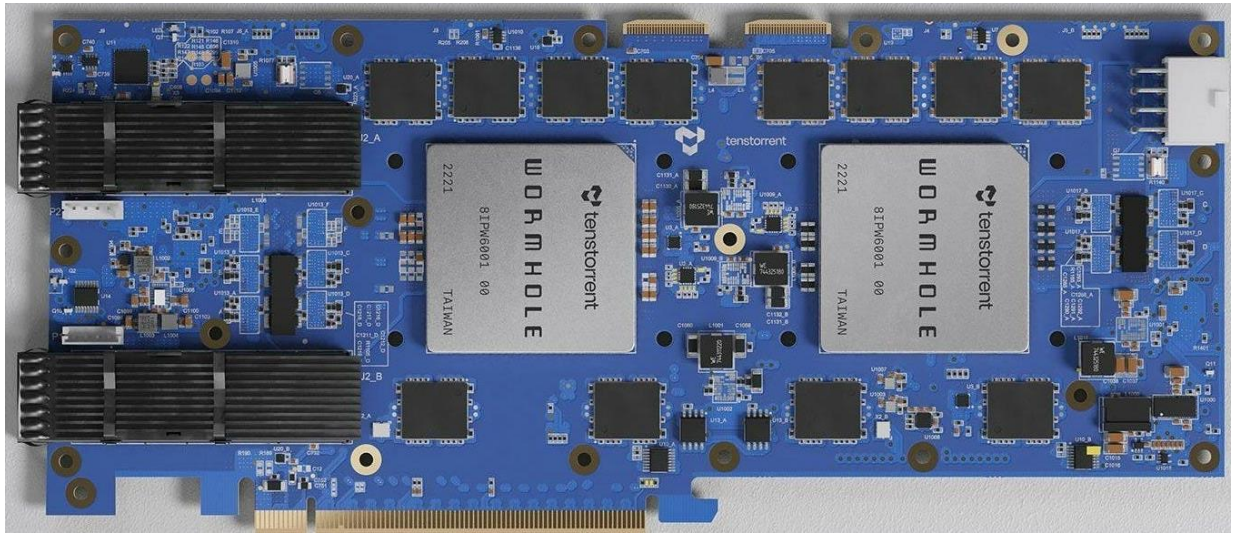
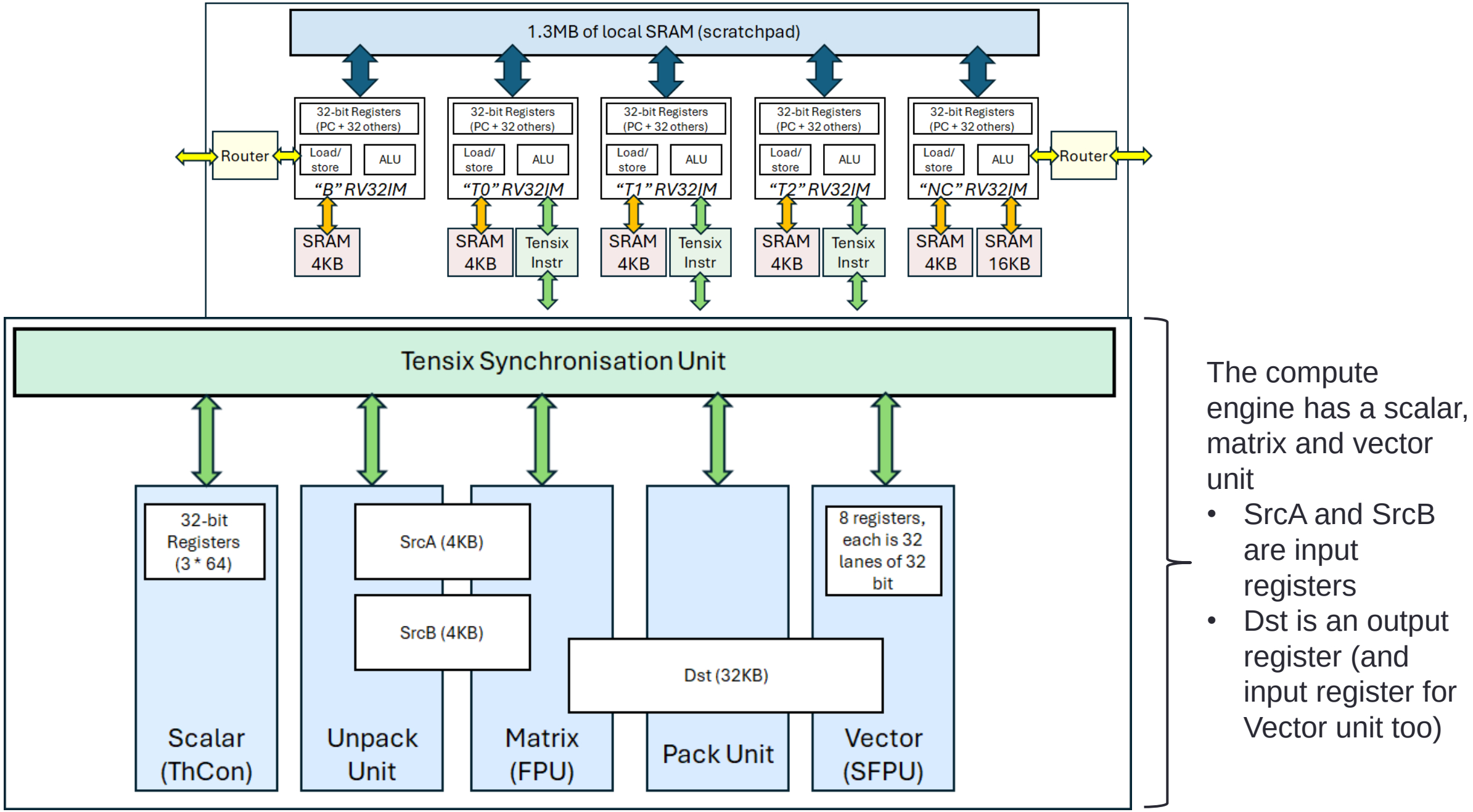
Overview of TT-Metalium SDK: Compute



