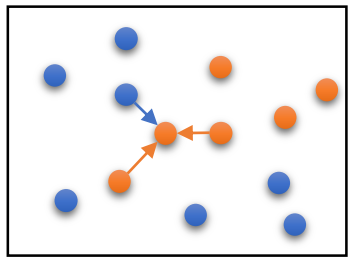
FPGA BASED LOW LATENCY, LOW POWER STREAM PROCESSING AI

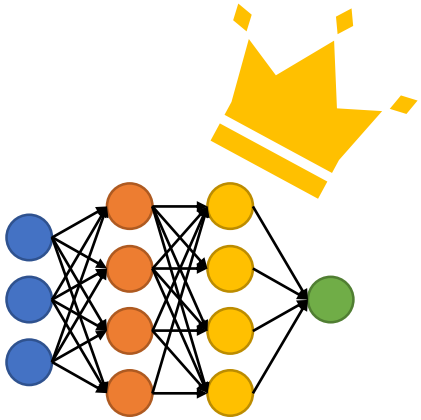
Domenik Helms[(1)], Mark Kettner[(1)], Behnam Razi Perjikolaei[(1)], Lukas Einhaus[(2)], Christopher Ringhofer[(2)], Chao Qian[(2)], Gregor Schiele[(2)]

[(1)] OFFIS, [(2)] University of Duisburg-Essen

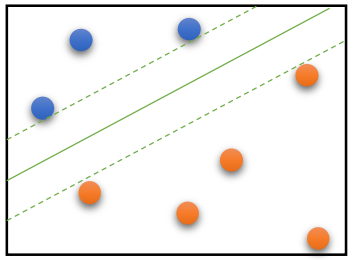
Motivation



nearest neighbor

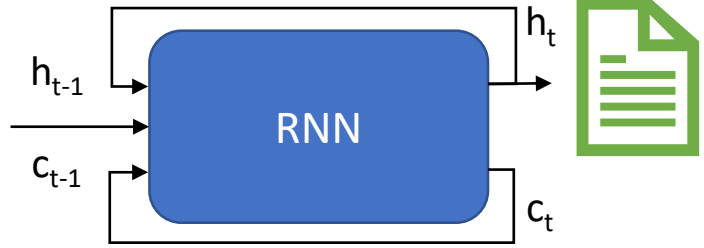


neural network



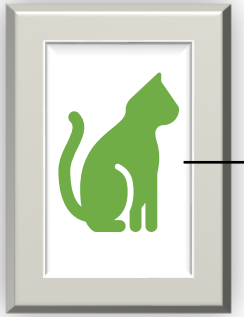
support vector machine

etc. …



recurrent neural network

“heart attack”



“cat”

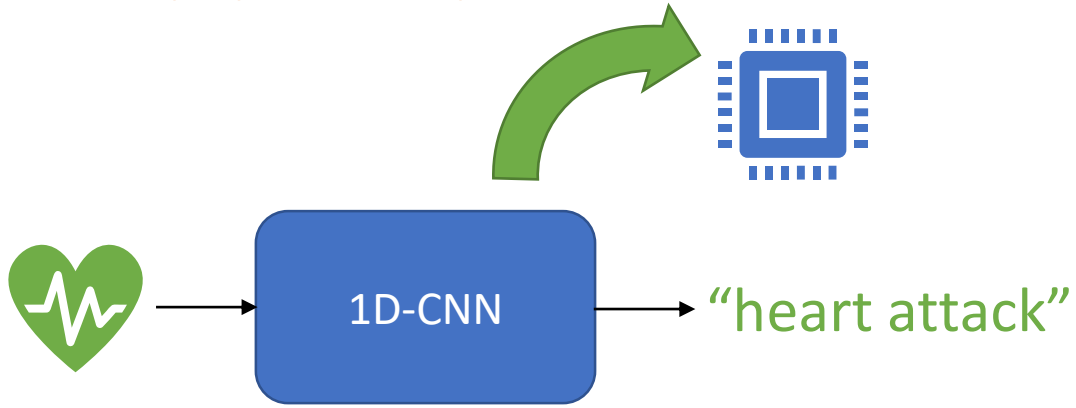
convolutional neural network

Motivation

- Wanted:
  - high throughput
    - ↯ limited mainly by data access bandwidth
  - power efficiency
    - ↯ limited mainly by data access cost
  - radiation hardness
    - ↯ limited mainly by weak sequential cells

