
Kendall Stewart
University of Washington


Thierry Moreau
University of Washington


Luis Ceze
University of Washington

with Armin Alaghi, Adrian Sampson, Andre Baixo, Mark Wyse,
Jacob Nelson, Ben Ransford, Hadi Esmaeilzedeh and Mark Oskin

End-to-End Demonstration of an Approximate Computing System

Computation Tasks — Theme 2384.001

Approximate Computing Methodologies - Task 1.4

ACCEPT: C/C++ Compiler for Approximate Programs

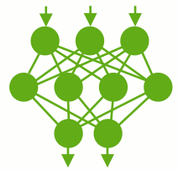
```
APPROX float a;
float p;

a = p;  ✓        p + p;  ← precise add
p = a;  ✗        a + p;  ← approx add
                 a + a;  ←
```

ACCEPT **automatically** identifies **safely** approximable code based on **lightweight** data annotations

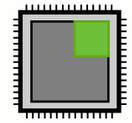
A quality **autotuner** helps programmers **navigate** the quality/ performance **tradeoff** space

SNNAP: Hardware Support for Neural Acceleration

Neural Acceleration **approximates** compute-intensive functions with **neural networks** which can then be efficiently evaluated on a **hardware accelerator**



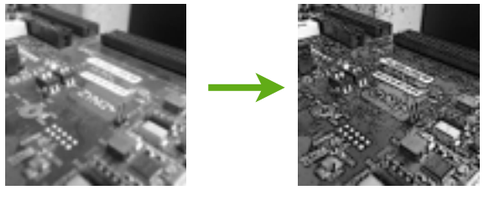
We leverage off-the shelf programmable SoCs to implement an **efficient** and **reconfigurable** Neural Processing Unit



programming model

The programmer marks data that is **approximable** using type **annotations**. In addition the programmer provides a **quality metric** and a set of representative **test inputs**.

Example: Image “Cartoonization"



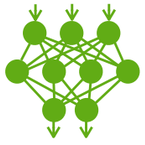
```
APPROX float cartoonize(float w[3][3]) {
  bool mask = diff_of_gauss(w) < -3e-3;
  return bilateral(w) * mask;
}

float** src;
APPROX float** dst;

while (true) {
  src = read_from_camera();
  for (y=0; y < h; ++y) {
    for (x=0; x < w; ++x) {
      dst[y][x] =
        cartoonize(window(src, y, x));
    }
  }
  display(ENDORSE(dst));
}
```

“push button” approximate compilation

test inputs

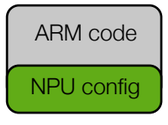
Instrumented runs, neural network training

NN config



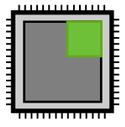
NPU microcode generation, program instrumentation

hybrid binary

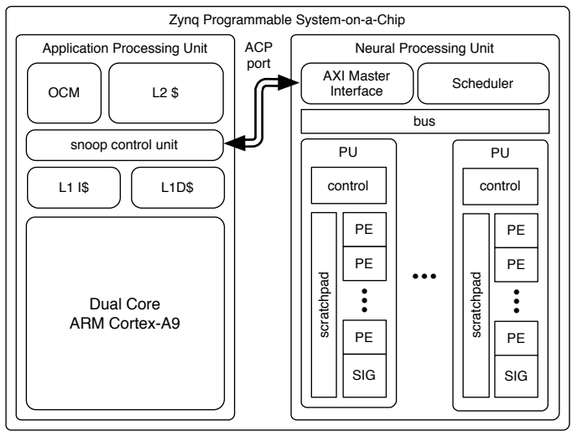


Dynamic NPU configuration, neural network offloading

neural acceleration



hardware design



Tight coupling allows **low-latency, coherent communication** with the core.

Processing Elements arranged in a systolic array make **efficient** use of DSPs

Replicated structure can take advantage of **parallelism** in the application

NPU micro-coded schedule and weights can be dynamically loaded into local memory for **fast reconfiguration**

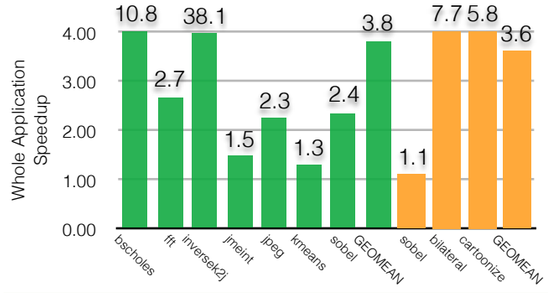
evaluation results

average application speedup

Baremetal	Linux
3.8×	3.6×
range: 1.3× – 38×	range: 1.1× – 7.7×

average energy reduction

2.8×	2.8×
range: 0.87× – 28×	range: 0.86× – 5.65×



tech transfer

This work was partly funded by Qualcomm through their Innovation Fellowship program.

ACCEPT is an **open-sourced** compiler accessible at http://accept.rocks

Publications & Reports:
Sampson et. al, *ACCEPT: A Programmer-Guided Framework for Practical Approximate Computing*, UW-TR
Moreau et al., *SNNAP: Approximate Computing on Programmable SoCs via Neural Acceleration*, HPCA2015