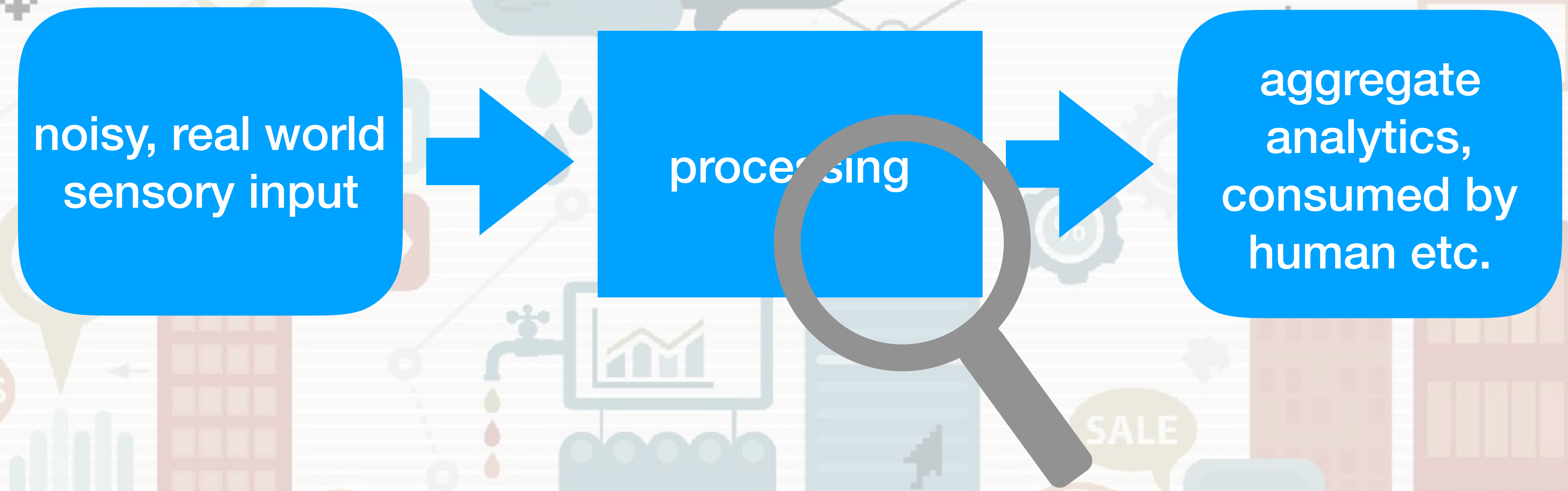
Exploiting Quality-Efficiency Tradeoffs with Arbitrary Quantization

Special Session - CODES+ISSS

Thierry Moreau, Felipe Augusto, Patrick Howe
Armin Alaghi, Luis Ceze

Internet of Things Revolution

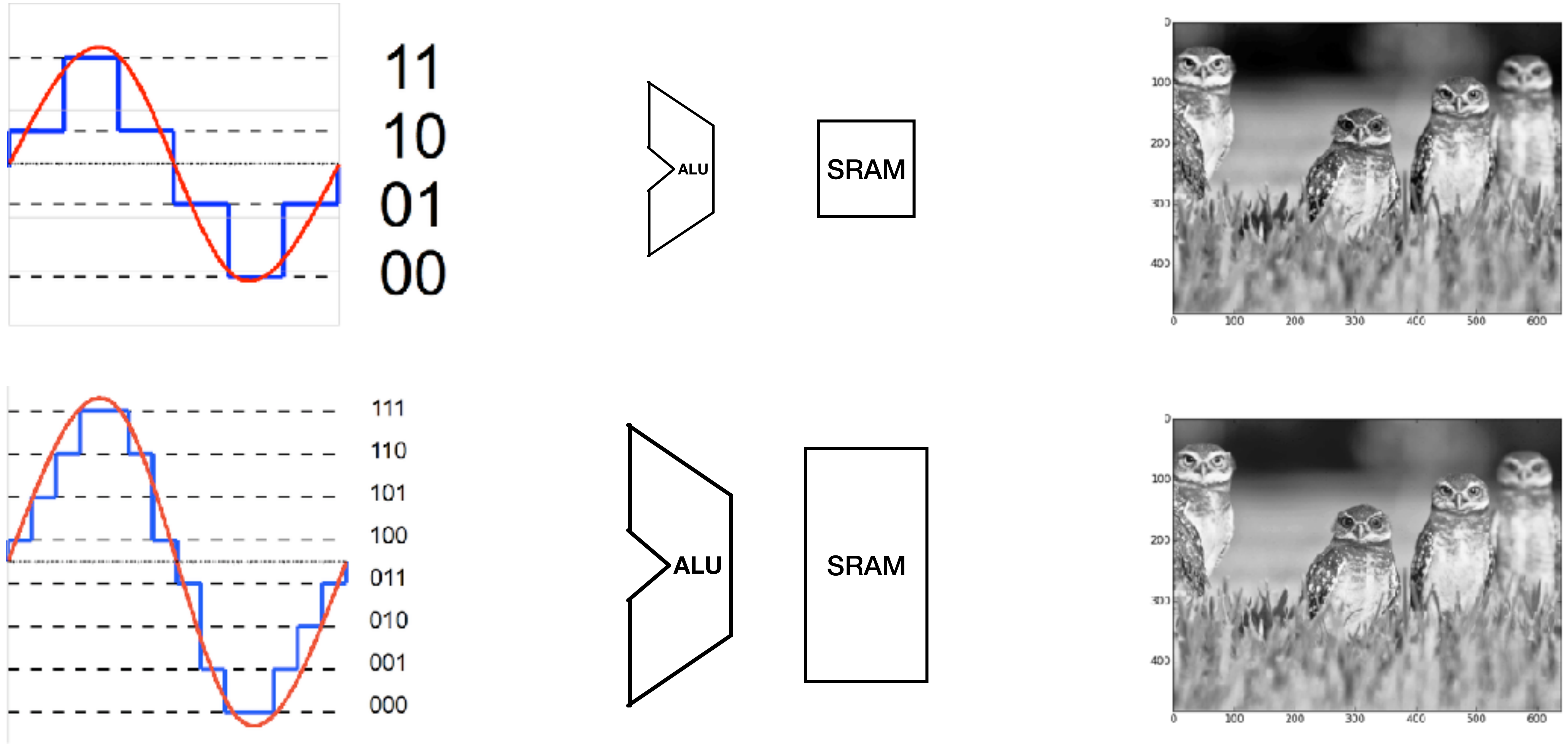



```
…
double temp = sensor_acquire();
…
```

