iACT: A Software-Hardware Framework for Understanding the Scope of Approximate Computing

Asit K. Mishra   Rajkishore Barik   Somnath Paul

Intel Labs

Motivation

- Bottom-Up
  - Devices are becoming less reliable
    - We operate devices at limits of reliability
  - Error detection/correction is expensive
    - Opportunity for significant energy improvement

Motivation

- Bottom-Up
  - Devices are becoming less reliable
    - We operate devices at limits of reliability
  - Error detection/correction is expensive
    - Opportunity for significant energy improvement
- Top-Down
  - Important applications can be approximate
    - Computer Vision
    - Graphics (raster and ray tracing)
    - Speech
    - Signal processing
    - Machine learning
    - Any lossy codec (video encode / decode)

State of Current Research

- Lots of recent proposals advocating the potential of in this area
- But..
  - Ad hoc techniques

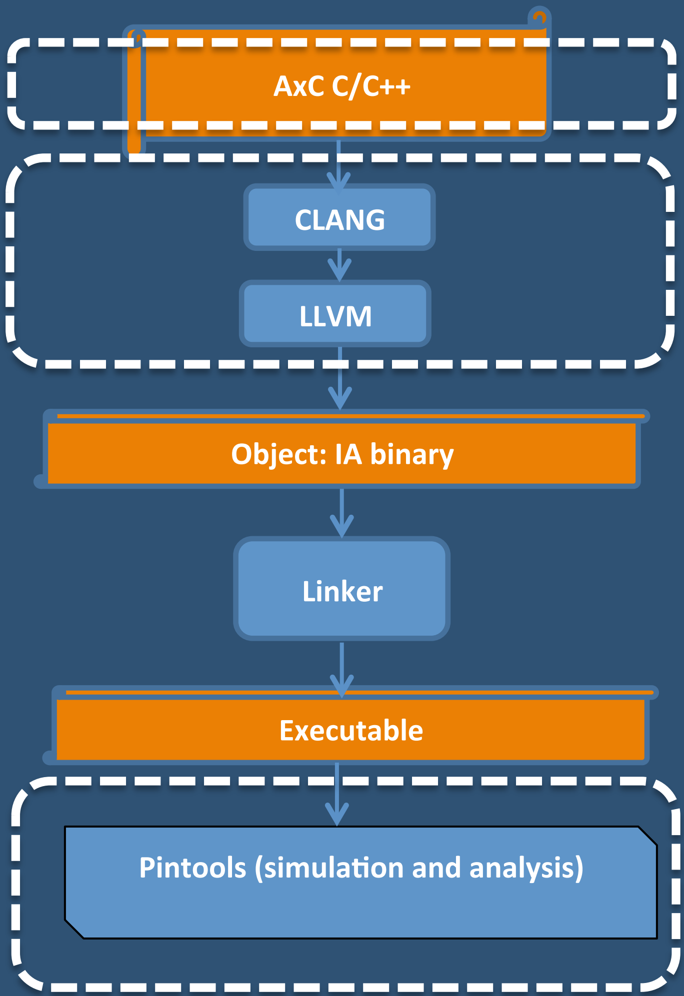
    e.g. code perforation with sparse data didn't work
  - Small number of applications
  - Simulation environment

State of Current Research

- Lots of recent proposals advocating the potential of in this area
- But..
  - Ad hoc techniques

    e.g. code perforation with sparse data didn't work
  - Small number of applications
  - Simulation environment
- How does one do a 1st order analysis to study the scope of approximations in an application
  - Intel's Approximate Computing Toolkit (iACT)!

