n_in = 16
n_out = 8

data

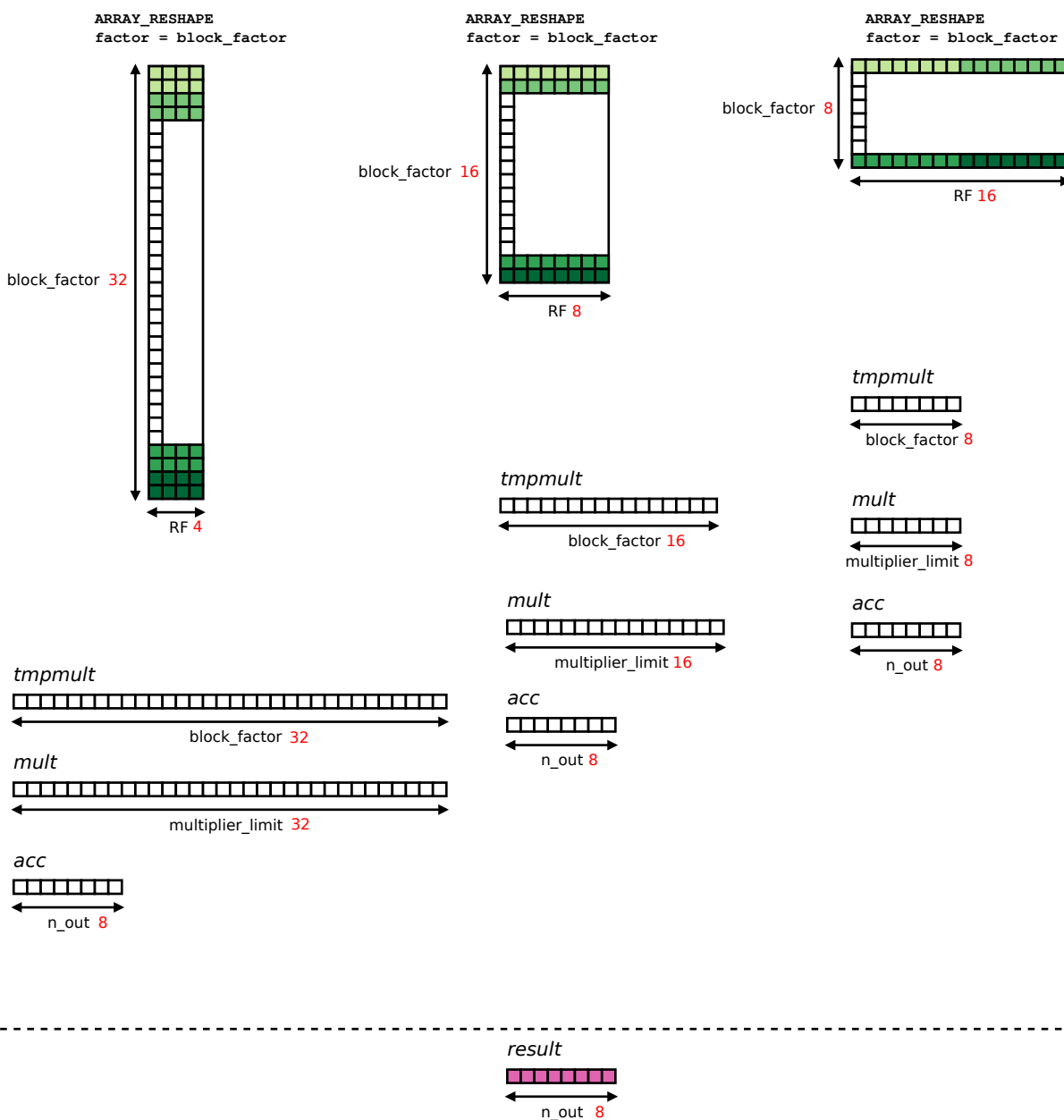n_in

weights

n_out

n_in

n_in * n_out

biases

n_out

$$block\_factor = \left\lceil \frac{n\_in * n\_out}{RF} \right\rceil$$

