Scalar Arithmetic Multiple Data Customizable Precision for Deep Neural Networks

Andrew Anderson and Michael Doyle and David Gregg

Lero, Trinity College Dublin
{**aanderso**,mjdoyle,dgregg}@tcd.ie

ARITH, Kyoto
June 2019

DNN Convolution

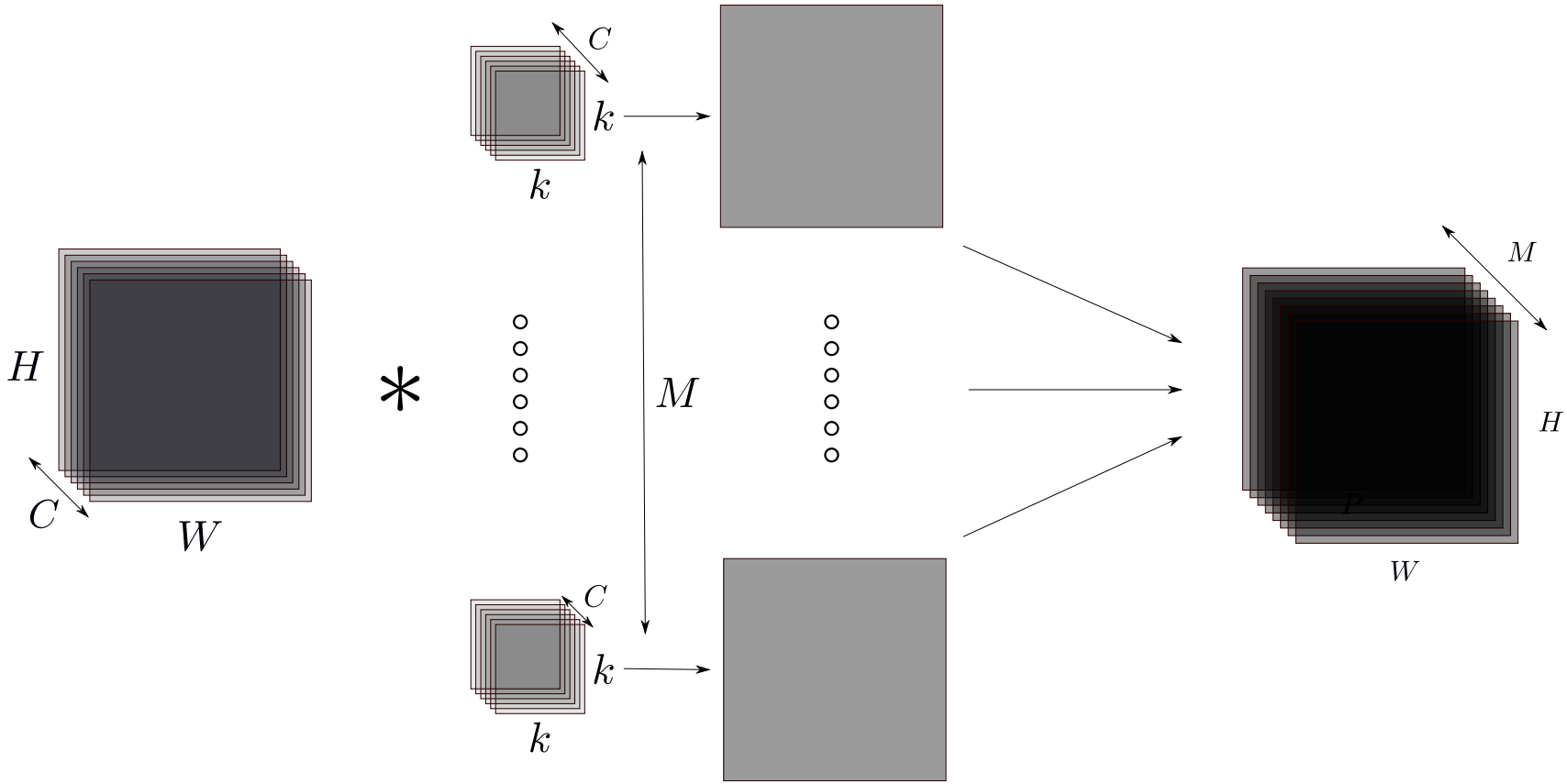

Figure: Multi-channel multi-kernel convolution

DNN Convolution

```
for (unsigned m = 0; m < kernels; m++)
  for (unsigned h = 0; h < img_h/stride_h; h++)
    for (unsigned w = 0; w < img_w/stride_w; w++)
      for (unsigned c = 0; c < channels; c++)
        for (unsigned y = 0; y < k; y++)
          for (unsigned x = 0; x < k; x++)
            output[m][h][w] +=
            input[c][((h * stride_h) + y) - (k/2)]
                    [((w * stride_w) + x) - (k/2)]
            *
            kernel[m][c][y][x];
```

Quantized Arithmetic

DNN weights occupy huge amounts of space in FP32
VGG-19 Network: 548 MB

