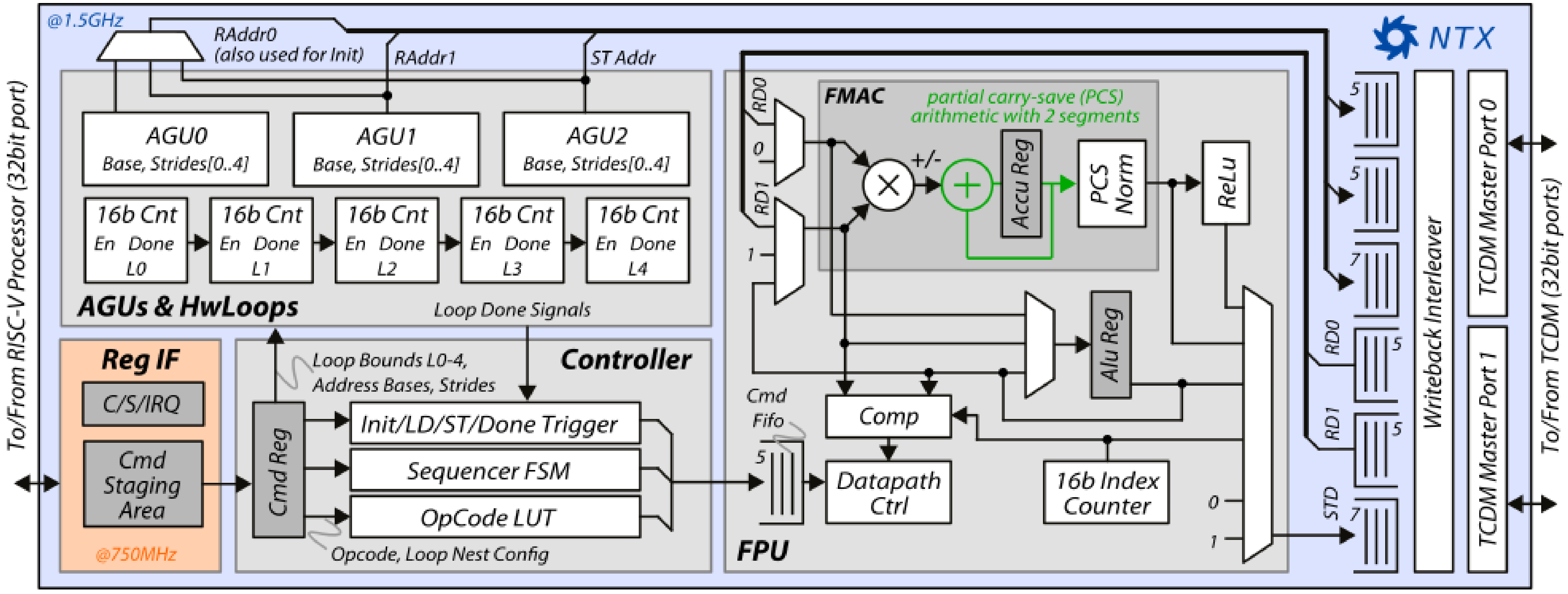
The RISC-V based Stencil Tensor Accelerator of EPI

Matheus Cavalcante
Ph.D. Student, ETH Zürich
4 May 2022

Agenda

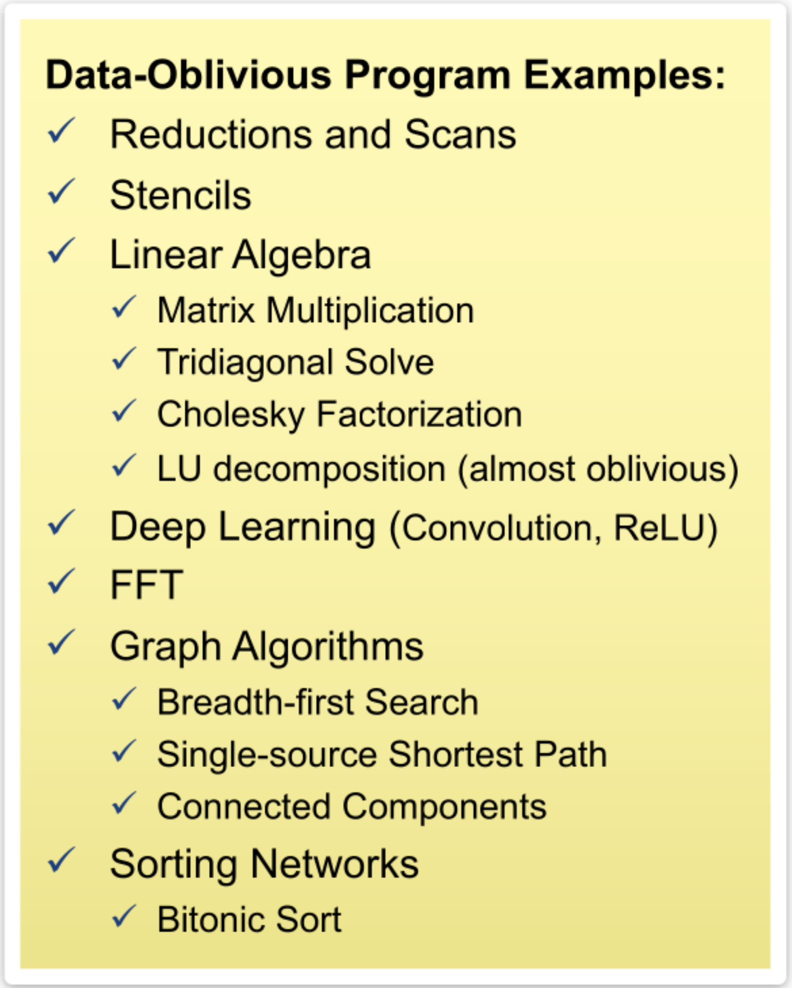
1. NTX Accelerator
2. From NTX to STX: towards stencil acceleration with the STX
3. Physical Implementation

NTX: A 260 Gflop/sW Streaming Accelerator for Oblivious Floating-Point Algorithms



The specialization challenge

- Machine-learning cloud workloads are the norm
- Key challenge: **training algorithms change!**
  - Sparsity in DNNs
  - Novel number formats
- **Avoid overspecialization!**
- GPUs are succesful because they remain flexible
  - Reduction of the VNB thanks to SIMT
  - Memory latency tolerance
- Our approach:
  - **Design an architecture for a large class of programs!**
  - Data-Oblivious algorithms