epcc

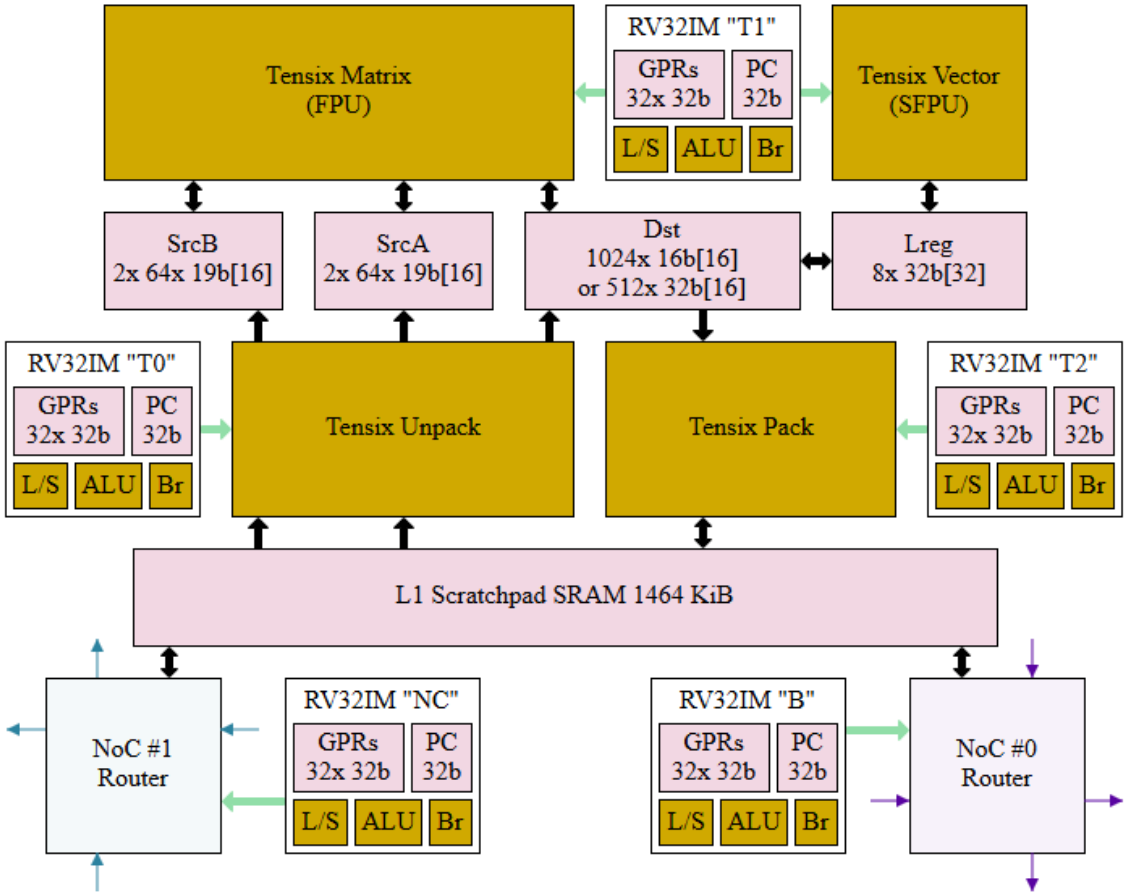
tenstorrent



The compute engine has a scalar, matrix and vector unit

- SrcA and SrcB are input registers
- Dst is an output register (and input register for Vector unit too)

A more accurate diagram….



