BrainTTA: A 28.6 TOPS/W Compiler Programmable Transport Triggered NN SoC

IEEE INTERNATIONAL CONFERENCE ON COMPUTER DESIGN - 2023

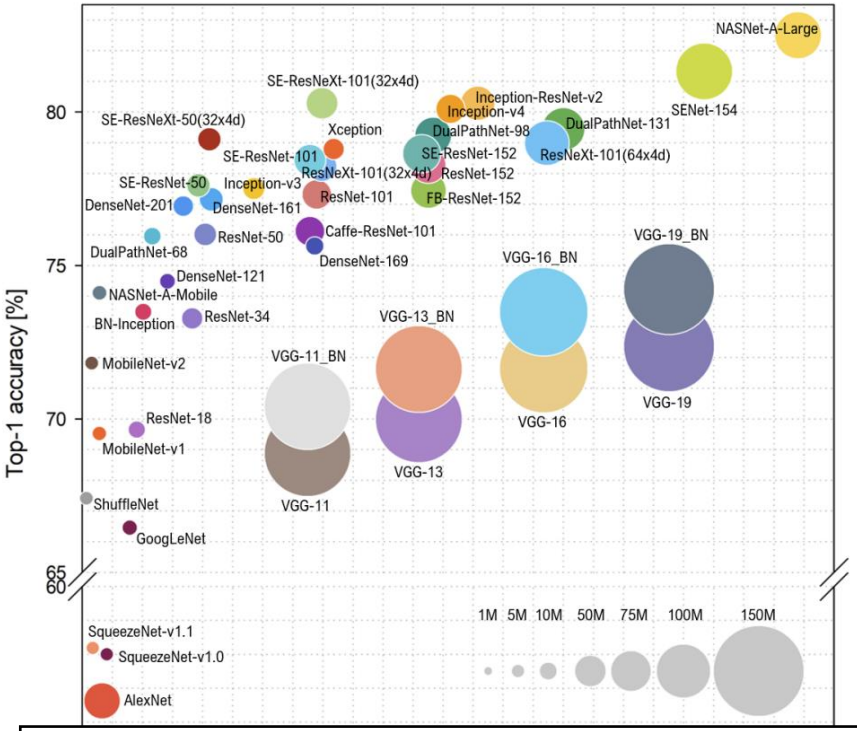
Maarten J. Molendijk[1,2], Floran A.M. de Putter[1], Manil Dev Gomony[1], Pekka Jääskeläinen[3], Henk Corporaal[1]

[1]Eindhoven University of Technology, the Netherlands
[2]NXP Semiconductors, the Netherlands
[3]Tampere University, Finland

Tampere University

TU/e EINDHOVEN UNIVERSITY OF TECHNOLOGY

NN Architecture and Hardware Development Cost

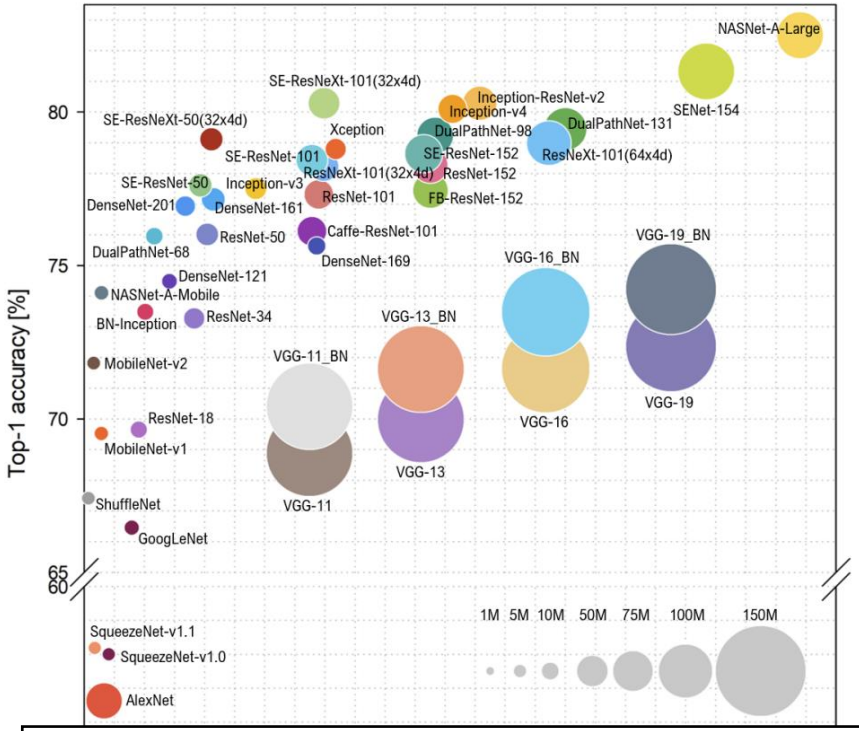


Source: [2]

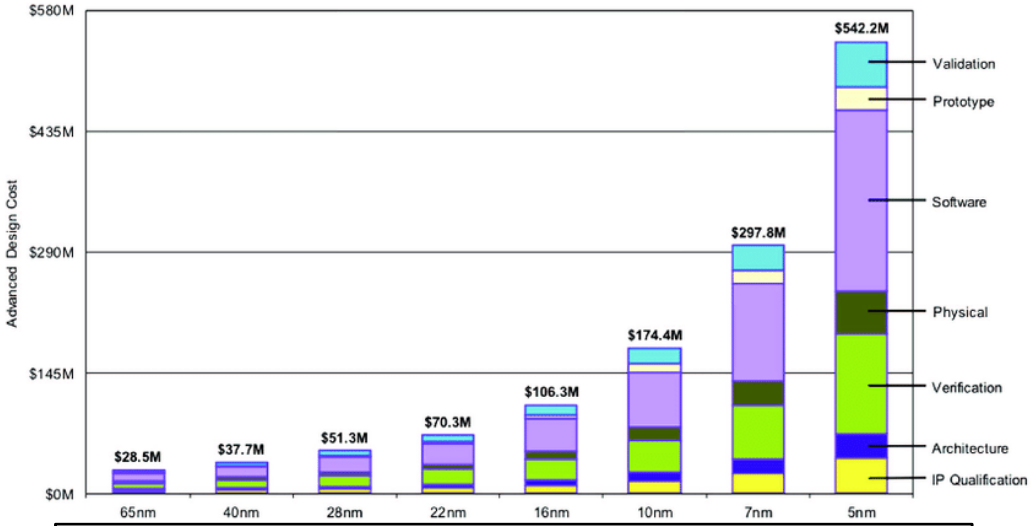
- Large **variety** of NN architectures
- Rapidly evolving

[1] S. Hashemi et. al. "Understanding the impact of precision quantization on the accuracy and energy of neural networks," DATE 2017
[2] https://www.researchgate.net/figure/Chip-Design-and-Manufacturing-Cost-under-Different-Process-Nodes-Data-Source-from-IBS_fig1_340843129

