ArmSME Dialect Proposal

Department name: Toronto Heterogeneous Compiler Lab
Author's name: Frank (Fang) Gao
Date: June 21, 2023
Rational for a Dialect

• Focusing on Outer Product instructions - MOPA/MOPS

• Challenges in bridging the gap between tensor/vector and SME ZA tiles
  > LLVM’s representation of ZA tiles is an immediate integer
  > Some sort of Register Allocation necessary
  > May be better suited to do in LLVM?

• Representation of outer-product accumulate (MOPA/MOPS instructions)
  > Predication in two dimensions
    - `vector.mask` can only handle take one masking vector
  > Alternative: Directly translate `vector.matrix_multiply` to SME Intrinsics?
    - How much do we want to abstract away from the hardware at this level?
    - We would prefer to keep some flexibility of using MOPA instructions explicitly
Design

• By no means final, open to the use of vector ops

• Proposed new operations
  > arm_sme.zero // Allocates Tile
  > arm_sme.load_tile // Allocates Tile
  > arm_sme.store_tile // Frees Tile
  > arm_sme.mopa // Outer Product Accumulate
  > arm_sme.mops // Outer Product Subtract

• Potential future additions
  > arm_sme.save // Spill tiles
  > arm_sme.restore // Re-load tiles
Allocating (Virtual) Tiles

- Lowering pass will keep an internal mask for tiles currently in use

- Zero or Loading will allocate a tile based on the type given

- Do we want to represent the tile as a vector or an opaque construct?
  - SSA with vectors could incur unintentional copying and spilling
  - … but could also make it easier to interface with other vector ops

// Since tiles should be initialized with either a sme.zero or a load,
// we can allocate tiles upon those operations

// Maps to 0x01 for tile enumeration. tilesInUse = 0x01
%tile0 = arm_sme.load_tile %C[%i, %j], %hmask, %vmask

// Tries to map to 0x11, fails because inUse = (tilesInUse & 0x11) = true
// Tries next tiles 0x22, succeeds. tilesInUse = 0x23
%tile1 = arm_sme.zero : vector<[4x4]xf32>

// Try 0x01, 0x10, 0x02, 0x20... in sequence
// Gets 0x10. tilesInUse = 0x33
%tile2 = arm_sme.zero : vector<[2x2]xf64>
Use of (Virtual) Tiles

```c
// Overwrites %tile0 - use of %tile0 will be invalid past this op? Perhaps we
// may need to introduce a sme.copy which will need to allocate another tile?
%tile0_new = arm_sme.mopa %tile0, %lhs, %rhs, %hmask, %vmask

// Emit error? - reference to %tile0 after it has already been overwritten
%tile0_new_new = arm_sme.mopa %tile0, %lhs, %rhs, %hmask, %vmask

// ...

// Deallocates 0x01. tilesInUse = 0x32
arm_sme.store_tile %C[i, j], %tile0_new, %hmask, %vmask
    : memref<x?xf64>, vector<[2x2]xf64>
```

- If using vector representation – Emit error after RAW (First iteration)
- Stores releases tiles?
- Would it make sense to introduce LLVM intrinsic to allocate tiles?
Summary

• Tradeoff in flexibility and “generality”
  > MOPA/MOPS ops vs. outerproduct, matmul, etc.

• Allocation ops makes tile management easier, but not strictly necessary
  > zero, load_tile, store_tile
  > Slightly more complex translation but nothing too bad

• Challenges in tile allocation
  > Pseudo RA in MLIR?

• Representation of tiles
  > Vector type with restrictions vs. opaque type