Figure: But we want to use them on this!

OpenMV Cam – 512 KB RAM, 2 MB ROM, 216 MHZ Cortex-M7

Quantized Arithmetic

In Deep Learning we have it very easy!

- Network training compensates for arithmetic error
- Often, noisy arithmetic actually **helps**! (with overfitting)

Lots of research about how harshly DNN weights can be quantized

- Can go to integer (eventually!)
- Can go down to one (1) bit ('binarized' nets)
- But we don't want to do all our work on FPGA...
- In fact, commodity hardware is ideal.

The Simple Approach

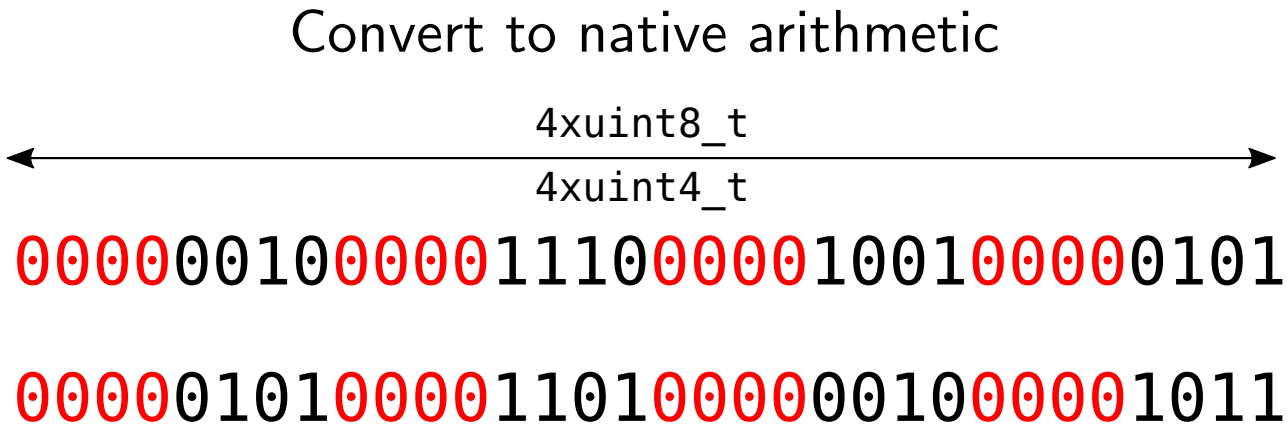

Figure: uint4_t expanded to uint8_t

- Can use native SIMD
- Space overhead only in registers (not memory)
- Extra precision in intermediate results (for free)
- Easy to mix and match number formats (e.g. $uint6\_t + uint4\_t$)

Quantized Arithmetic

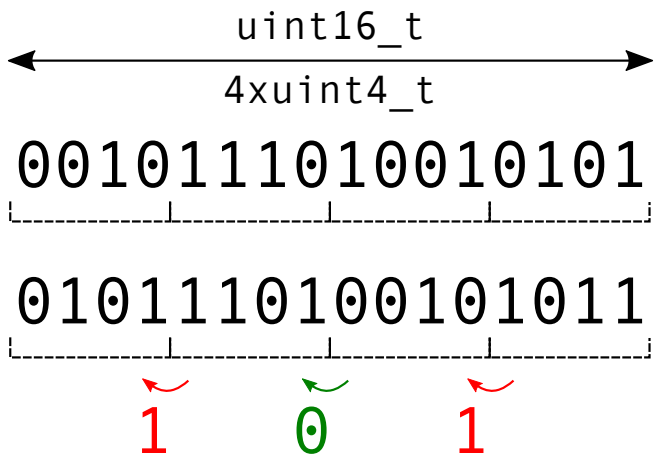

Figure: Example SWAR operation.