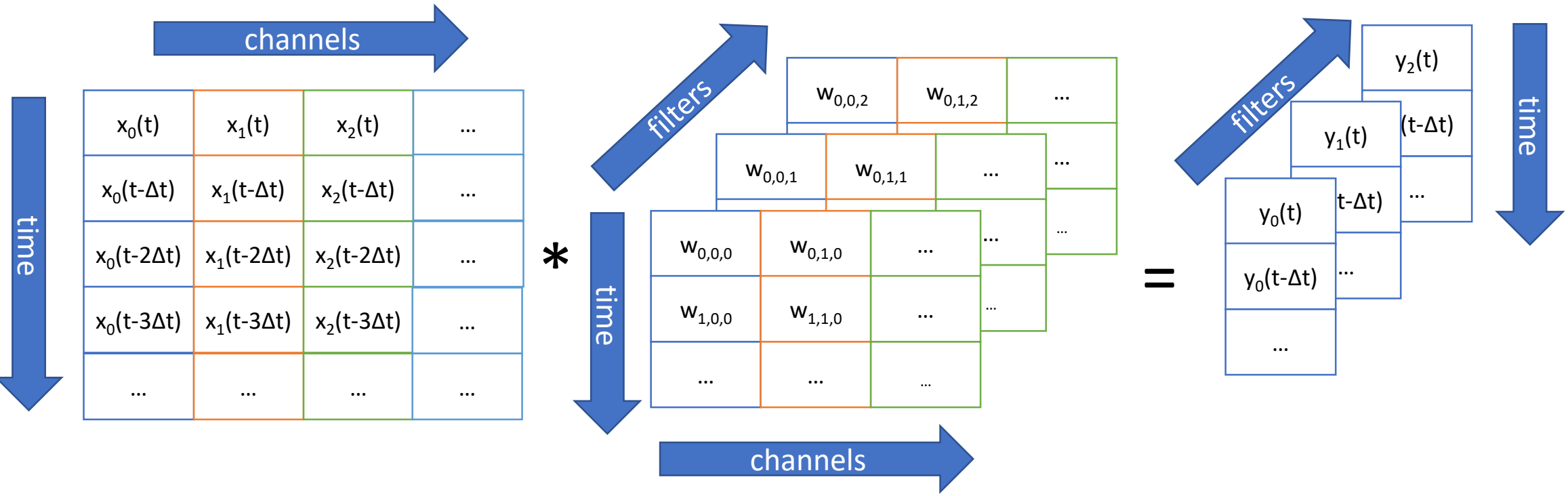
→ limited mainly by memory!!!

do we need memory?

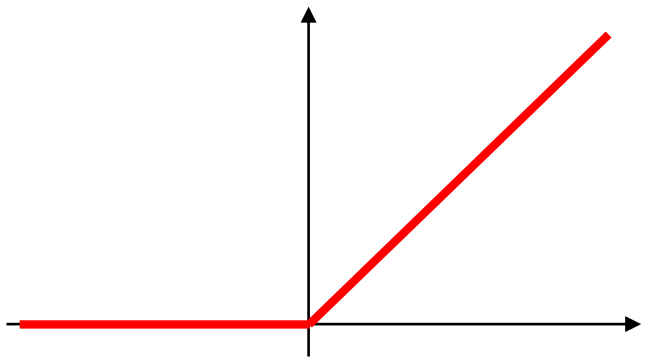


How to build simple NN without memory?

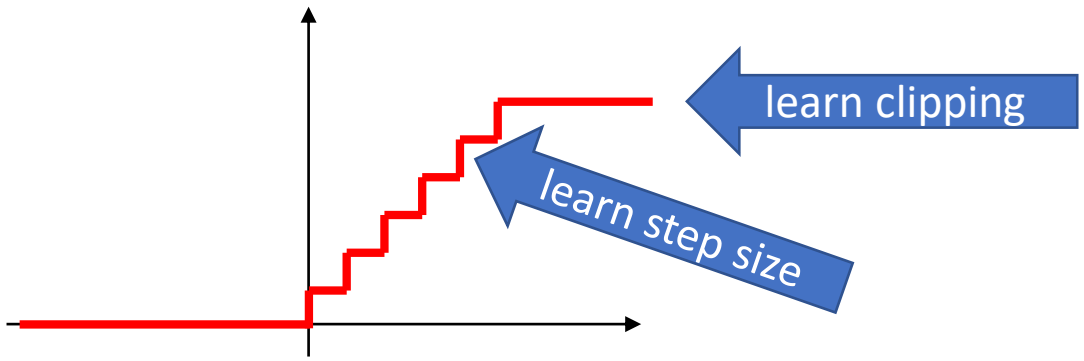
1D-Convolution: $$y_f(t) = max\left(0, \sum_{c=0}^{C-1} \sum_{i=0}^{N-1} w_{i,c,f} \cdot x_c(t - i \cdot \Delta t) + b_f\right)$$