NN Architecture and Hardware Development Cost



- Large **variety** of NN architectures
- Rapidly evolving

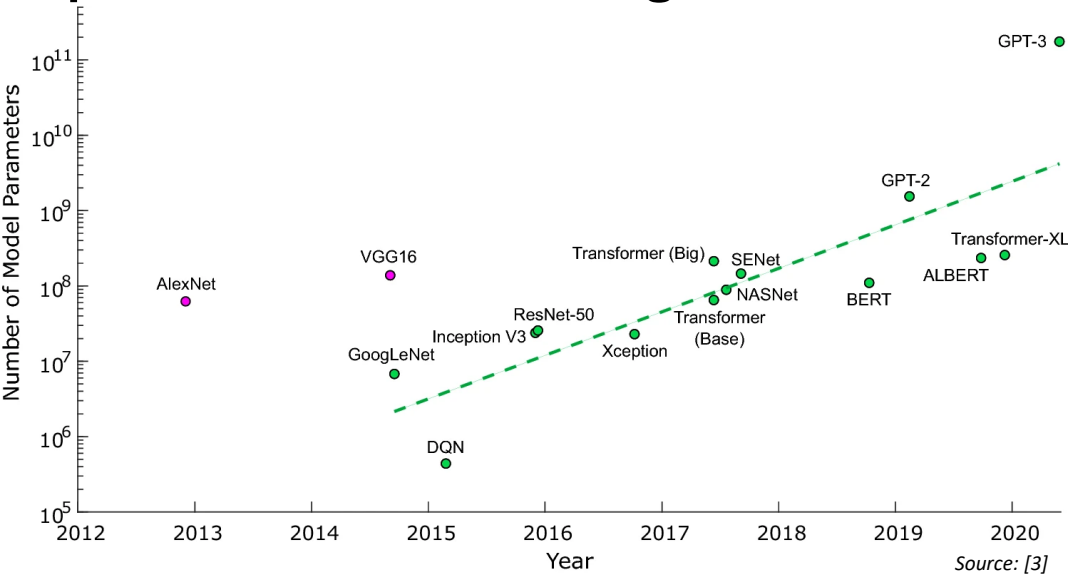


- NN memory + compute reqs
- Increased HW development cost

→ **reconfigurable solution**

[1] S. Hashemi et. al. "Understanding the impact of precision quantization on the accuracy and energy of neural networks," DATE 2017
[2] https://www.researchgate.net/figure/Chip-Design-and-Manufacturing-Cost-under-Different-Process-Nodes-Data-Source-from-IBS_fig1_340843129

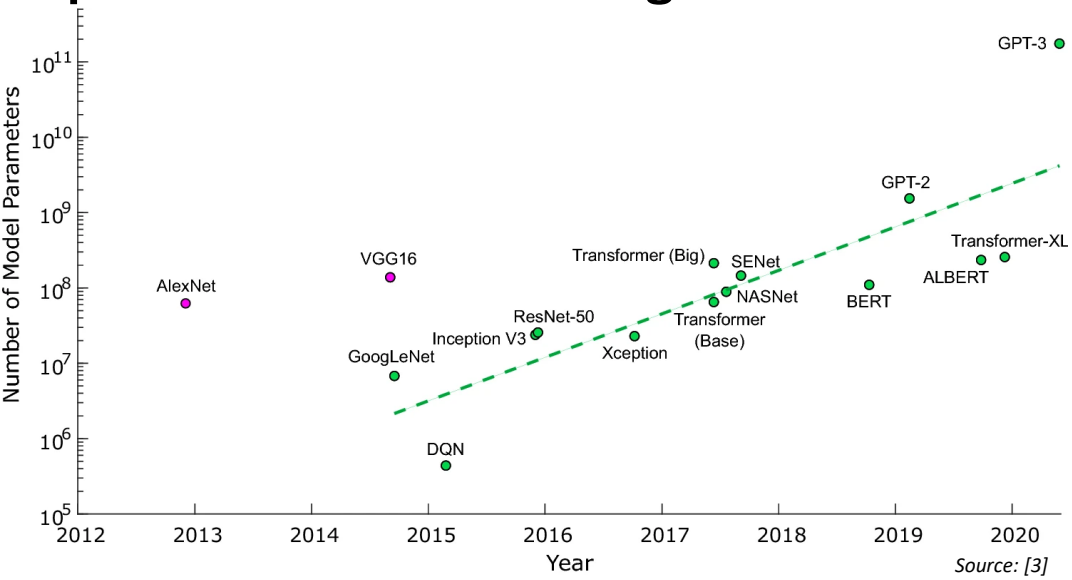
Operand Precision Scaling

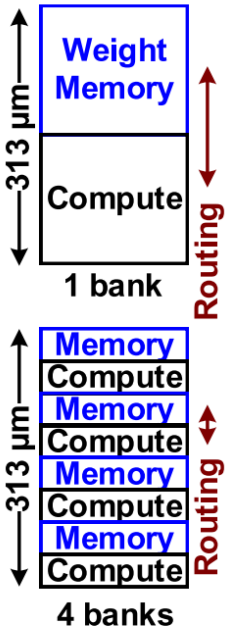


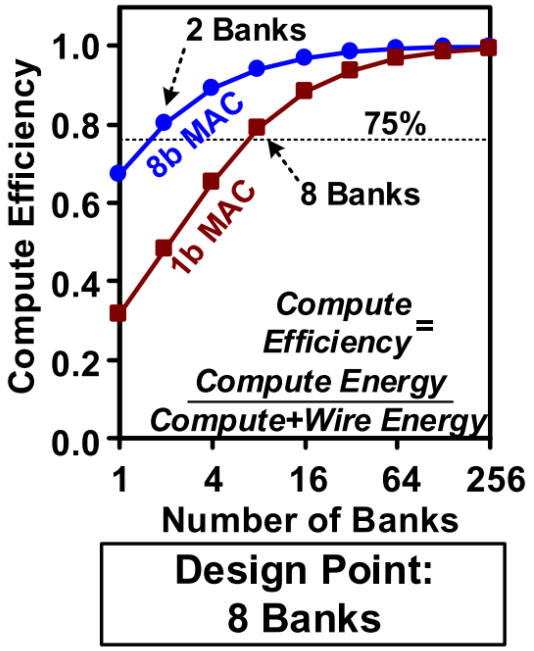
- Deployment on edge devices → **Quantization**
- Operand width ↓
  - MAC HW **superlinearly ↓↓**
  - Overhead **(sub)linearly ↓**

[3] https://www.researchgate.net/figure/Number-of-parameters-ie-weights-in-recent-landmark-neural-networks1-2-31-43_fig1_349044689
[4] P. C. Knag et al., "A 617-TOPS/W All-Digital Binary Neural Network Accelerator in 10-nm FinFET CMOS," IEEE J. Solid-State Circuits (JSSC), 2021.

