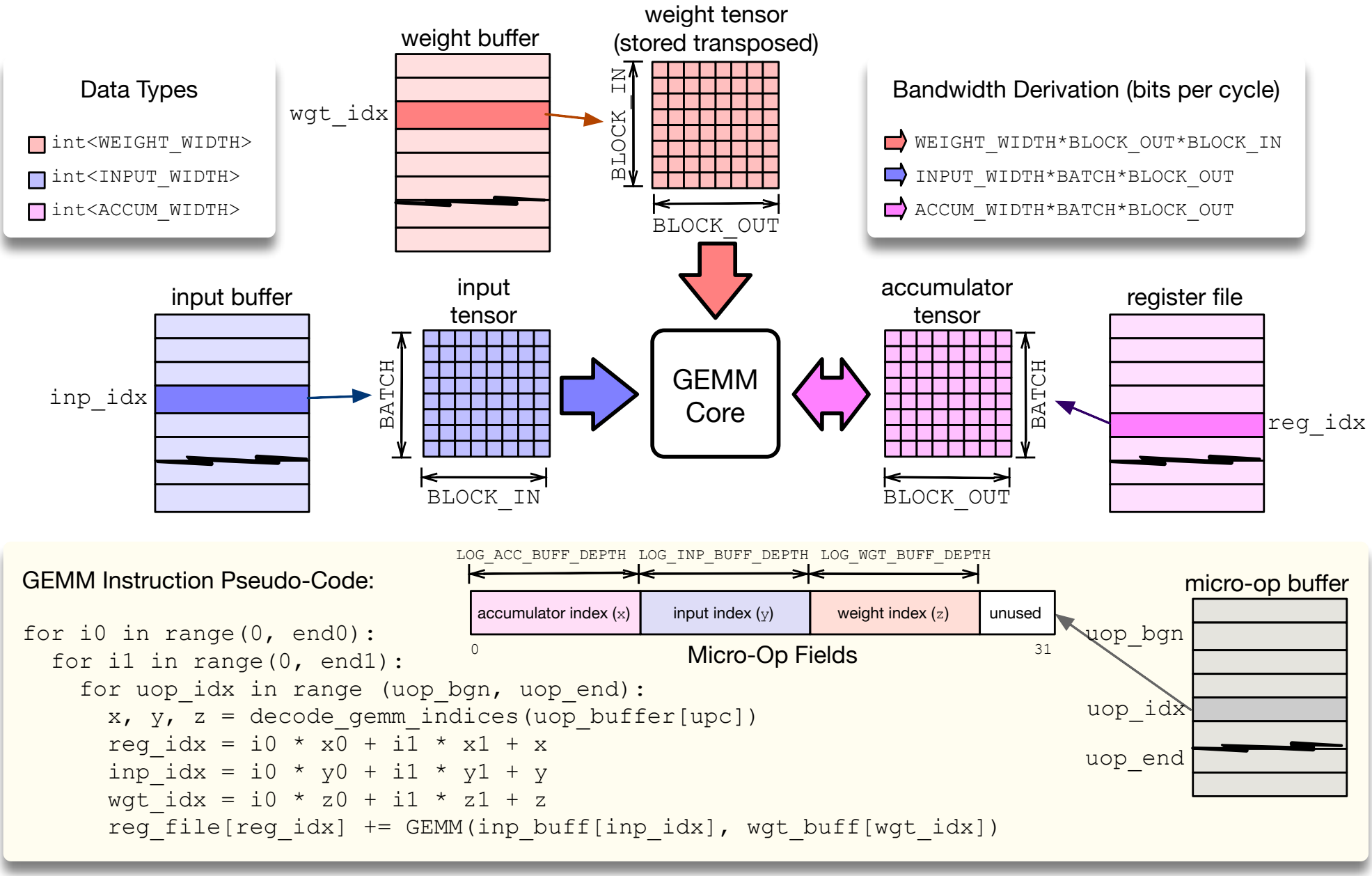
Data Types
int<WEIGHT_WIDTH>
int<INPUT_WIDTH>
int<ACCUM_WIDTH>
weight buffer
wgt_idx
weight tensor
(stored transposed)
BLOCK_IN
BLOCK_OUT
Bandwidth Derivation (bits per cycle)
WEIGHT_WIDTH*BLOCK_OUT*BLOCK_IN
INPUT_WIDTH*BATCH*BLOCK_OUT
ACCUM_WIDTH*BATCH*BLOCK_OUT
input buffer
inp_idx
input
tensor
BATCH
BLOCK_IN
GEMM
Core
accumulator
tensor
BATCH
BLOCK_OUT
register file
reg_idx
GEMM Instruction Pseudo-Code:
for i0 in range(0, end0):
for i1 in range(0, end1):
for uop_idx in range (uop_bgn, uop_end):
x, y, z = decode_gemm_indices(uop_buffer[upc])
reg_idx = i0 * x0 + i1 * x1 + x
inp_idx = i0 * y0 + i1 * y1 + y
wgt_idx = i0 * z0 + i1 * z1 + z
reg_file[reg_idx] += GEMM(inp_buff[inp_idx], wgt_buff[wgt_idx])
LOG_ACC_BUFF_DEPTH
LOG_INP_BUFF_DEPTH
LOG_WGT_BUFF_DEPTH
accumulator index (x)
input index (y)
weight index (z)
unused
0
31
Micro-Op Fields
micro-op buffer
uop_bgn
uop_idx
uop_end