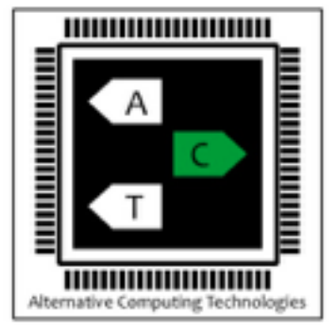
SNNAP:
Approximate Computing on Programmable SoCs via Neural Acceleration

Thierry Moreau
Mark Wyse
Jacob Nelson
Adrian Sampson

Hadi Esmaeilzadeh
Luis Ceze
Mark Oskin



Approximate Computing

Expose quality-performance trade-offs

Approximate Computing

Expose quality-performance trade-offs

✅ Accurate

❌ Expensive

❌ Approximate

✅ Cheap

Approximate Computing

Expose quality-performance trade-offs

✅ Accurate
❌ Expensive

✅ Approximate
✅ Cheap

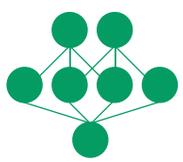
Domains include image processing, machine learning, search, physical simulation, multimedia etc.

Neural Acceleration

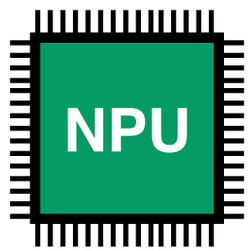
Neural Acceleration

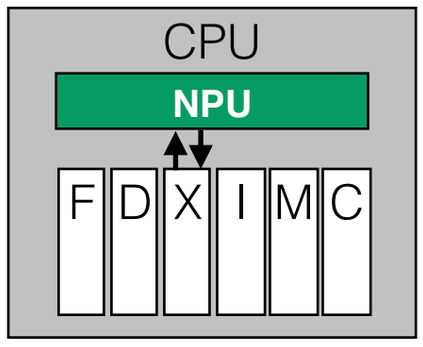
```
float foo (float a, float b)
{
  …
  return val;
}
```

approximation

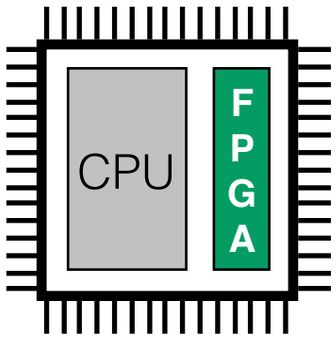


acceleration





Esmaeilzadeh et al. [MICRO 2012]



SNNAP

SNNAP

A neural processing unit on off-the-shelf Programmable SoCs

3.8x speedup and 2.8x efficiency gains

offers an alternative to HLS tools for neural acceleration

Talk Outline

Introduction

Programming model

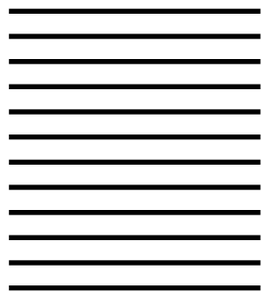
SNNAP design:

- Efficient neural network evaluation
- Low-latency communication

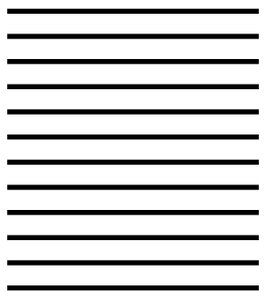
Evaluation & Comparison with HLS

Background: Compilation

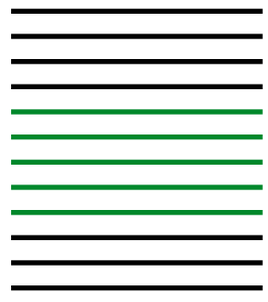
1. Region detection



code annotation

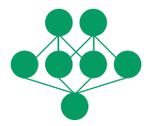


region detection & program instrumentation

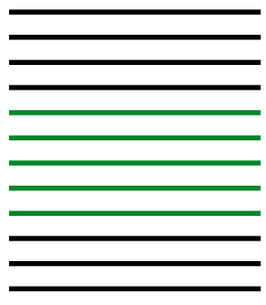


2. ANN Training

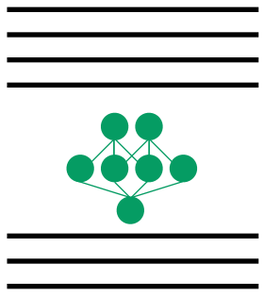
`[training.data]` → back prop. & topology search →

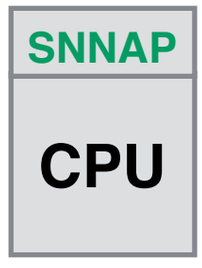


3. Code Generation



binary generation





Programming Model

sobel

```
float sobel (float* p);
. . .

Image src;
Image dst;

while (true) {
    src = read_from_camera();
    for (y=0; y < h; ++y) {
        for (x=0; x < w; ++x) {
            dst.p[y][x] = sobel(& src.p[y][x]);
        }
    }
    display(dst);
}
```

Programming Model

sobel ↓

```
APPROX float sobel (APPROX float* p);
. . .

APPROX Image src;
APPROX Image dst;

while (true) {
    src = read_from_camera();
    for (y=0; y < h; ++y) {
        for (x=0; x < w; ++x) {
            dst.p[y][x] = sobel(& src.p[y][x]);
        }
    }
    display(dst);
}
```

✅ no side effects

✅ executes often

ACCEPT: compilation framework for approximate programs

Talk Outline

Introduction

Programming model

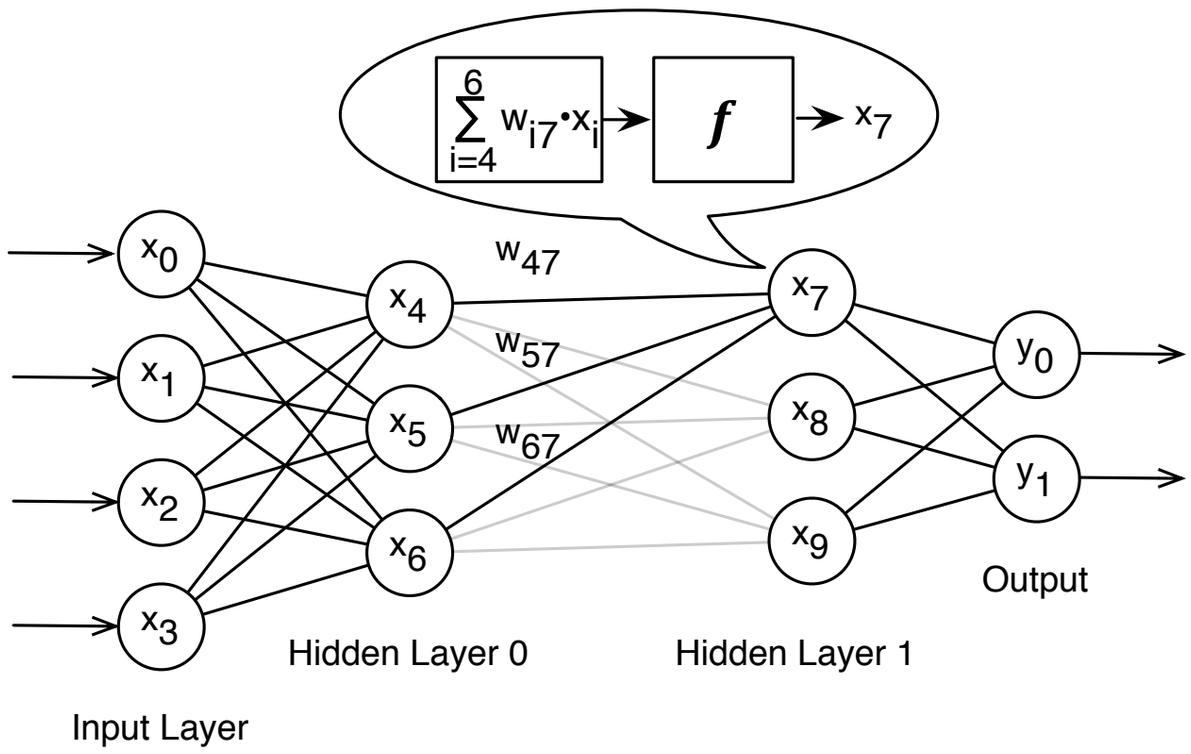
SNNAP design:

- **Efficient neural network evaluation**
- Low-latency communication

Evaluation & Comparison with HLS

Background: Multi-Layer Perceptrons

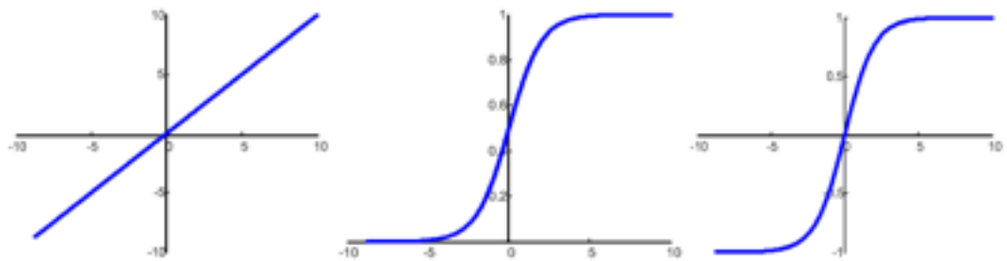
neural network



computing a single layer

$$\begin{bmatrix} x_7 \\ x_8 \\ x_9 \end{bmatrix} = f\left(\begin{bmatrix} w_{67} & w_{57} & w_{47} \\ w_{68} & w_{58} & w_{48} \\ w_{69} & w_{59} & w_{49} \end{bmatrix} \begin{bmatrix} x_6 \\ x_5 \\ x_4 \end{bmatrix} \right)$$

activation function f

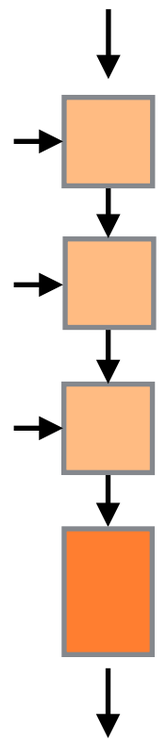


Background: Systolic Arrays

computing a single layer

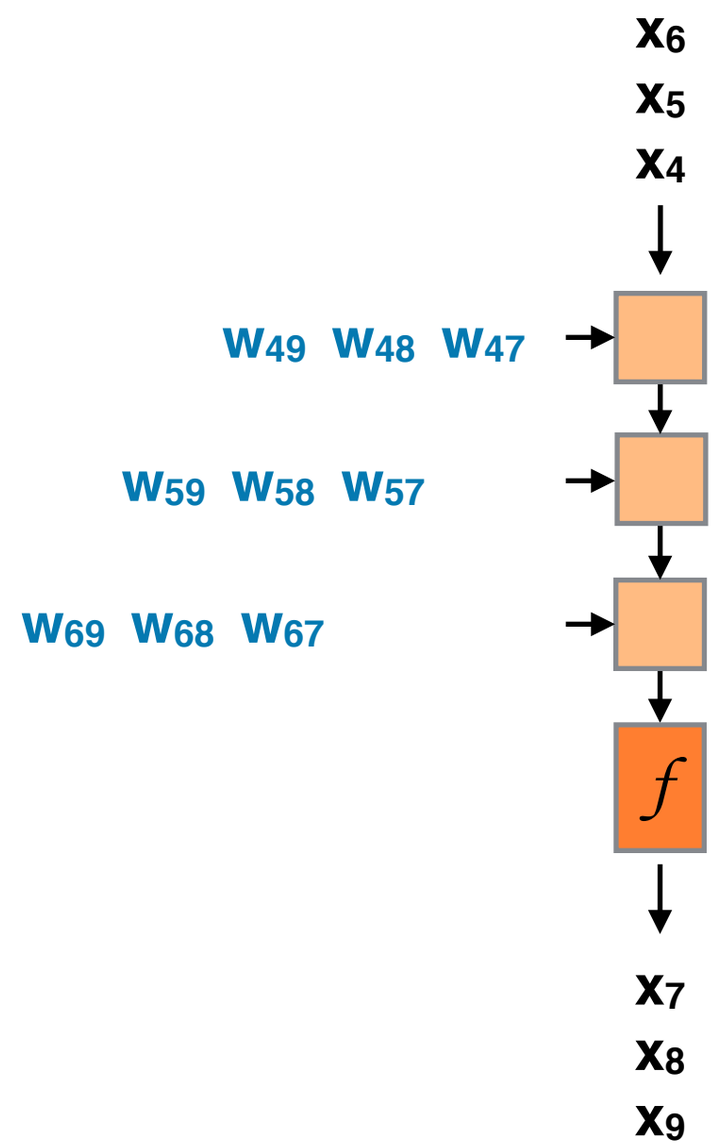
$$\begin{bmatrix} x_7 \\ x_8 \\ x_9 \end{bmatrix} = f\left(\begin{bmatrix} w_{67} & w_{57} & w_{47} \\ w_{68} & w_{58} & w_{48} \\ w_{69} & w_{59} & w_{49} \end{bmatrix} \begin{bmatrix} x_6 \\ x_5 \\ x_4 \end{bmatrix} \right)$$