Operand Precision Scaling





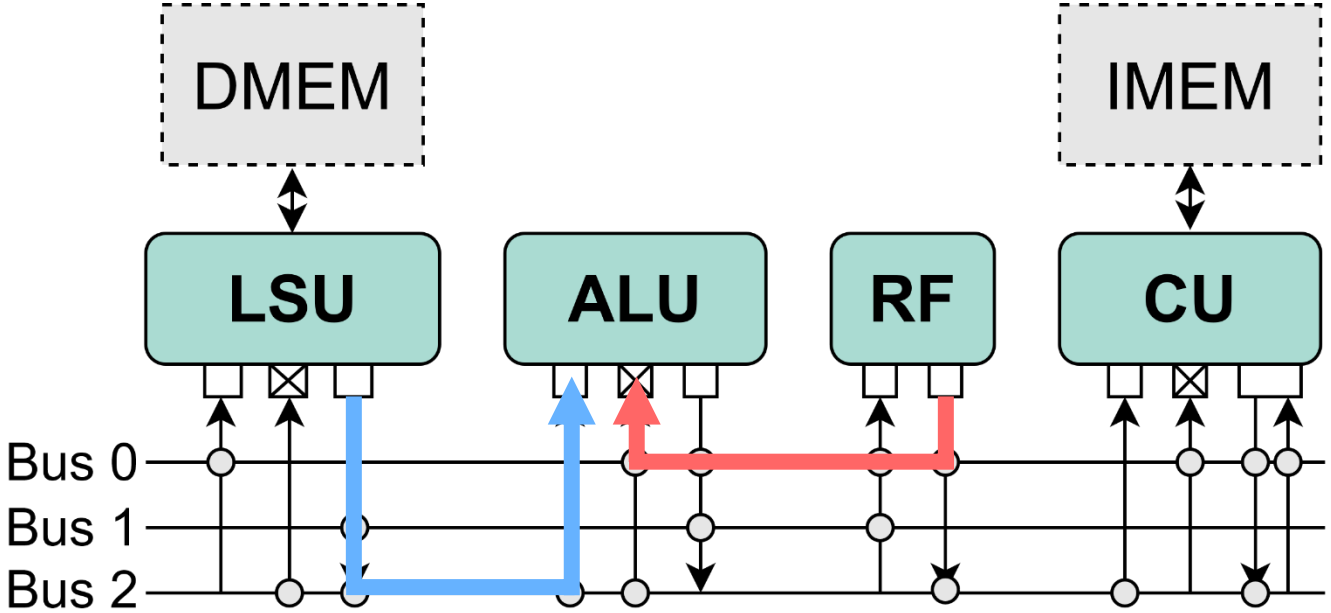


Source: [4]

- Deployment on edge devices → **Quantization**
- Operand width ↓
  - MAC HW **superlinearly ↓↓**
  - Overhead **(sub)linearly ↓**
- Efficient data reuse + minimized data movement

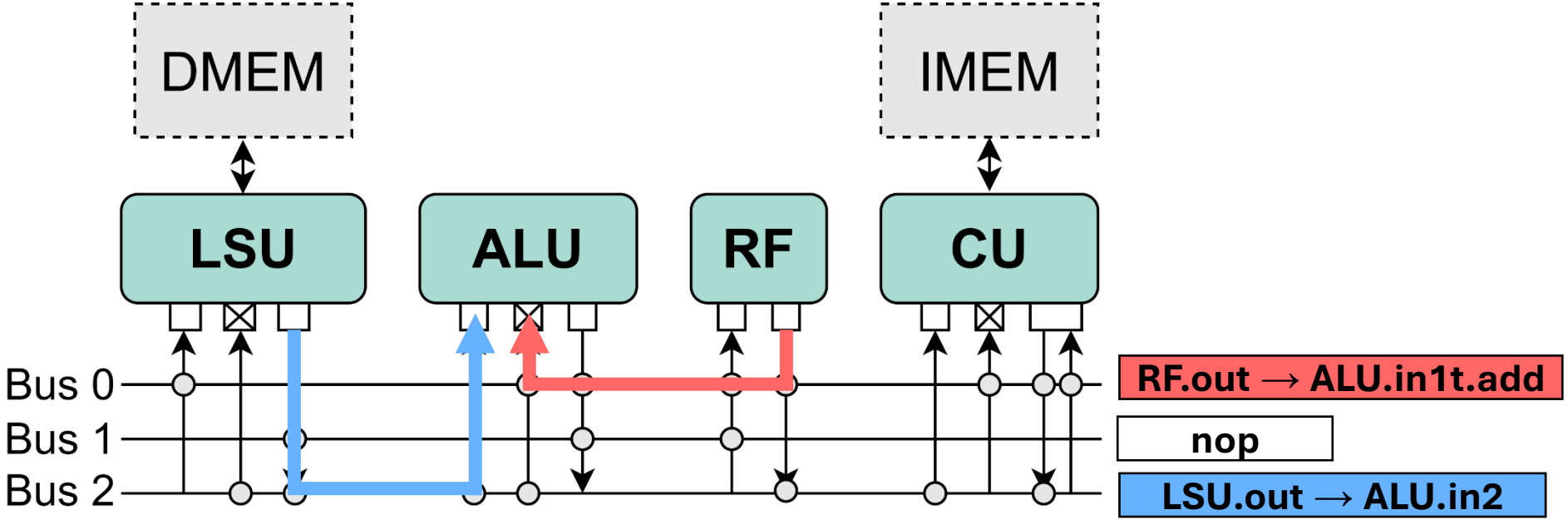
[3] https://www.researchgate.net/figure/Number-of-parameters-ie-weights-in-recent-landmark-neural-networks1-2-31-43_fig1_349044689
[4] P. C. Knag et al., "A 617-TOPS/W All-Digital Binary Neural Network Accelerator in 10-nm FinFET CMOS," IEEE J. Solid-State Circuits (JSSC), 2021.

Transport-Triggered Architecture



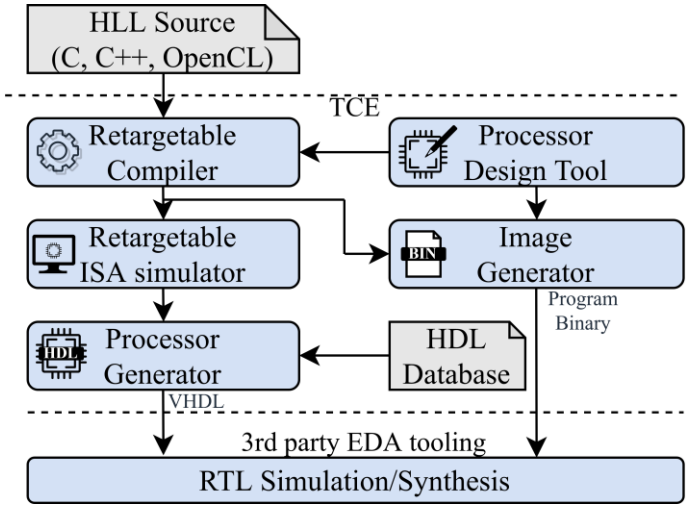
RF.out → ALU.in1t.add	nop	LSU.out → ALU.in2

Transport-Triggered Architecture



+ Compile-time configurable → flexible schedule
+ Exposed datapath → RF bypassing
+ Exposed datapath → Operand sharing