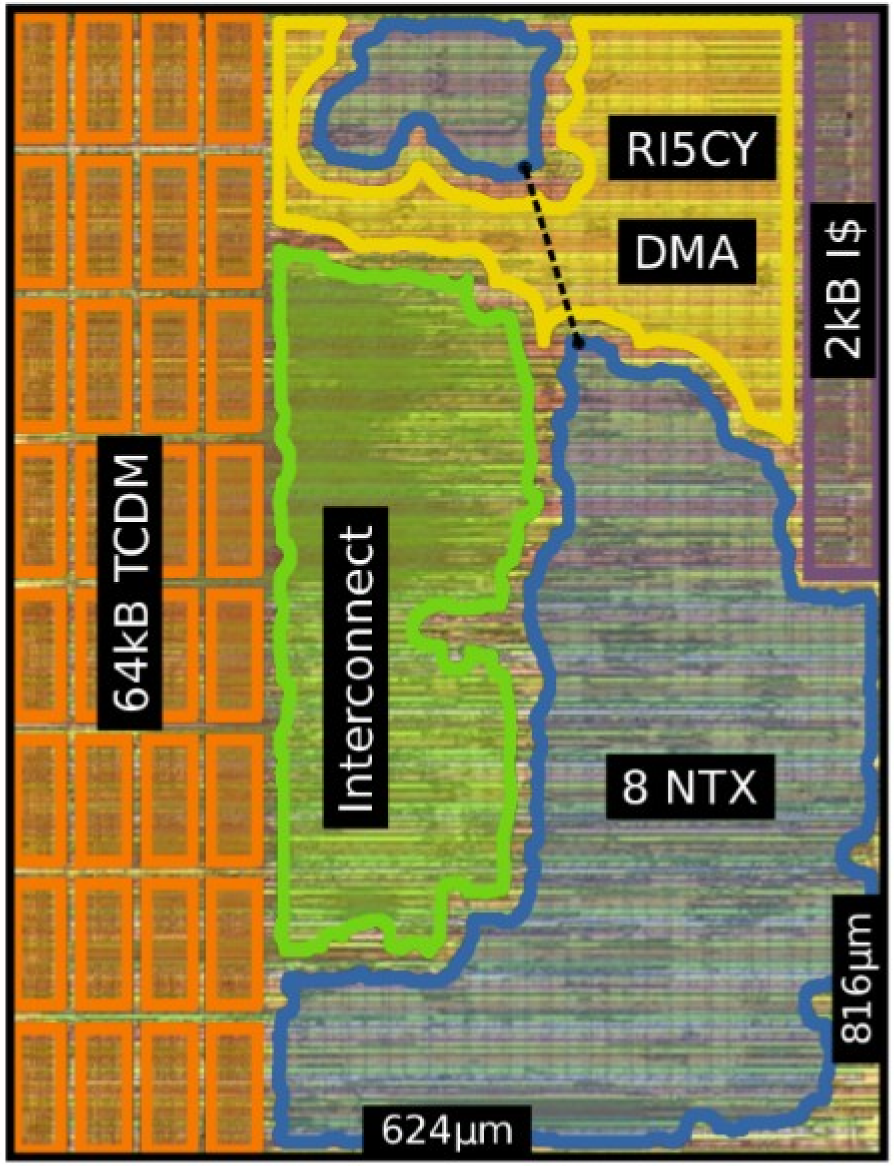


Data-Oblivious Program Examples:

✓ Reductions and Scans
✓ Stencils
✓ Linear Algebra
  ✓ Matrix Multiplication
  ✓ Tridiagonal Solve
  ✓ Cholesky Factorization
  ✓ LU decomposition (almost oblivious)
✓ Deep Learning (Convolution, ReLU)
✓ FFT
✓ Graph Algorithms
  ✓ Breadth-first Search
  ✓ Single-source Shortest Path
  ✓ Connected Components
✓ Sorting Networks
  ✓ Bitonic Sort

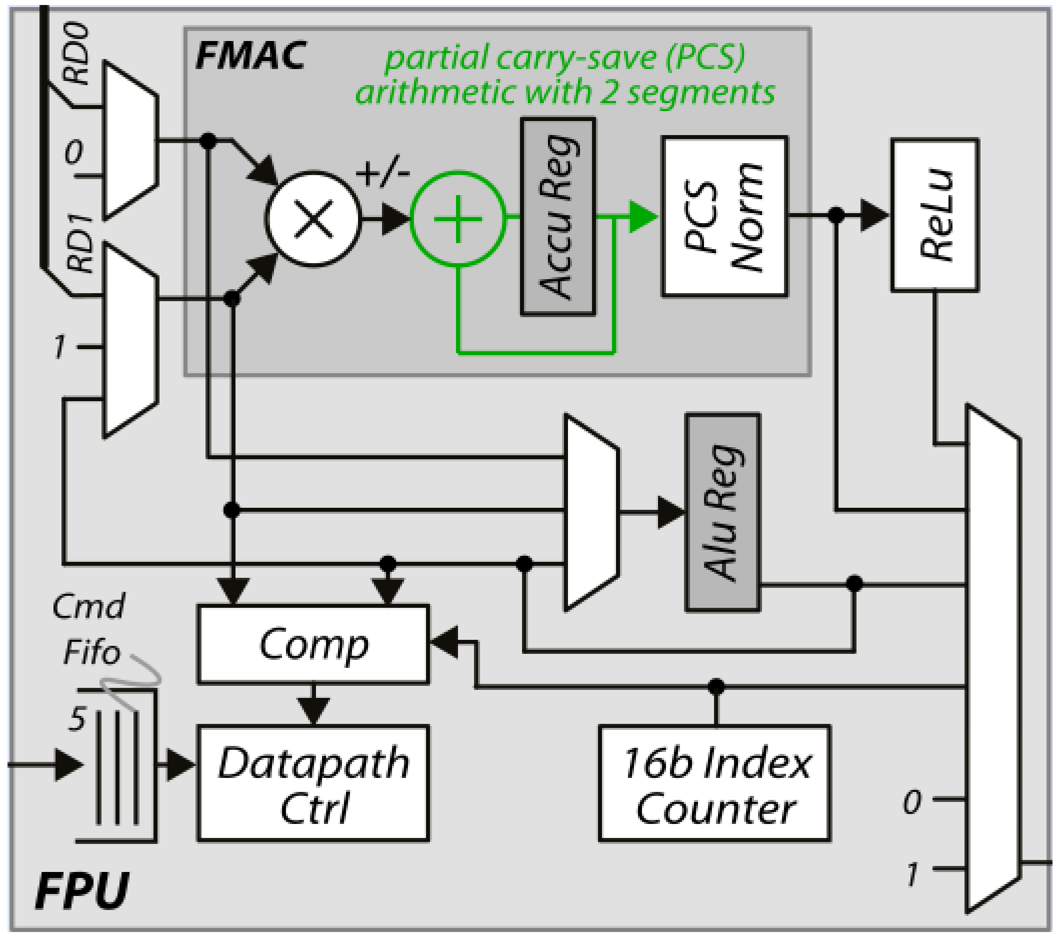
NTX at a glance

- "Network Training Accelerator"
  - 32 bit float streaming co-processor (IEEE 754 compatible)
  - Custom 300 bit "wide-inside" Fused Multiply-Accumulate
- Manufactured in Globalfoundries 22FDX
  - 1 RISC-V core ("RI5CY/CV32E40P") and DMA
  - 8 NTX co-processors
  - 64 kB L1 scratchpad memory
  - 0.5 mm^2, 1.25 GHz worst-case, 166 mW, 0.8 V
- Key ideas to increase hardware efficiency:
  - Reduction of von Neumann bottleneck (**load/store elision through streaming**)
  - Latency hiding through DMA-based **double-buffering**



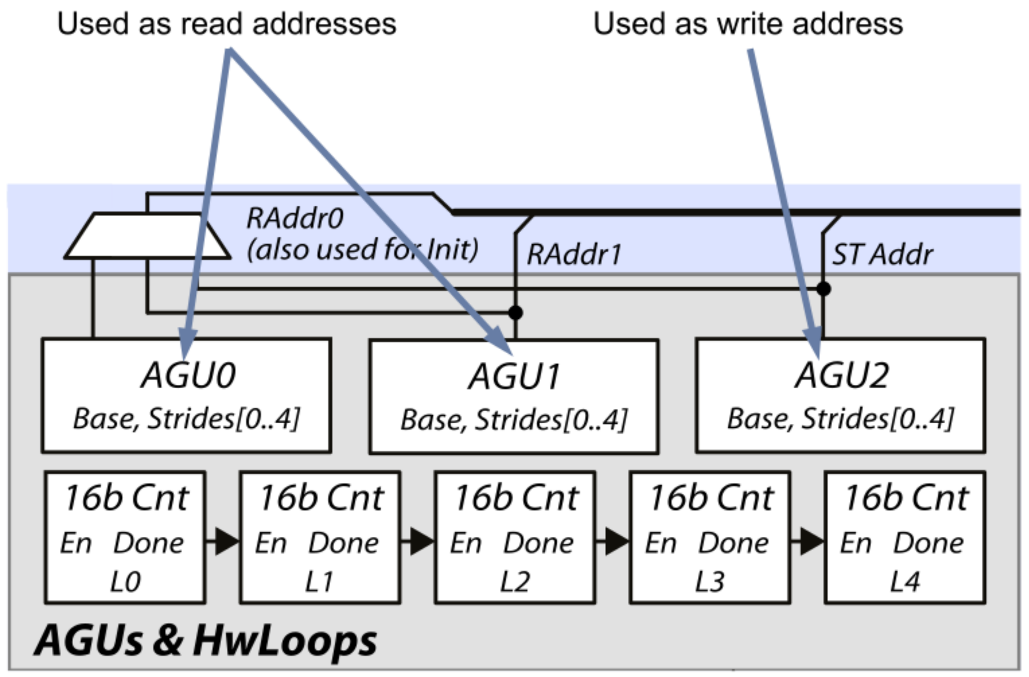
Architecture: Datapath

- FMA operands arrive as **memory streams**
  - Maskable to 0/1 to disable add/mul
- Optional **ReLU** on FMA result
- **Fire-and-forget datapath**
  - Command pushed into FIFO
  - Consumes fixed number of input iter
  - Produces fixed number of output iter