iACT

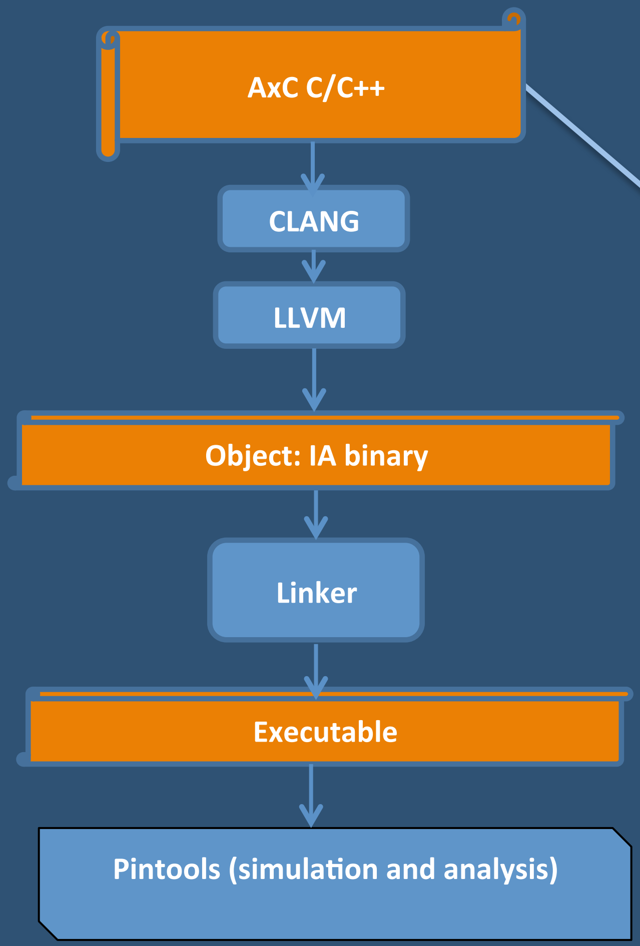


Sample applications

A compiler and runtime framework

A simulated hardware testbed

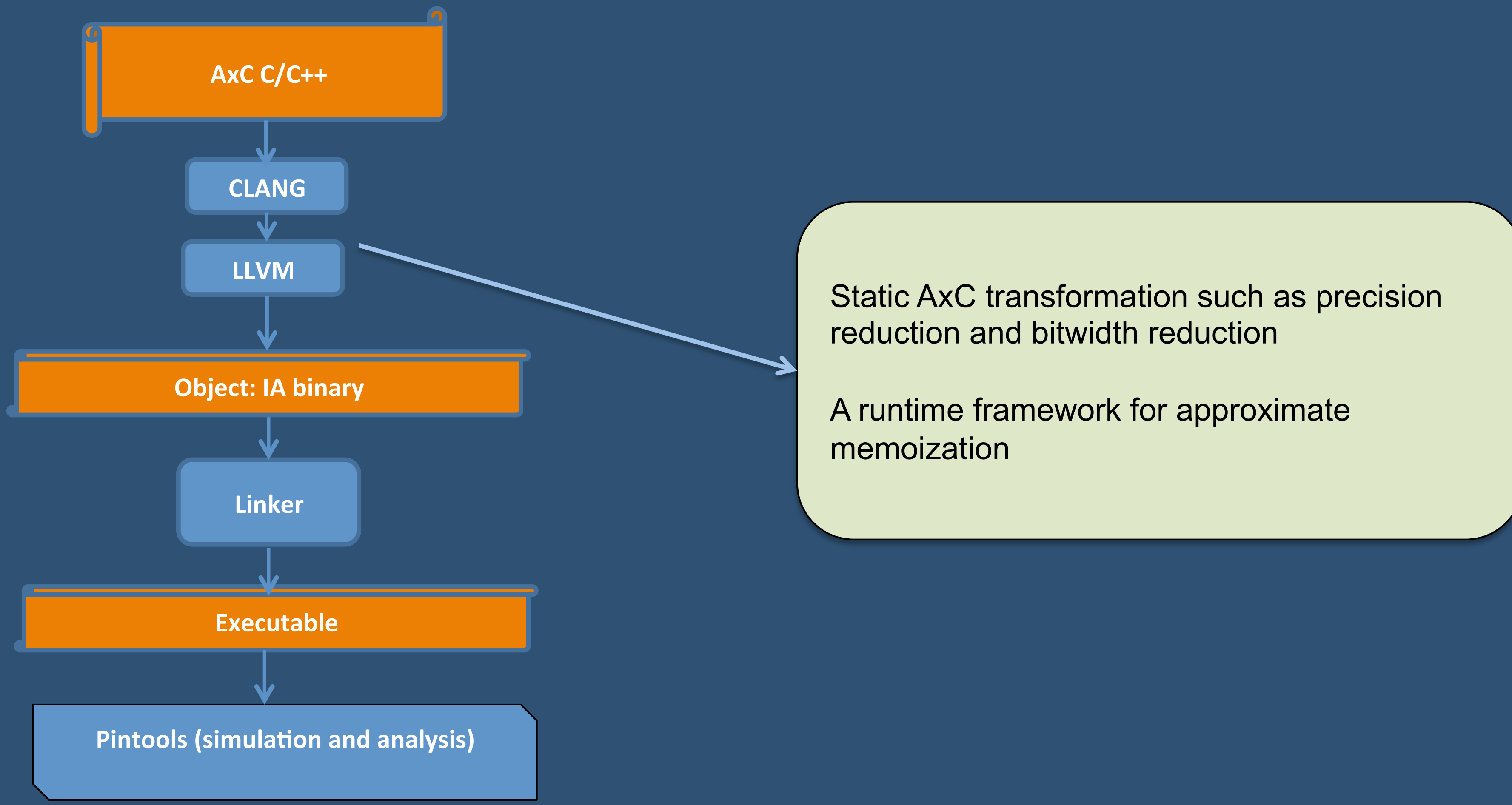