BrainTTA: Toolchain and System

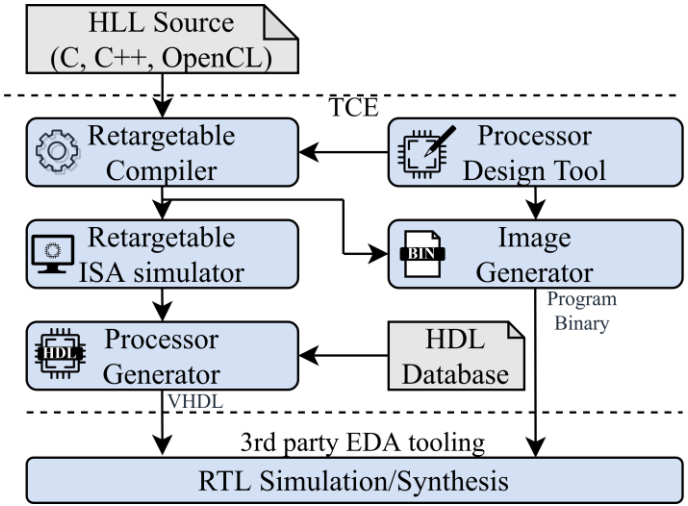


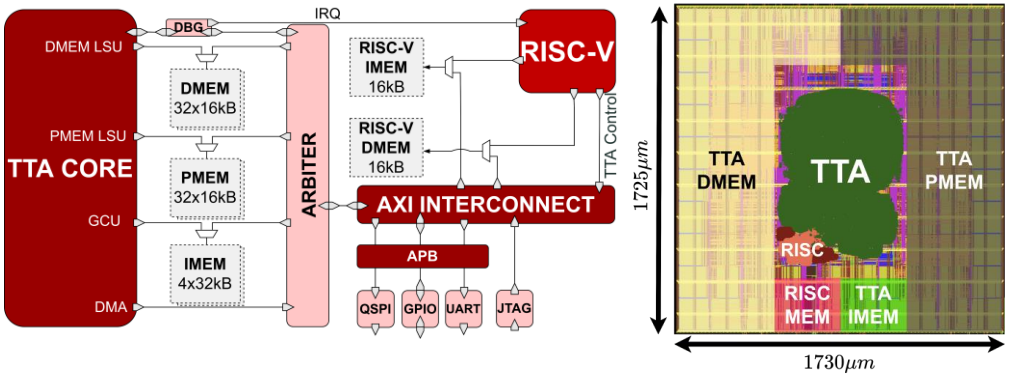
Design Flow

- **OpenASIP, retargetable** [1]
- LLVM-based compiler
- ISA simulator
- HDL Database → custom units

[1] http://openasip.org/cpc/publications.html

BrainTTA: Toolchain and System





Design Flow

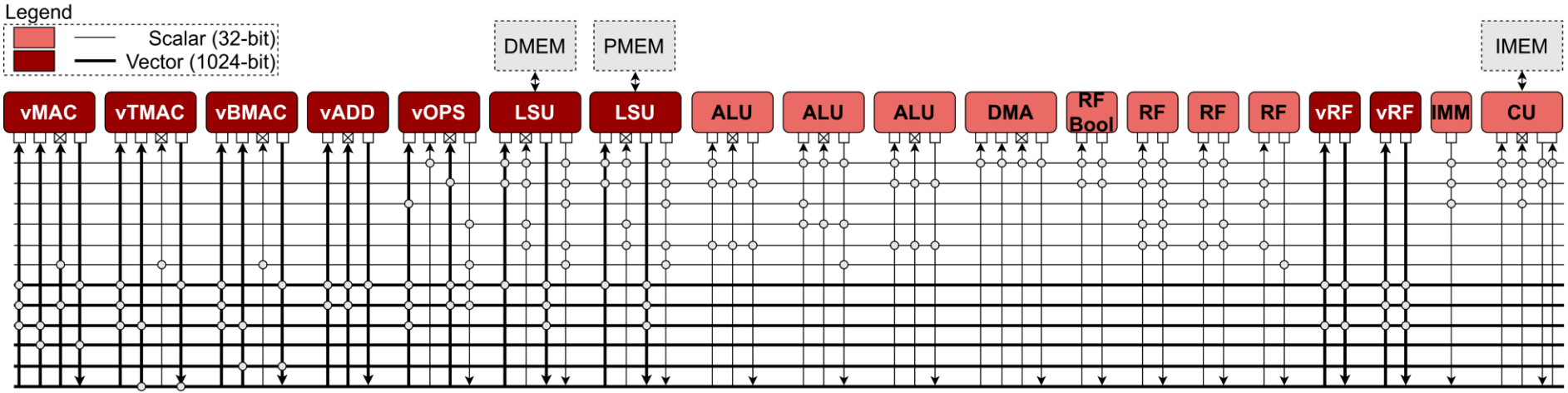
- **OpenASIP, retargetable** [1]
- LLVM-based compiler
- ISA simulator
- HDL Database → custom units

Architecture design:

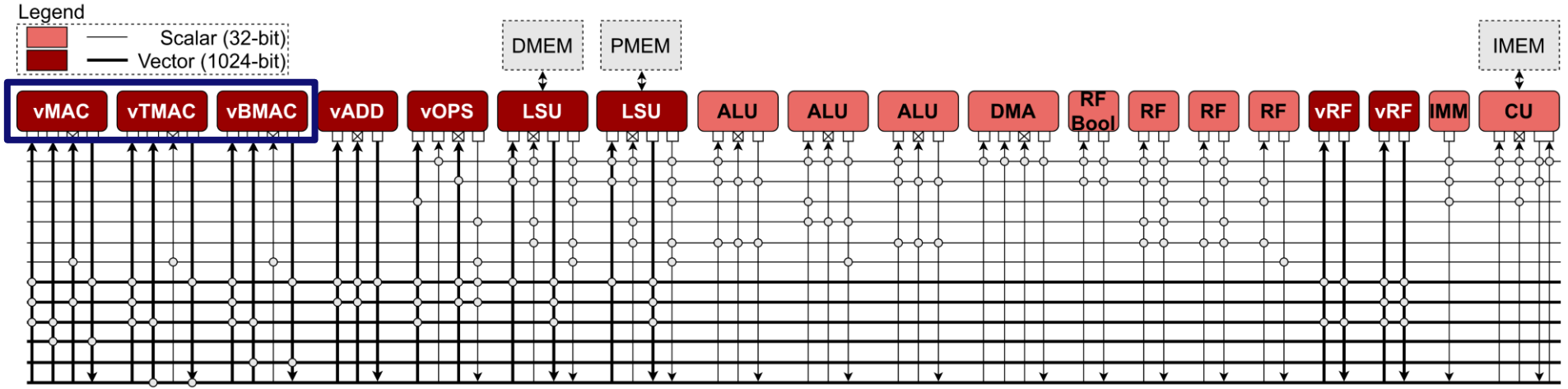
- Tech: GF 22nm FDX
- RISC-V + peripherals
- DMEM/PMEM split + banked access
- IMEM + HW loopbuffer