Architecture: Address generator

- 5 nested hardware loop counters
  - 16 bit counter register
  - Configurable number of iterations
  - Once last iteration reached:
    - Reset counter to 0
    - Enable next counter for one cycle
- 3 address generation units
  - 32 bit address register
  - Each has 5 configurable strides, one per loop
  - One stride added to register per cycle
  - Stride corresponds to the highest enabled loop
- Allows for complex address patterns



Programming Model: Loops

- Up to 5 nested loops can be offloaded to NTX
  - Loops should describe a reduction for best performance
  - Covers convolutions, fully connected layers, and more
- Accumulator initialization and writeback is configurable
- For example a DNN convolution:

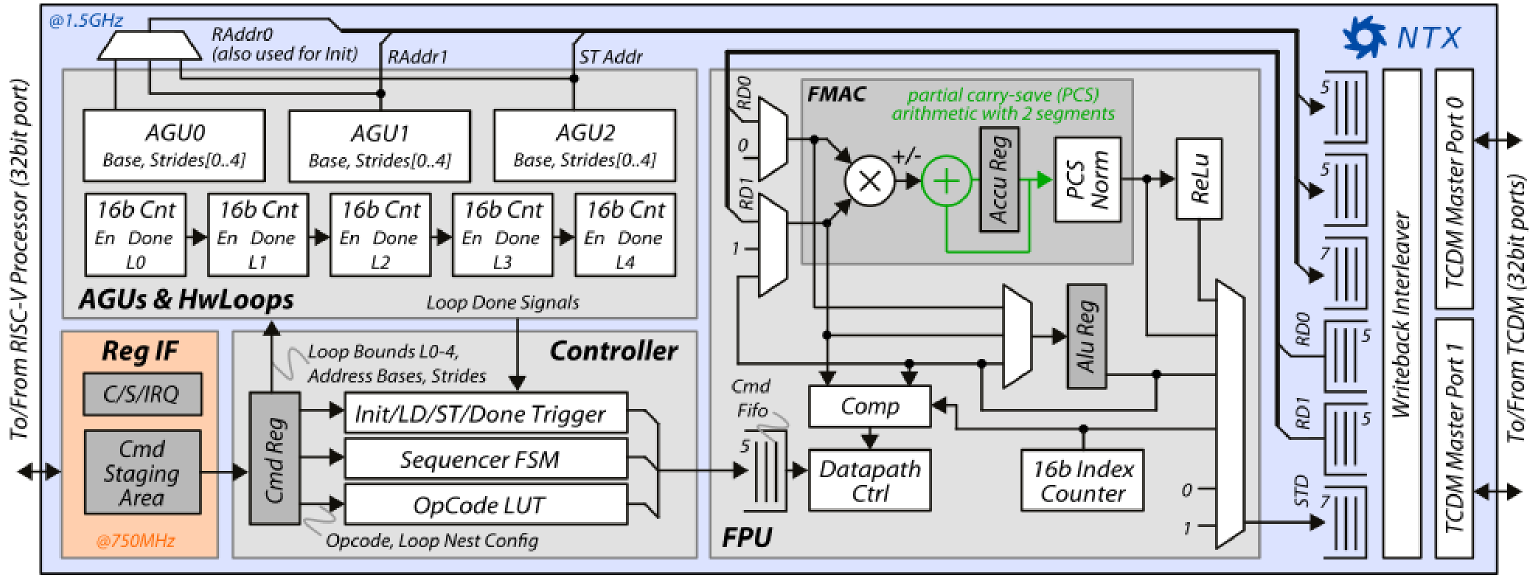
Perform outermost loop level on processor core.

```
for (int k = 0; k < K; ++k)
for (int n = 0; n < N; ++n) Level 4
for (int m = 0; m < M; ++m) { Level 3
    float a = b[k];                        Init Level = 3
    for (int d = 0; d < D; ++d) Level 2
    for (int u = 0; u < U; ++u) Level 1
    for (int v = 0; v < V; ++v) { Level 0
        a += x[d][n+u][m+v] * w[k][d][u][v];
    }
    y[k][n][m] = a;                        Store Level = 3
}
```

NTX

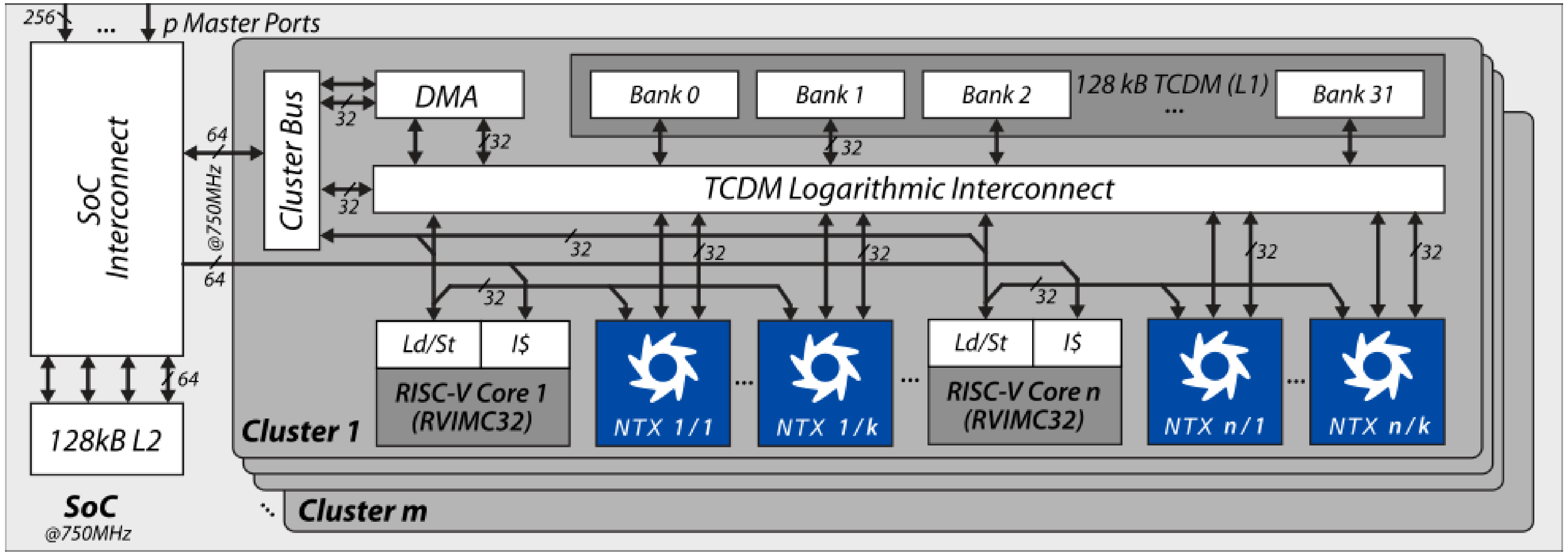
Architecture: NTX

- Processor configures operation via **memory-mapped registers**
- Controller issues AGU, HWL, and FPU micro-commands based on configuration
- Reads/writes data via 2 memory ports (2 operand and 1 writeback streams)
- FIFOs help buffer data path and memory latencies

