Outer-product CPU Instruction for NPU-level ML Throughput

Houxiang (hj14), Husnain (mubarik3) and Nam (hn5)

Motivation

- DNN accelerators usually accelerate **matrix-multiplications** (MM)

- Accelerators are hard to program --> poor optimization

- Extend CPU ISA to support **outer-product**
  - Compute MM as **sum of outer-products**
  - **High compute density** with low memory cost

Motivation: MM as Sum of Outer-products

Matrix Multiplication: C = AXB

Traditional Way: $C_{ij} = \sum_k A_{ik} * B_{kj} = A_i^T B_j$ (Inner-products)

$C = \sum_i (A^T)_i \otimes (B_i) = \sum_i (A^T)_i (B_i)^T$ (sum of outer products)

$$\begin{bmatrix} a_1 \\ a_2 \\ \vdots \\ a_n \end{bmatrix} \times [b_1\ b_2 \cdots b_n] = \begin{bmatrix} a_1b_1 & a_1b_2 & \cdots & a_1b_n \\ a_2b_1 & a_2b_2 & & a_2b_n \\ \vdots & & \ddots & \vdots \\ a_nb_1 & a_nb_2 & \cdots & a_nb_n \end{bmatrix}$$

ISA Specification (n=16, ARM based)

Usage: opa C, a, b ($C = a \otimes b + C$)

a, b: vector registers of 16 INT8 values (128-bit) (v0~v16)

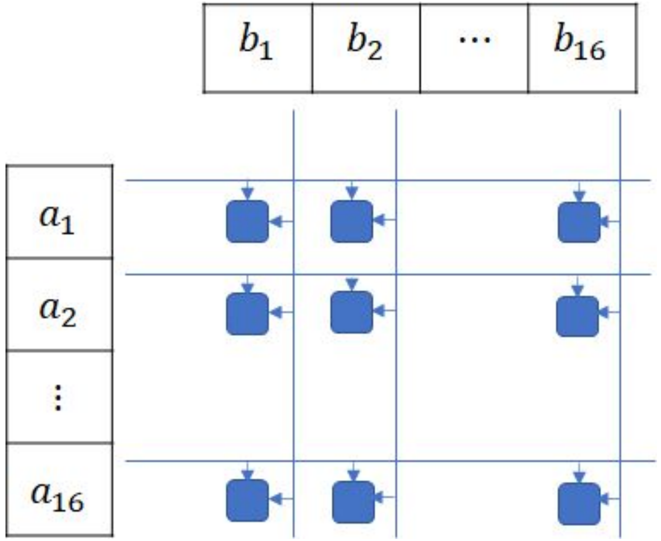
C: **tile register** of 16X16 INT32 accumulators (8192-bit) (m0~m3)

256 MACs per cycle -> 5 TOPs @ 2.5 GHz 4-core (> 4 TOPs of Edge TPU)

tld/tst C, a, #n (s) : shifted load/store n-th row of the tile C from/to the vector a

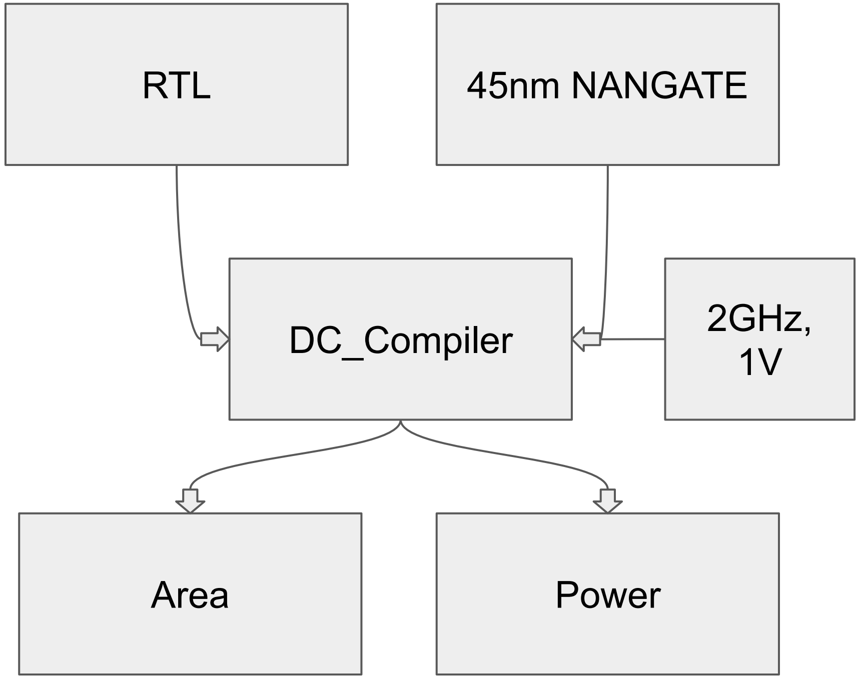
Architecture Details (HW)