systolic array



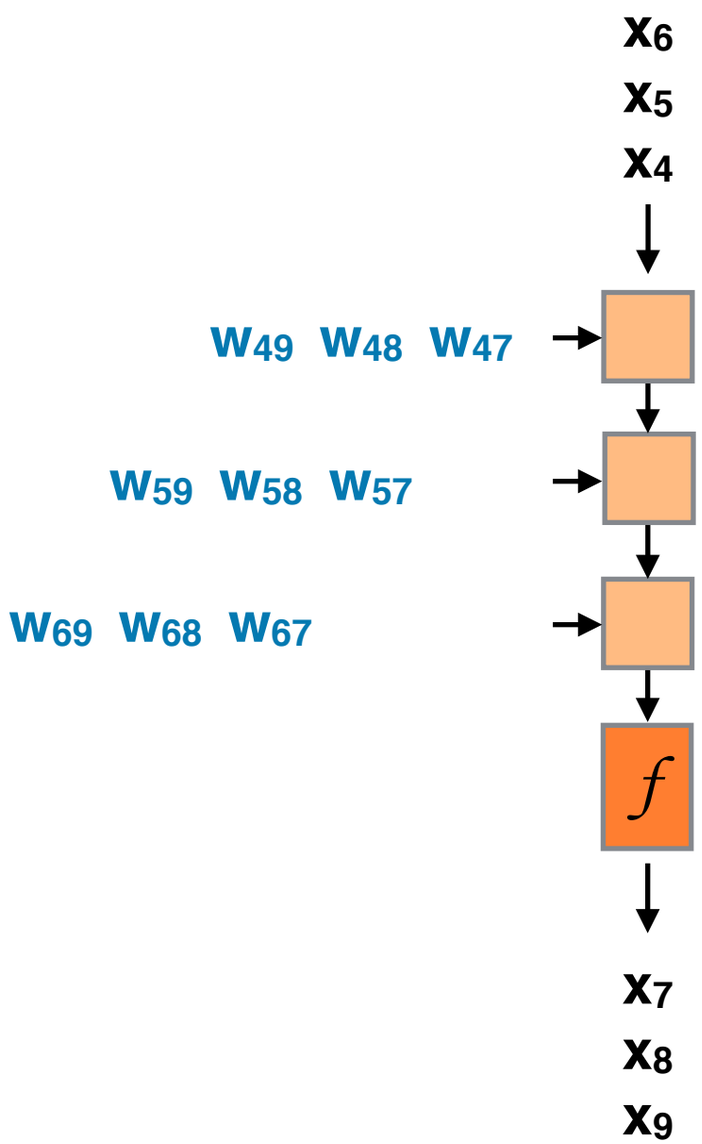
Background: Systolic Arrays

systolic array

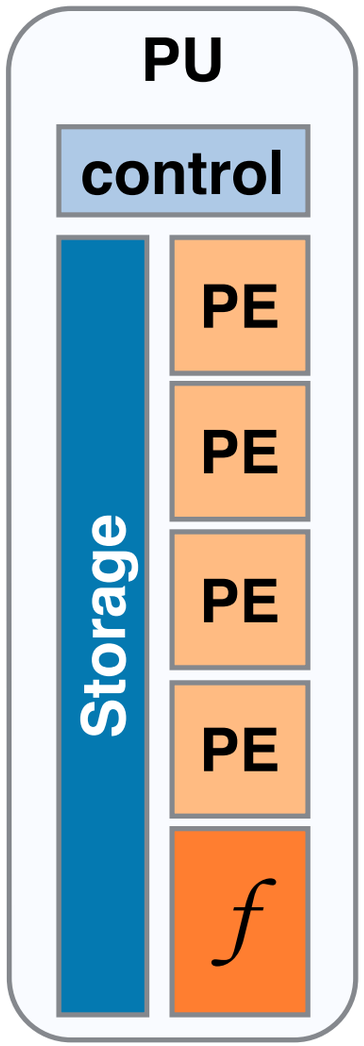


PU Micro-Architecture

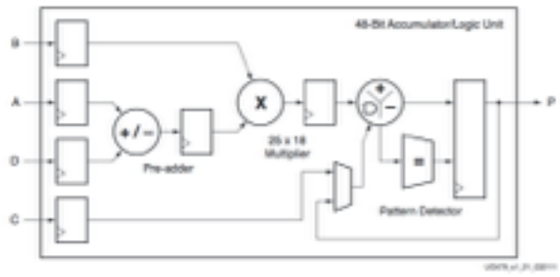
systolic array



processing unit

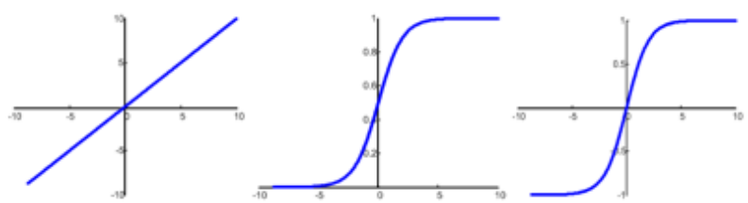


1 - processing elements in DSP logic



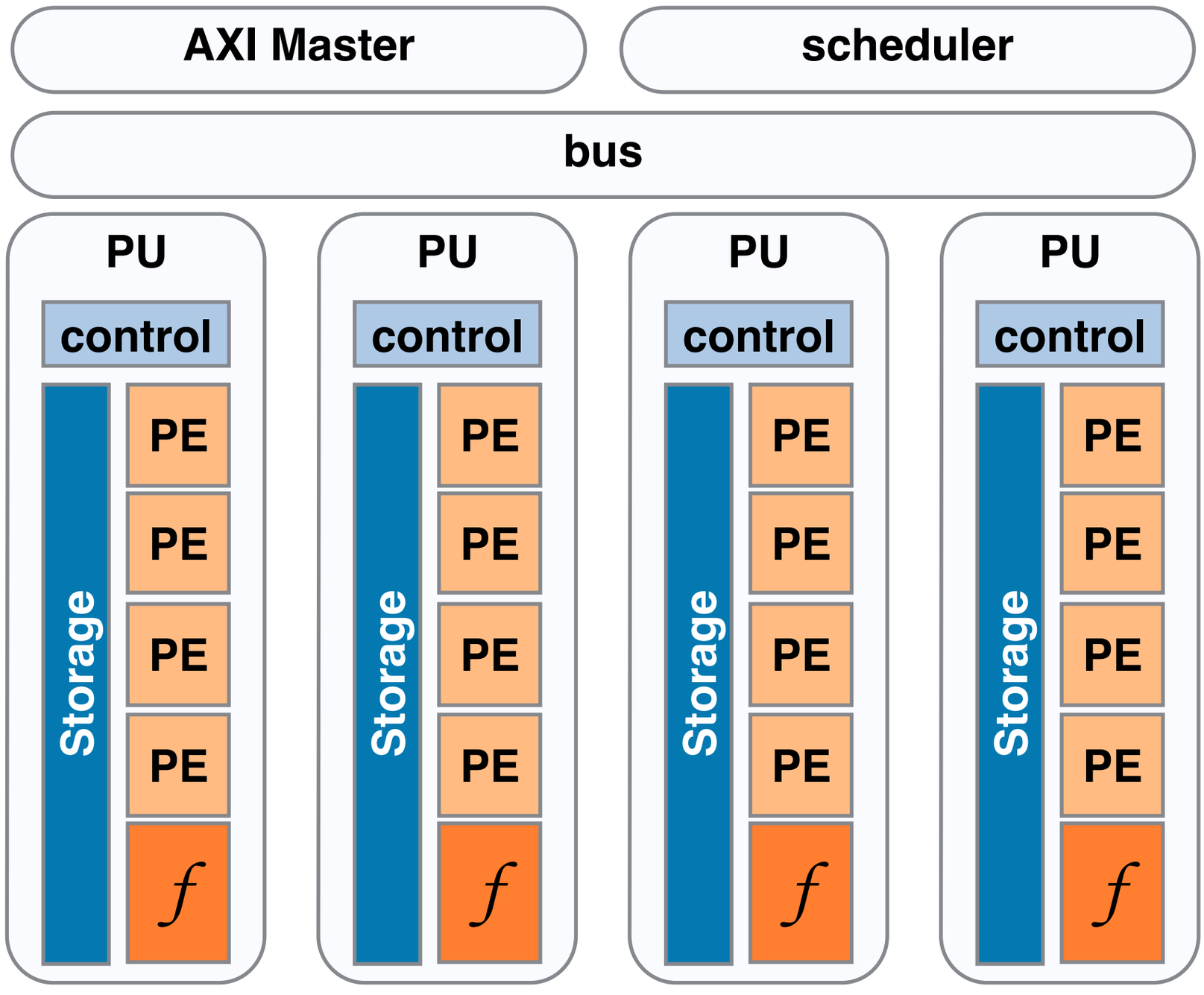
2 - local storage for synaptic weights

3 - sigmoid unit implements non-linear activation functions



4 - vertically micro-coded sequencer

Multi-Processing Units



Talk Outline

Introduction

Programming model

SNNAP design:

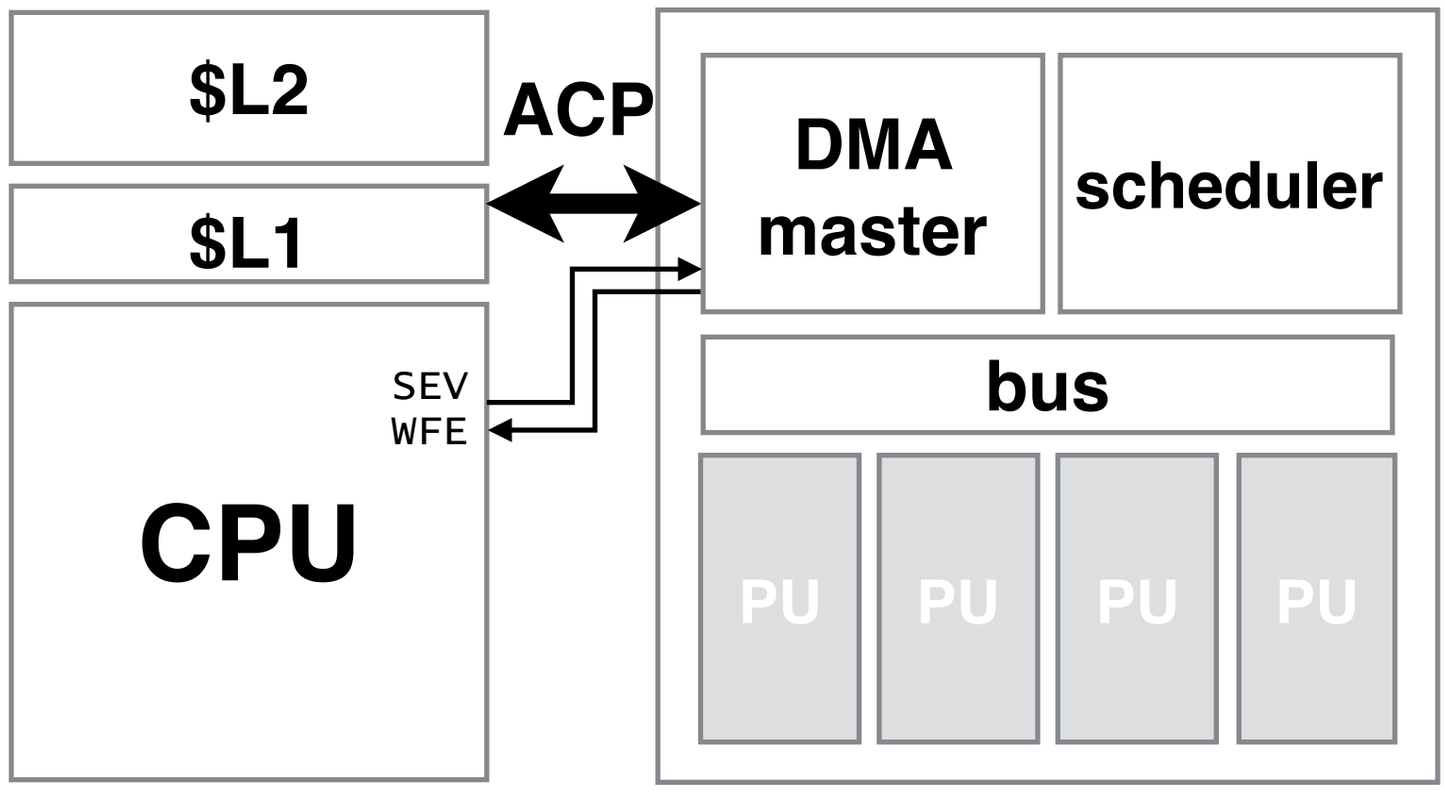
- Efficient neural network evaluation
- **Low-latency communication**

Evaluation & Comparison with HLS

CPU-SNNAP Integration

Interface requirements:

- Low-latency data transfer
- Fast signaling

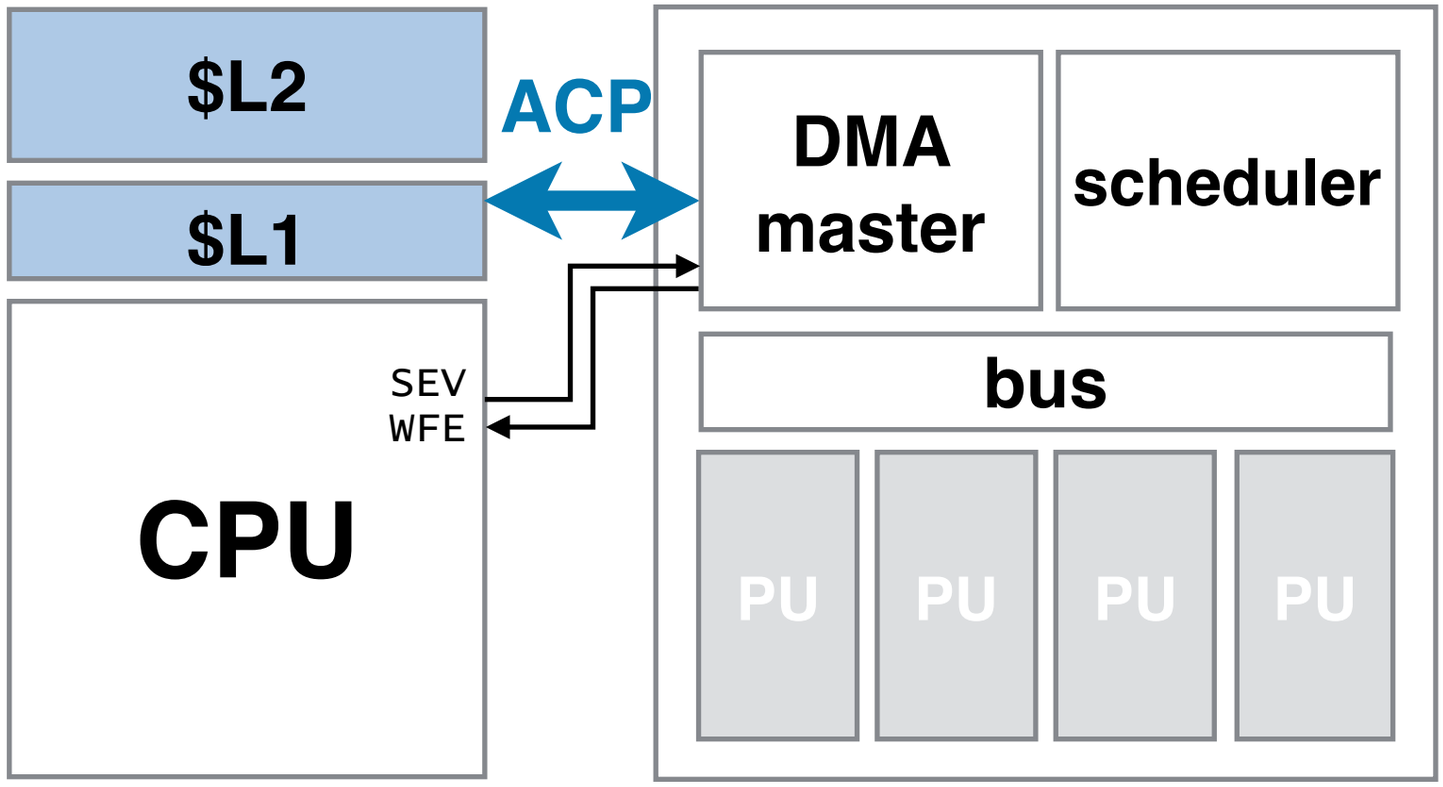


CPU-SNNAP Integration

Interface requirements:

- Low-latency data transfer
- Fast signaling

coherent reads
& writes
with accelerator
coherency port



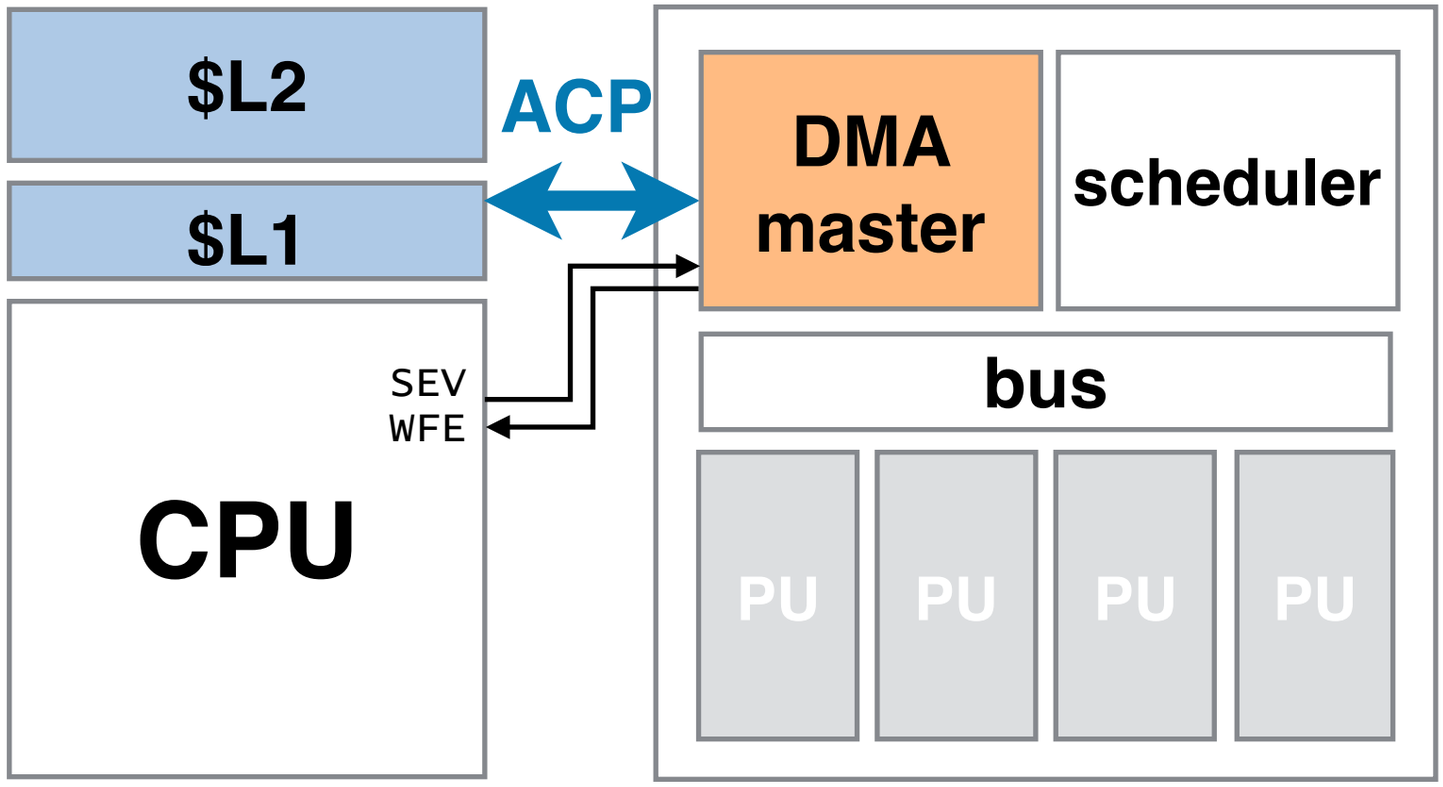
CPU-SNNAP Integration

Interface requirements:

- Low-latency data transfer
- Fast signaling

coherent reads & writes with accelerator coherency port

custom mastering interface



CPU-SNNAP Integration

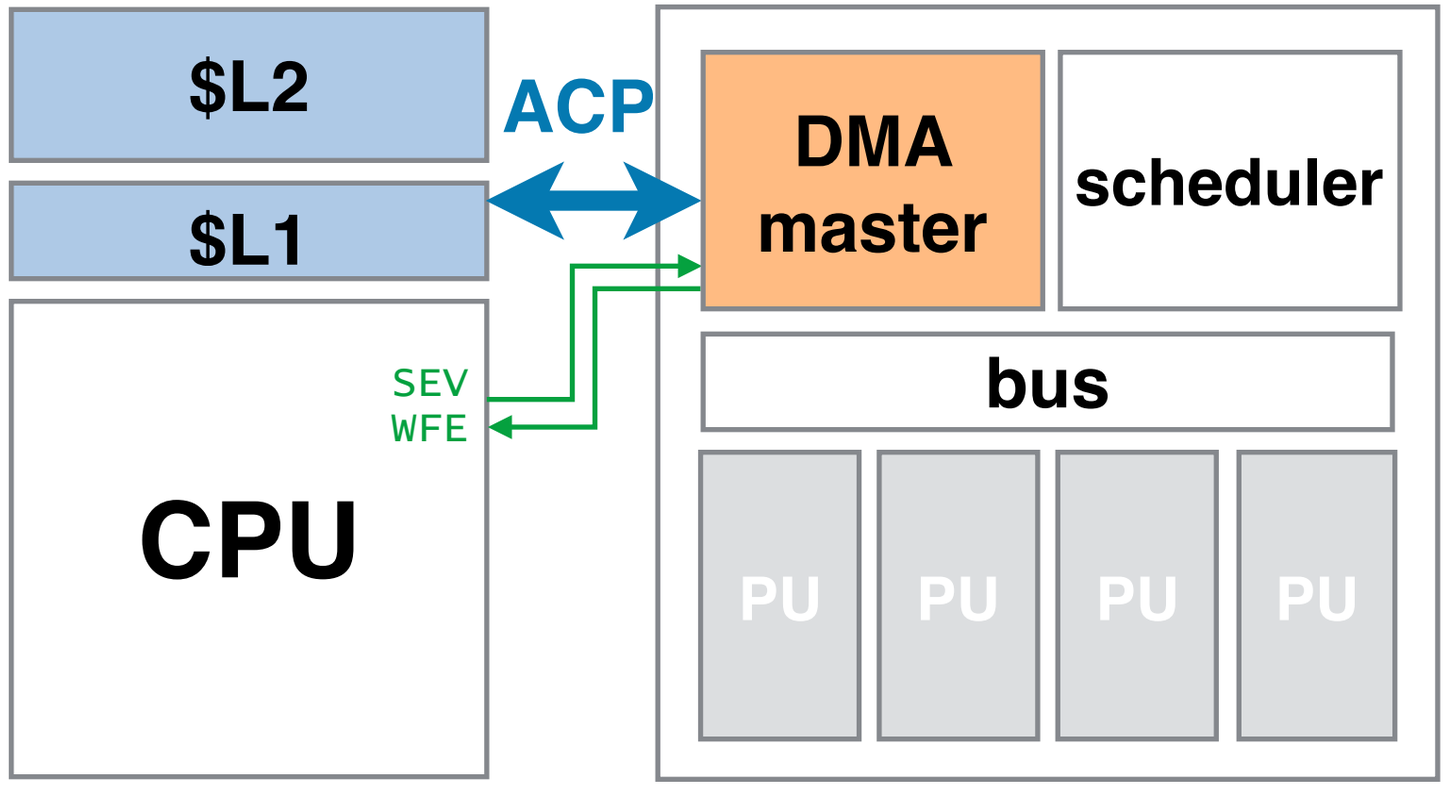
Interface requirements:

- Low-latency data transfer
- Fast signaling

coherent reads & writes with accelerator coherency port

custom mastering interface

low-latency event signaling, sleep & wakeup



Talk Outline

Introduction

Programming model

SNNAP design:

- Efficient neural network evaluation
- Low-latency communication

Evaluation & Comparison with HLS

Evaluation

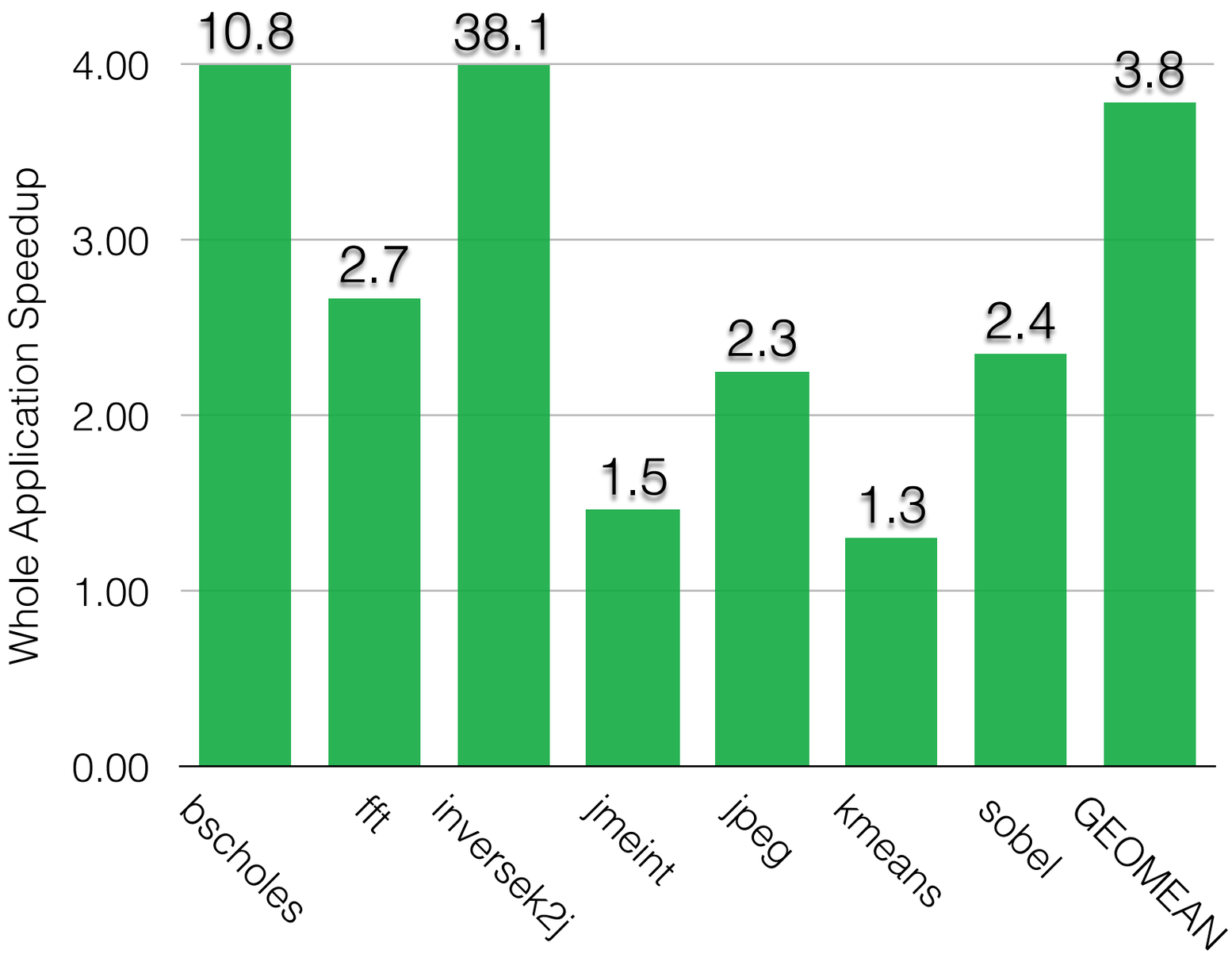
Neural acceleration on SNNAP (8x8 configuration, clocked at 1/4 of f_{CPU}) vs. precise CPU execution

Evaluation

Neural acceleration on SNNAP (8x8 configuration, clocked at 1/4 of f_{CPU}) vs. precise CPU execution