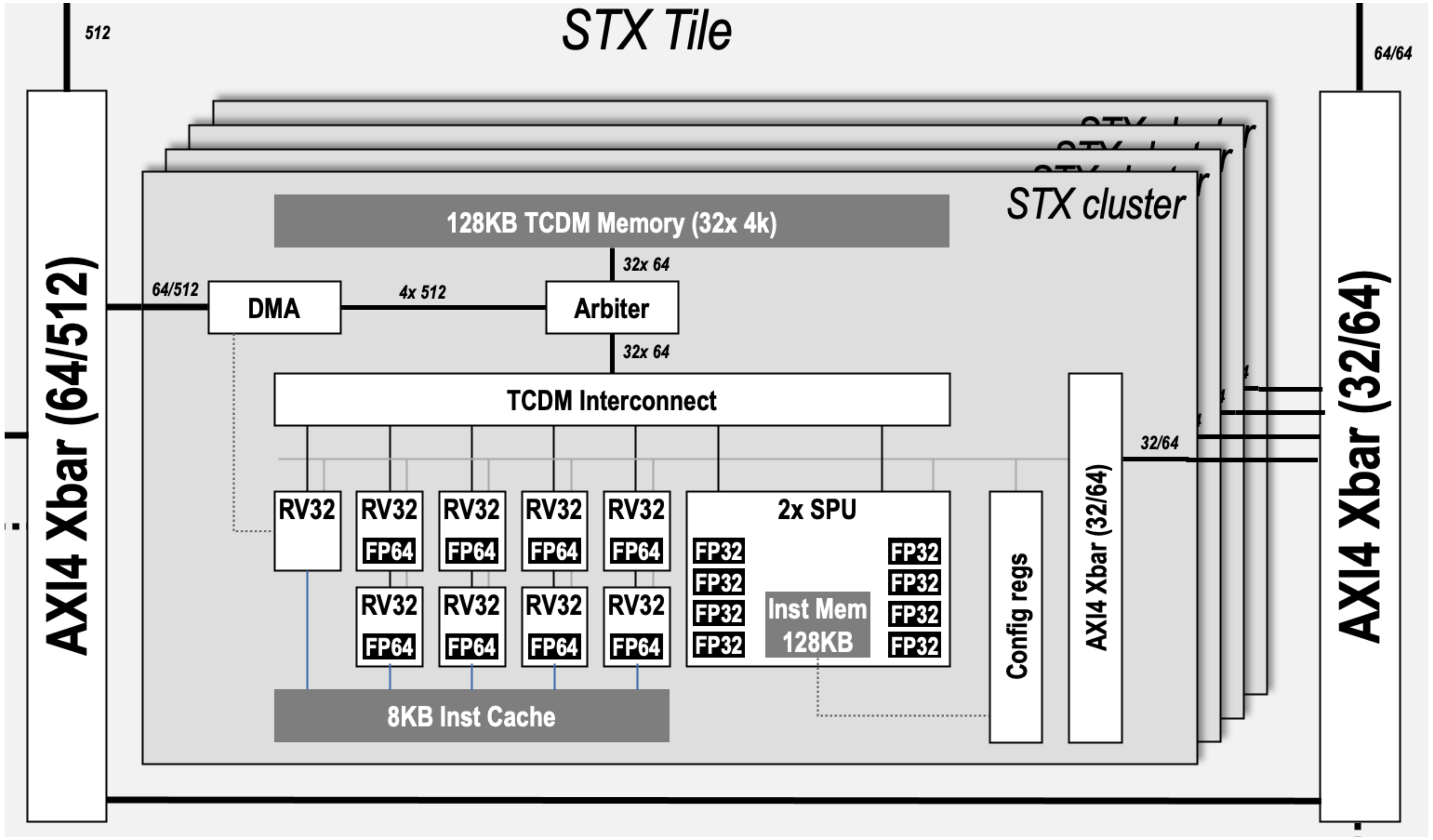


Architecture: Processing cluster

- **1 processor core controls 8 NTX coprocessors**
- Attached to 128 kB shared **TCDM** via a logarithmic interconnect
- **DMA** engine used to transfer data (double buffering)
- Multiple clusters connected via interconnect (crossbar/NoC)

