.

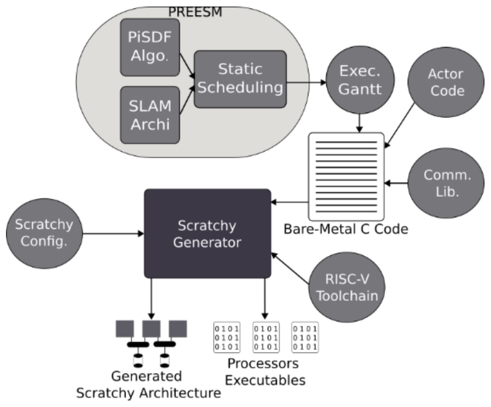
Macro-pipeline structure: read-execute-write. Our contribution focuses on the read and write stages.



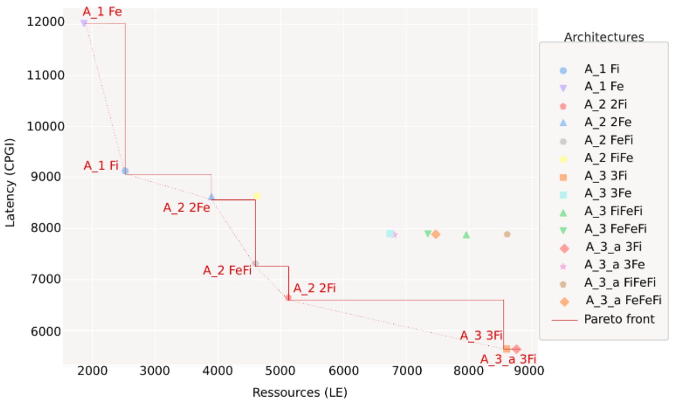
Optimization of data movements inside the computing chip

Exploration of customizable interconnects and organizations

Scratchy is a class of software-managed communication multi-RISC-V adaptable architectures designed for streaming applications. Scratchy uses scratchpad memories and offers customizable interconnect topology and storage options to optimize synchronization and communication time. The custom interconnects can provide many topologies with arbitrary numbers of busses and scratchpad memories. A middleware for Synchronous DataFlow (SDF) applications is provided through the LiteX and PREESM tools for static scheduling. This work demonstrates Scratchy capabilities through a design space exploration test case that aims to derive an efficient multicore topology for executing two SDF-described applications. Additionally, customizing the communication for a 3-core Scratchy only adds 2% resource overhead. The implementations presented in the article [1] were developed using a small Intel MAX10 FPGA with only 205 kB of BRAM. Among the architectures implemented, the most resource-intensive takes less than 5 minutes to synthesize.



Design flow proposed in Scratchy.



Resources vs Cycles Per Graph Iterations (CPGI)

Publications

[1] Joseph W Faye, Naouel Haggui, Florent Kermarrec, Kevin J M Martin, Shuvra Bhattacharyya, Jean-François Nezan, Maxime Pelcat. Scratchy : A Class of Adaptable Architectures with Software-Managed Communication for Edge Streaming Applications. DASIP 2024.

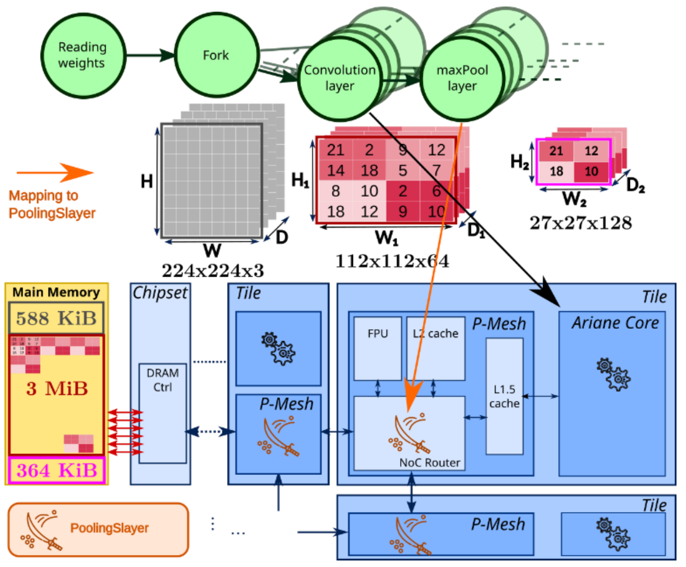
[2] Corentin Ferry, Nicolas Derumigny, Steven Derrien, and Sanjay Rajopadhye. "An Irredundant and Compressed Data Layout to Optimize Bandwidth Utilization of FPGA Accelerators." arXiv preprint 2024

Programmability

Starting from a dataflow model of computation, it is possible to automatically detect and derive the actors candidates for computing in network. An ongoing work focuses on how to automatically transform an SDF graph into a PAFG. The PAFG features allow to map the corresponding actors onto the computing routers.

PoolingSlayer

PoolingSlayer: a Computing-in-Network Approach to Accelerate Pooling Layers of AI applications



The hardware solution is embedded inside the routers of the network-on-chip, but is yet programmable through a simple software mechanism that allows a core to configure it for max pooling or token pooling. The experimental results on an OpenPiton platform shows up to 41% performance improvement compared to the baseline approach for vision transformer application, with an energy consumption nearly halved.

Combined Layer	Input Dimensions	Baseline (ms)	PoolingSlayer (ms)	Perf Gain (%)
SqueezeNet				
Conv1 + MaxPool1	224 × 224 × 3	62	44	29.0%
Fire3 + MaxPool2	55 × 55 × 96	55	36	34.5%
Fire5 + MaxPool3	27 × 27 × 128	59	36	39.0%
ResNet-50				
Conv1 + MaxPool1	224 × 224 × 3	120	90	25.0%
VGG-16				
Conv1_1-2 + MaxPool1	224 × 224 × 3	181	122	32.6%
Conv2_1-2 + MaxPool2	112 × 112 × 64	180	121	32.8%
Conv3_1-3 + MaxPool3	56 × 56 × 128	225	140	37.8%
Conv4_1-3 + MaxPool4	28 × 28 × 256	223	140	37.2%
Conv5_1-3 + MaxPool5	14 × 14 × 512	220	142	35.5%
ViT				
Attention + Pool1	16 × 16 tokens × D features	124	74	40.3%
Attention + Pool2-12	8 × 8 tokens × D features	1325	778	41.3%