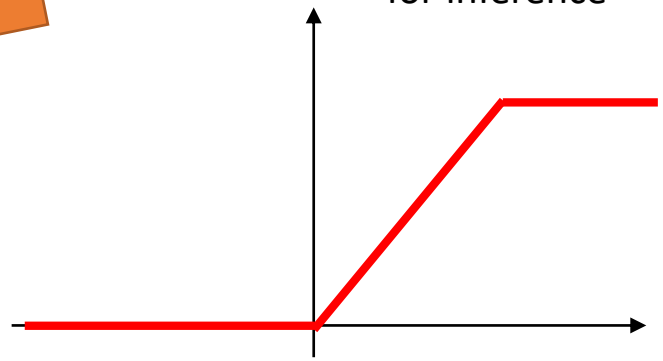
Step 1: Quantization



ReLU

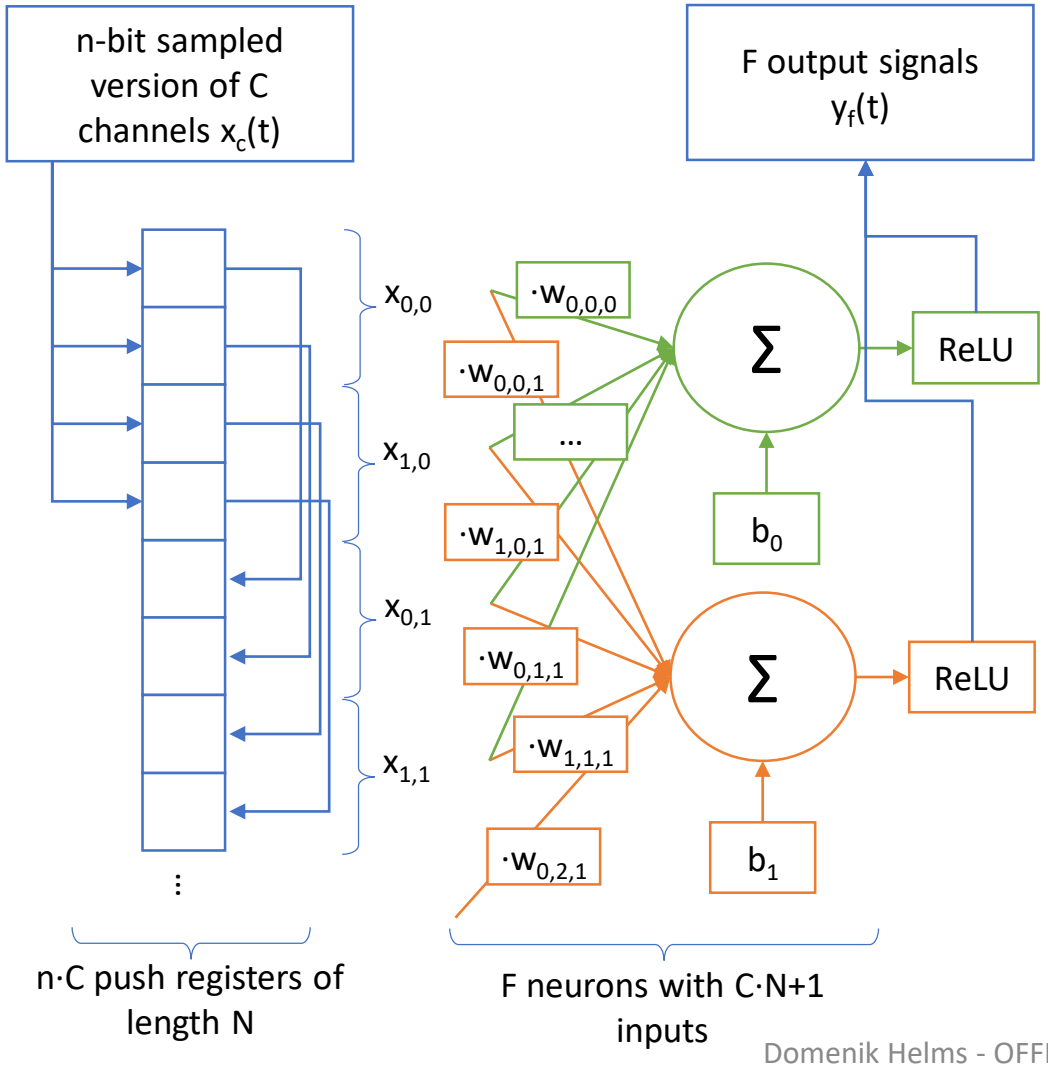


DoReFa and PACT
for inference

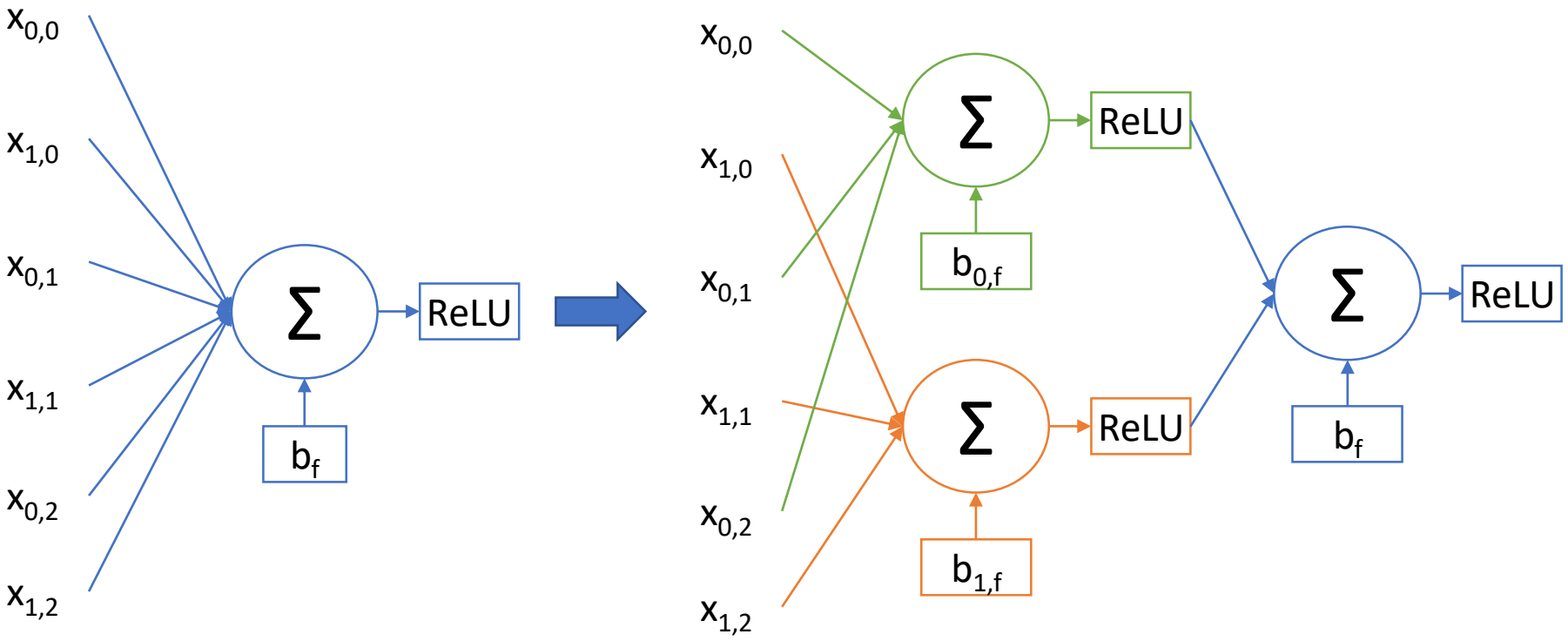


straight through estimator
for backpropagation

Step 2: Convolution



Step 3: Reducing the input count