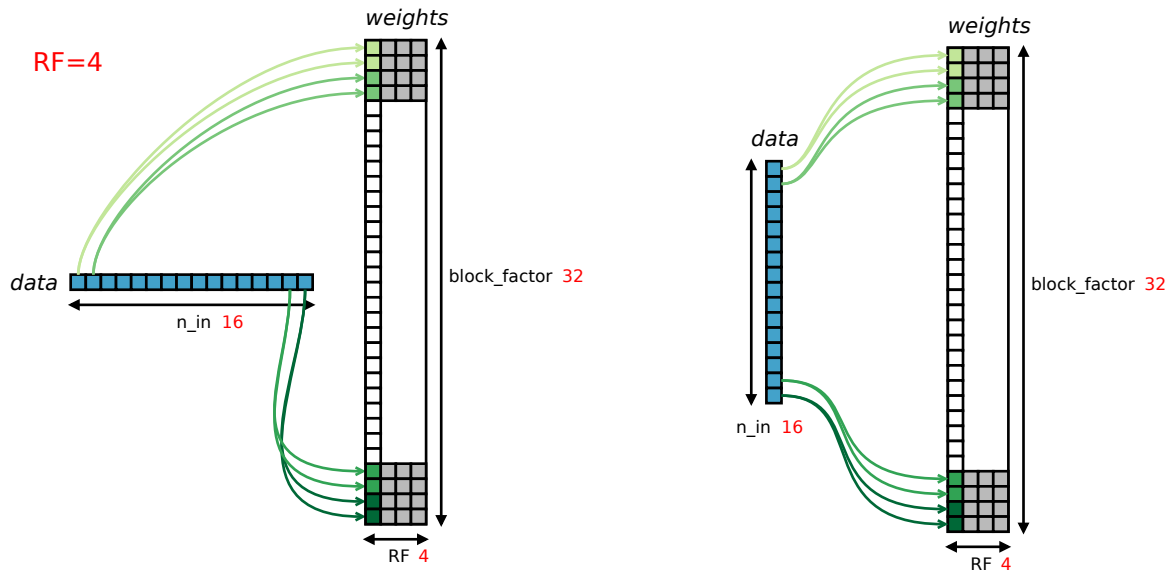$$multiplier\_limit = \left\lceil \frac{n\_in * n\_out}{min(n\_in, RF)} \right\rceil$$
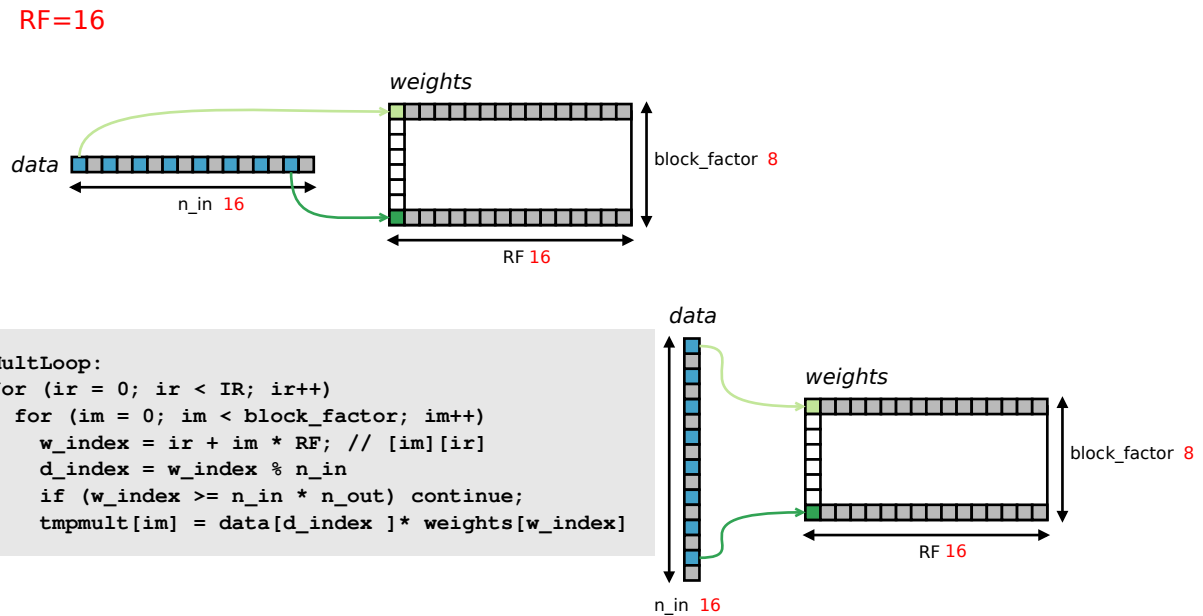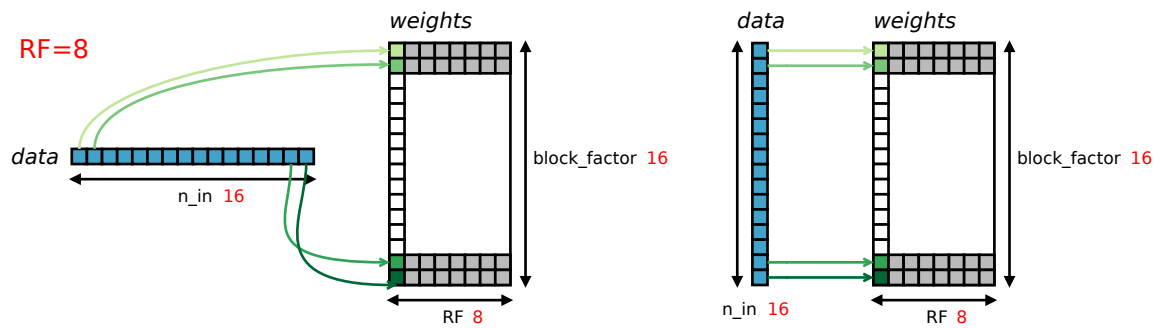
RF=4   **RF=8**   **RF=16**