$4 \times$ 4-bit words packed into a 16-bit scalar register

SIMD Within A Register (SWAR)

Dealing with overflow

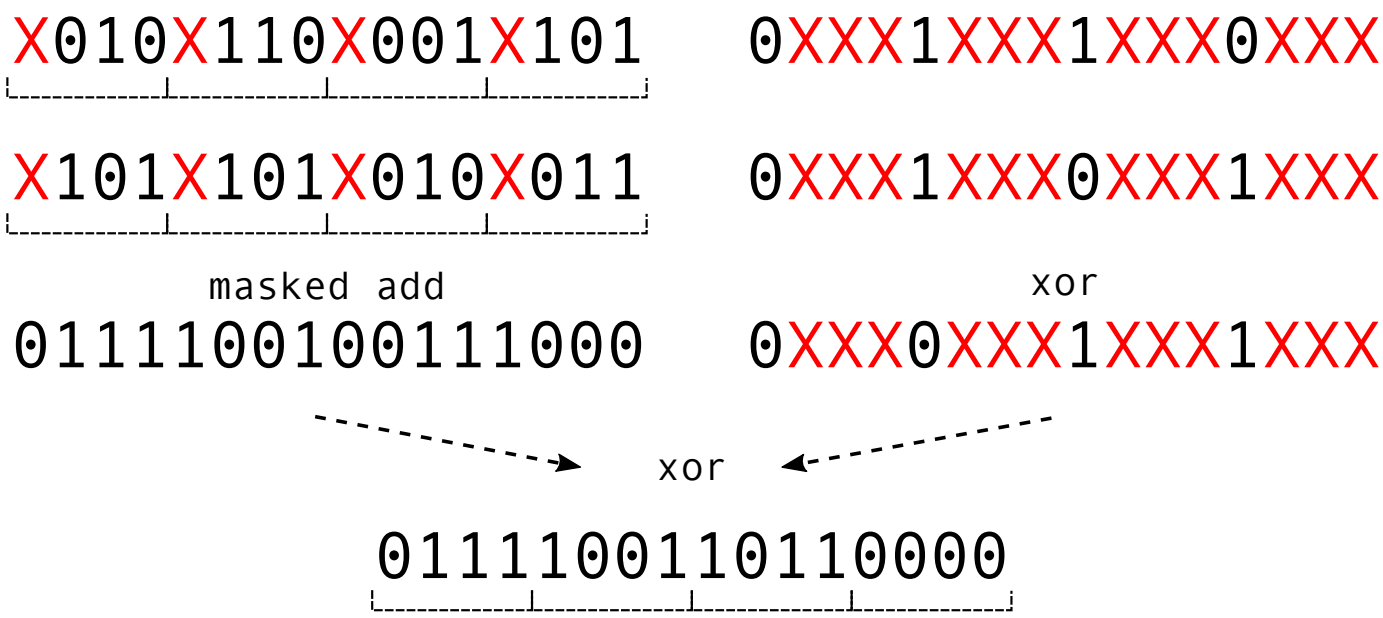

Figure: Spacer bits

Temporary spacer bits are spacer bits in intermediate values that don't get written to the data format in memory.