Approximate computing: eliminate inefficiencies in systems by producing just-the-right quality

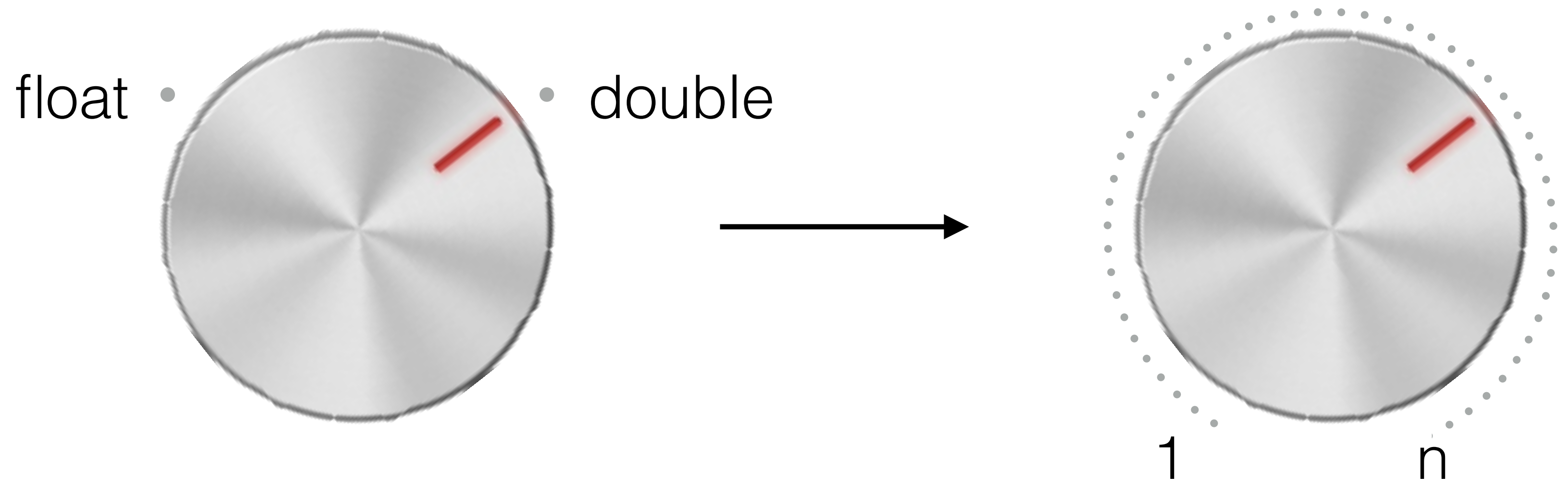
Quantization: going back to basics

noisy, real world sensory input → processing → aggregate analytics, consumed by human etc.



This Talk: A “Limit Study” on Precision Scaling

Assumption: *hardware that can **dynamically** and **arbitrarily** scale its precision*