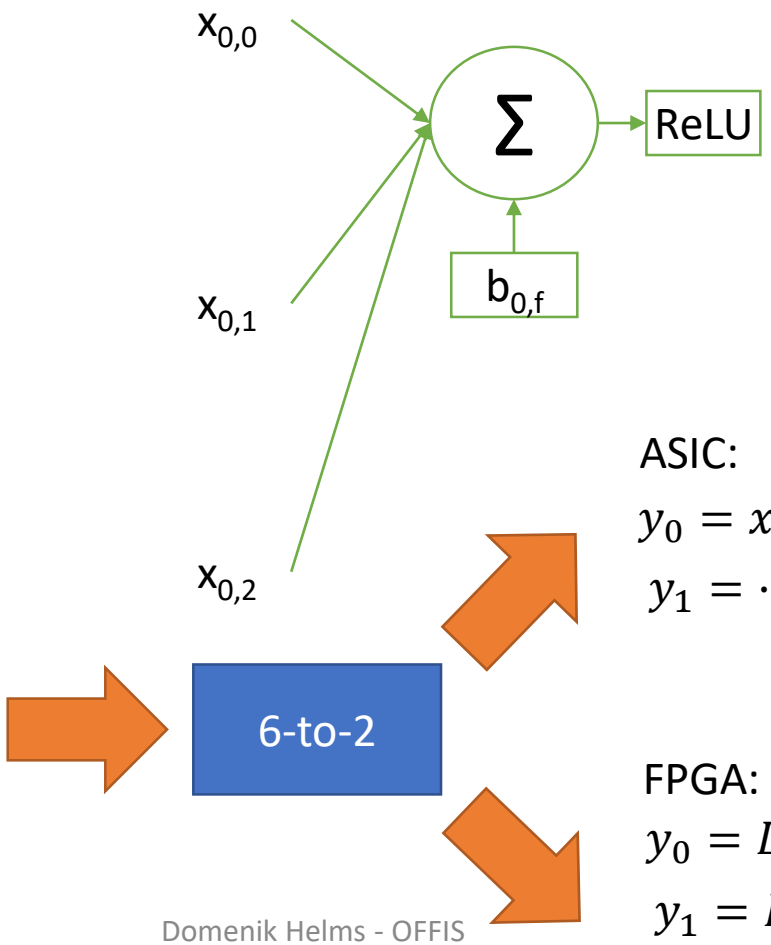
Step 4: Precompute multiplication

$$y = max\left(0, \sum_{i=0}^{N-1} w_i \cdot x_i + b\right)$$

$$w_0 = 0.43, w_1 = -0.24$$
$$w_2 = 0.65, b = 0.44$$

x_0	x_1	x_2	y	$\lfloor y \rfloor$
0	0	0	0.44	0
0	0	1	1.09	1
0	0	2	1.74	2
0	0	3	2.39	2
0	1	0	0.2	0
0	1	1	0.85	1