[1] http://openasip.org/cpc/publications.html

BrainTTA: TTA Core



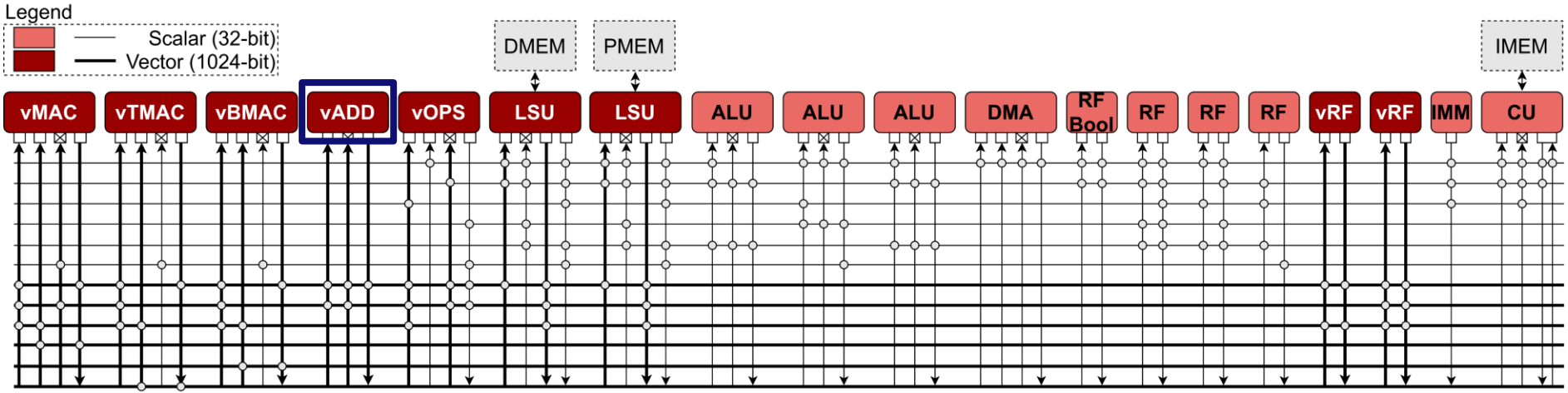
BrainTTA: TTA Core



vMAC

- 8-bit MAC
  - Scalar-Vector MAC
  - Vector-Vector MAC
- Binary MAC
- Ternary MAC

BrainTTA: TTA Core



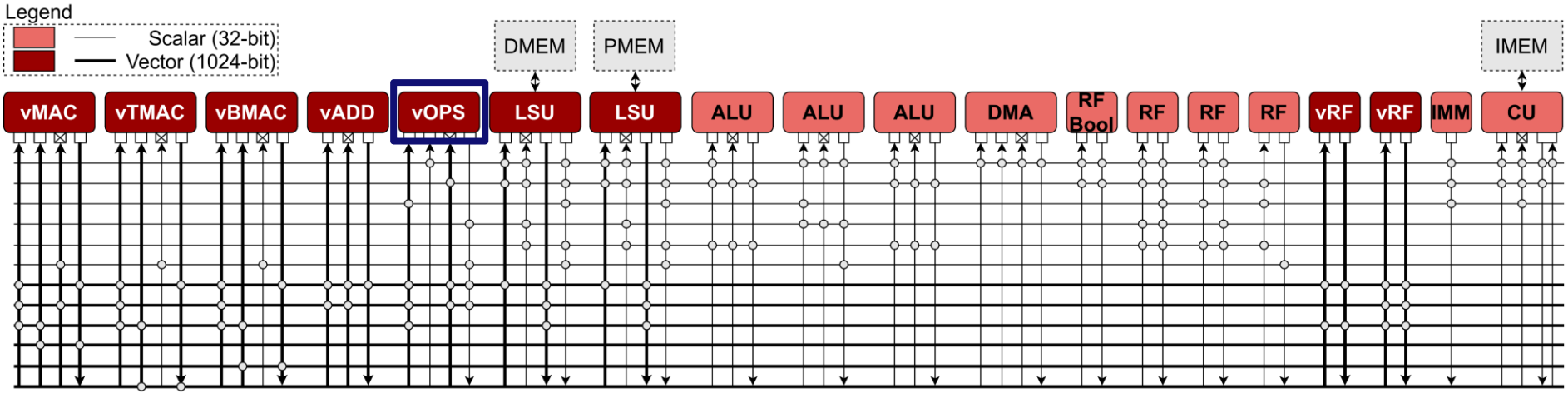
vMAC

- 8-bit MAC
  - Scalar-Vector MAC
  - Vector-Vector MAC
- Binary MAC
- Ternary MAC

vADD

- Vector-Vector addition
- Residual support

BrainTTA: TTA Core



vMAC

- 8-bit MAC
  - Scalar-Vector MAC
  - Vector-Vector MAC
- Binary MAC
- Ternary MAC

vADD

- Vector-Vector addition
- Residual support

vOPS

- Requantization
- Binarization
- Ternarization
- MaxPool
- Auxiliary ops

Application Mapping

Quantized NNs →

OS schedule

```
for h in [0, H − R + 1]:                    Output feature map height
  for w in [0, W − S + 1]:                  Output feature map width
    for m in [0, M]:                        Ouput channels
      accu = bias[m]
      for c in [0, C]:                      Input channels
        for r in [0, R]:                    Kernel height
          for s in [0, S]:                  Kernel width
            accu += in[h+r][w+s][c] * w[c][r][s][m]
      output[h][w][m] = act_function(accu)
```

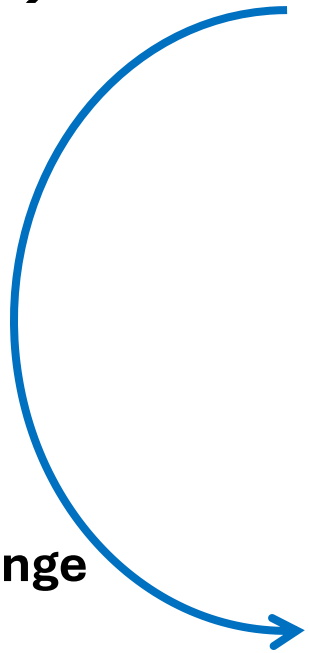
Application Mapping

Quantized NNs →

OS schedule

Loop tiling

Loop interchange



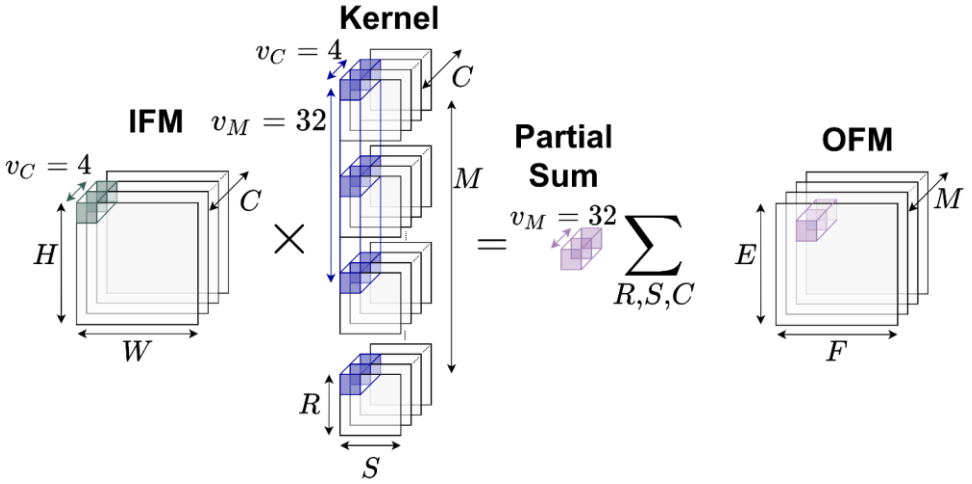
```
for h in [0, H − R + 1]:                    Output feature map height
  for w in [0, W − S + 1]:                  Output feature map width
    for m in [0, M]:                        Ouput channels
      accu = bias[m]
      for c in [0, C]:                      Input channels
        for r in [0, R]:                    Kernel height
          for s in [0, S]:                  Kernel width
            accu += in[h+r][w+s][c] * w[c][r][s][m]
      output[h][w][m] = act_function(accu)
```

```
for h in [0, H − R + 1]:                    Output feature map height
  for w in [0, W − S + 1]:                  Output feature map width
    for m in [0, M/32]:                     Ouput channels (v_M = 32)
      accu = bias[32*m]
      for c in [0, C/4]:                    Input channels (v_C = 4)
        for r in [0, R]:                    Kernel height
          for s in [0, S]:                  Kernel width
            for tm in [0, 31]:
              for tc in [0, 3]:
                accu += in[h+r][w+s][4*c+tc]
                  * w[4*c+tc][r][s][32*m+tm]
      output[h][w][m] = act_function(accu)
```

