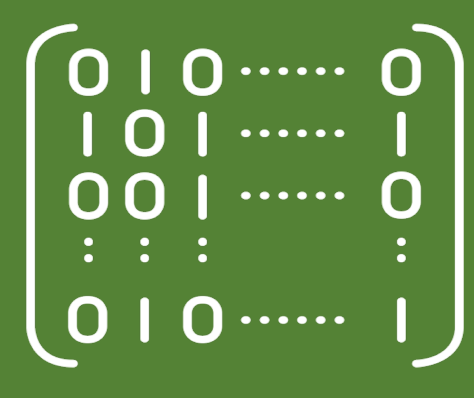


Efficient Matrix Multiplications on RISC-V MCUs



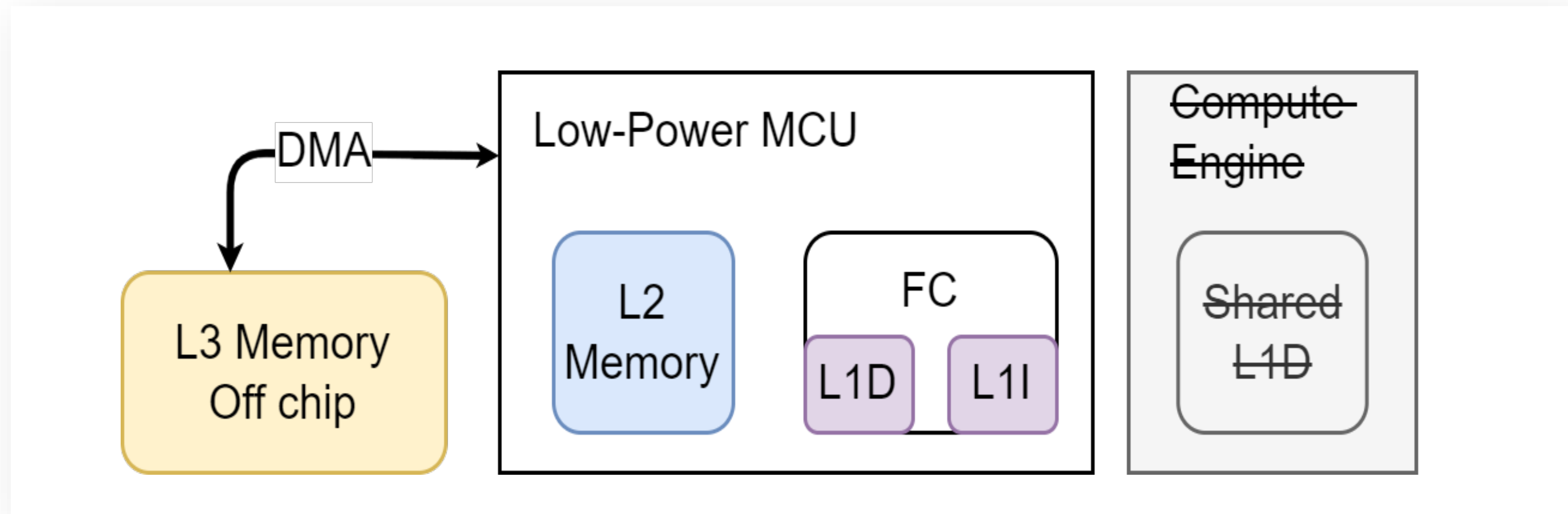
- > GAP8
 - > Leverage Dot product **vector instructions**
 - > New Multi-level **memory hierarchy**
 - > Approximate Computing
 - > Modified BLIS
- (BLAS-like Library Instantiation Software)