iACT



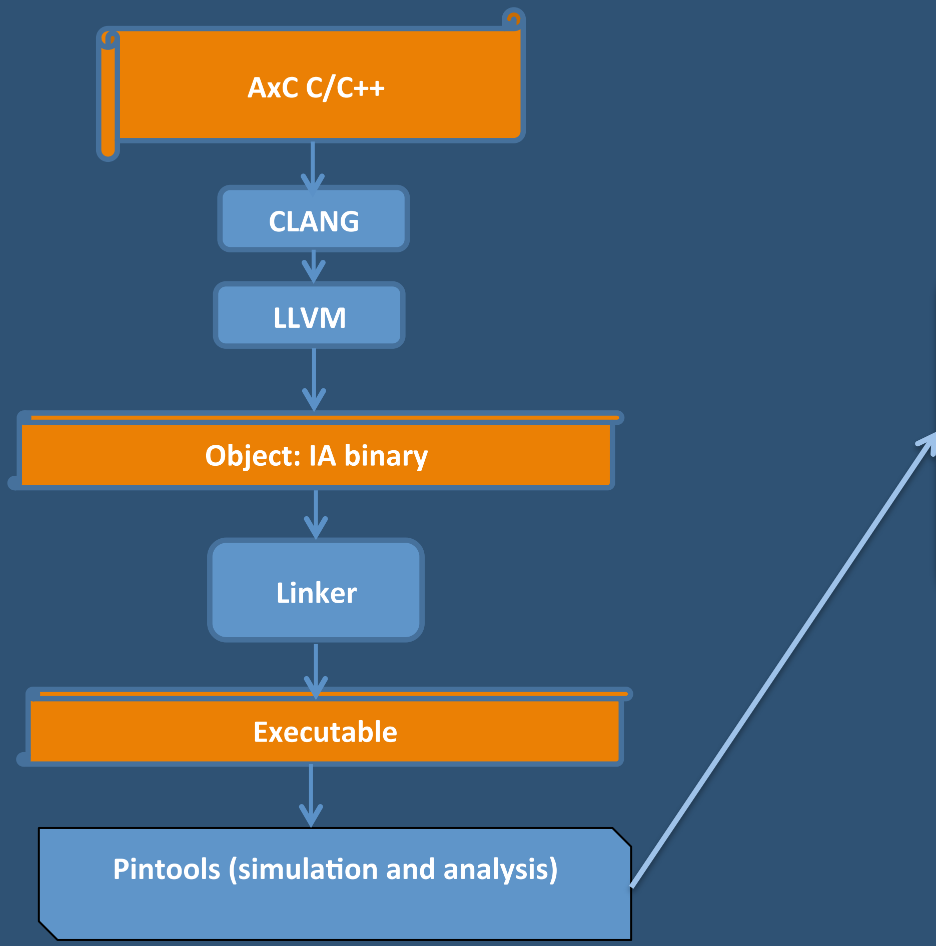
Language pragmas
#pragma axc
#pragma axc_precision_reduce
#pragma axc memoize [(arg, err), …]
{out_vars}