SIMD Within A Register (SWAR)

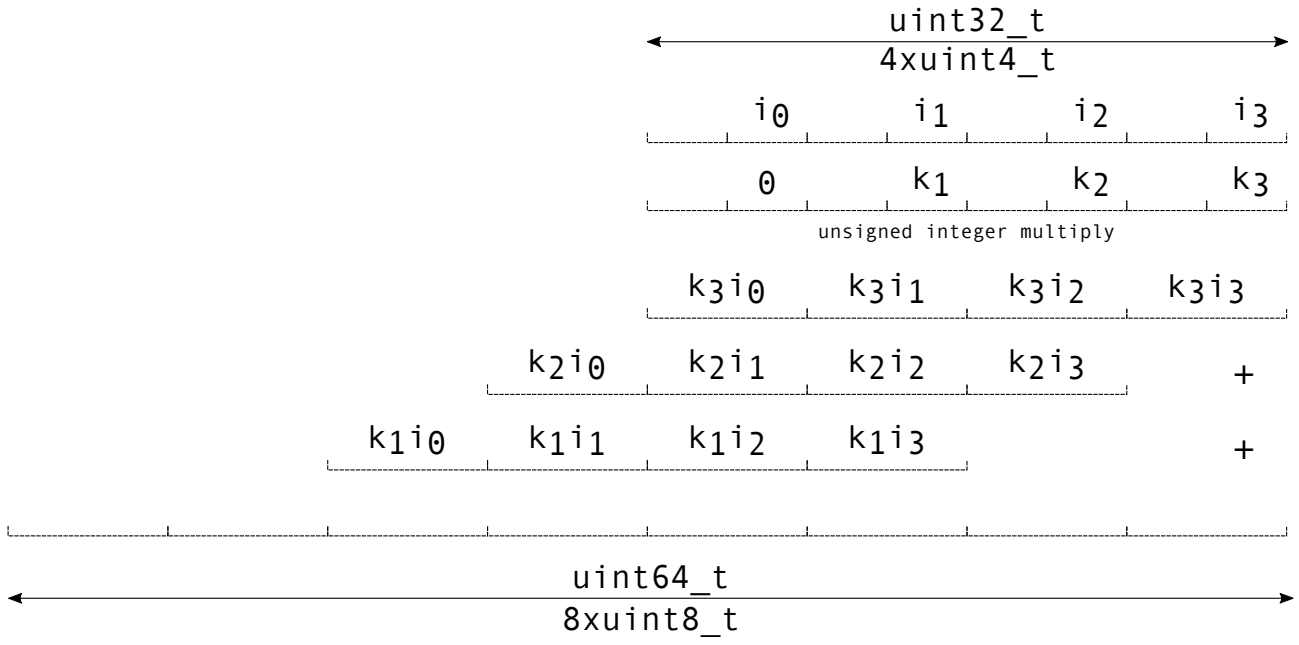

Figure: Convolutional substructure in scalar integer multiplication

Long multiplication is discrete convolution over digit sequences

SIMD Within A Register (SWAR)