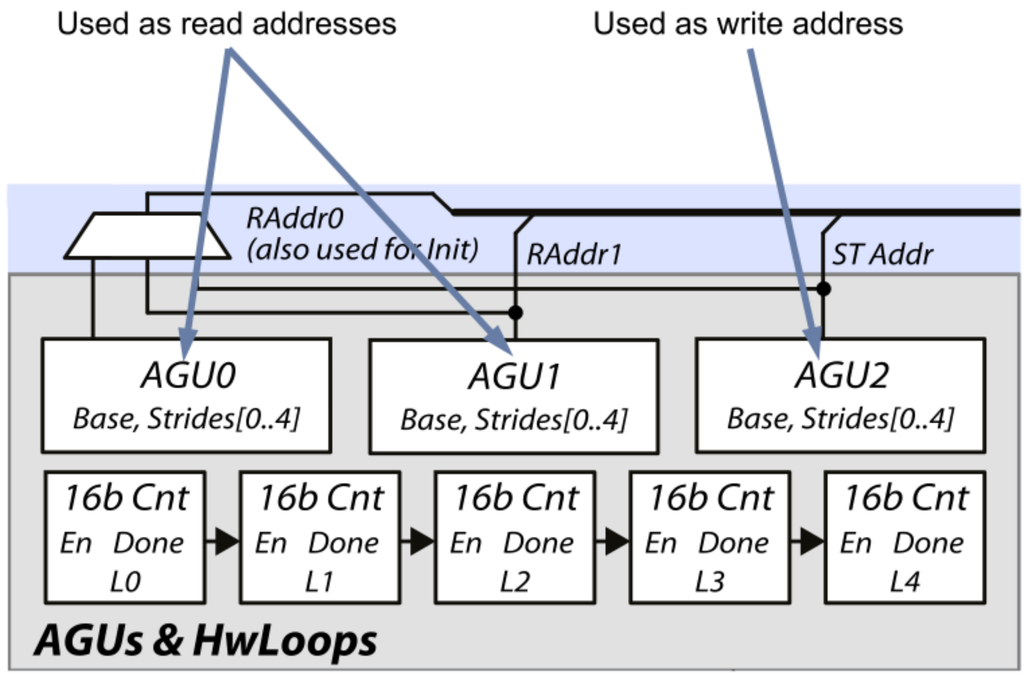


From NTX to STX with SPU and SSRs

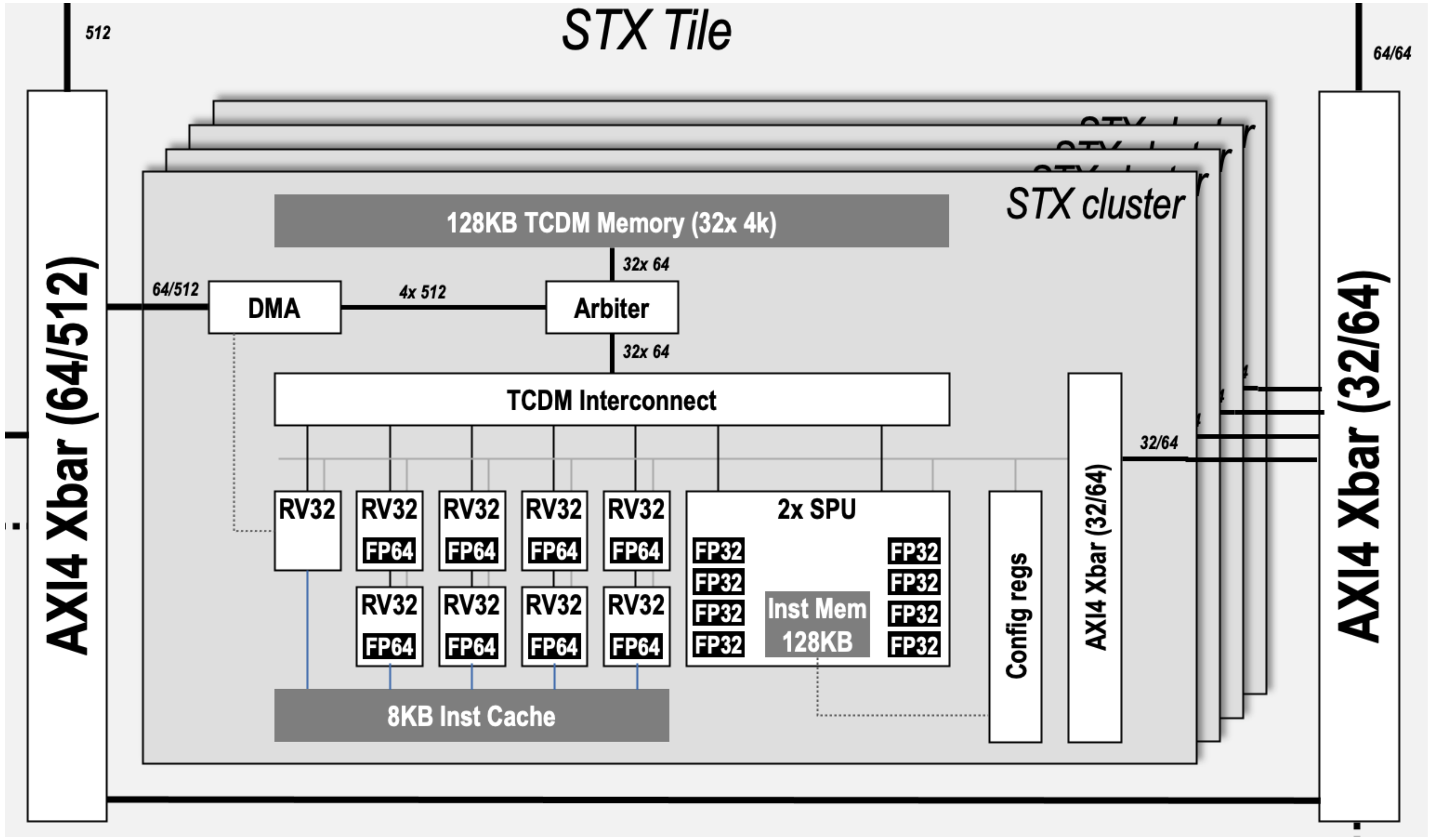


NTX Limitations

- The accelerator itself calculates the offsets
  - Highly efficient
  - But highly constrained!
- Complex access patterns do not map well onto NTX
  - Strided Stencil Operations
  - Sparse Operations

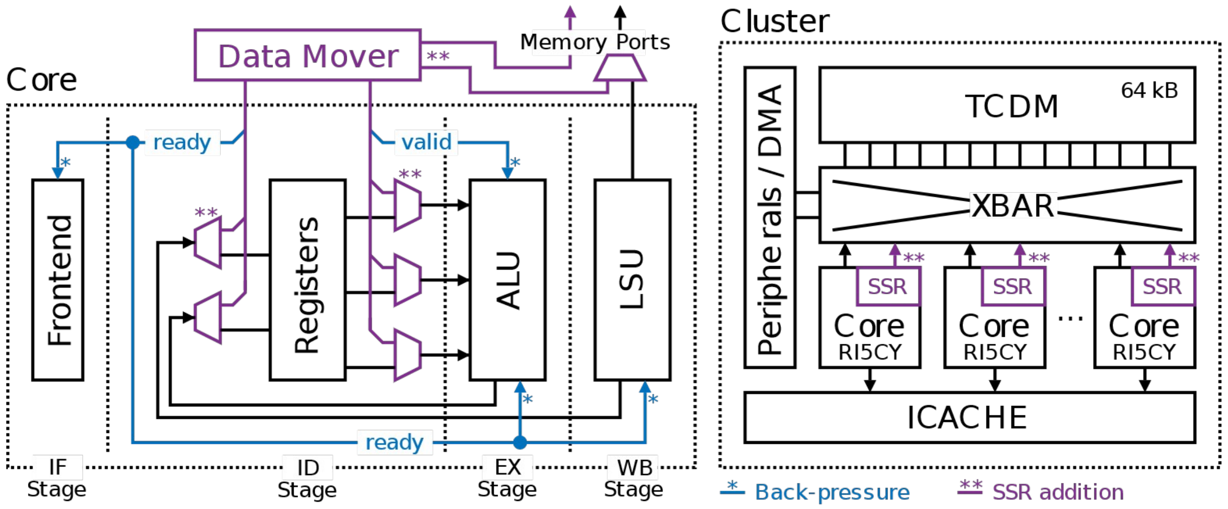


The STX Tile


STX Tile
512
64/64
AXI4 Xbar (64/512)
AXI4 Xbar (32/64)
STX cluster
128KB TCDM Memory (32x 4k)
32x 64
64/512
DMA
4x 512
Arbiter
32x 64
TCDM Interconnect
RV32
RV32
FP64
2x SPU
FP32
Inst Mem 128KB
Config regs
AXI4 Xbar (32/64)
32/64
8KB Inst Cache

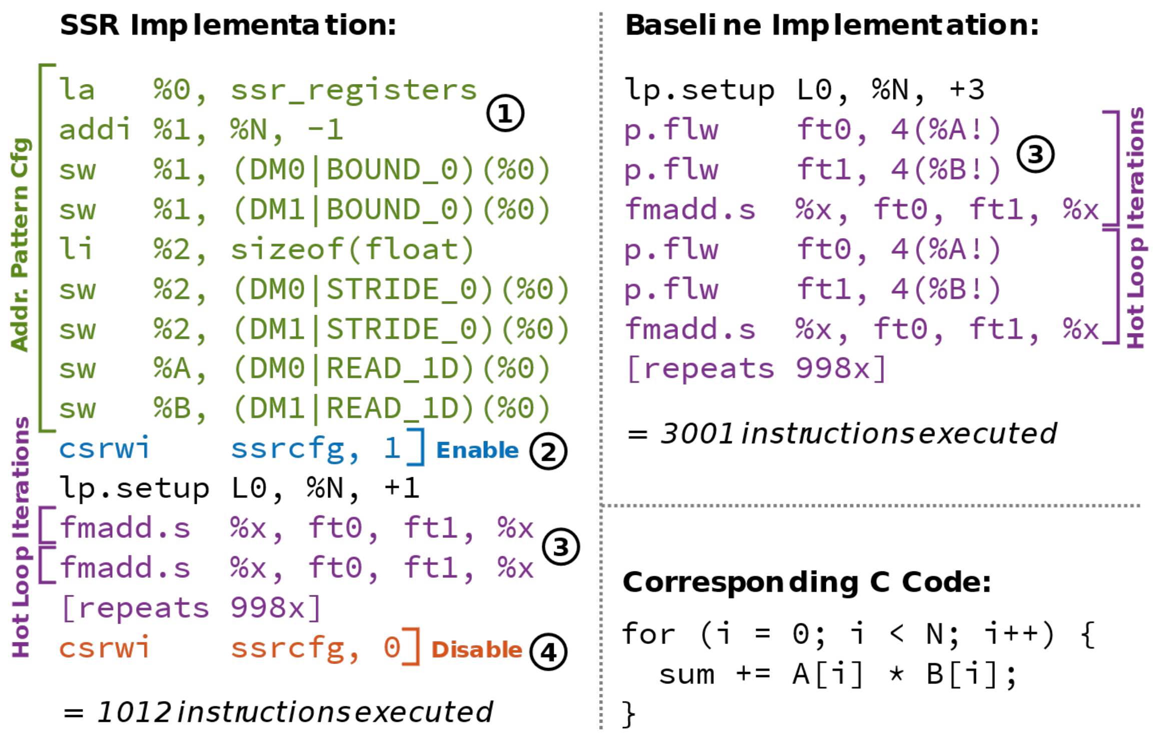
SSRs

- Stream Semantic Registers
  - **Implicitly** encode memory accesses as register reads/writes
  - Boost utilization of FPU from 33% to close to 100%
- Key ideas to increase hardware efficiency:
  - Reduction of von Neumann bottleneck (**load/store elision** through streaming)
  - Latency hiding through DMA-based **double-buffering**