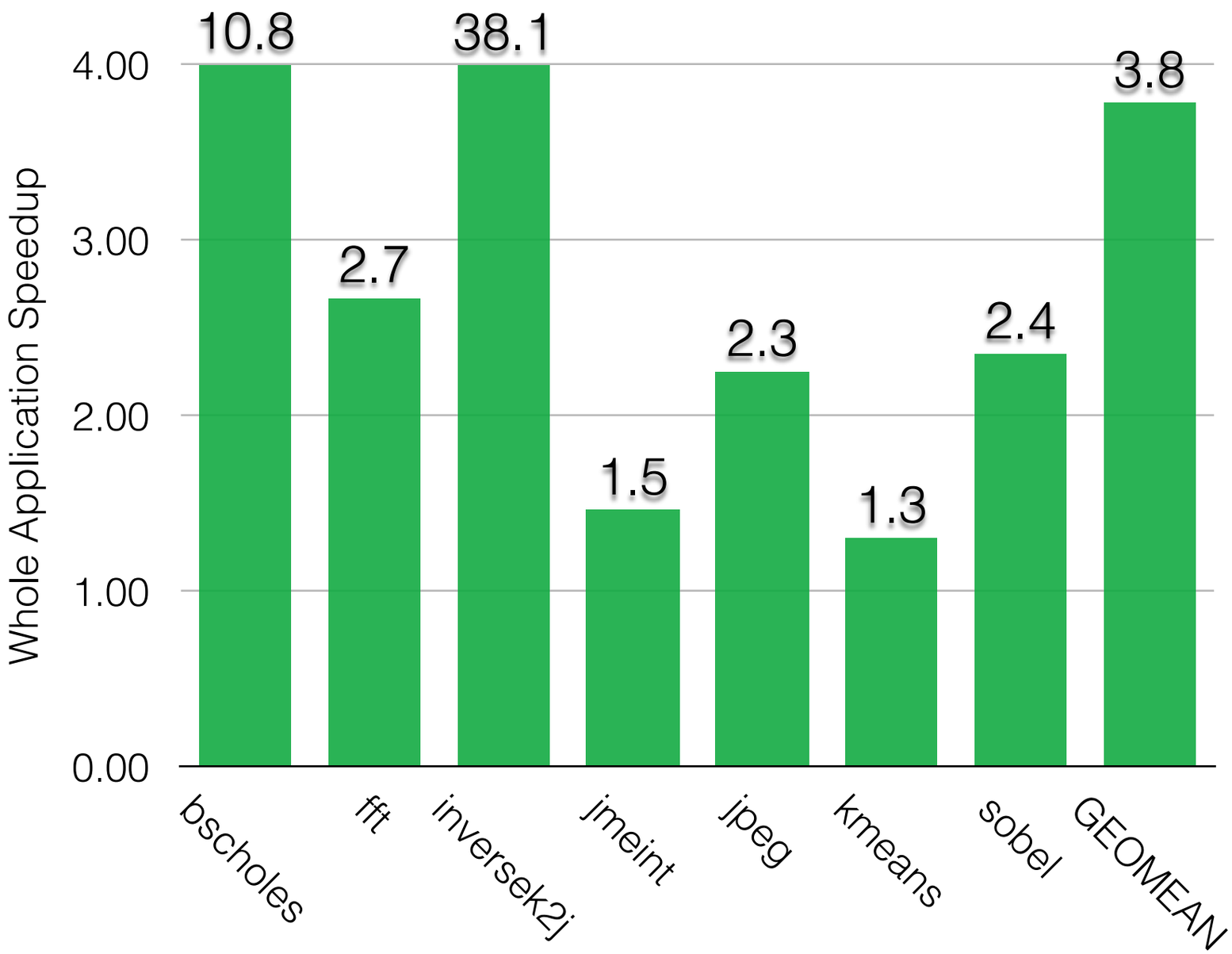
application	domain	error metric
blackscholes	option pricing	MSE
fft	DSP	MSE
inversek2j	robotics	MSE
jmeint	3D-modeling	miss rate
jpeg	compression	image diff
kmeans	ML	image diff
sobel	vision	image diff

Speedup



Factors:
- Amdahl's Speedup
- Cost of instructions on CPU vs. cost of NN on SNNAP

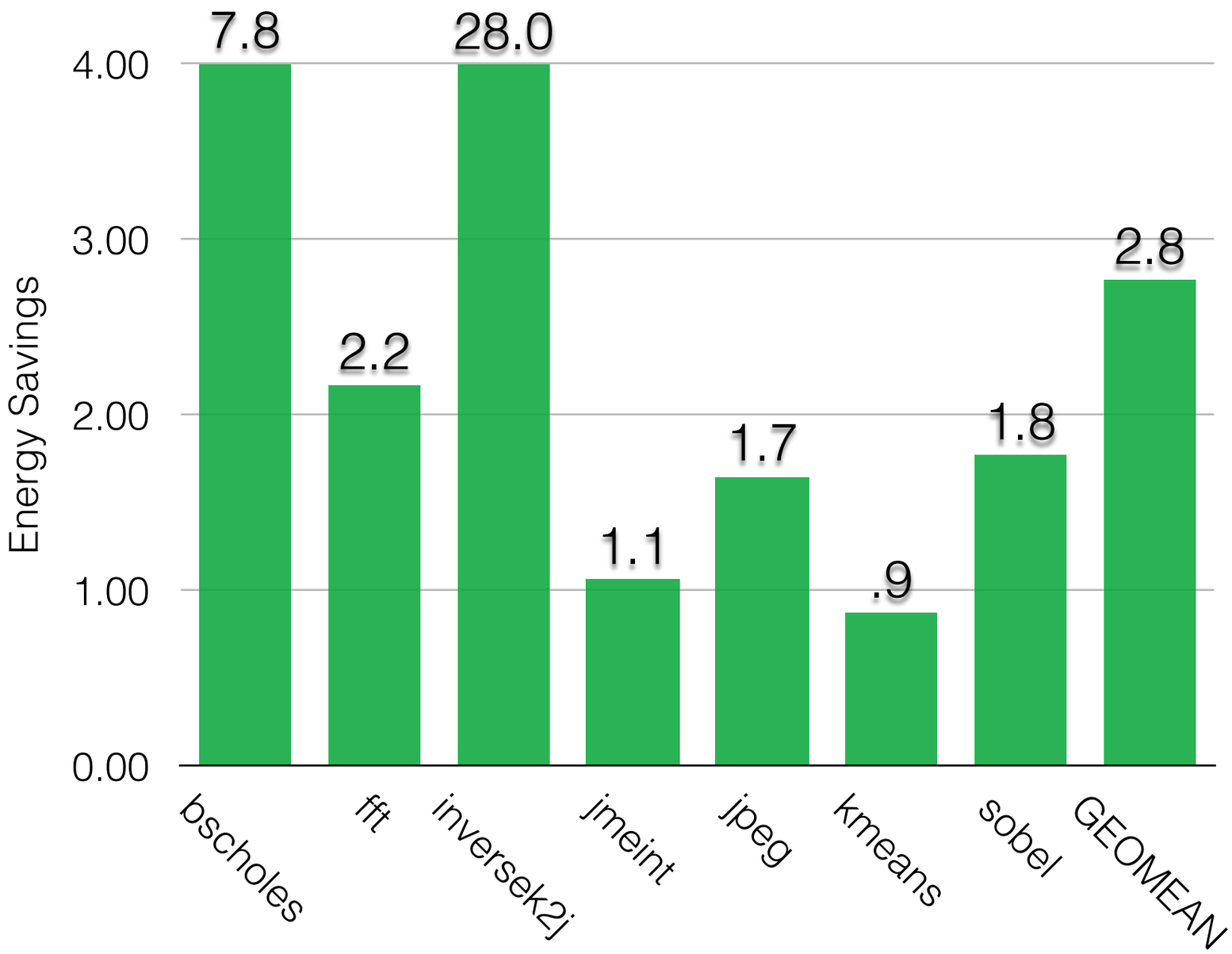
Speedup



Factors:
- Amdahl's Speedup
- Cost of instructions on CPU vs. cost of NN on SNNAP

	inversek2j	kmeans
Amdahl's Speedup	>100x	1.47x
CPU cost	1660 cycles	29 cycles
NN hidden layers	1	2