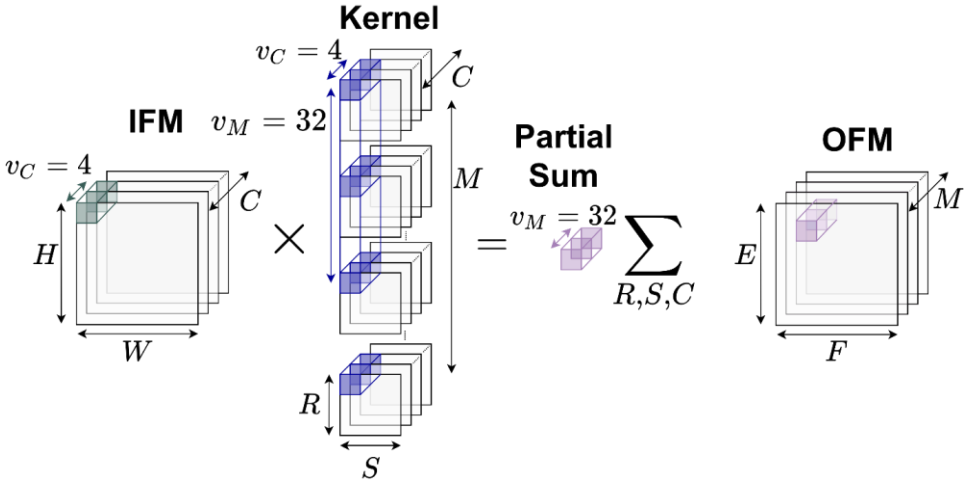
Application Mapping

```
for h in [0, H - R + 1]:                    Output feature map height
  for w in [0, W - S + 1]:                  Output feature map width
    for m in [0, M/32]:                     Ouput channels (v_M = 32)
      accu = bias[32*m]
      for c in [0, C/4]:                    Input channels (v_C = 4)
        for r in [0, R]:                    Kernel height
          for s in [0, S]:                  Kernel width
            for tm in [0, 31]:
              for tc in [0, 3]:
                accu += in[h+r][w+s][4*c+tc]
                  * w[4*c+tc][r][s][32*m+tm]
      output[h][w][m] = act_function(accu)
```