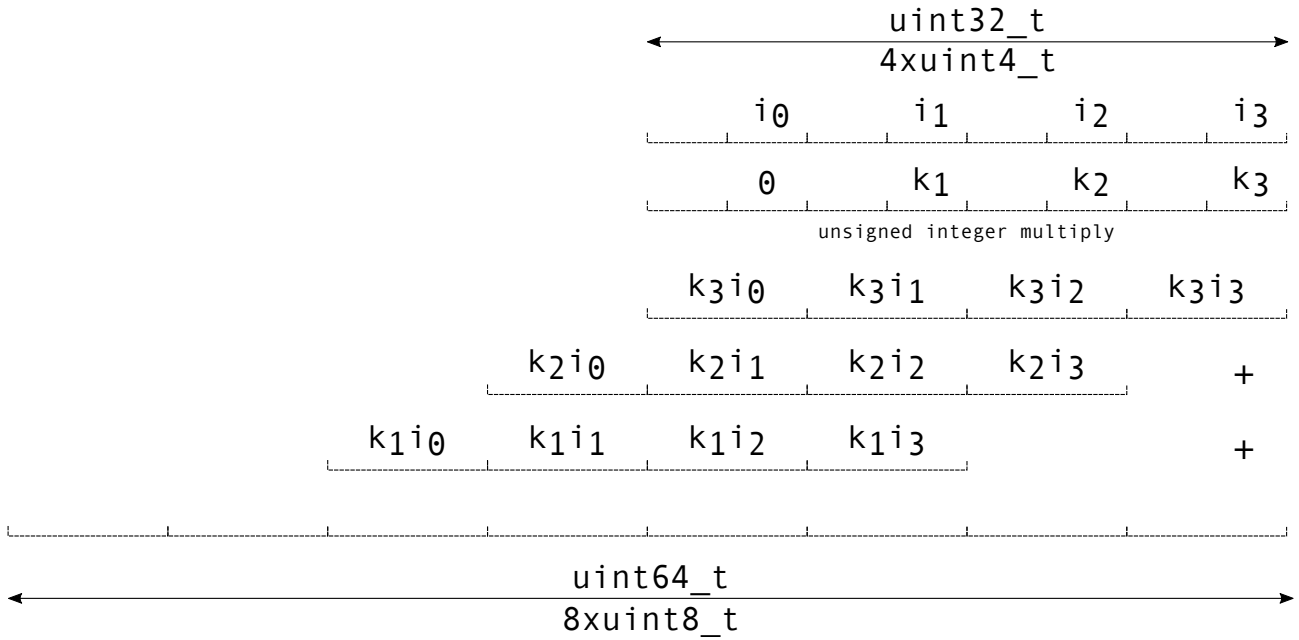

Figure: Convolution

$k \times i$ subword multiplies and $(k-1) \times (i-1)$ additions with a single instruction