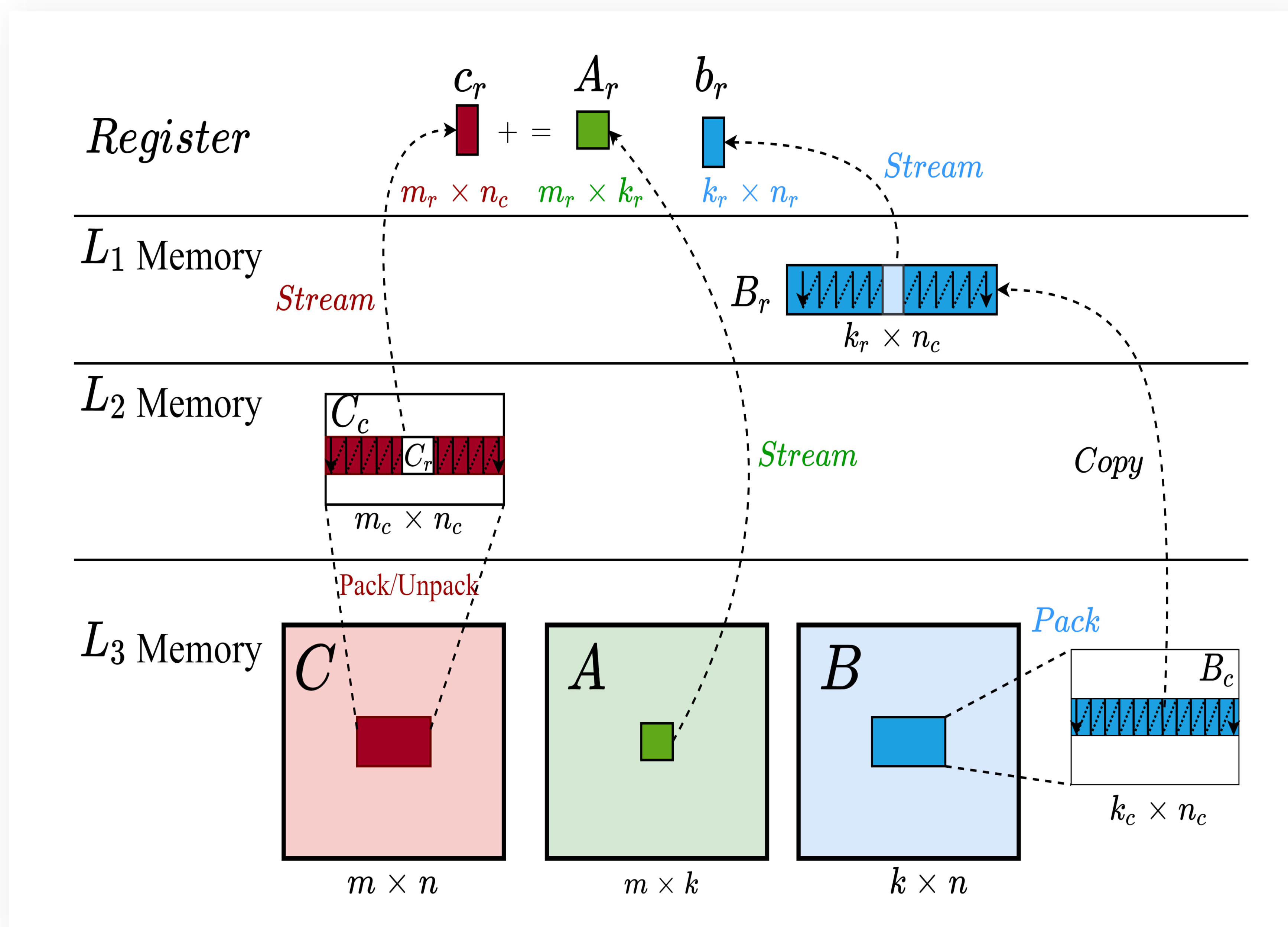
Full Text & More

Paper: Cristian Ramírez, Adrián Castellón, Enríque S. Quintana-Ortí. Poster: Cristian Ramírez, Jie Lei

GAP8 Memory Architecture



Matrix Packing, Copying, Streaming



Proposed GEMM algorithm (B3C2A0)

```

L1 for ( j_c = 0; j_c < n_c; j_c += n_c )
L2   for ( p_c = 0; p_c < k; p_c += k_c ) {
L3     B_c ← B(p_c : p_c + k_c - 1, j_c : j_c + n_c - 1); // Pack
L4     for ( i_c = 0; i_c < m; i_c += m_c ) {
L5       C_c ← C(i_c : i_c + m_c - 1, j_c : j_c + n_c - 1); // Pack
L6       for ( p_r = 0; p_r < k_r; p_r += k_r )
           for ( i_r = 0; i_r < m_r; i_r += m_r )
           for ( j_r = 0; j_r < n_r; j_r += 1 ) // Micro-kernel
               C_c(i_r : i_r + m_r - 1, j_r)
                   += A_c(i_r : i_r + m_r - 1, p_r : p_r + k_r - 1)
                       · B_c(p_r : p_r + k_r - 1, j_r);
           C_c(i_c : i_c + m_c - 1, j_c : j_c + n_c - 1) ← C_c; // Unpack
       }
   }