- -

ARRAY_RESHAPE
factor = block_factor

ARRAY_RESHAPE
factor = block_factor

ARRAY_RESHAPE
factor = block_factor

block_factor 8

RF 16

block_factor 32

block_factor 16

RF 8

RF 4

tmpmult

block_factor 8

mult

multiplier_limit 8

acc

n_out 8

tmpmult

block_factor 16

mult

multiplier_limit 16

acc

n_out 8

tmpmult

block_factor 32

mult

multiplier_limit 32

acc

n_out 8

- -

result

n_out 8

Two different visualizations of MultLoop

RF=4 — weights — block_factor 32 — RF 4 — data — n_in 16

DEPRECATED

RF=8 — weights — block_factor 16 — RF 8 — data — n_in 16

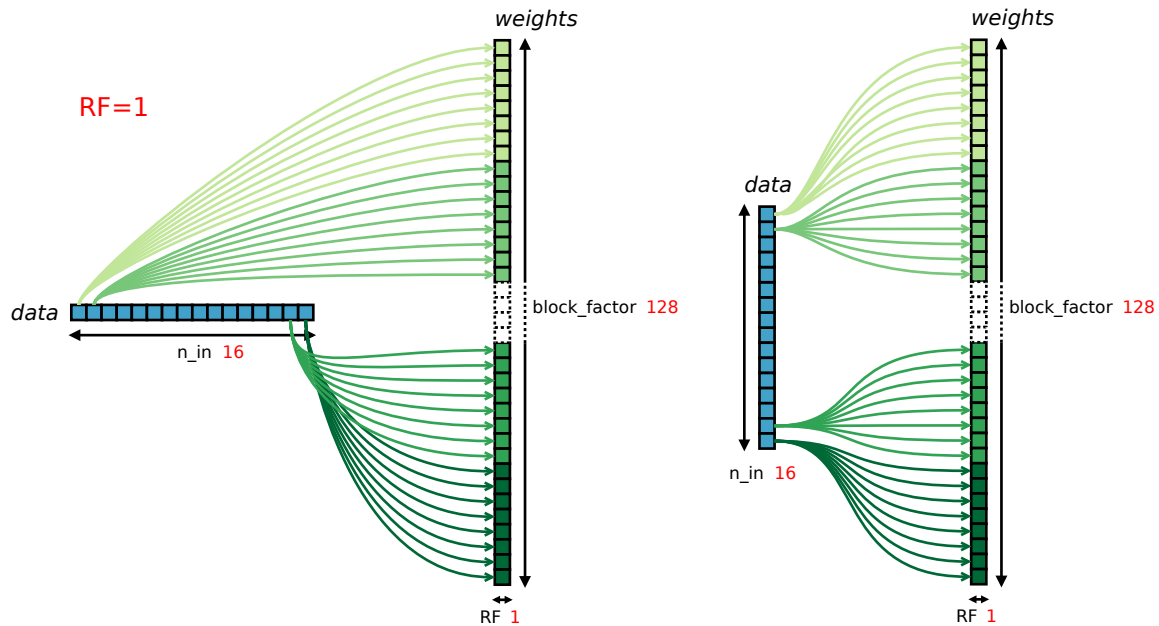RF=16 — weights — block_factor 8 — RF 16 — data — n_in 16

```
MultLoop:
for (ir = 0; ir < IR; ir++)
  for (im = 0; im < block_factor; im++)
    w_index = ir + im * RF; // [im][ir]
    d_index = w_index % n_in
    if (w_index >= n_in * n_out) continue;
    tmpmult[im] = data[d_index ]* weights[w_index]
```