Programmer provided per-function checker functions

iACT


AxC C/C++
CLANG
LLVM
Object: IA binary
Linker
Executable
Pintools (simulation and analysis)
Static AxC transformation such as precision reduction and bitwidth reduction
A runtime framework for approximate memoization

iACT

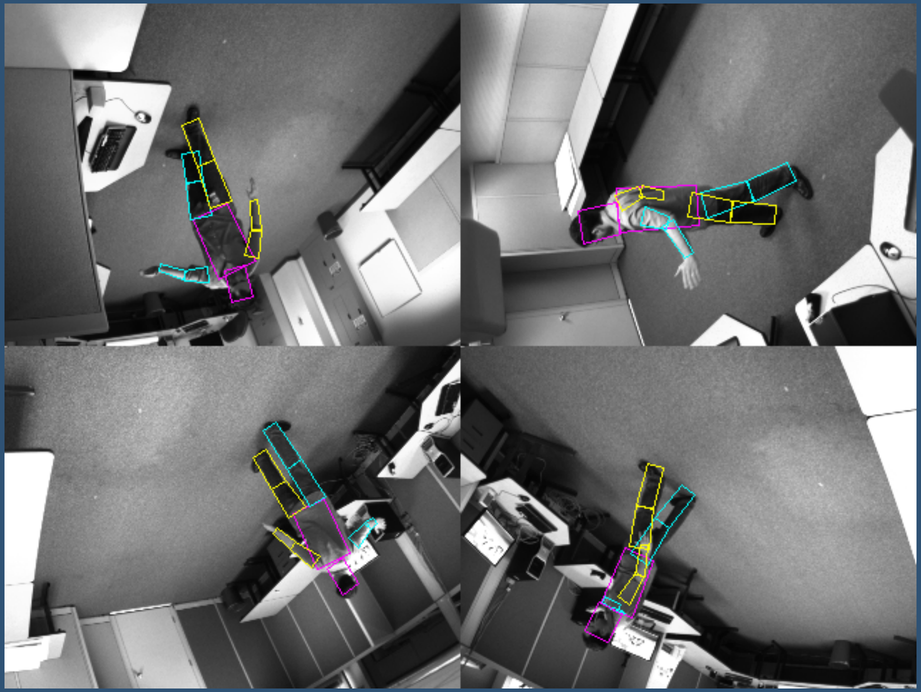


- Precision reduction
- Hardware memoization (WIP)
- Noisy computation
- Noisy memory modules
- Noisy network channels