- Functional Element
  - 256 MAC units (int8)
  - Accumulated to the tile register
- Tile Accumulator Register
  - 16X16 32 bit accumulator register
  - Row load/store unit from/to 128-bit vector



Evaluation Method - Power/Area

- Handwritten, synthesizable RTL
- Synopsys DC_Compiler
- NanGate 45nm Library
- Area and Power reports



Current Results - Power/Area

- Technology node: 45 nm
- Cortex A72 Area: 50 mm2
- Cortex A72 Power: 4.76 W

	Frequency	Area mm2	Power mW	Area overhead vs A72	Power overhead vs A72
macBrick	2GHz	0.23	458	0.45%	9.65%

SW Usage

Inner-product

```
for A_t, B_t, C_t in tile(A,B,C):
  tmp = [0, ..., 0]
  for i in 0...16:
    for j in 0...16:
      tmp = fma16(A_t[i][:], B_t[:][j])
      C_t[i][j] = C_t[i][j] + sum(tmp)
```

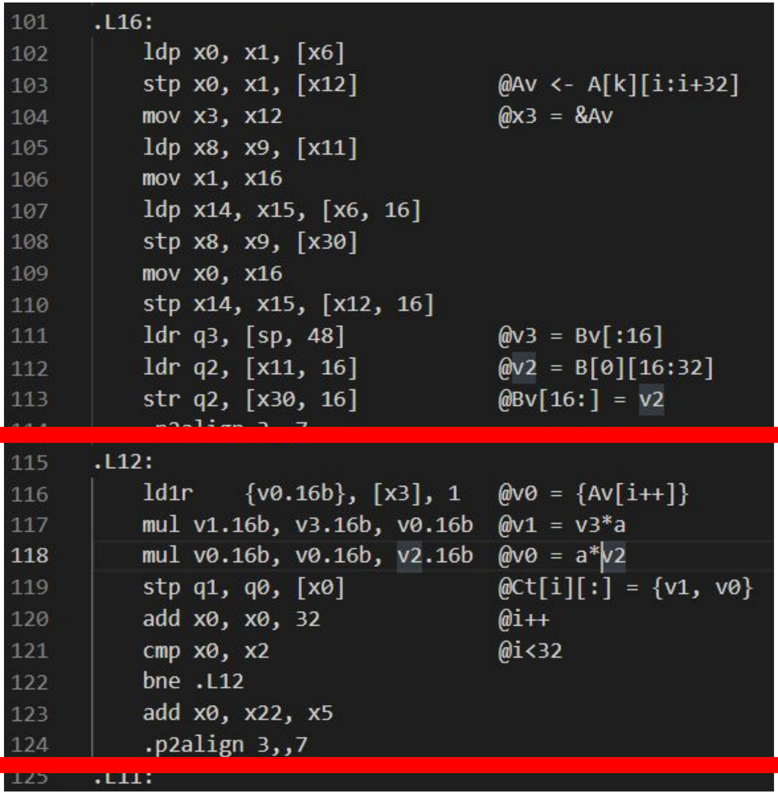
Outer-product using FMA

```
for A_t, B_t, C_t in tile(A,B,C):
  for k in 0...16:
    for i in 0...16:
      tmp = [A_t[i][k], A_t[i][k], ...]
      C_t[i] = fma16(C_t[i], tmp, B_t[k][:])
```

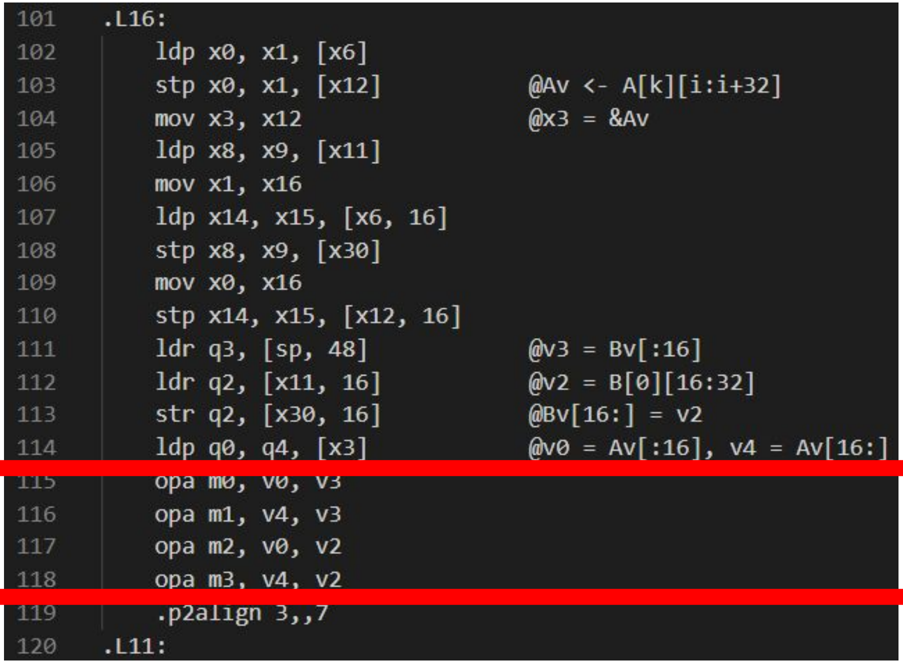
Ours (16X acceleration)

```
for A_t, B_t, C_t in tile(A,B,C):
  for k in 0...16:
      C_t = opa16(C_t, A_t[:][k], B_t[k][:])
```