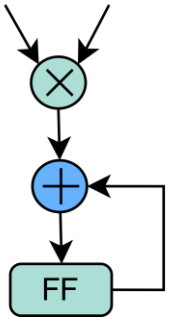
Application Mapping



Application Mapping



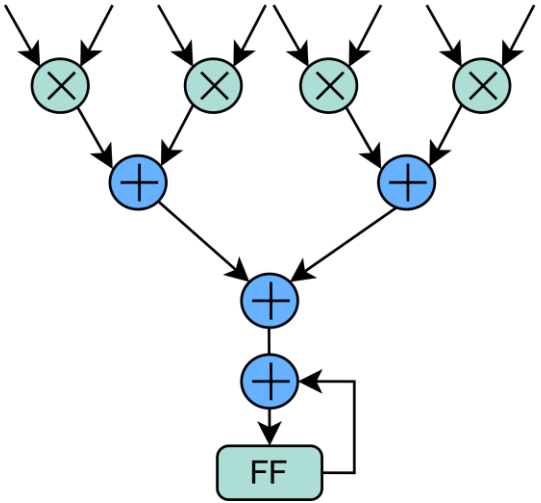
$$v_C = 1$$
$$v_M = 1$$

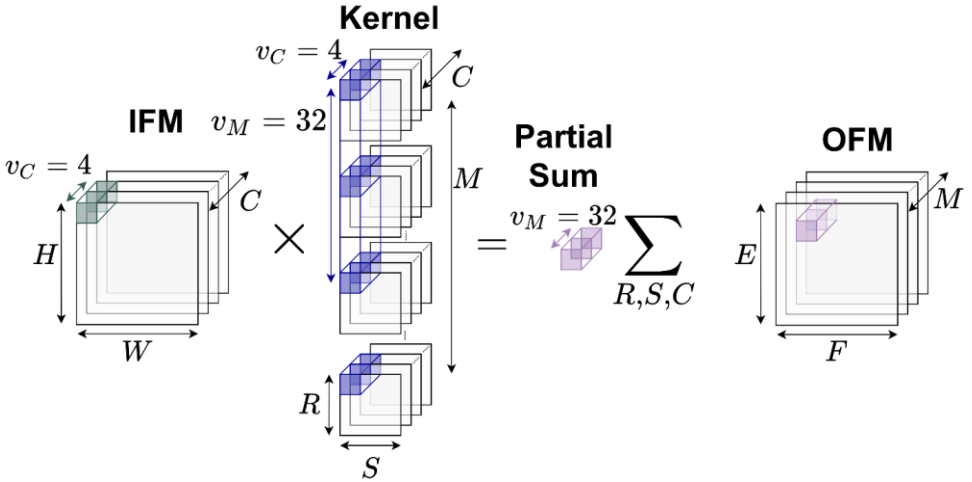


ACCU cost >> MUL cost →
wider reduction tree

Application Mapping

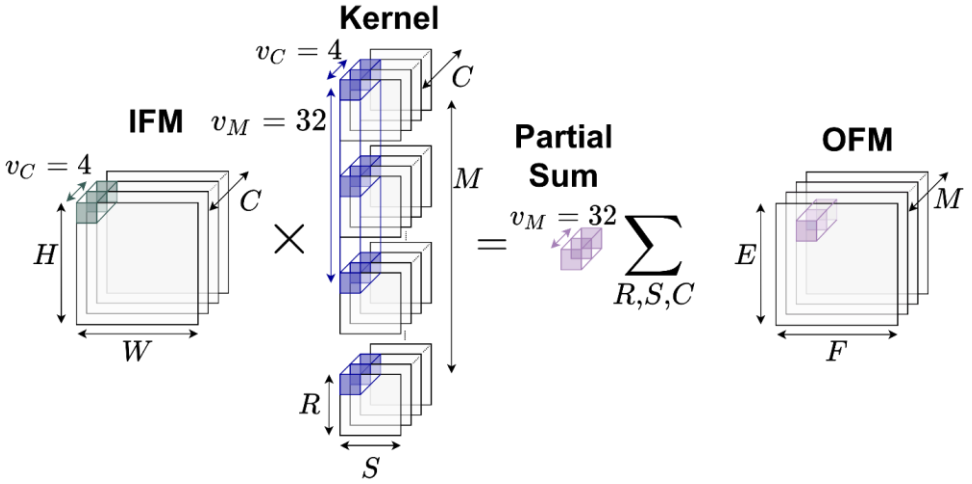
$$v_C = 4$$
$$v_M = 1$$



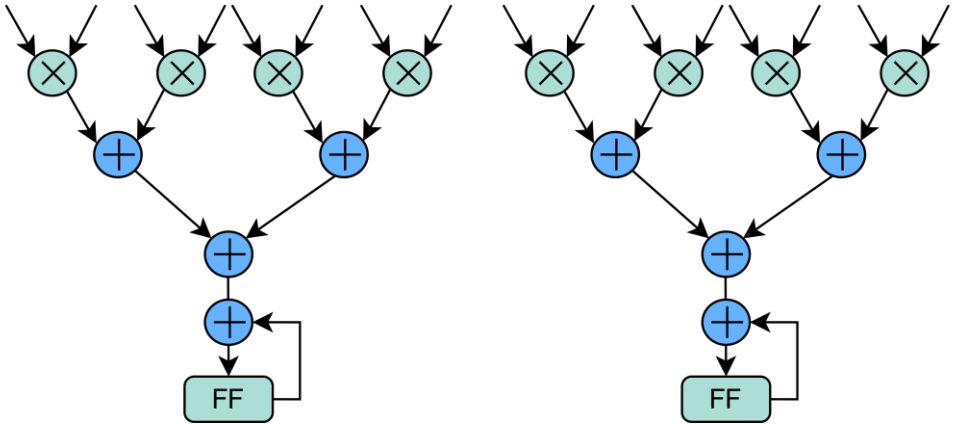


ACCU cost >> MUL cost →
wider reduction tree

Application Mapping



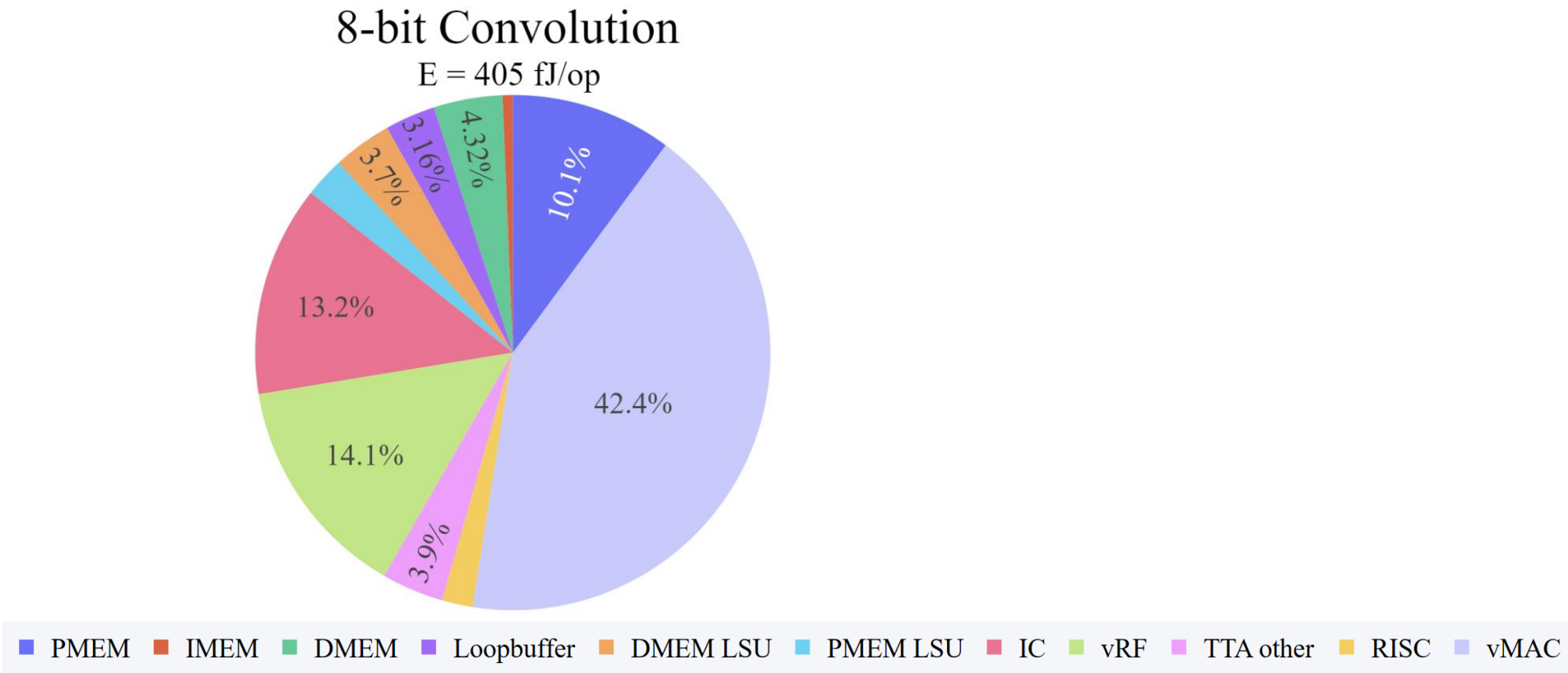
$$v_C = 4$$
$$v_M = 2$$



ACCU cost >> MUL cost → **wider reduction tree**

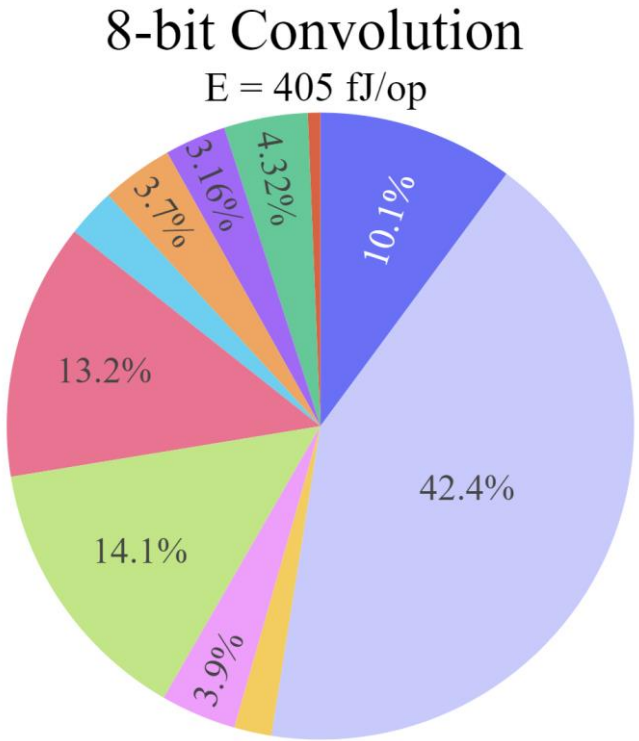
IFM broadcast → **multiple reduction trees**

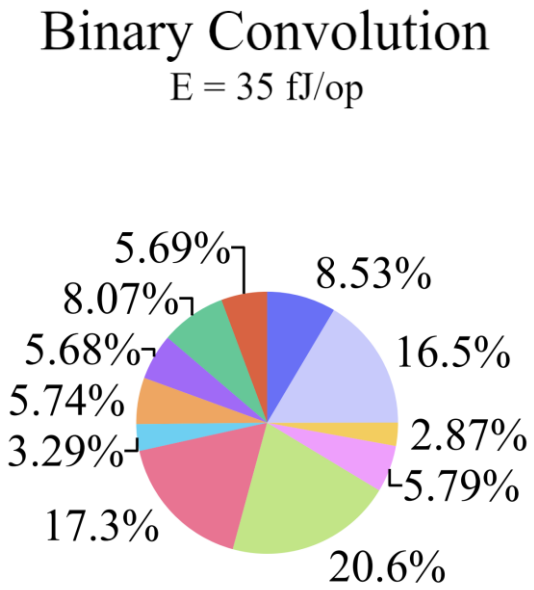
Post-layout Energy Consumption [GF 22nm FDX]