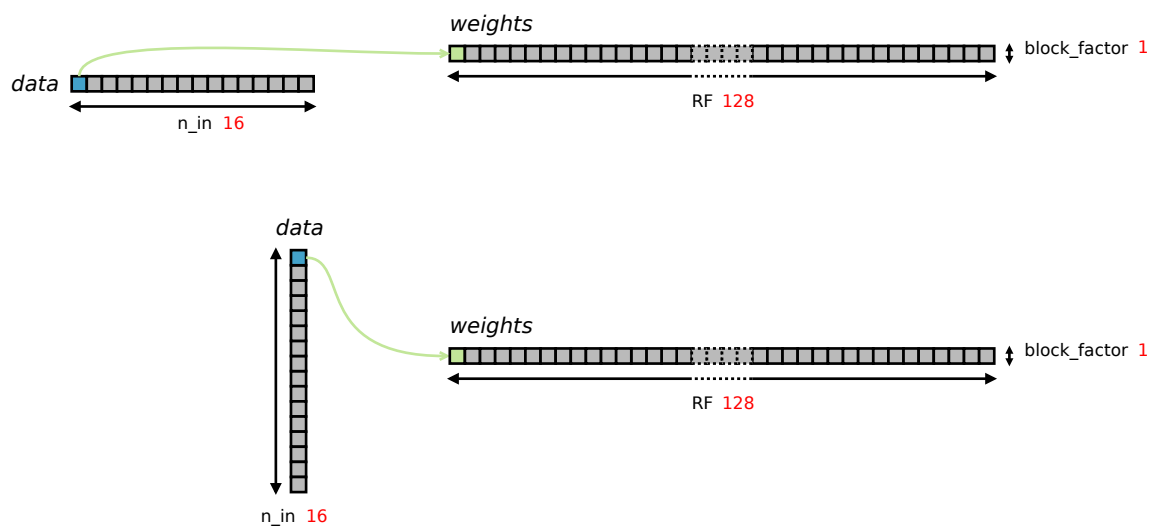Two extreme situations: (RF = 1, block_factor = 128) and (RF = 128, block_factor = 1)
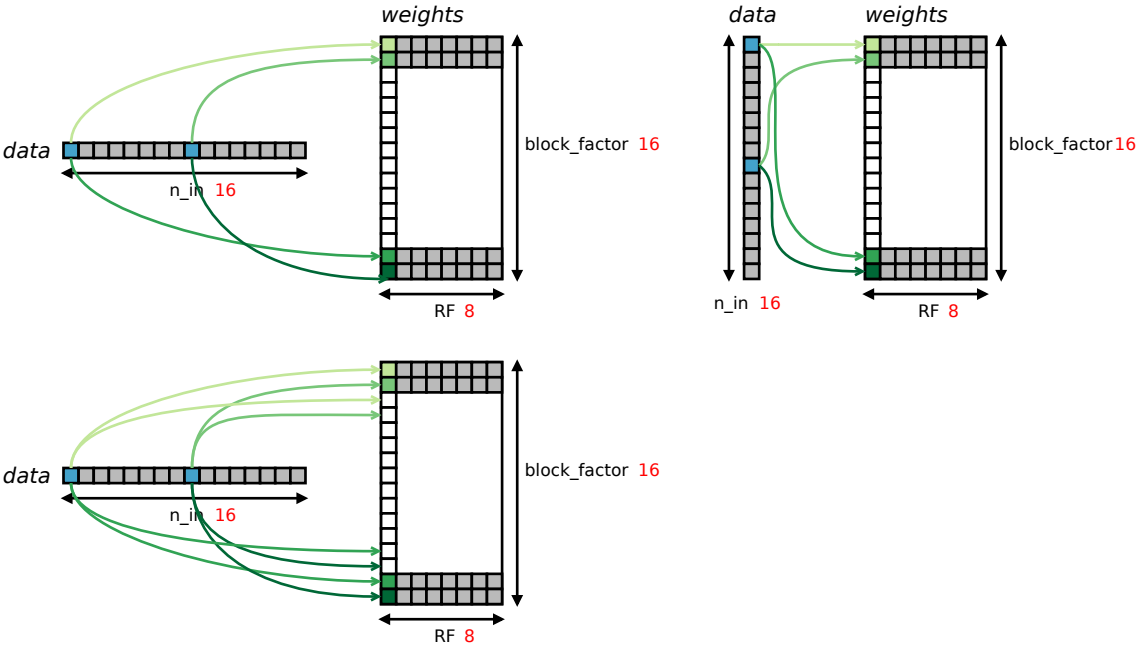

DEPRECATED

```
MultLoop:
for (ir = 0; ir < IR; ir++)
  for (im = 0; im < block_factor; im++)
    w_index = ir + im * RF; // [im][ir]
    d_index = w_index % n_in
    if (w_index >= n_in * n_out) continue;
    tmpmult[im] = data[d_index] * weights[w_index]
```

RF=8

weights

data

n_in 16

block_factor 16

RF 8

data *weights*

block_factor 16

n_in 16

RF 8

block_factor 16

data

n_in 16

RF 8

DEPRECATED