Motivation: Von Neumann Bottleneck

- SSR helps alleviate the von Neumann bottleneck
  - No explicit load/store instructions
  - No explicit address calculation instructions
- Simple example: Dot product over 1000 elements
- With single RV32IF:
  - **3001** instructions executed
- With single core + SSR (plus RV32I):
  - **1012** instructions executed



SSR Implementation:

```
la     %0, ssr_registers
addi   %1, %N, -1
sw     %1, (DM0|BOUND_0)(%0)
sw     %1, (DM1|BOUND_0)(%0)
li     %2, sizeof(float)
sw     %2, (DM0|STRIDE_0)(%0)
sw     %2, (DM1|STRIDE_0)(%0)
sw     %A, (DM0|READ_1D)(%0)
sw     %B, (DM1|READ_1D)(%0)
csrwi      ssrcfg, 1
lp.setup L0, %N, +1
fmadd.s    %x, ft0, ft1, %x
fmadd.s    %x, ft0, ft1, %x
[repeats 998x]
csrwi      ssrcfg, 0
```

= 1012 instructions executed

Baseline Implementation:

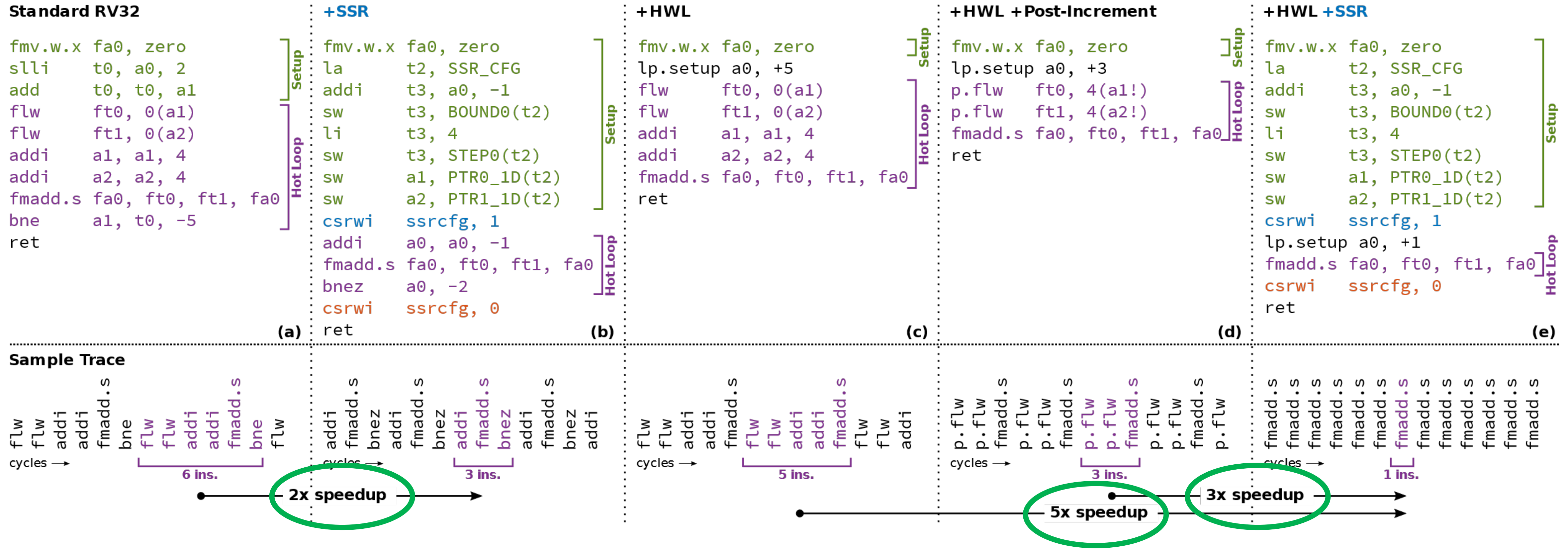
```
lp.setup L0, %N, +3
p.flw     ft0, 4(%A!)
p.flw     ft1, 4(%B!)
fmadd.s   %x, ft0, ft1, %x
p.flw     ft0, 4(%A!)
p.flw     ft1, 4(%B!)
fmadd.s   %x, ft0, ft1, %x
[repeats 998x]
```

= 3001 instructions executed

Corresponding C Code:

```
for (i = 0; i < N; i++) {
  sum += A[i] * B[i];
}
```

Comparison of different ISA Extensions



Standard RV32

```
fmv.w.x fa0, zero
slli    t0, a0, 2
add     t0, t0, a1
flw     ft0, 0(a1)
flw     ft1, 0(a2)
addi    a1, a1, 4
addi    a2, a2, 4
fmadd.s fa0, ft0, ft1, fa0
bne     a1, t0, -5
ret
```

(a)

+SSR

```
fmv.w.x fa0, zero
la      t2, SSR_CFG
addi    t3, a0, -1
sw      t3, BOUND0(t2)
li      t3, 4
sw      t3, STEP0(t2)
sw      a1, PTR0_1D(t2)
sw      a2, PTR1_1D(t2)
csrwi   ssrcfg, 1
addi    a0, a0, -1
fmadd.s fa0, ft0, ft1, fa0
bnez    a0, -2
csrwi   ssrcfg, 0
ret
```

(b)

+HWL

```
fmv.w.x fa0, zero
lp.setup a0, +5
flw     ft0, 0(a1)
flw     ft1, 0(a2)
addi    a1, a1, 4
addi    a2, a2, 4
fmadd.s fa0, ft0, ft1, fa0
ret
```

(c)

+HWL +Post-Increment

```
fmv.w.x fa0, zero
lp.setup a0, +3
p.flw   ft0, 4(a1!)
p.flw   ft1, 4(a2!)
fmadd.s fa0, ft0, ft1, fa0
ret
```

(d)

+HWL +SSR

```
fmv.w.x fa0, zero
la      t2, SSR_CFG
addi    t3, a0, -1
sw      t3, BOUND0(t2)
li      t3, 4
sw      t3, STEP0(t2)
sw      a1, PTR0_1D(t2)
sw      a2, PTR1_1D(t2)
csrwi   ssrcfg, 1
lp.setup a0, +1
fmadd.s fa0, ft0, ft1, fa0
csrwi   ssrcfg, 0
ret
```

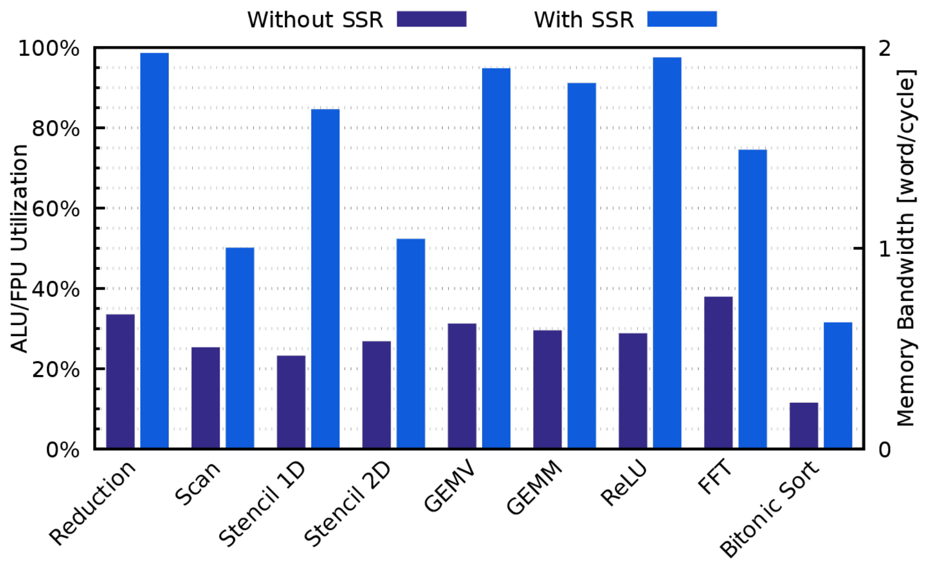
(e)