From Pete Cawley's blog at https://www.corsix.org/content/tt-wh-part7

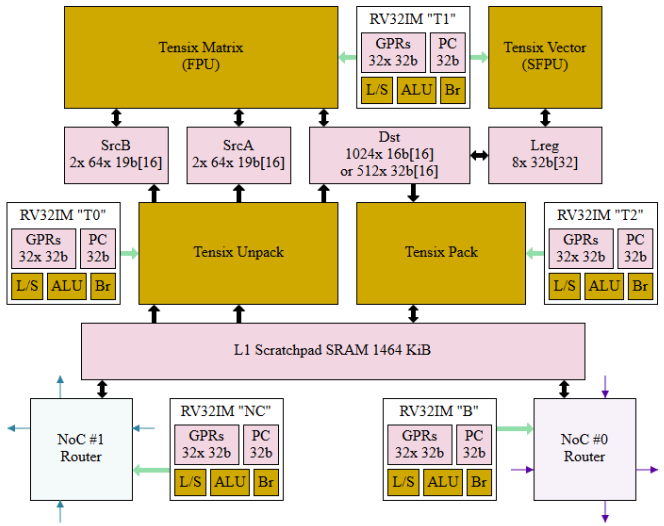
- The matrix unit can perform 2048 FP multiplies and 2048 FP additions per cycle
- The vector unit can perform 256 FP32 maths operations per second

- This is FP32(ish)
  - The matrix unit makes several sacrifices to IEEE compliance to achieve this performance
  - The vector unit is more accurate – so for HPC workloads is probably the one we would use

Issuing compute operations to matrix unit

To get input data in

- Wait for two CBs (LHS and RHS) to be available via *cb_wait_front* API call



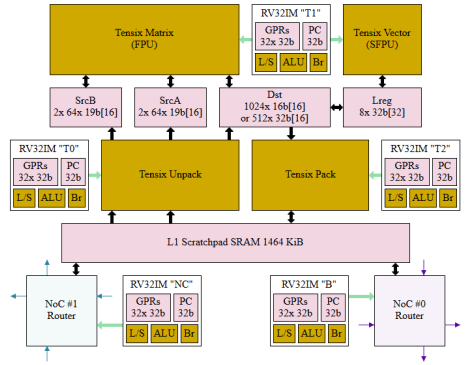
To compute

- Acquire exclusive compute lock on DST (target) registers using *tile_regs_aqcuire* API call
- Issue corresponding matrix API call such as *add_tiles, sub_tiles, mul_tiles* with CB index as input
- Release exclusive compute lock on DST (target) registers using *tile_regs_commit* API call

To get results out

- Acquire exclusive pack lock on DST (target) registers using *tile_regs_wait* API call
- Copy results from dst register to target CB via *pack_tile* API call
- Release exclusive pack lock on DST (target) registers using *tile_regs_release* API call

Issuing compute operations to vector unit



To get input data in

- Wait for two CBs (LHS and RHS) to be available via *cb_wait_front* API call
- Acquire exclusive pack lock on DST (target) registers using *tile_regs_wait* API call
- Copy both input tiles into *dst* register using segment index
- Release exclusive pack lock on DST (target) registers using *tile_regs_release* API call

To compute

- Acquire exclusive compute lock on DST (target) registers using *tile_regs_aqcuire* API call
- Issue corresponding vector API call such as *add_binary_tile, sub_binary_tile* with segment index determining inputs (first input overwritten with results)
- Release exclusive compute lock on DST (target) registers using *tile_regs_commit* API call