```

Proposed micro-Kernel (Loop 6)

```

gemm_ukernel_ABresident_gap8( int nc, signed char *A, int lda,
                               signed char Br, signed char Cc
){
    int jr, baseCB = 0;
    v4s A0, A1, A2, A3, // Columns of the 4x4 micro-tile Ar
        br, cr; // Columns of Br, Cr

    // Load the columns of the 4x4 micro-tile Ar into vector registers
    // Simulated in software using the v4s datatype
    A0 = ((v4s) (&A[0])); A1 = ((v4s) (&A[lda]));
    A2 = ((v4s) (&A[2*lda])); A3 = ((v4s) (&A[3*lda]));

    // Transposition of Ar omitted for brevity
    // ...
    for ( jr = 0; jr < nc; jr++ ) { // Loop L6
        // Load the jr-th columns of Cr, Br into two vector registers
        cr = ((v4s) (&Cr[baseCB]));
        br = ((v4s) (&Br[baseCB]));

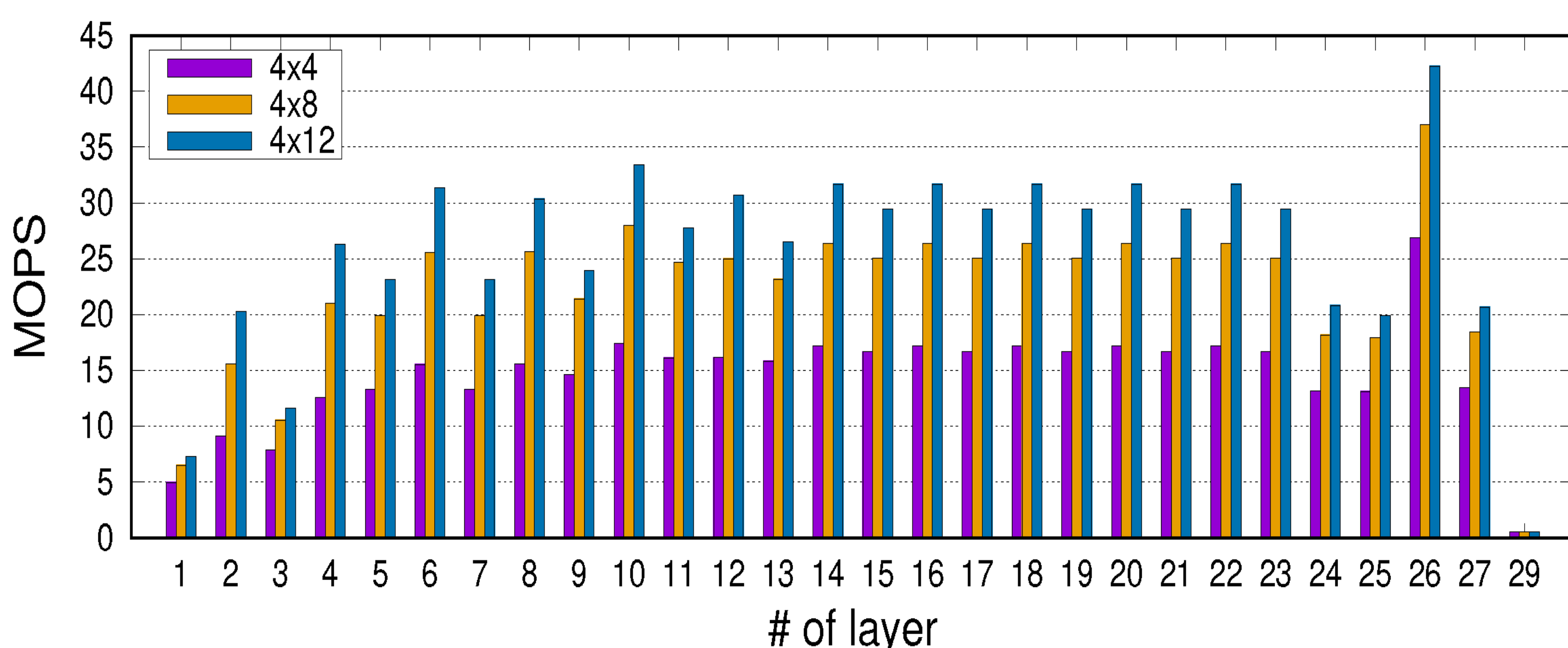