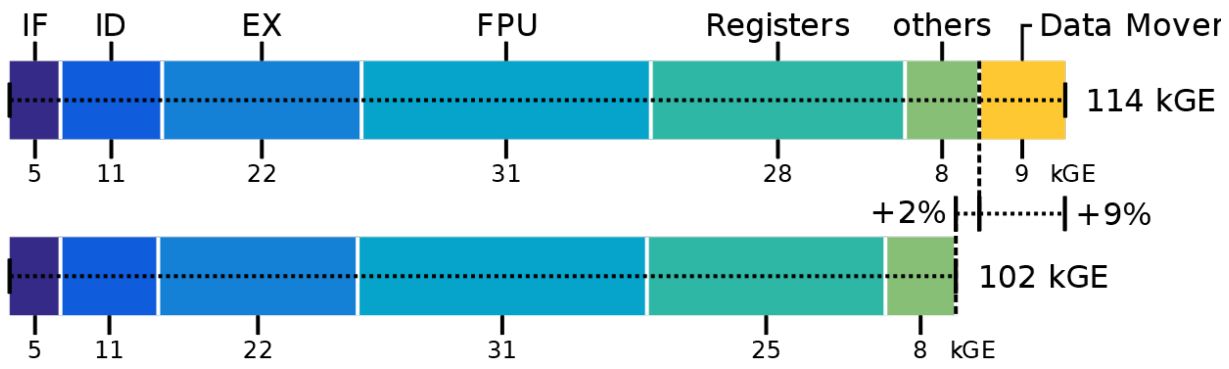
Sample Trace

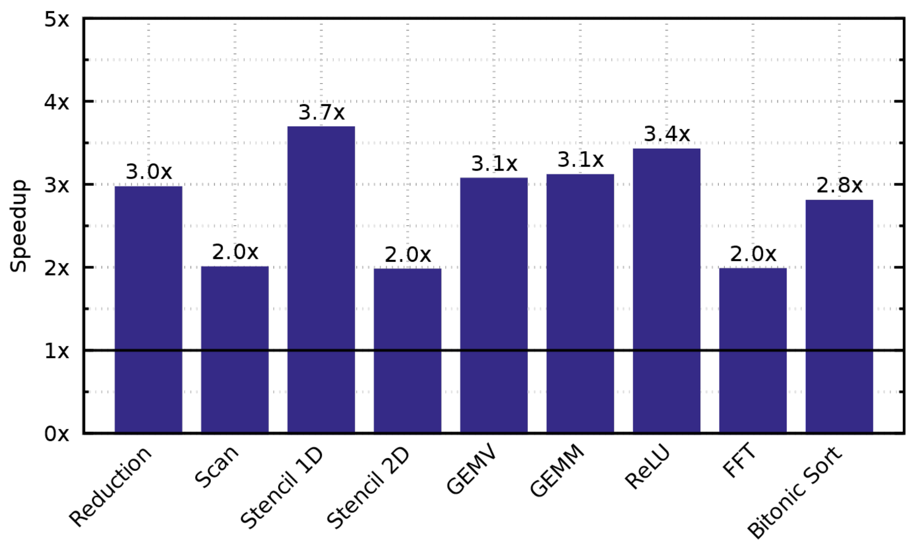
Setup
Hot Loop
flw flw addi addi fmadd.s bne flw flw addi addi fmadd.s bne flw
cycles →
6 ins.
addi fmadd.s bnez addi fmadd.s bnez addi fmadd.s bnez addi fmadd.s bnez addi
3 ins.
2x speedup
flw flw addi addi fmadd.s flw flw addi addi fmadd.s flw flw addi
5 ins.
p.flw p.flw fmadd.s p.flw p.flw fmadd.s p.flw p.flw fmadd.s p.flw p.flw fmadd.s p.flw
3 ins.
5x speedup
3x speedup
fmadd.s fmadd.s fmadd.s fmadd.s fmadd.s fmadd.s fmadd.s fmadd.s fmadd.s fmadd.s fmadd.s fmadd.s fmadd.s
1 ins.

Single core utilization/speed-up

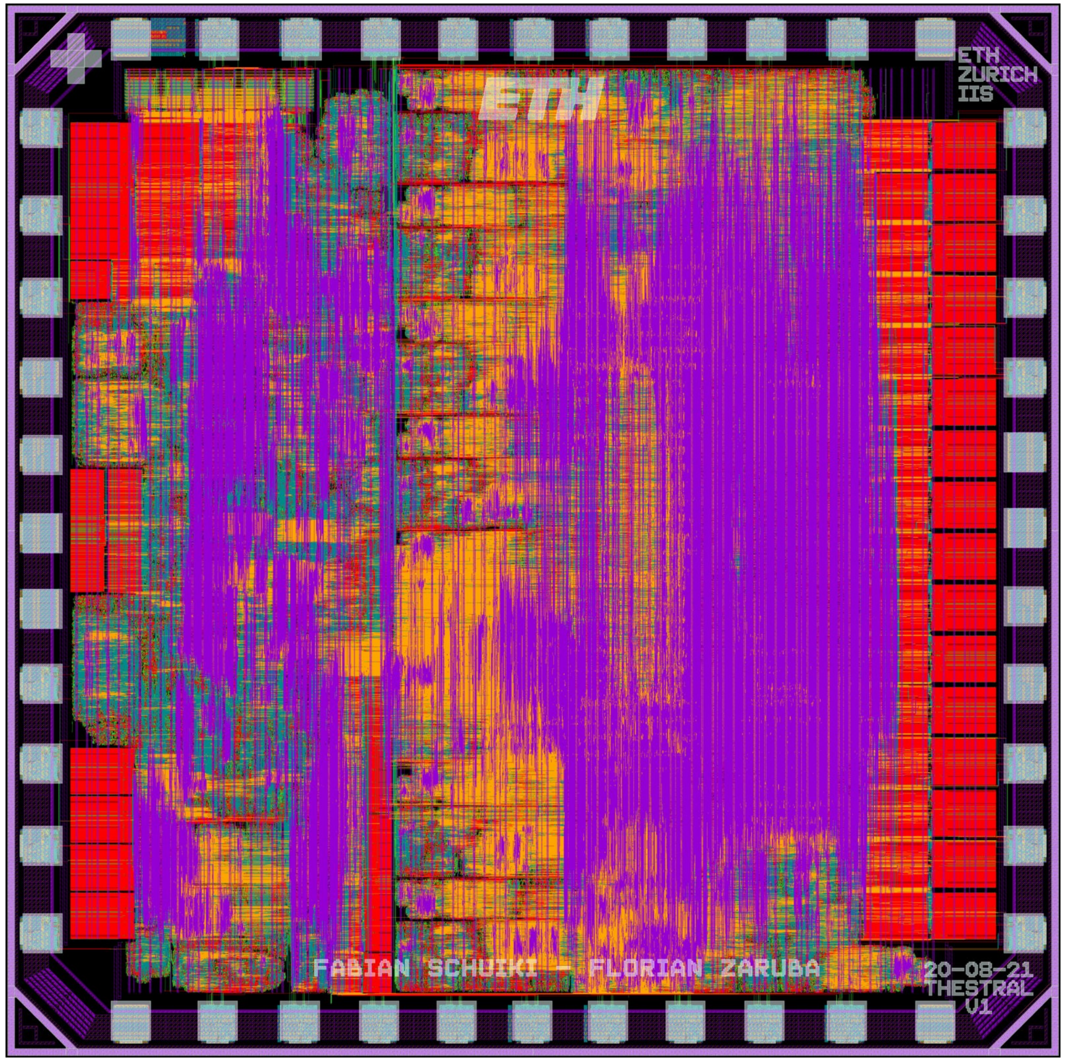
- Up to **> 95% FPU/ALU** optimization
- Up to **3.7x speedup** single core case
- Minimal area impact