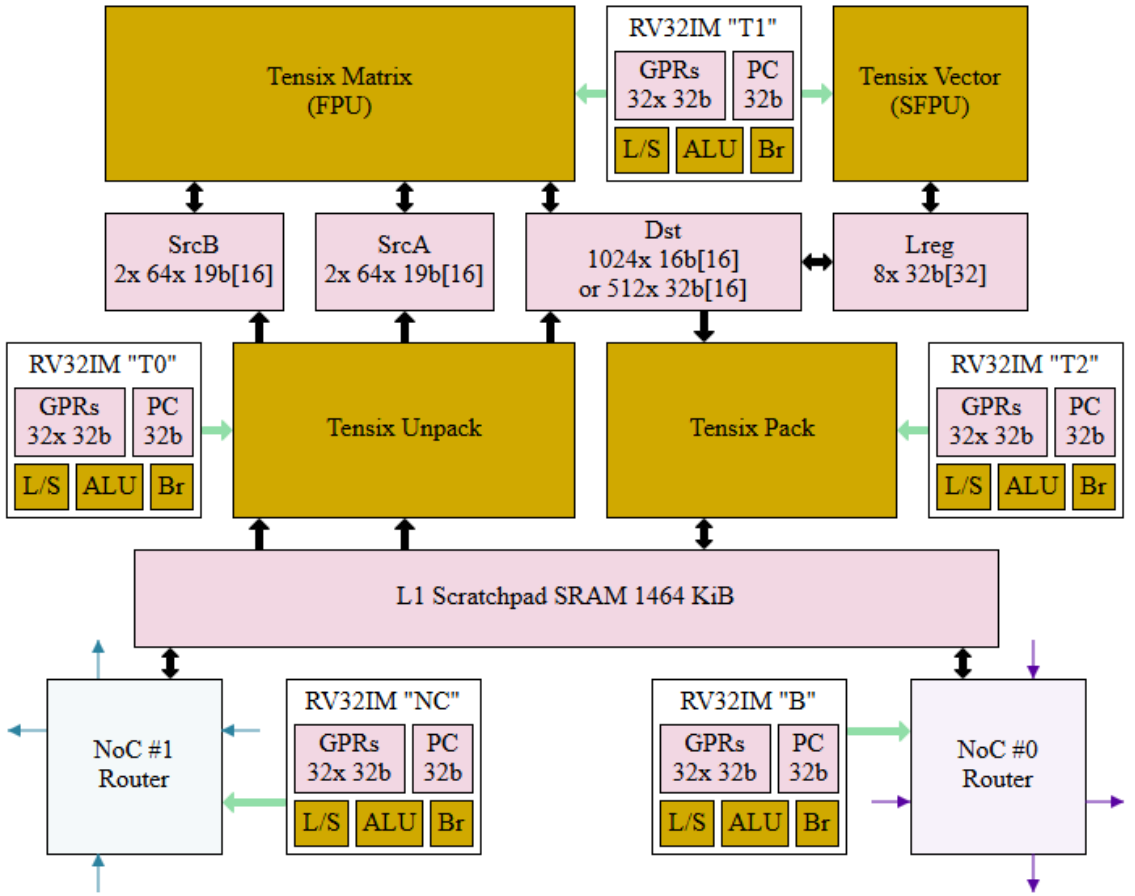
To get results out

- Acquire exclusive pack lock on DST (target) registers using *tile_regs_wait* API call
- Copy results from dst register to target CB via *pack_tile* API call
- Release exclusive pack lock on DST (target) registers using *tile_regs_release* API call

Vector unit uses the dst registers for both inputs and output

The key points

- Need to initialise with the data type
  - And reinitialise if change this
- Inputs are CBs and the output is a CB
- dst register is split into 16 segments
  - Matters more when using the vector unit
- Need to acquire locks on the dst register as this coordinates instructions from the pack, compute and unpack RISC-V cores



Most common maths calls

Matrix unit

- add_tiles
- sub_tiles
- mul_tiles
- matmul_tiles
- reduce_tile
- transpose_wh_tile

This also explains why the vector unit consumes from dst, as a common ML use-case is to execute with the matrix unit and then run another operation on results via the vector unit