Energy Savings



+36%

Energy = Power on (DRAM + SoC) * Runtime

HW Acceleration

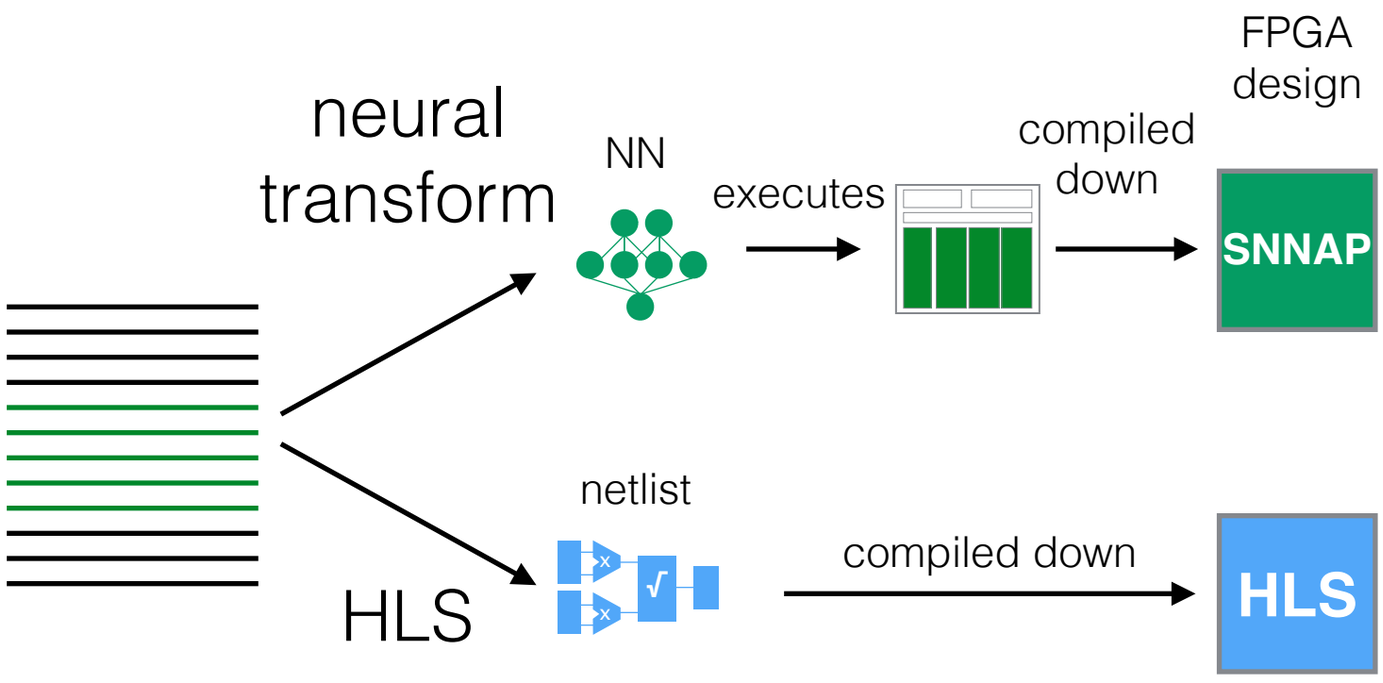
Neural Acceleration with SNNAP

vs.

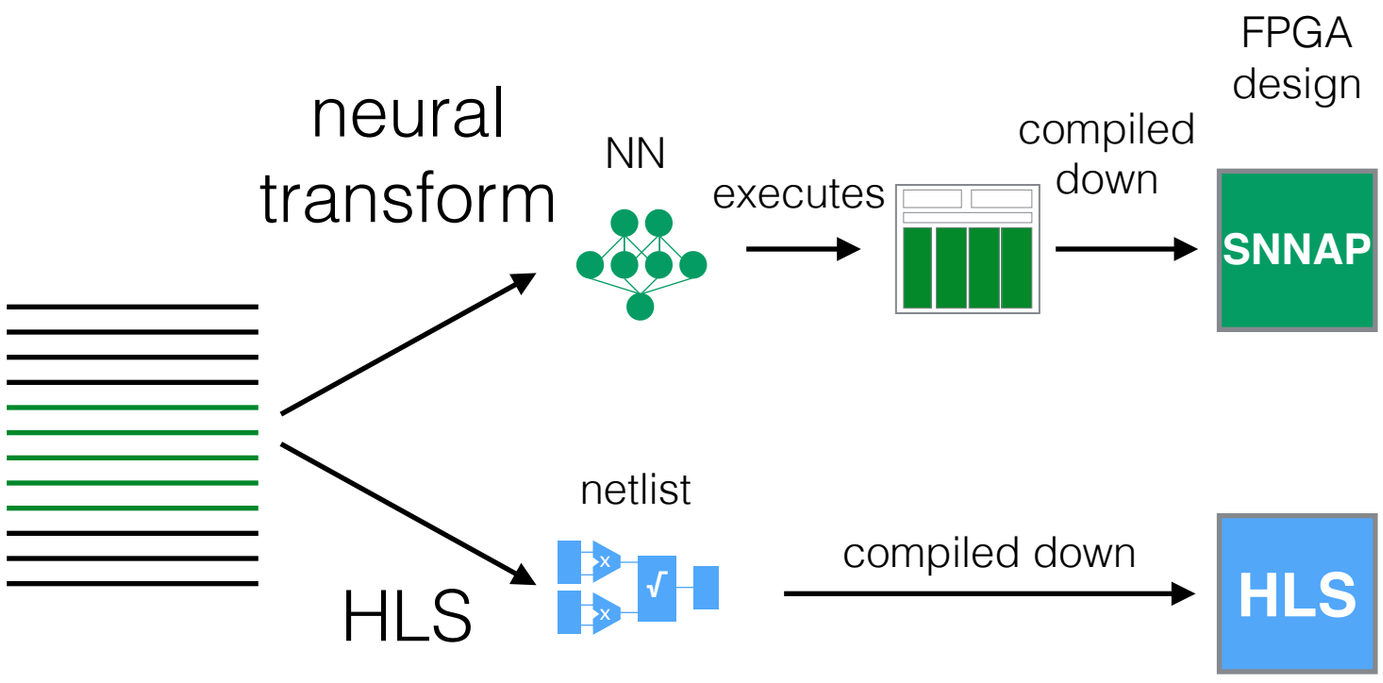
High Level Synthesis Compilers

which one should you use?

HLS Comparison Study



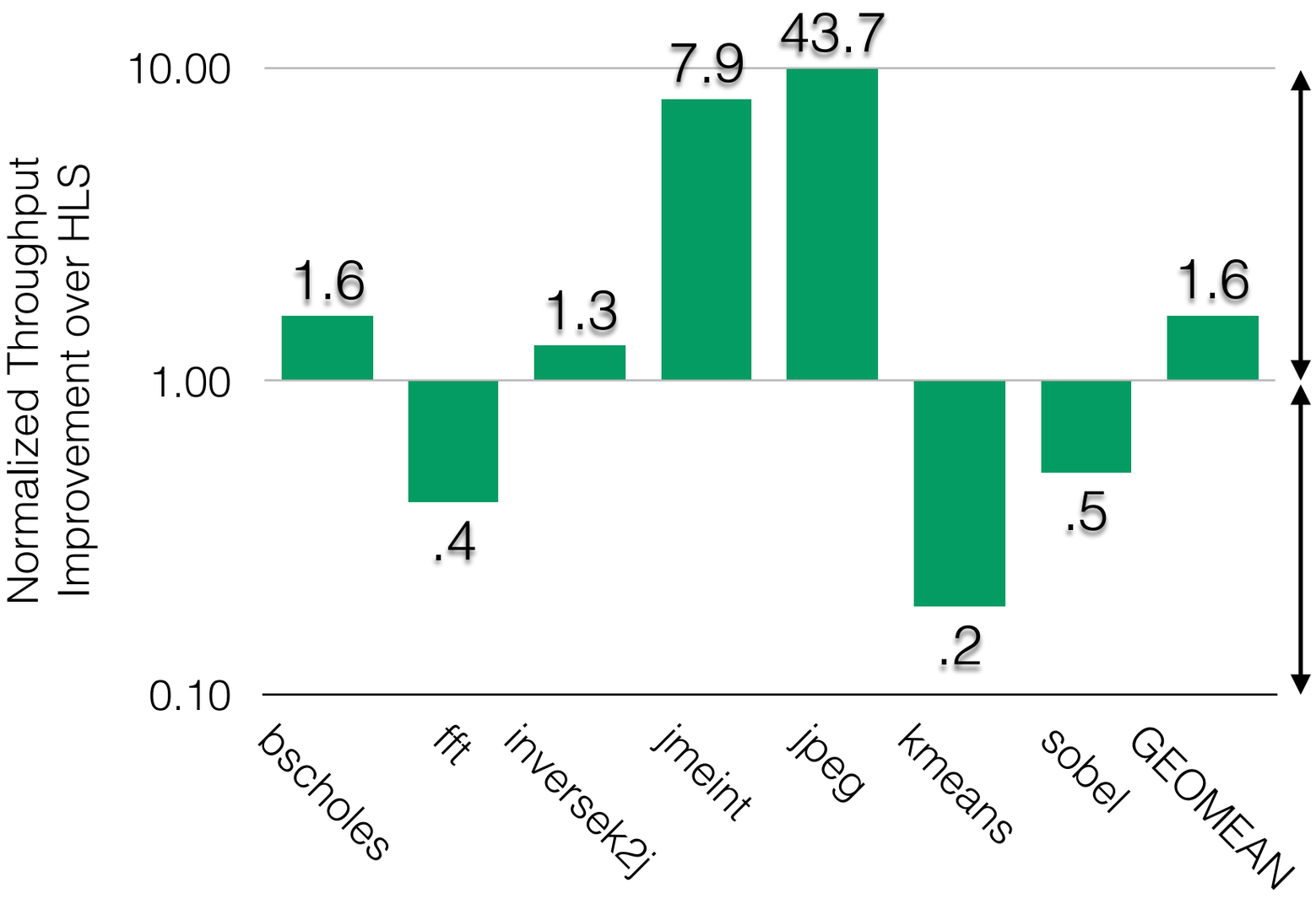
HLS Comparison Study



Resource-normalized throughput:

- pipeline invocation interval
- maximum frequency
- resource utilization

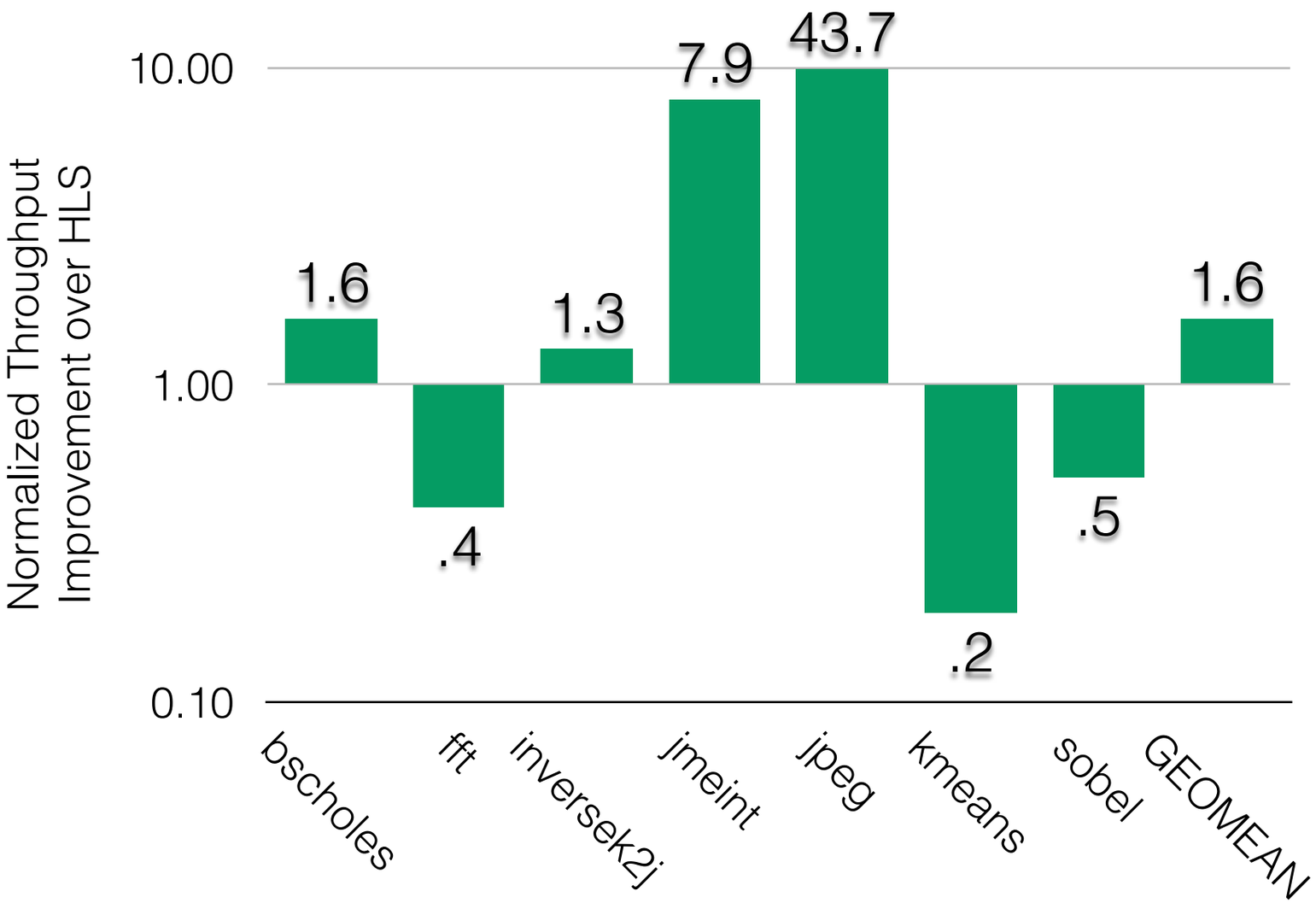
HLS Comparison Study



Neural Acceleration is better

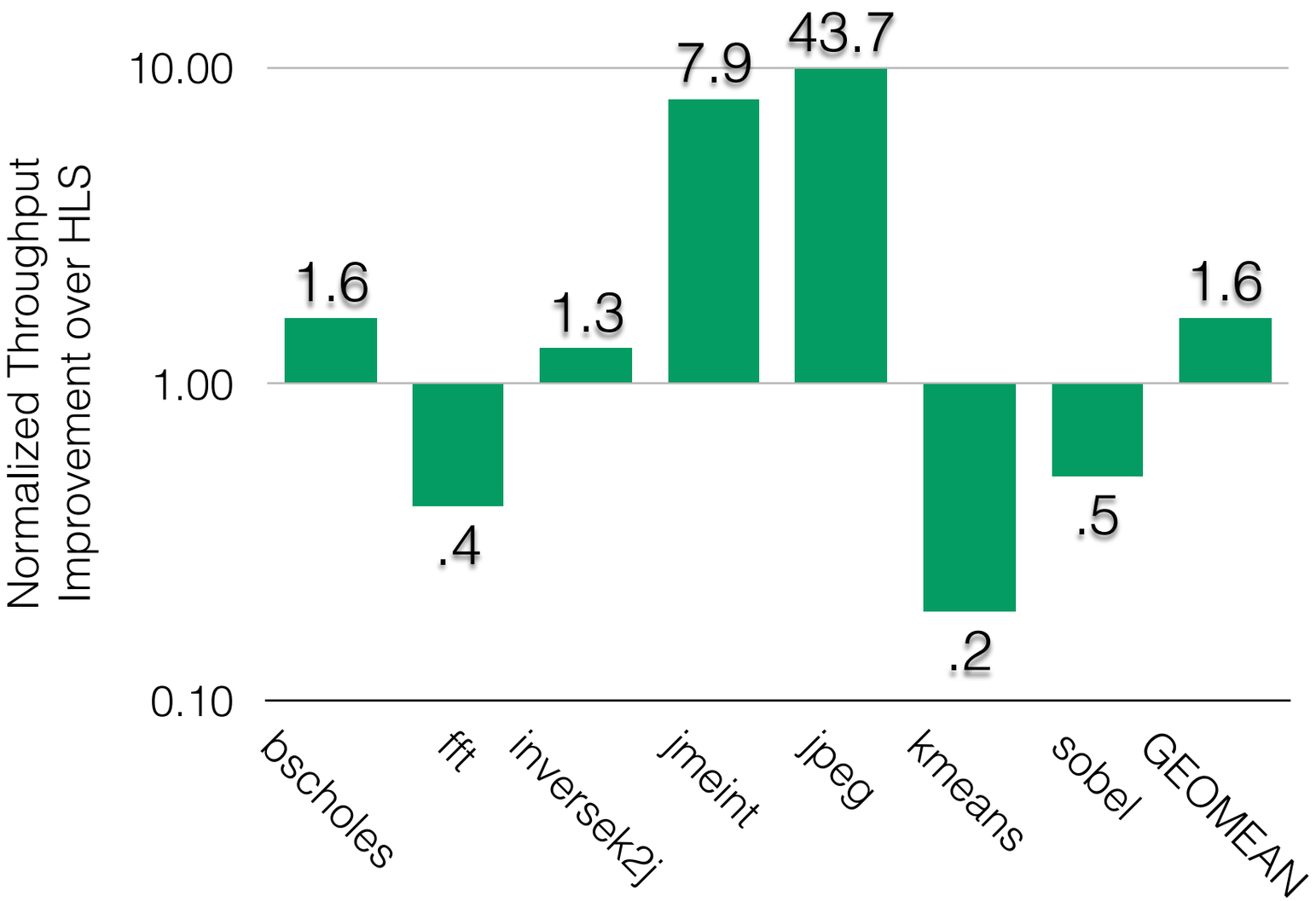
HLS is better

HLS Comparison Study



	Neural Accel.	HLS
Precision		✅
Virtualization	✅	
Performance		
Programmability		

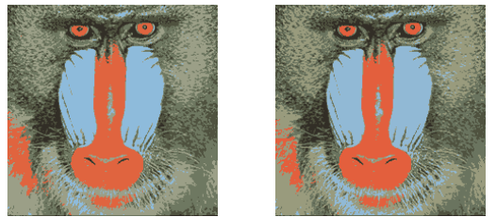
HLS Comparison Study



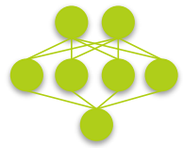
	Neural Accel.	HLS
Precision		✅
Virtualization	✅	
Performance	~	~
Programmability	✅	

Conclusion

SNNAP: apply **approximate computing** on **programmable SoCs** through **neural acceleration**



```
float foo (float a, float b) {
   …
   return r;
}
```



3.8x speedup & 2.8x energy savings

neural acceleration is a viable alternative to HLS

SNNAP:
Approximate Computing on Programmable SoCs via Neural Acceleration

Thierry Moreau: moreau@uw.edu

Mark Wyse
Jacob Nelson
Adrian Sampson

Hadi Esmaeilzadeh
Luis Ceze
Mark Oskin

http://sampa.cs.washington.edu/