Operating conditions:TT corner,, T=25 °C, Vdd=0.5V
Conv params:W=H=16, M=C=128, R=S=3

Post-layout Energy Consumption [GF 22nm FDX]





PMEM IMEM DMEM Loopbuffer DMEM LSU PMEM LSU IC vRF TTA other RISC vMAC

Operating conditions:TT corner,, T=25 °C, Vdd=0.5V
Conv params:W=H=16, M=C=128, R=S=3

Comparison to SotA

	Eyeriss v2	XNE	SamurAI	XPULPNN	Dustin	This work
Tech	65nm	22nm	28nm	22nm	65nm	22nm
Progamm ability	Configurable	ASM	ASM	**Compiler**	**Compiler**	**Compiler**
Energy efficiency	252 GOPS/W(8b)[1]	8.7 TOPS/W (1b)	1.3 TOPS/W (8b)	2.2 TOPS/W (8b) **6.1 TOPS/W (2b)**	606 GOPS/W (8b) **2304 GOPS/W (2b)**	**2.5 TOPS/W (8b)** **14.9 TOPS/W (T)** **28.6 TOPS/W (1b)**
Memory Cap. [kB]	246	520	464	640	80	1024[2]
Area Eff. [GOPS/mm2]	5.5 (8b)[3]	28.9 (1b)	0.6 (8b)	21.7 (8b) 70.7 (2b)	0.9 (8b) 3.46 (2b)	25.8 (8b) 103.0 (T) 206.0 (1b)

*after technology scaling

[1]Average on AlexNet.
[2]Excluding instruction memory.
[3]Area estimated using gatecount diff. EyerissV1, EyerissV2.

Comparison to SotA

	Eyeriss v2	XNE	SamurAI	XPULPNN	in	This work
Tech	65nm	22nm	28nm	22nm	65nm	22nm
Progamm ability	Configurable	ASM	ASM	**Compiler**		**Compiler**
Energy efficiency	252 GOPS/W(8b)[1]	8.7 TOPS/W (1b)	1.3 TOPS/W (8b)	2.2 TOPS/W (8b) **6.1 TOPS/W (2b**	606 GOPS/W (8b) **2304 GOPS/W (2b**	**2.5 TOPS/W (8b)** **14.9 TOPS/W (T)** **28.6 TOPS/W (1b)**
Memory Cap. [kB]	246	520	464	640	80	1024[2]
Area Eff. [GOPS/mm2]	5.5 (8b)[3]	28.9 (1b)	0.6 (8b)	21.7 (8b) 70.7 (2b)	0.9 (8b) 3.46 (2b)	25.8 (8b) 103.0 (T) 206.0 (1b)

+14%

+39%*

X 2.7

X 3.8

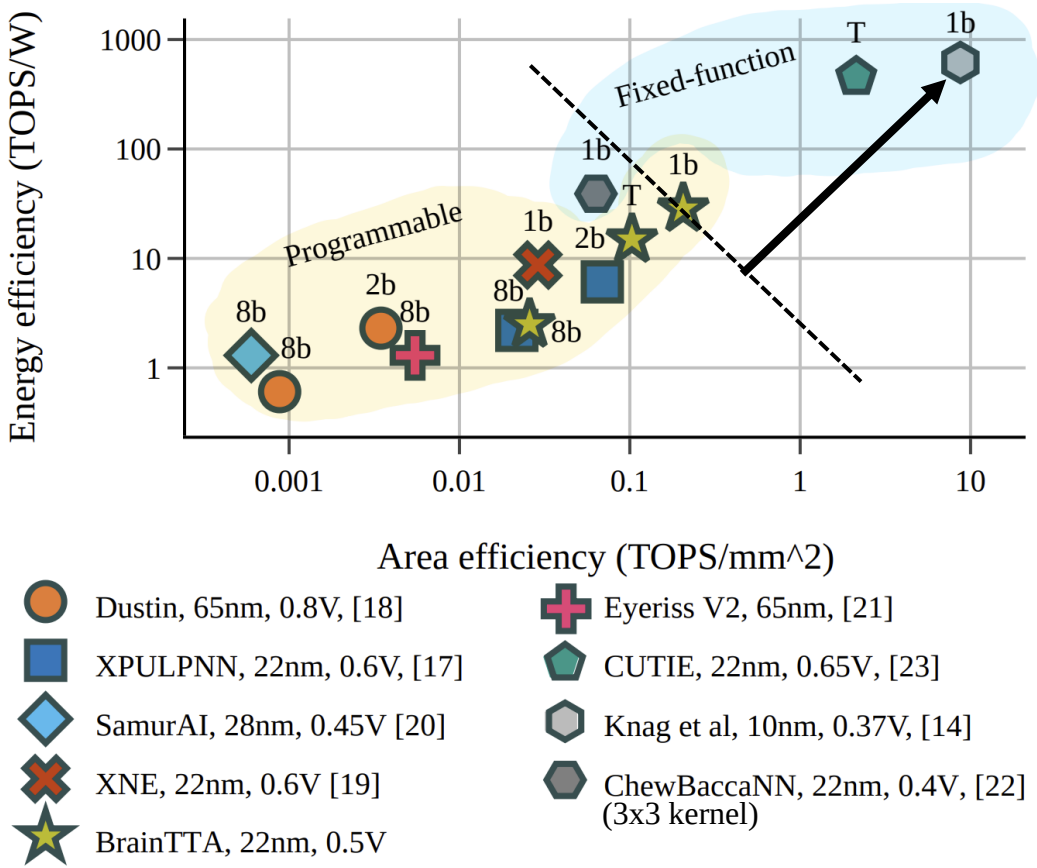
X 11.4

*after technology scaling

[1]Average on AlexNet.
[2]Excluding instruction memory.
[3]Area estimated using gatecount diff. EyerissV1, EyerissV2.

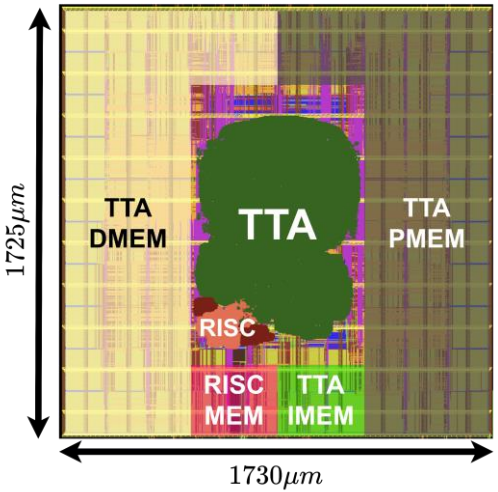
Programmable vs. Fixed-Function Trade-off

- Programmable architectures
- Fixed-function
  - Spatial loop unrolling
  - Fixed FM/weight bufferring



BrainTTA - Conclusion

- Efficient and flexible NN inference engine:
  - Mixed-precision
  - Compile-time reconfigurable
  - Eff: **2.47** / **14.9** / **28.6** [TOPS/W]
  - Throughput: **77** / **307** / **614** [GOPS]
- Superlinear energy eff. scaling
  - 8-bit → ternary: **x5.96**
  - 8-bit → binary: **x11.4**



Q&A