SW Scope: *compute heavy, regular applications*

HW Scope: *hardware accelerators*

Talk Overview

1. How much precision is needed at different stages of a program?

2. How much energy can be saved (upper bound)?

3. How does this inform approximate computing research?

Talk Overview

1. How much precision is needed at different stages of a program?

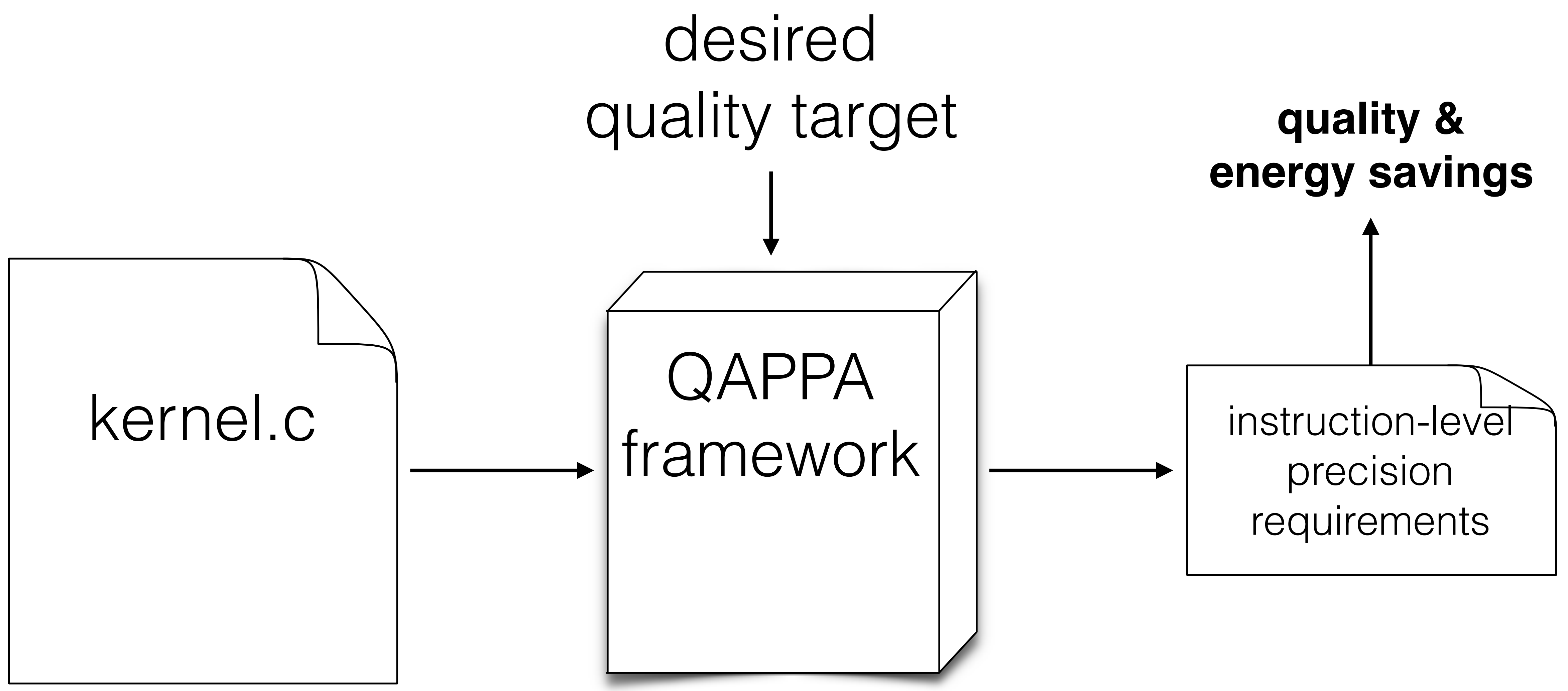
QAPPA - Precision Autotuner

2. How much energy can be saved?

3. How does this inform approximate computing research?

QAPPA: Quality Autotuner for Precision-Programmable Accelerators

Goal: Minimize instruction-level precision requirements given a quality target

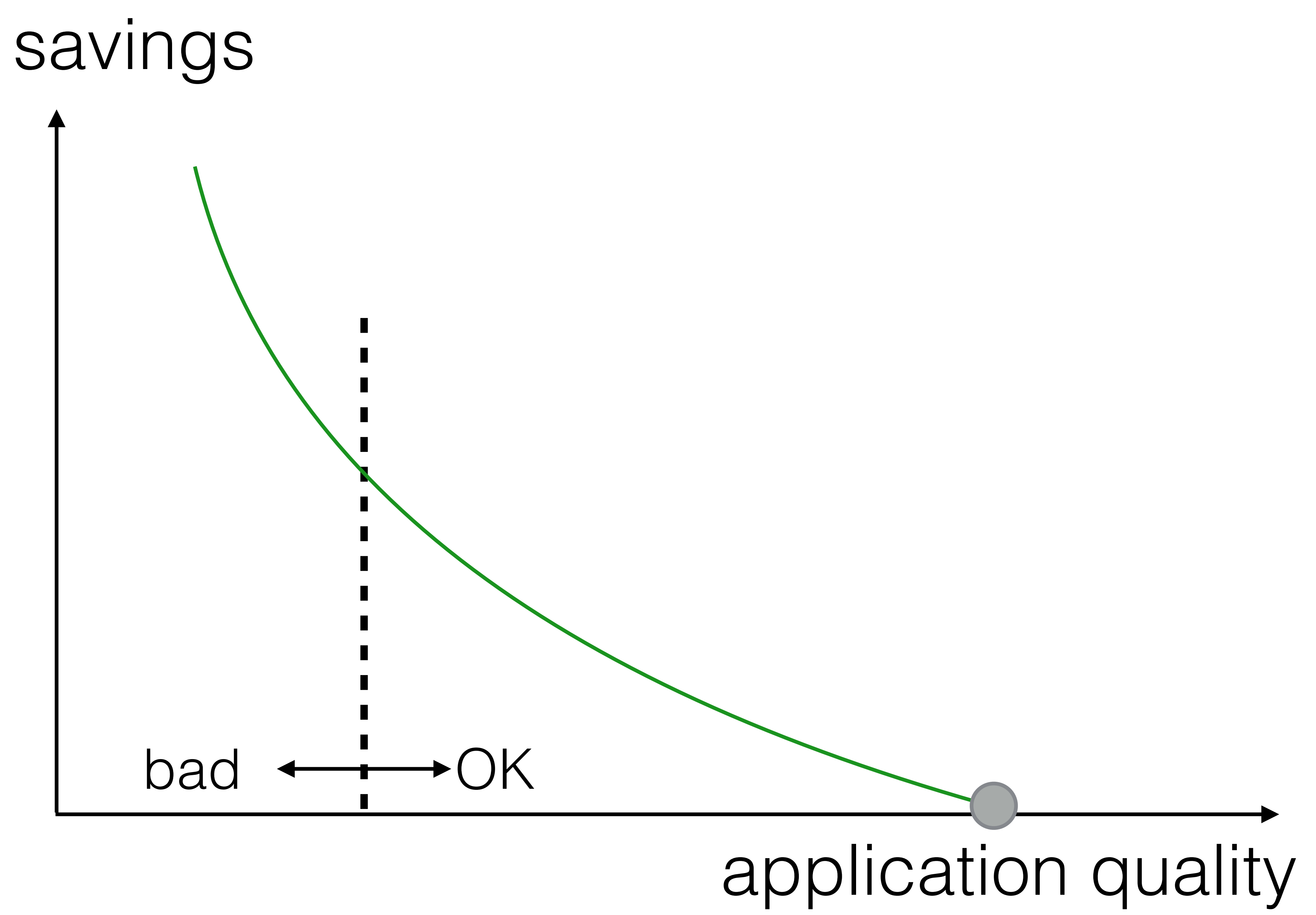


Built on top of **ACCEPT**, the approximate C/C++ compiler
http://accept.rocks

QAPPA Autotuner Overview

Default (no savings)