⋮	⋮	⋮	⋮	⋮
1	0	0	0.87	1
1	0	1	1.52	2
⋮	⋮	⋮	⋮	⋮
3	3	2	2.31	2
3	3	3	2.96	3

x						y	
0	0	0	0	0	0	0	0
0	0	0	0	0	1	0	1
0	0	0	0	1	0	1	0
0	0	0	0	1	1	1	0
0	0	0	1	0	0	0	0
0	0	0	1	0	1	0	1
0	1	0	0	0	0	0	1
0	1	0	0	0	1	1	0
1	1	1	1	1	0	1	0
1	1	1	1	1	1	1	1



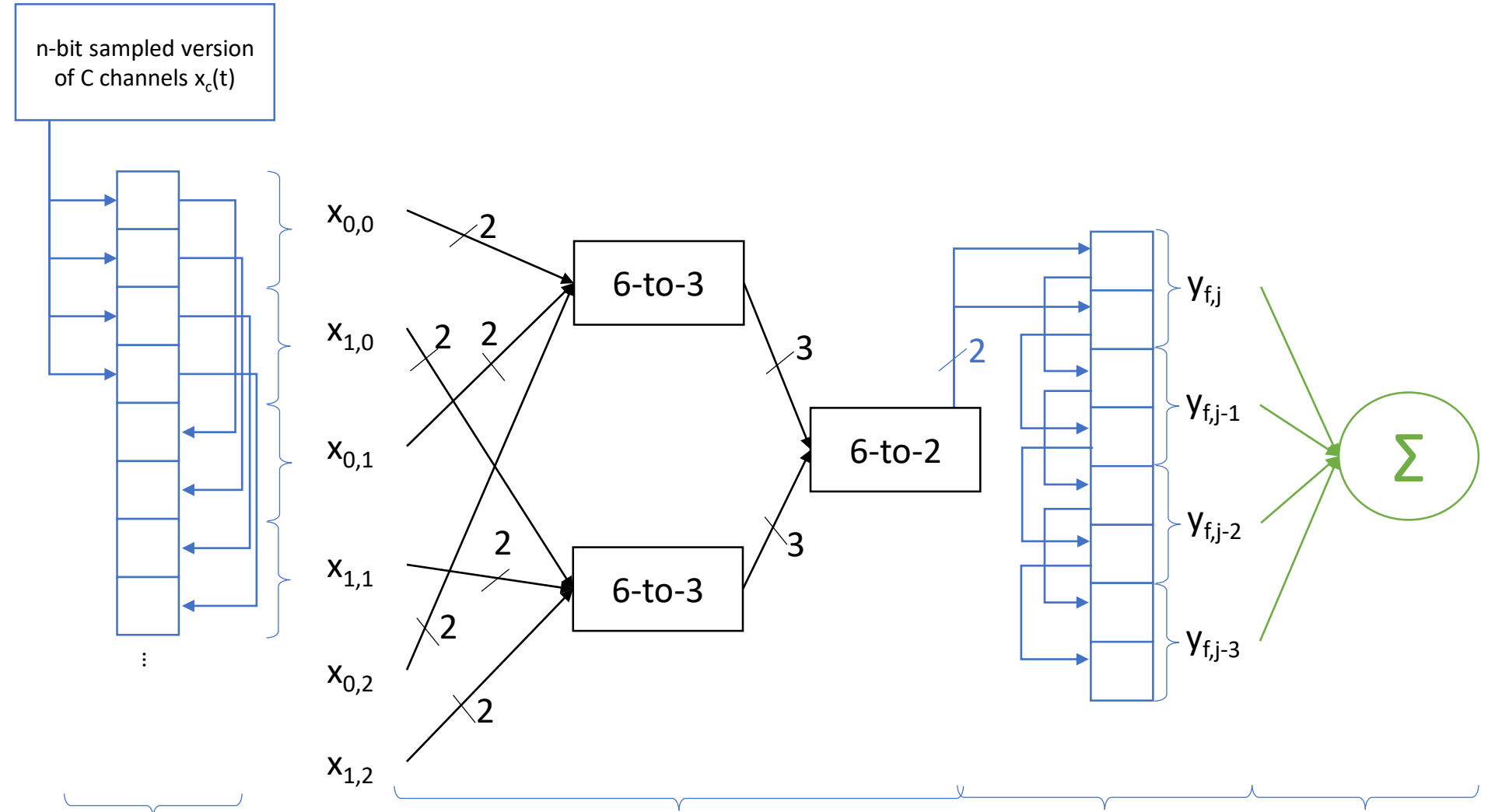
ASIC:

$$y_0 = x_0 \cap (\overline{x_1} \cup x_4) \cup \overline{x_3} \cap (x_5 \cup x_6) \cup \ \ldots$$
$$y_1 = \cdots$$

FPGA:

$$y_0 = LUT(x_0, x_1, \ldots)$$
$$y_1 = LUT(x_0, x_1, \ldots)$$

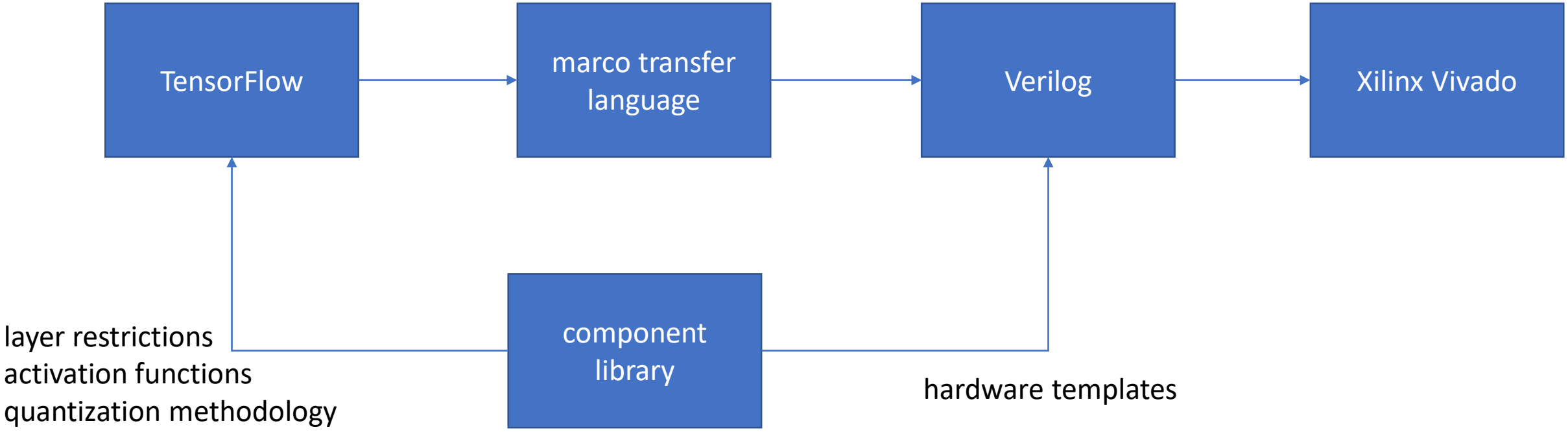
Step 5: Pooling



Results

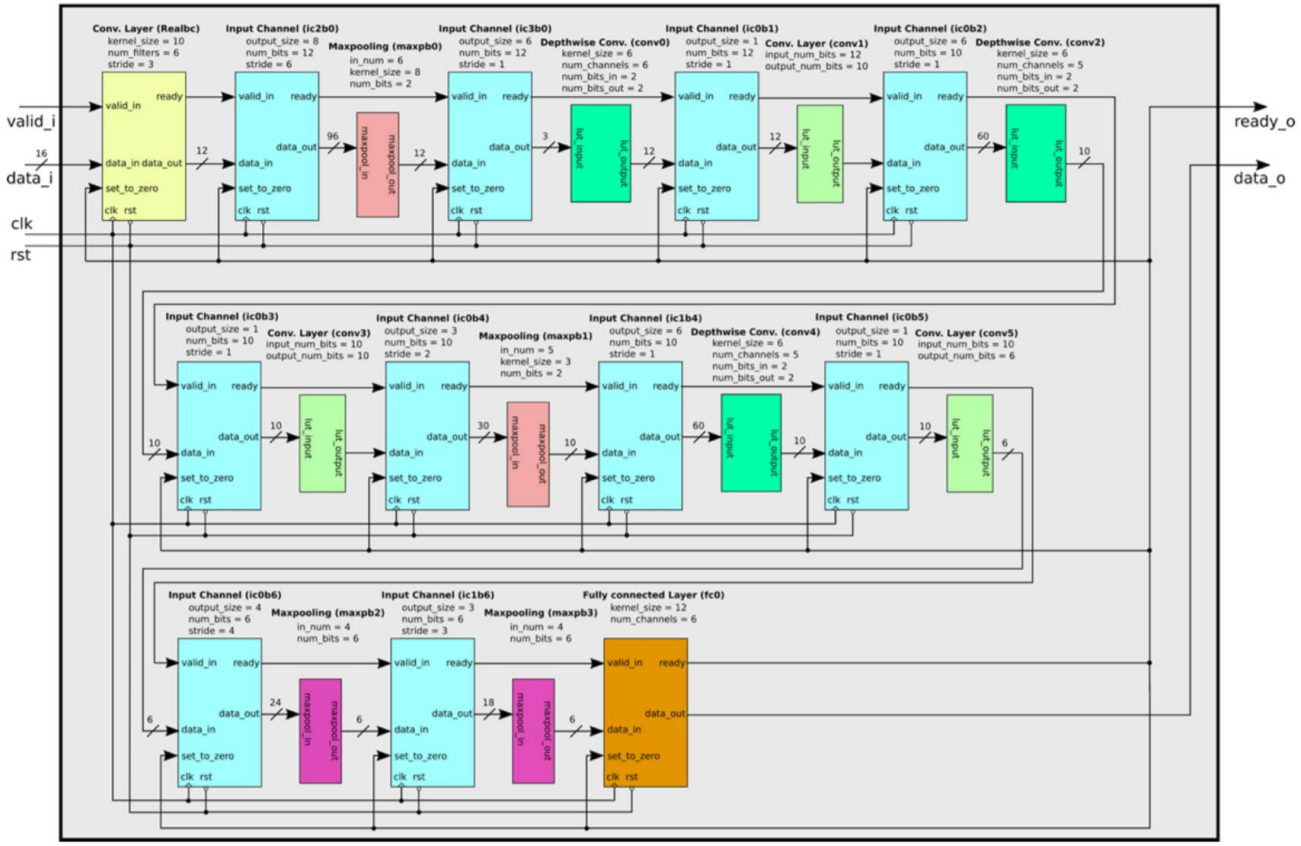
- All activations are quantized and stored in registers
  - quick, efficient and radiation hard
- All weights and biases are still floating values and implicitelly coded into the Boolean functions
  - better training
- No RAM access needed at all
- Each neuron is explicitly and exclusively implemented
  - ~300 transistors per neuron in ASIC
  - ~10 LUTs per neuron in FPGA
- But:
  - it is not possible to retrain ASIC (needs reproduction ➔ $)
  - FPGA can be retrained, but it contains 5T SRAM cells ➔ radiation)

Flow automation



Evaluation

- Spartan 7 S15
- 2155 LUT (26%)
- 26nJ per sample (full inference)
- 10MHz input sample rate