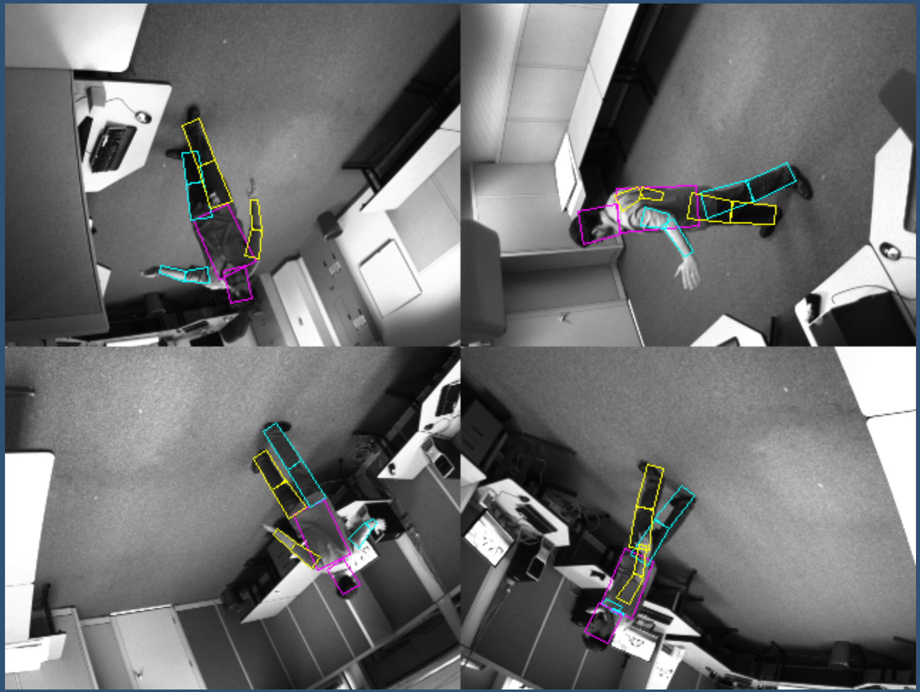
iACT – Sample Workloads

- Bodytracking (from PARSEC)
- Sobel filter
- Classification algorithm

Bodytracking Application

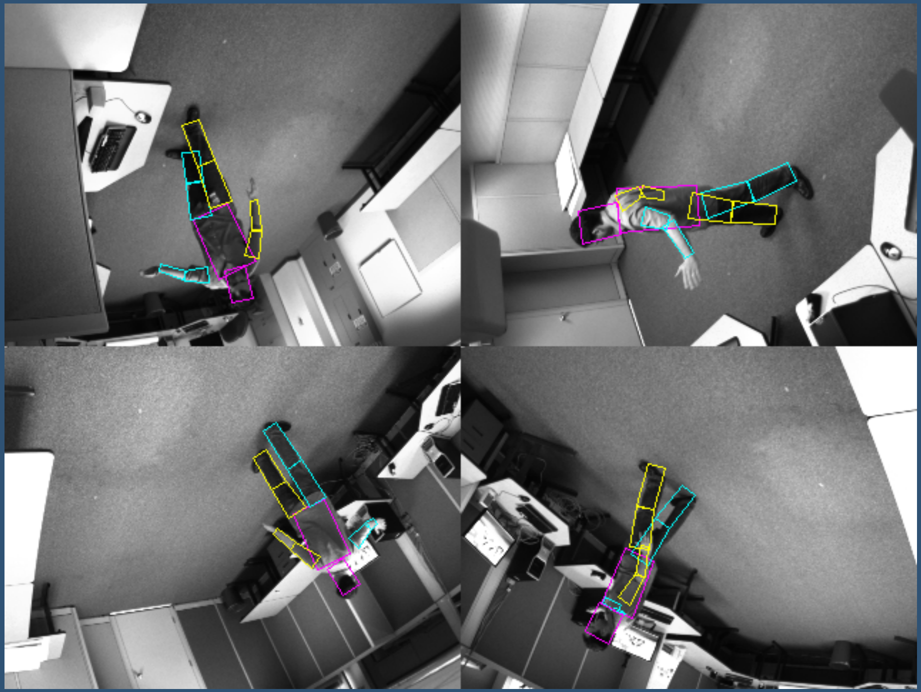


Tracking result 1

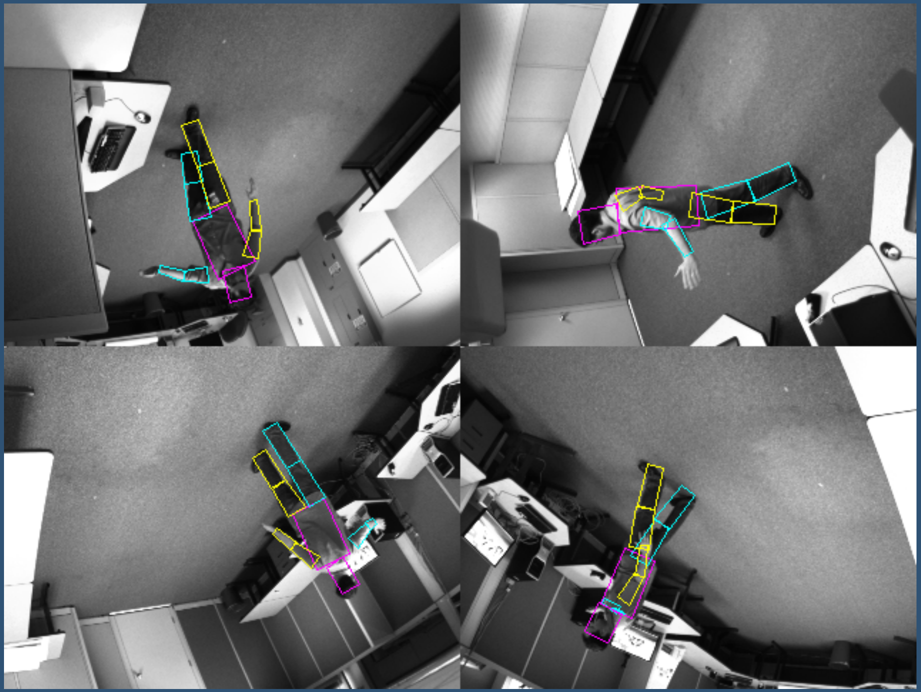


Tracking result 2

Bodytracking Application



Tracking result 3

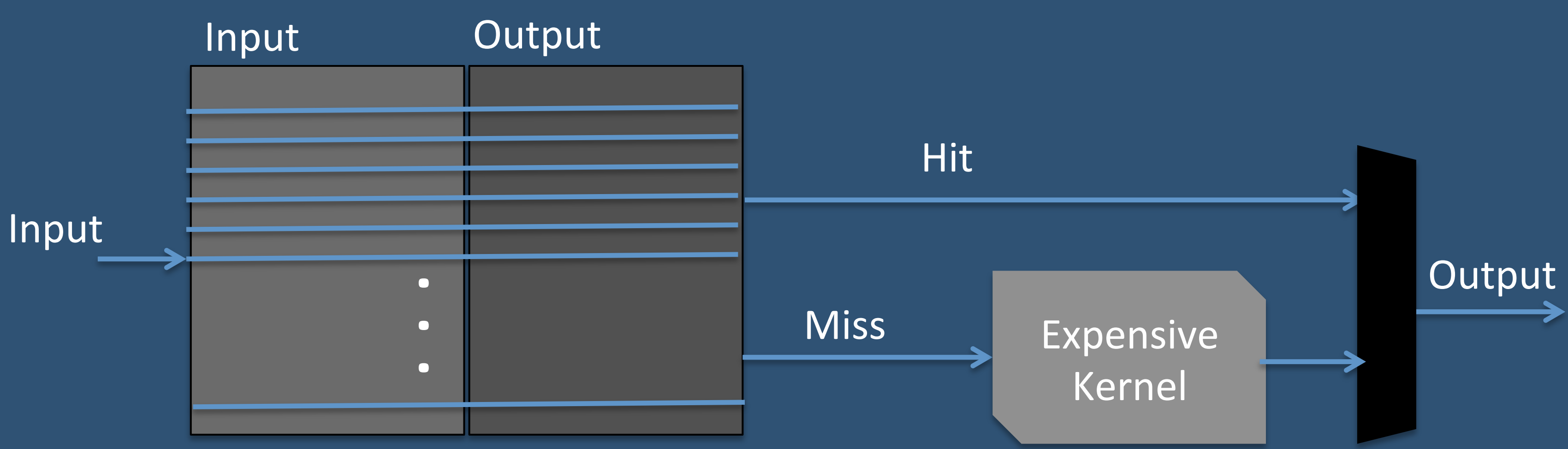


Tracking result 4

Sobel Filter

For "similar" window of pixels, read out the value from the memoization table

- If "miss" in table, then do the "expensive" computation and populate the table
- If "hit" in table then read out the result value



Sobel Filter

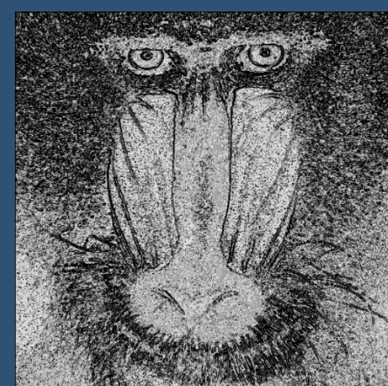
For "similar" window of pixels, read out the value from the memoization table

- If "miss" in table, then do the "expensive" computation and populate the table
- If "hit" in table then read out the result value