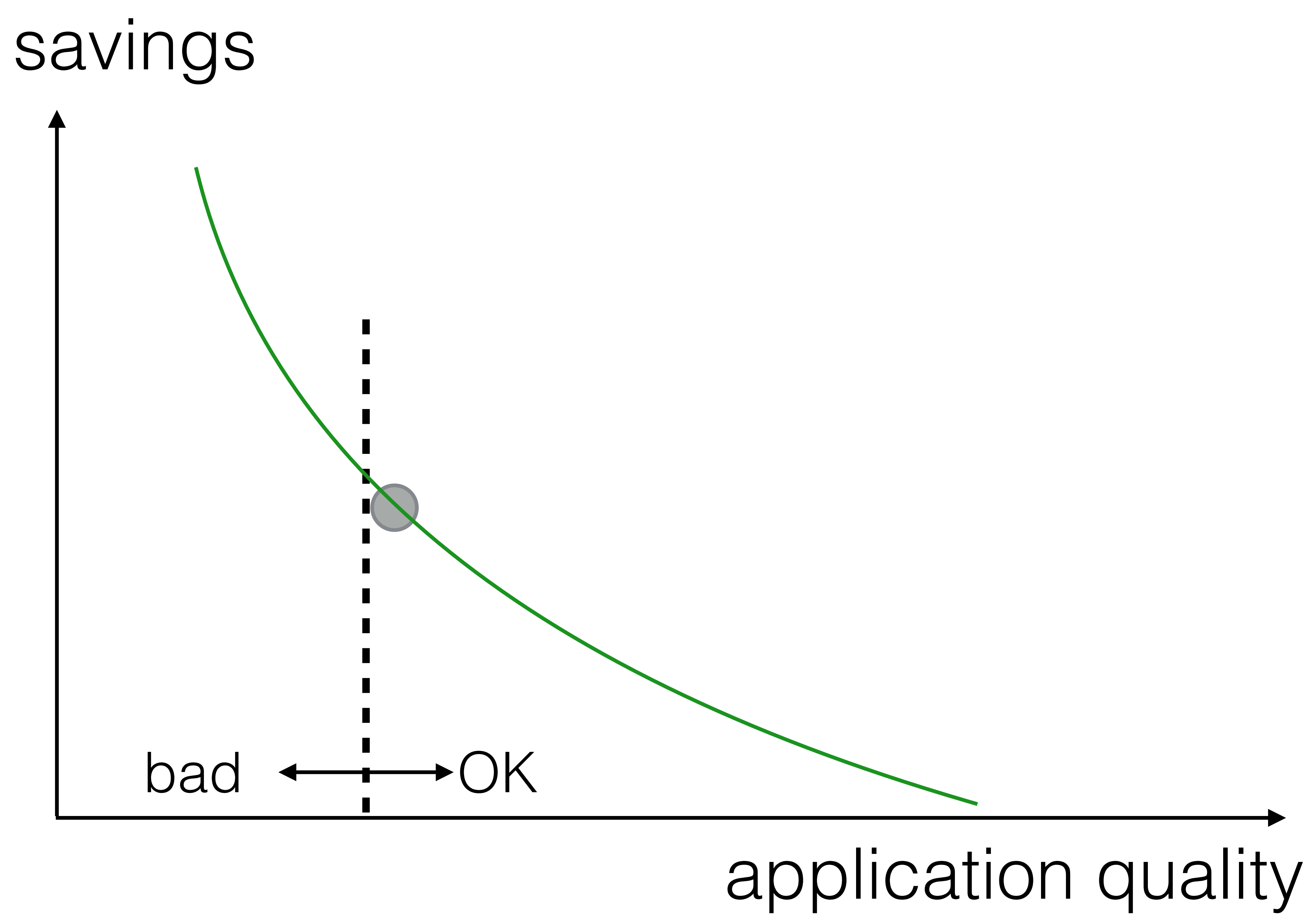
```
instruction 0
instruction 1
instruction 2
…
instruction n-1
instruction n
```