        // Update i-th entry of cr as cr[i] += Ai * br, i=0,1,2,3,
        // The GAP8 dot product is realized in software via the
        // gap8_dotp4 instruction
        cr[0] += gap8_dotp4(A0, br); cr[1] += gap8_dotp4(A1, br);
        cr[2] += gap8_dotp4(A2, br); cr[3] += gap8_dotp4(A3, br);

        // Store the column of Cr in memory. No vector support in GAP8
        Cr[baseCB+0] = cr[0]; Cr[baseCB+1] = cr[1];
        Cr[baseCB+2] = cr[2]; Cr[baseCB+3] = cr[3];

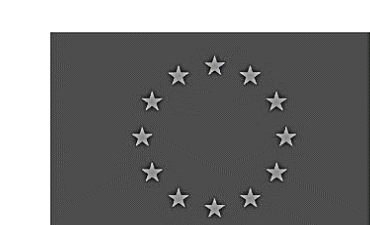
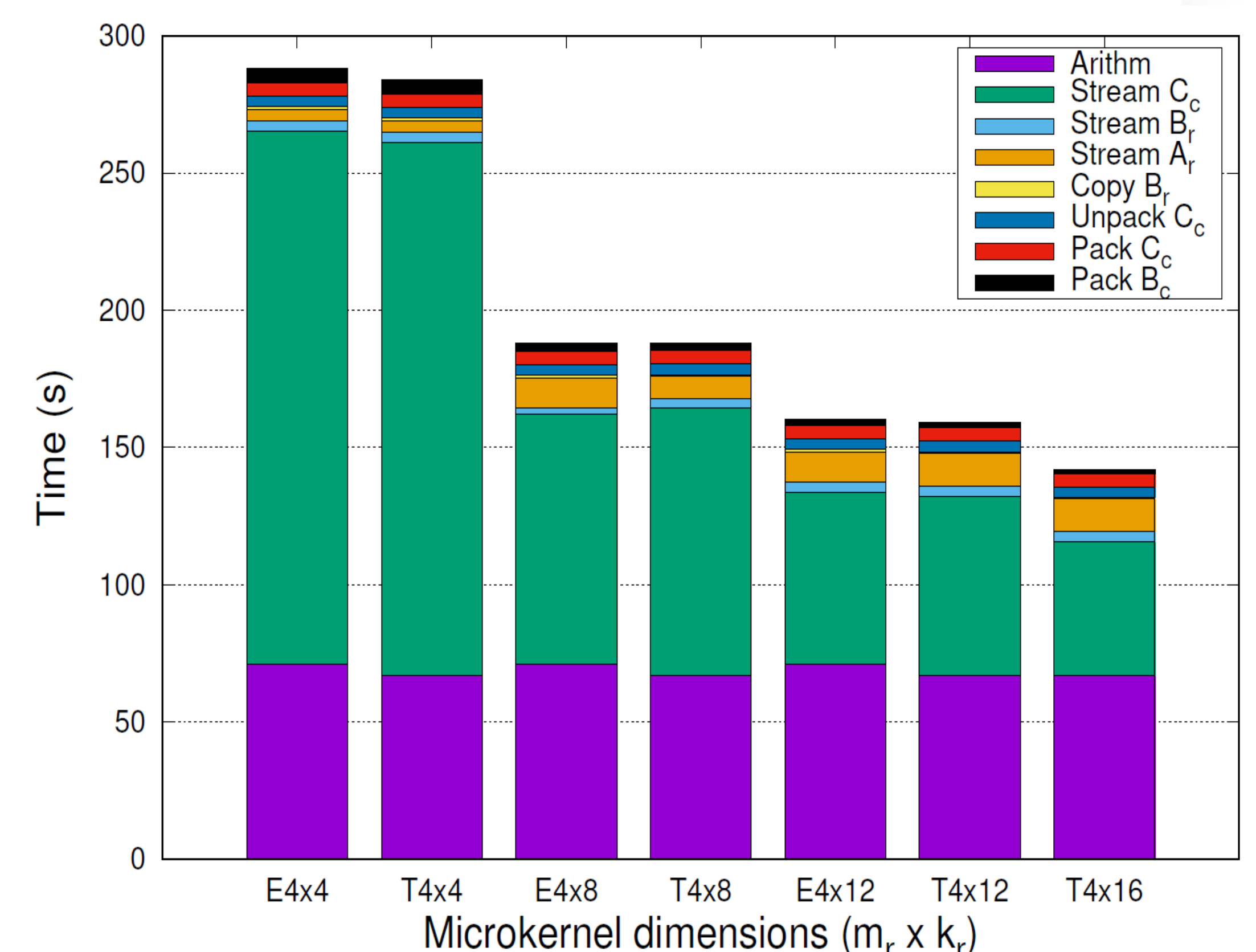
        baseCB += 4; // Prepare for next iteration
    }
}

```

Performance on Different Dimension of Microkernel and Convolutional Layers of MobileNet V1



Runtimes on Different micro-kernel dimensions



Funded by the Horizon 2020 Framework Programme of the European Union