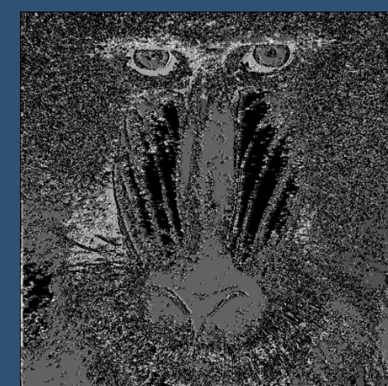
- Results show with "small" output quality degradation, we could have 60% hit in a small size table

Precise output

54b/entry



36b/entry



27b/entry

(Semi)Auto-generated Checker Functions - WIP

- The checker functions could be
  - programmer specified
  - auto generated by the runtime layer

A ML framework learns the relationship between approximation knobs and acceptable outputs

Also in the paper..

- Taxonomy of approximate computing
- Why an application could be amenable to approximate computing

Conclusions

- iACT toolkit
  - Language level constructs
  - Runtime framework
  - Approximate hardware simulation
  - Sample applications

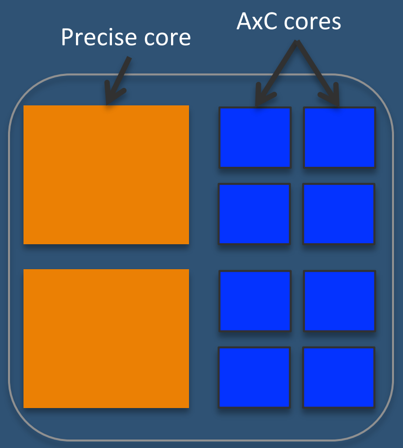
iACT Toolkit

https://github.com/IntelLabs/iACT

Asit K. Mishra

Backup

iACT - Hardware Simulation

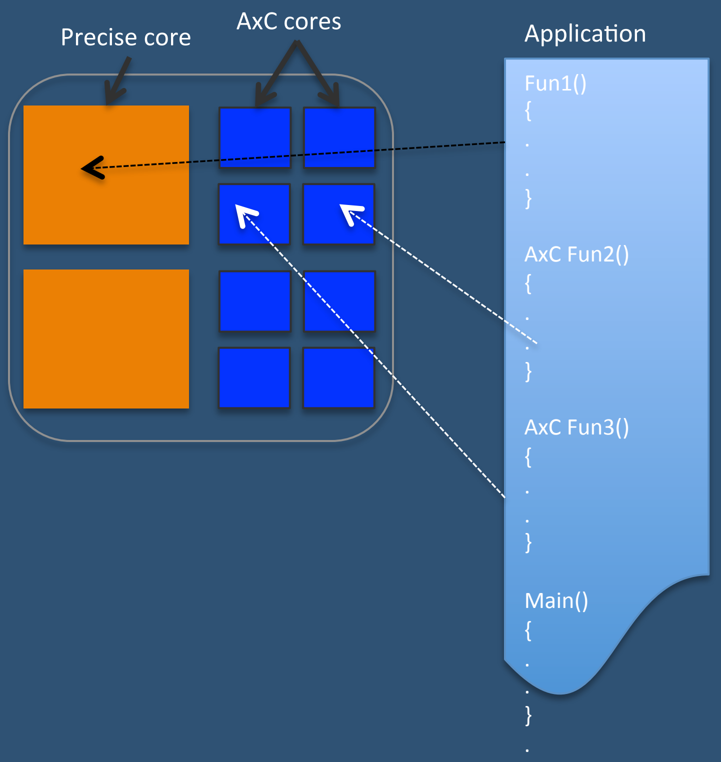


PIN based tool

Simulates a many core processor, few cores are "precise cores" and most other cores are "AxC cores"

iACT - Hardware Simulation

Functions annotated as “AxC tolerant” are executed on the AxC cores, rest of the code is executed on the “precise core”



iACT - Hardware Simulation

AxC cores would have knobs for

- Approximate ld/st to register files
- Caches operating at low voltage, storing imprecise values
- Imprecise but energy efficient functional units (for add, sub, mul, div, etc)
- Lossy interconnect
- Etc, etc (energy efficient knobs to enable AxC)