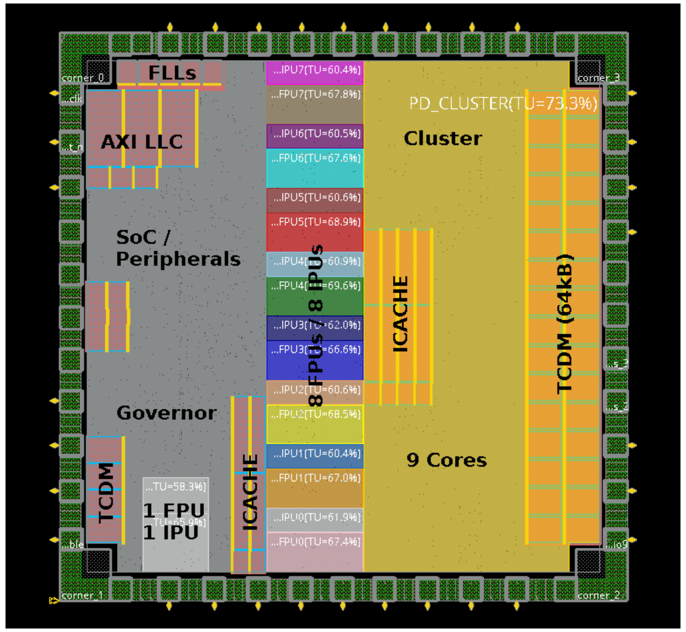


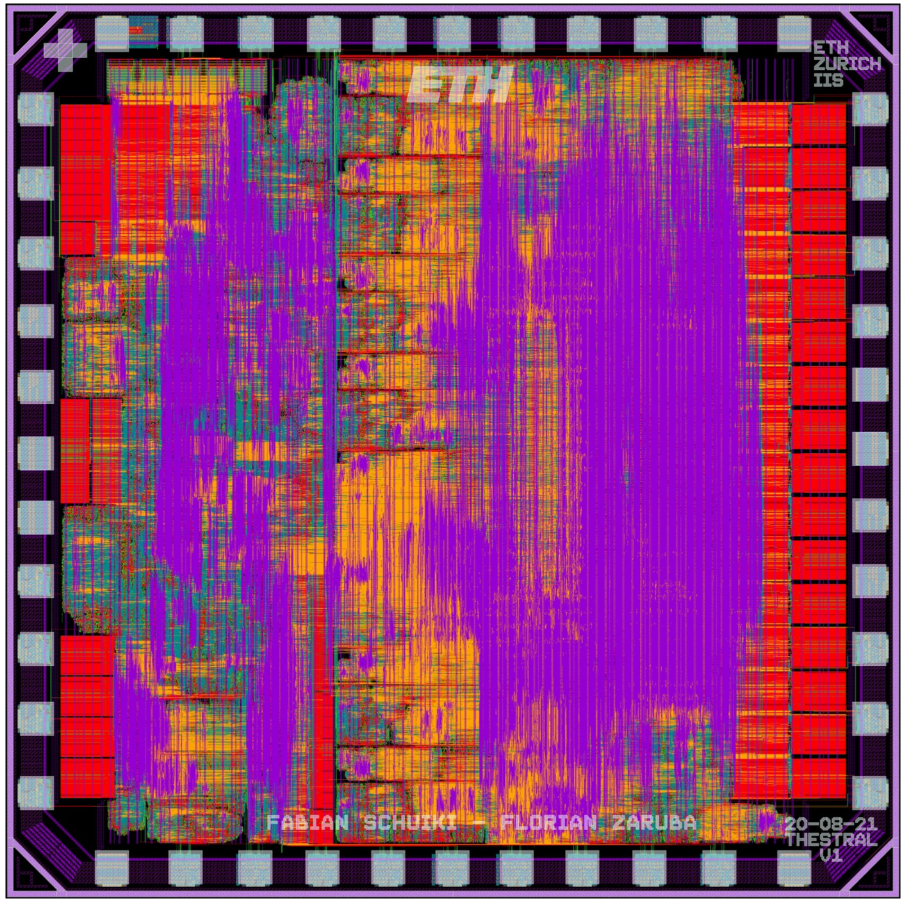
Physical Implementation



Thestral

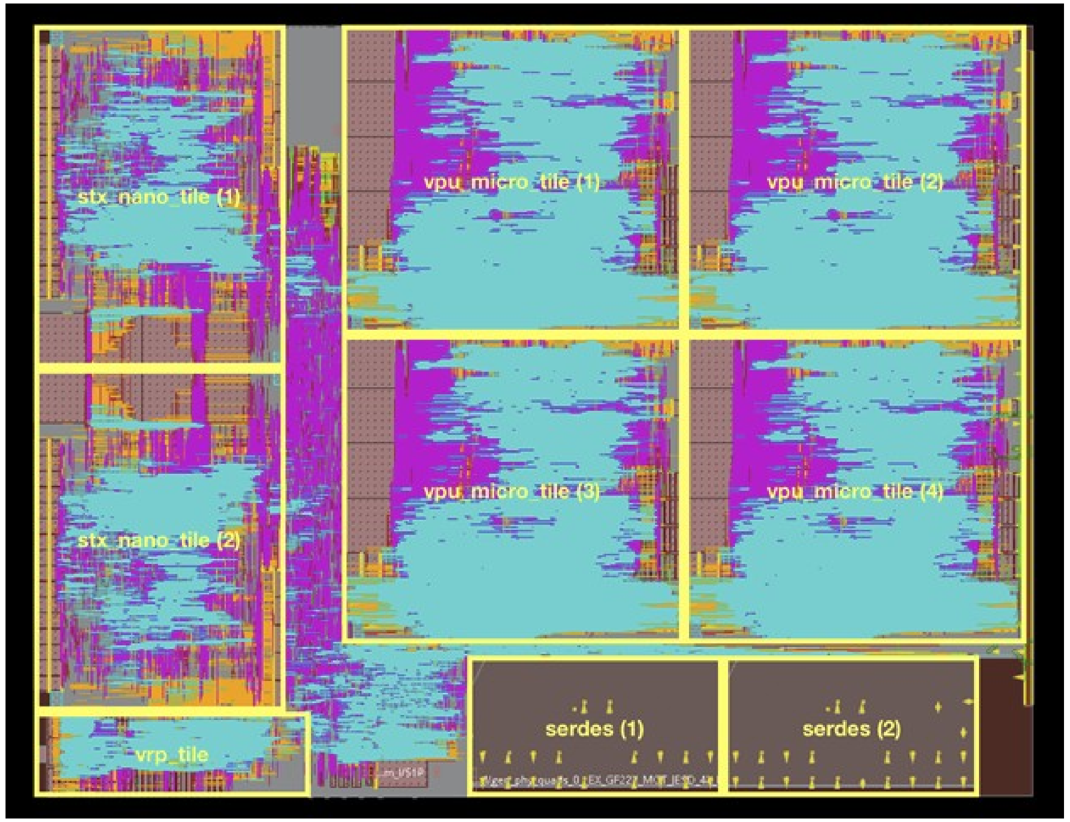
IoT/HPC
22nm FDSOI
1.56 mm^2
1 + 9 cores / 32b RISC-V
8 FPUs / 8 IPUs
910 MHz
0.6V - 0.9V
128kB (L1) / 24kB s.r. (L2)
Clock & Power Gating
Cluster / IPUs / FPUs
6.8 GOPs (32-bit)
13.6 GFLOPS (FP32)
10ns
118 GFLOPS/W @ 7.2 GFLOPS





EPAC

European Processor Initiative



ETH zürich

The RISC-V based Stencil Tensor Accelerator of EPI

Matheus Cavalcante
Ph.D. Student, ETH Zürich
4 May 2022