Results

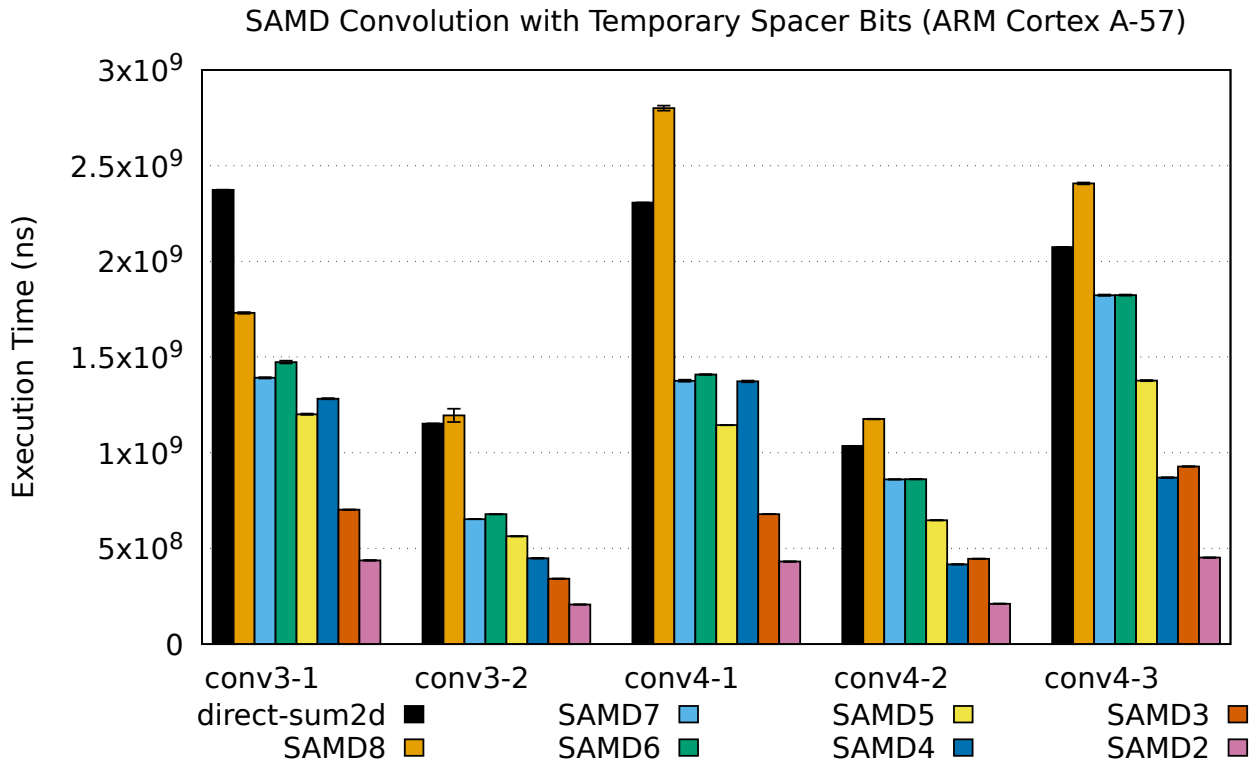

Figure: Performance with Temporary Spacer bits

Results

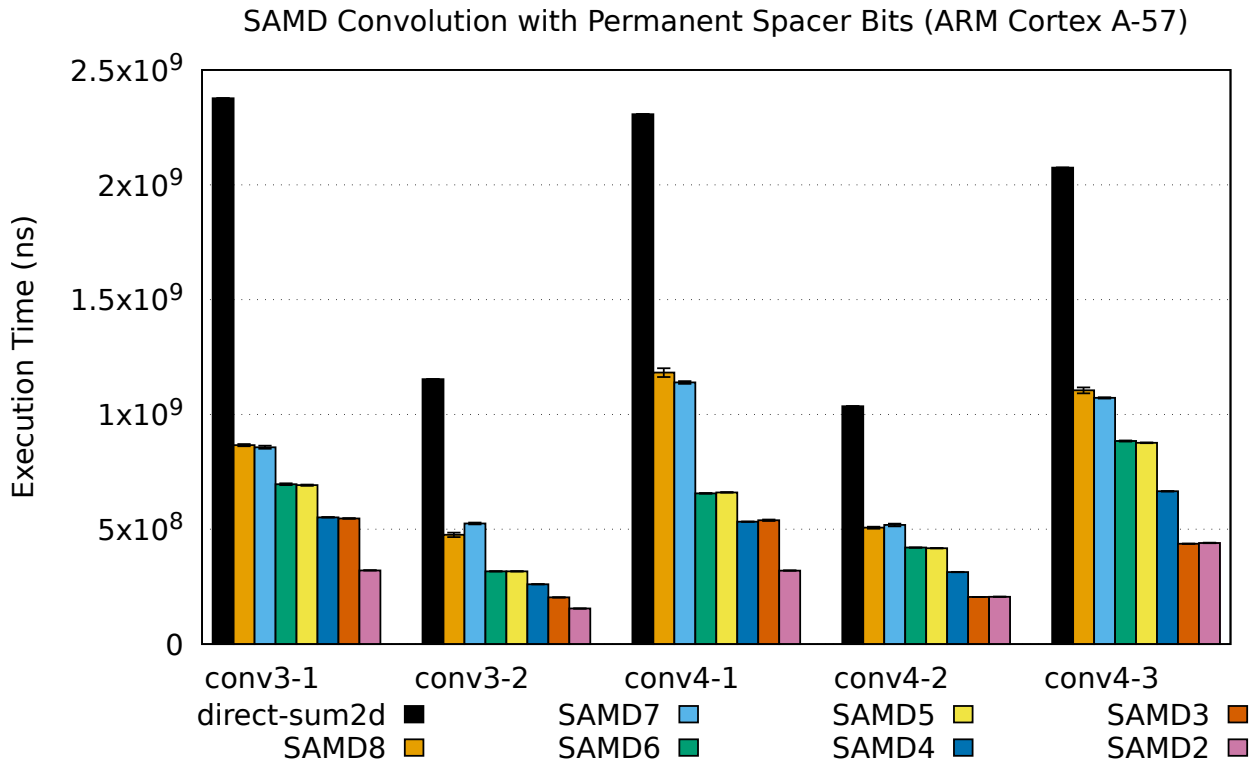

Figure: Performance with Permanent Spacer bits

Future Work

- All-SAMD network (nonlinearities & utility ops)
- Codesign HW Integer Support Instructions
- GPU (but microcontrollers don't have GPUs (yet!))

Thanks for listening!