Conclusion

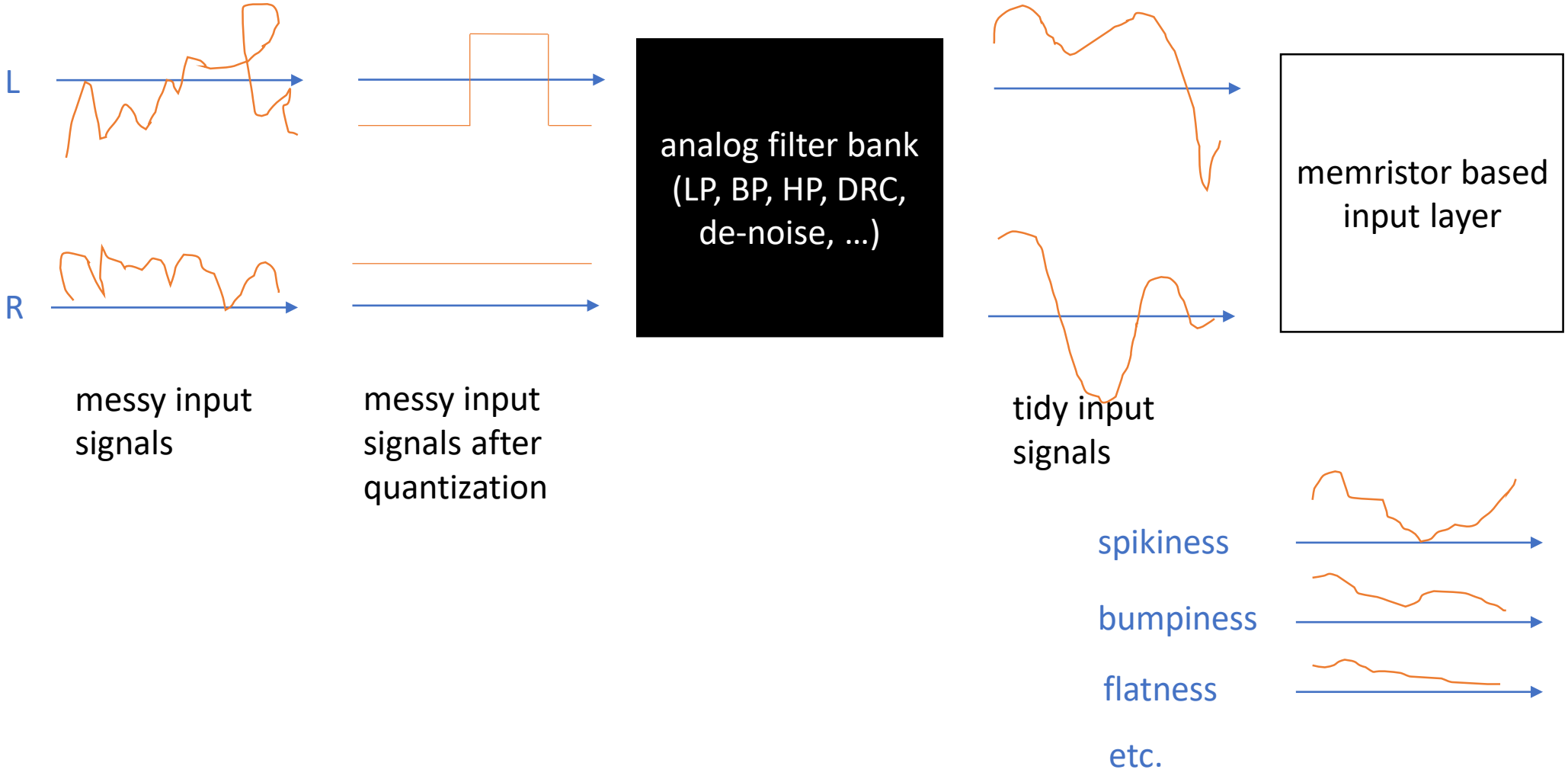
- The method has some fundamental limitations
  - low bitwidth
  - low synapse count
  - 1 dimensional convolution

- Within those limitations, it outperforms any other methodology
- An ASIC solution would improve performance and efficiency by at least an order of magnitude

The OCTOPUS project idea

1. Develop an analogue frontend chip
2. Improve the methodology
3. Medical application
4. Industry application

1 Develop an analogue frontend chip



2 Extend the methodology

available

- Conv1D
- Max Pooling 1D
- Dense
- Global average pooling
- quantization while training
- depthwise separation
- ReLU

planned

- Conv2D
- other std. activation
- nonlinear tensor functions
- width&height-wise separation

$$y = max\left(0, \sum_{i=0}^{N-1} v_i \cdot x_i^2 + w_i \cdot x_i + b\right)$$

3 Health application

- Use the system as is to detect blood pressure
  - determine blood speed by following the pressure peaks

4 Railway application

- Measure the sound (in air or in metal) of a train passing
- use on sensor AI to detect
  - axle traversing
  - weight (load) per axle
  - speed of train
  - type of train
- carefully!!! design and verify a system which uses unreliable AI components in a safety critical system
  - decide if the track is free based on our system's "best guess"