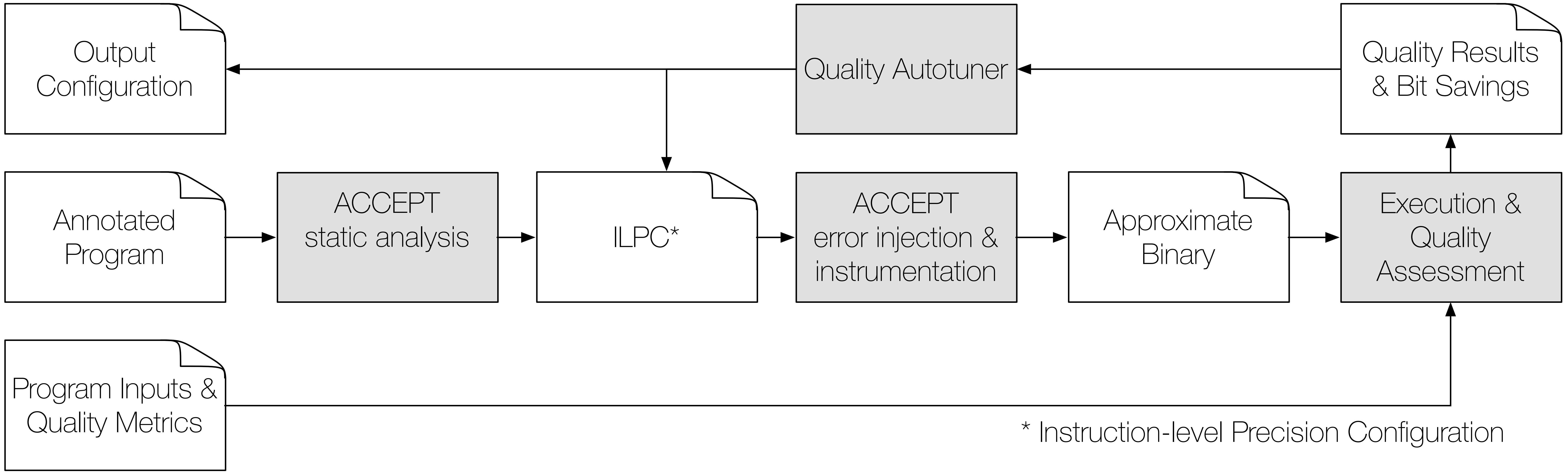
QAPPA Autotuner Overview

Optimized: extraneous precision is shaved off

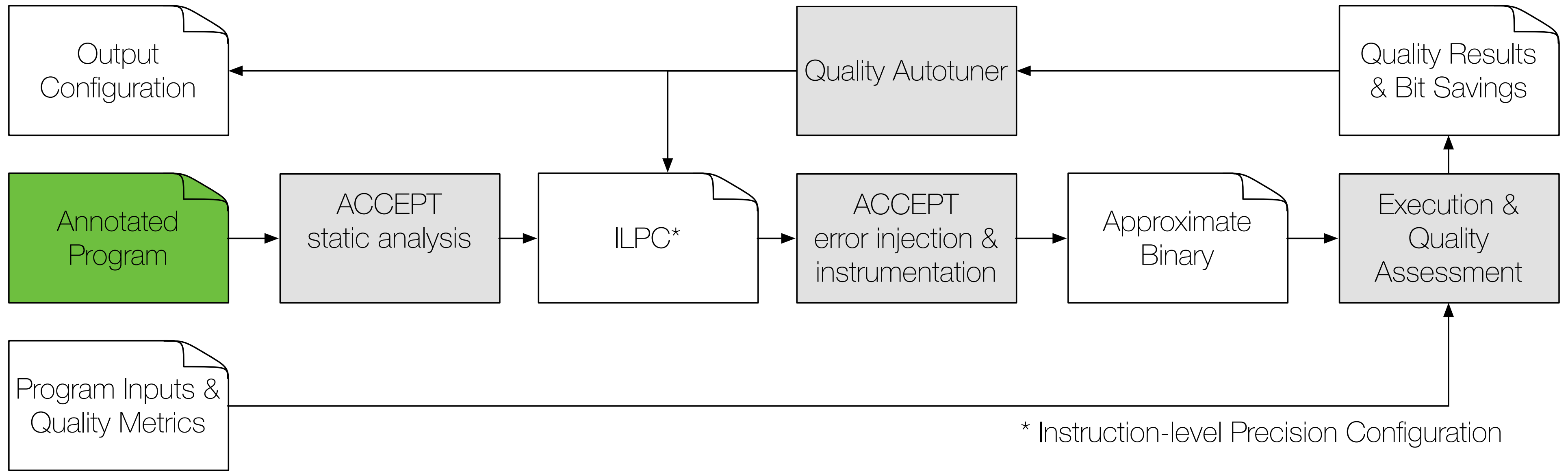
```
instruction 0
instruction 1
instruction 2
...
instruction n-1
instruction n
```



QAPPA 5-Step Description



1. Program Annotation

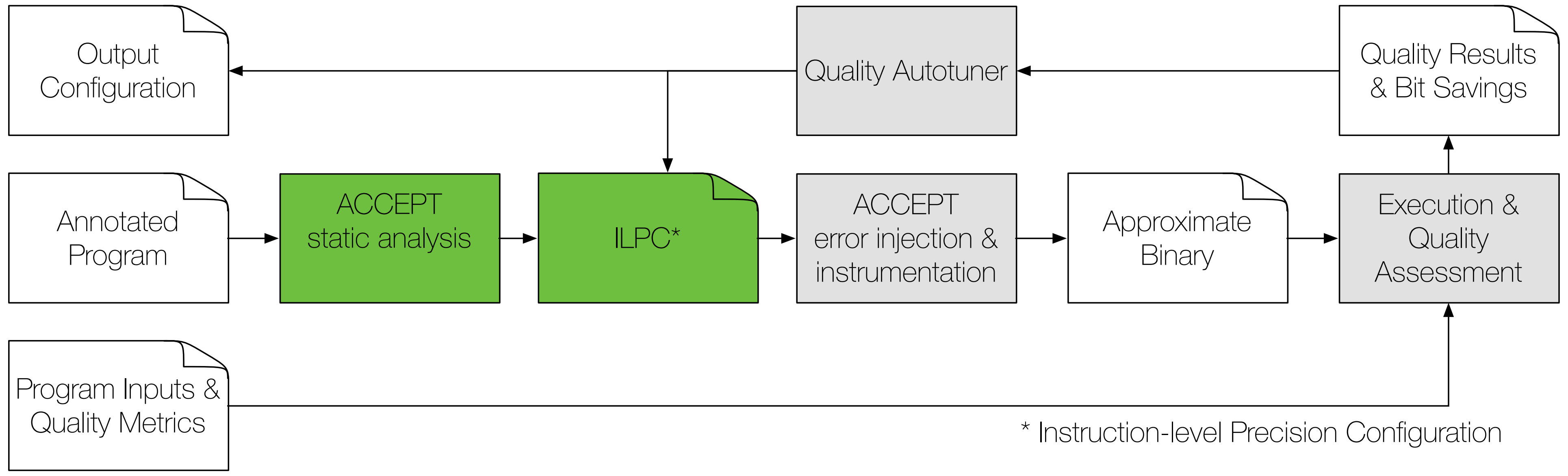



```
void
conv2d (APPROX pix *in, APPROX pix *out, APPROX flt *filter)
{
 for (row) {
  for (col) {
   APPROX flt sum = 0
   int dstPos = …
   for (row_offset) {
    for (col_offset) {
     int srcPos = …
     int fltPos = …
     sum += in[srcPos] * filter[fltPos]
    }
   }
   out[dstPos] = sum / normFactor
  }
 }
}
```

Key: use the **APPROX** type qualifier [*]

[*] EnerJ, Sampson et al., PLDI'11

2. Static Analysis




```
void
conv2d (APPROX pix *in, APPROX pix *out,
APPROX flt *filter)
{
 for (row) {
  for (col) {
   APPROX flt sum = 0
   int dstPos = …
   for (row_offset) {
    for (col_offset) {
     int srcPos = …
     int fltPos = …
     sum += in[srcPos] * filter[fltPos]
    }
   }
   out[dstPos] = sum / normFactor
  }
 }
}
```