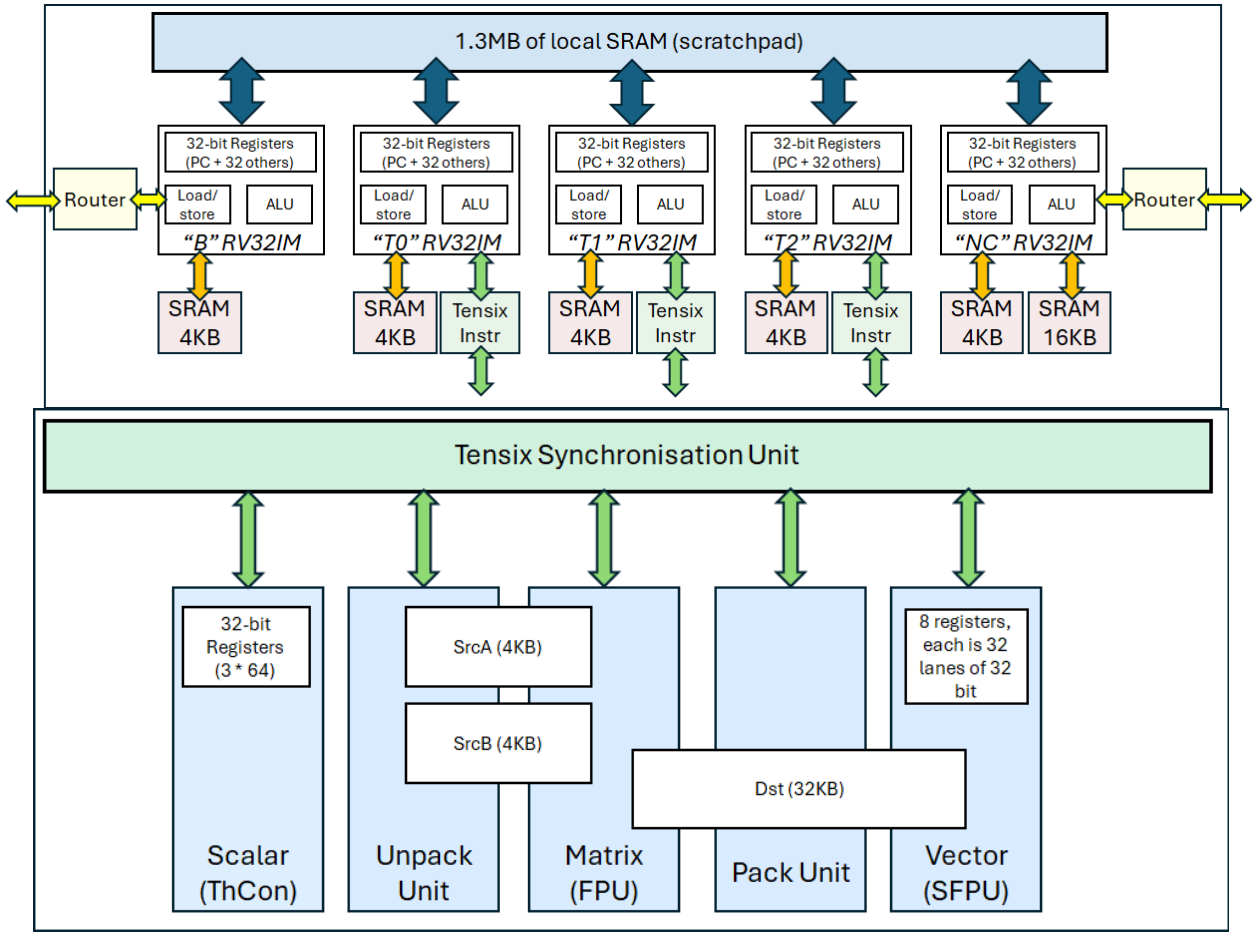
Vector unit

- add_binary_tile
- sub_binary_tile
- mul_binary_tile
- abs_tile
- exp_tile
- Isinf_tile
- Isfinite_tile
- Isnan_tile
- sqrt_tile
- square_time
- tan_tile
- sin_tile
- cos_tile
- atan_tile
- acos_tile
- asin_tile

- ltz_tile
- eqz_tile
- lez_tile
- gtz_tile
- neq_tile
- gez_tile
- unary_ne_tile
- unary_gt_tile
- unary_lt_tile
- unary_max_tile
- unary_min_tile

This last column are comparison operations

Three compute cores…



- Each Tensix unit has three RISC-V baby cores for compute
  - Unpacker drives the unpack unit
  - Maths drives FPU, SFPU, ThCon
  - Packer drives the pack unit
- Programmer's compute kernel is launched on all three cores which execute it concurrently
  - In the Metalium API there are explicit sections for different cores, where one (or more) cores will execute some code and additional synchronisation
    - But we don't really need to worry about this, however, it explains why there are locks on the dst register to avoid conflict between the math and pack cores

Tiling data to drive compute

- We have talked about bringing the FPU into play but the registers are only of a certain size
  - srcA and srcB are 4KB, so maximum 1024 FP32 and similar if you use the SFPU
- Therefore need to tile data across chunks
  - They use the terminology *tile* due to the architecture being designed for matrix multiplications, and chunk would be better as a tile can be 1D
- Practical three will explore how to do this, before using the matrix multiplication engine in practical four to perform the compute