Assembly Comparison



```
.L16:
    ldp x0, x1, [x6]
    stp x0, x1, [x12]            @Av <- A[k][i:i+32]
    mov x3, x12                  @x3 = &Av
    ldp x8, x9, [x11]
    mov x1, x16
    ldp x14, x15, [x6, 16]
    stp x8, x9, [x30]
    mov x0, x16
    stp x14, x15, [x12, 16]
    ldr q3, [sp, 48]             @v3 = Bv[:16]
    ldr q2, [x11, 16]            @v2 = B[0][16:32]
    str q2, [x30, 16]            @Bv[16:] = v2
    .p2align 3,,7
.L12:
    ld1r    {v0.16b}, [x3], 1    @v0 = {Av[i++]}
    mul v1.16b, v3.16b, v0.16b   @v1 = v3*a
    mul v0.16b, v0.16b, v2.16b   @v0 = a*v2
    stp q1, q0, [x0]             @Ct[i][:] = {v1, v0}
    add x0, x0, 32               @i++
    cmp x0, x2                   @i<32
    bne .L12
    add x0, x22, x5
    .p2align 3,,7
.L11:
```

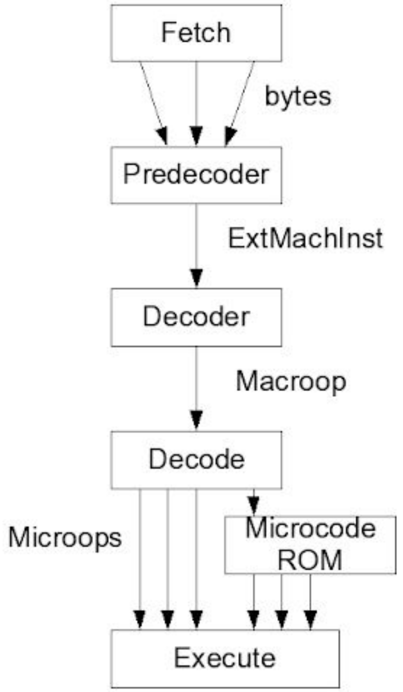


```
.L16:
    ldp x0, x1, [x6]
    stp x0, x1, [x12]            @Av <- A[k][i:i+32]
    mov x3, x12                  @x3 = &Av
    ldp x8, x9, [x11]
    mov x1, x16
    ldp x14, x15, [x6, 16]
    stp x8, x9, [x30]
    mov x0, x16
    stp x14, x15, [x12, 16]
    ldr q3, [sp, 48]             @v3 = Bv[:16]
    ldr q2, [x11, 16]            @v2 = B[0][16:32]
    str q2, [x30, 16]            @Bv[16:] = v2
    ldp q0, q4, [x3]             @v0 = Av[:16], v4 = Av[16:]
    opa m0, v0, v3
    opa m1, v4, v3
    opa m2, v0, v2
    opa m3, v4, v2
    .p2align 3,,7
.L11:
```

Entire loop (32-iter) to 4 instructions

Evaluation Method - Simulation

- Find an unallocated instruction
- Add customized OPA instruction to Gem5
- Compile the synthesized program with/without our OPA extension
- Run the simulations on Gem5



Progress and Plan

- Current Progress
  - Functional element Power/Area estimation (45nm)
  - ISA definition
  - Baseline matrix multiplication assembly code

- Apr 30 - May 12
  - Assembly code with/w.o. custom instruction
  - Gem5 instruction extension implementation and test
  - SpeedUp compared without extension
  - Edge TPU baseline results

Progress & Current Results

Please Put area, power (numbers we have now).