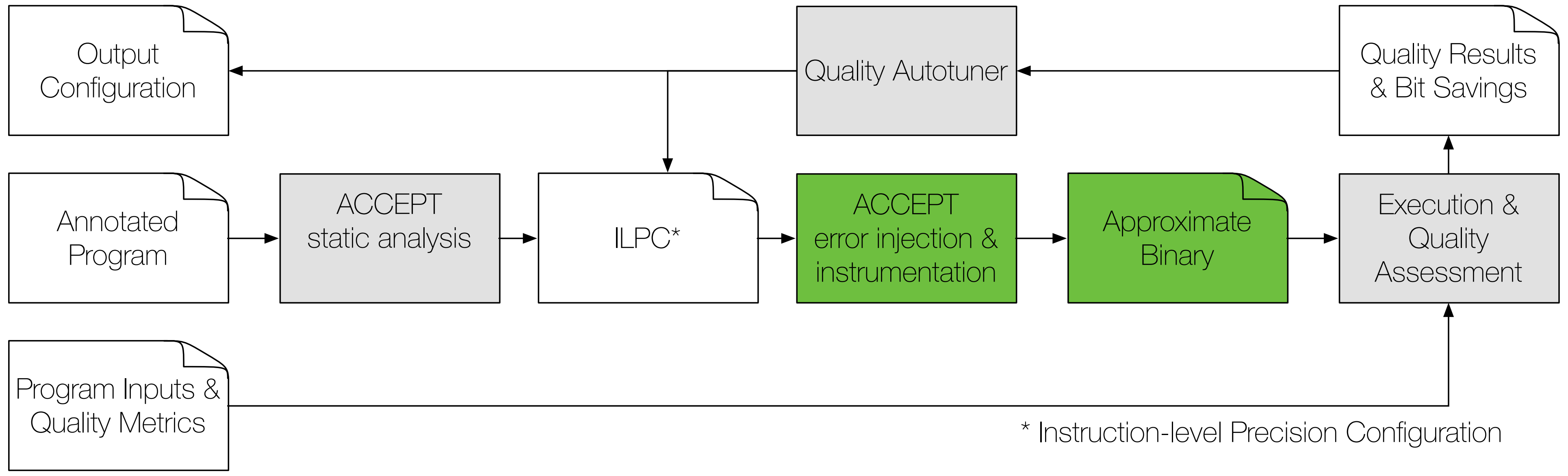
ACCEPT →

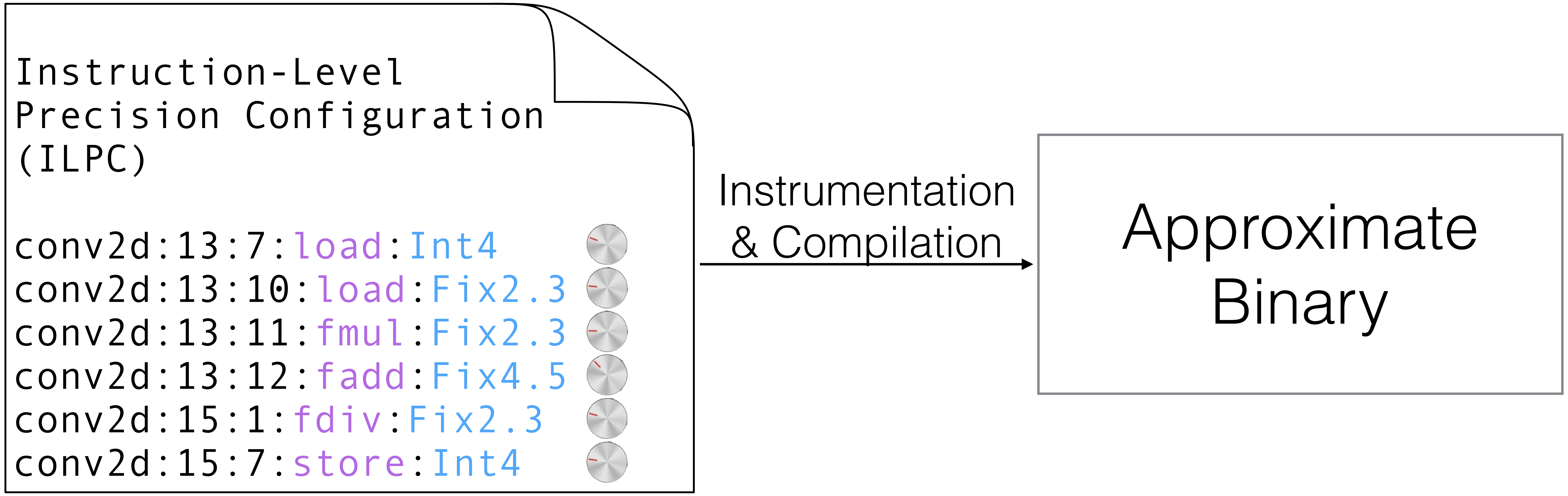
```
Instruction-Level
Precision Configuration
(ILPC)

conv2d:13:7:load:Int32
conv2d:13:10:load:Float
conv2d:13:11:fmul:Float
conv2d:13:12:fadd:Float
conv2d:15:1:fdiv:Float
conv2d:15:7:store:Int32
```

ACCEPT identifies safe-to-approximate instructions from data annotations using flow analysis

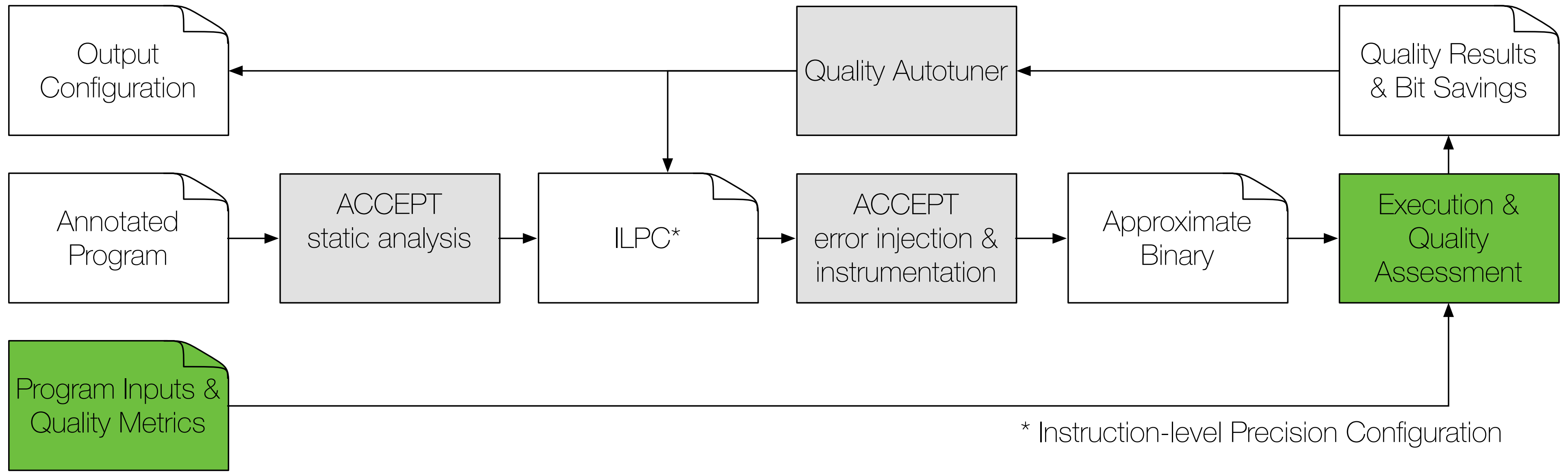
3. Error Injection

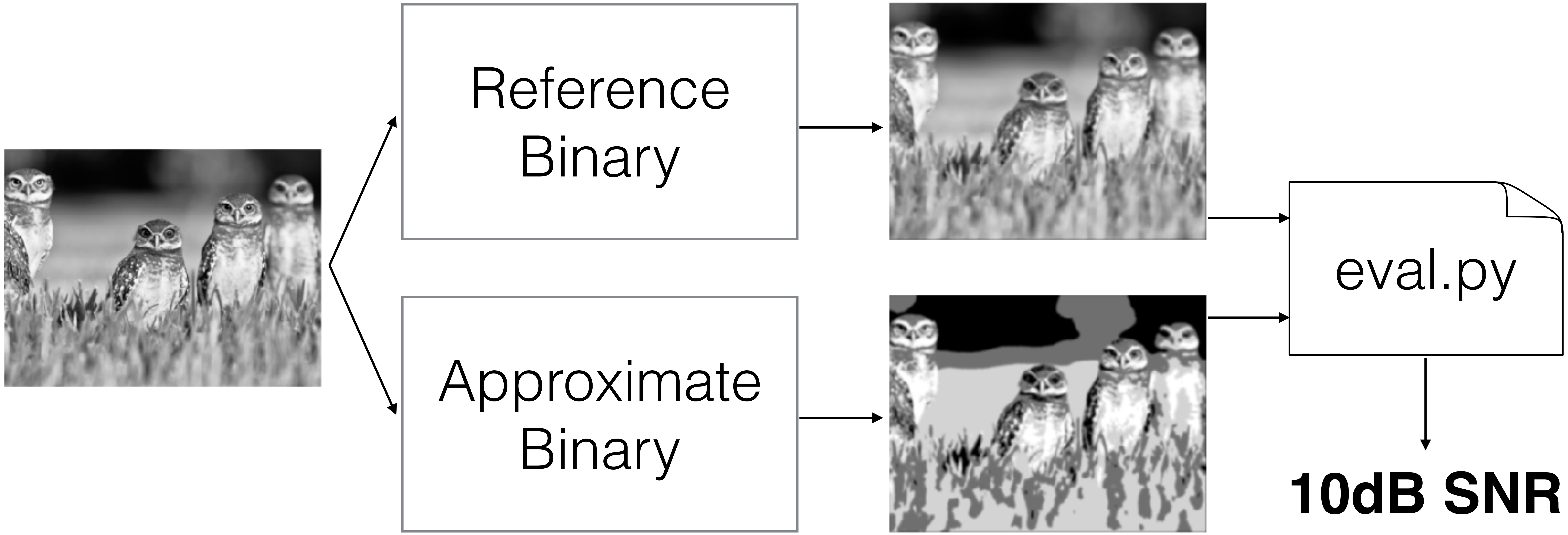




Each instruction in the ILCP acts as a quality knob that the autotuner can use to maximize bit-savings

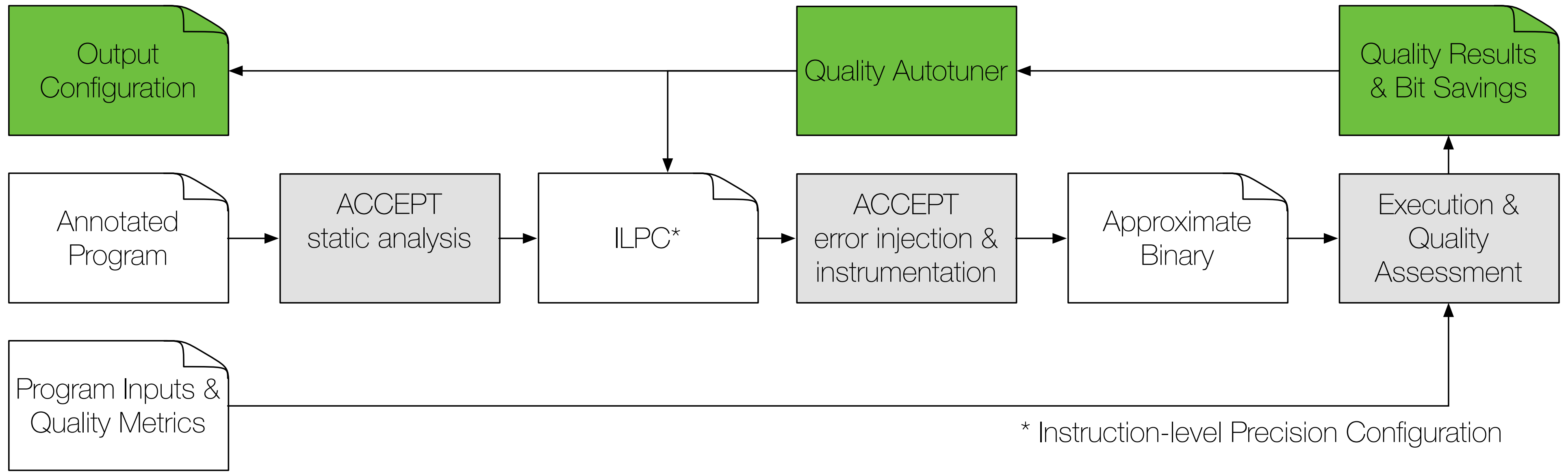
4. Quality Assessment



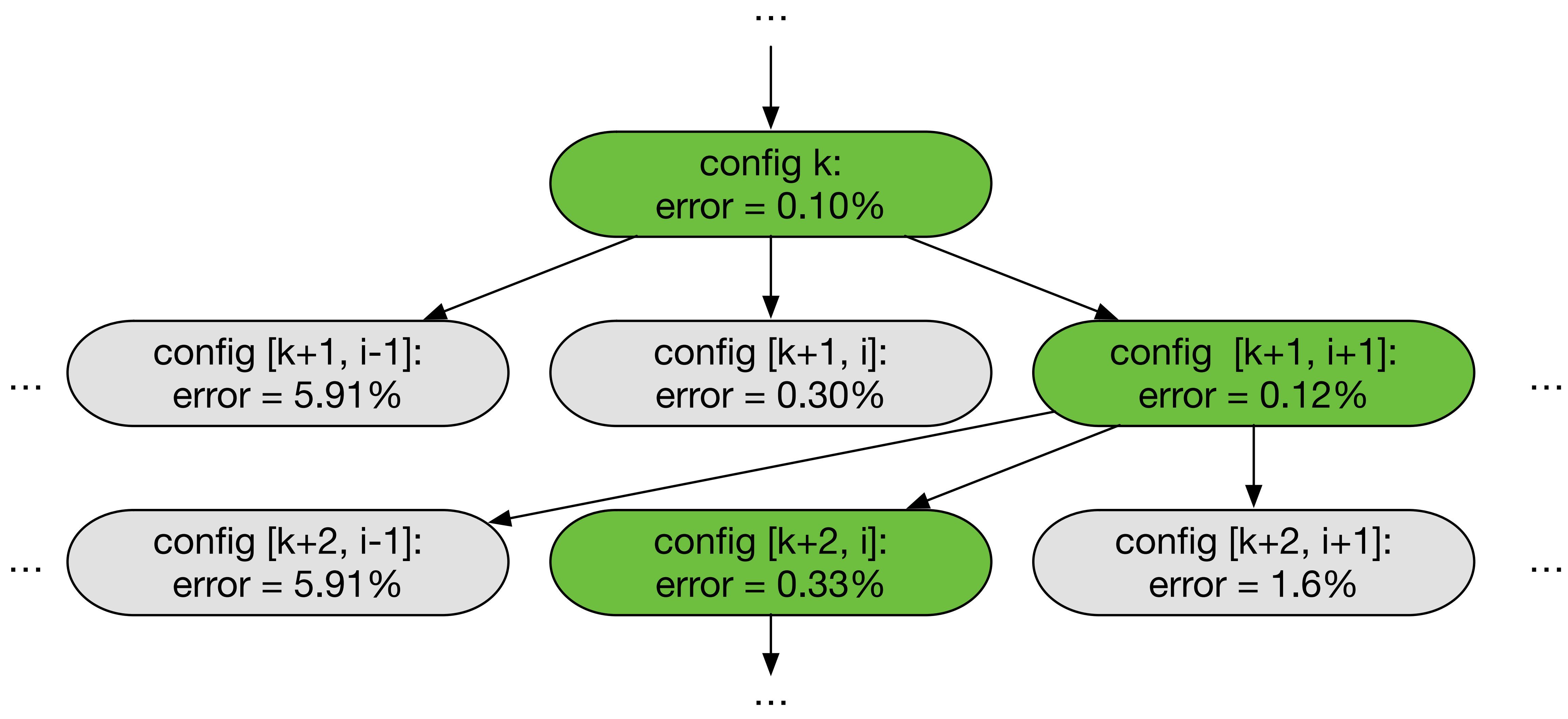


The programmer provides a quality assessment script to evaluate quality on the program output

5. Autotuning Algorithm



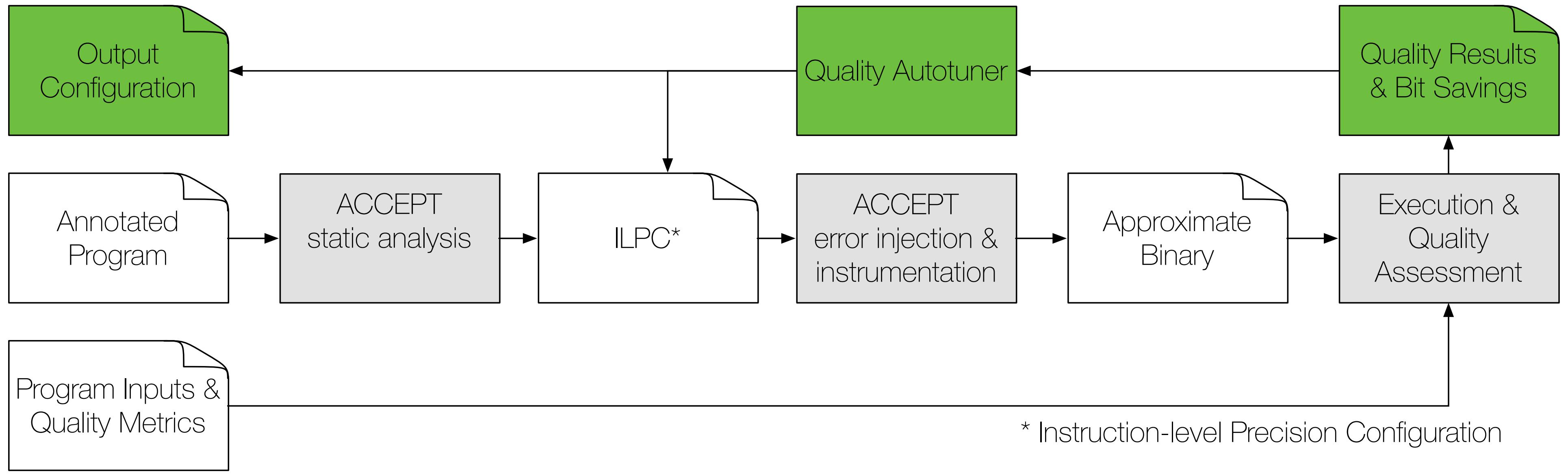
Greedy iterative algorithm [*]: reduces precision requirement of the instruction that impacts quality the least

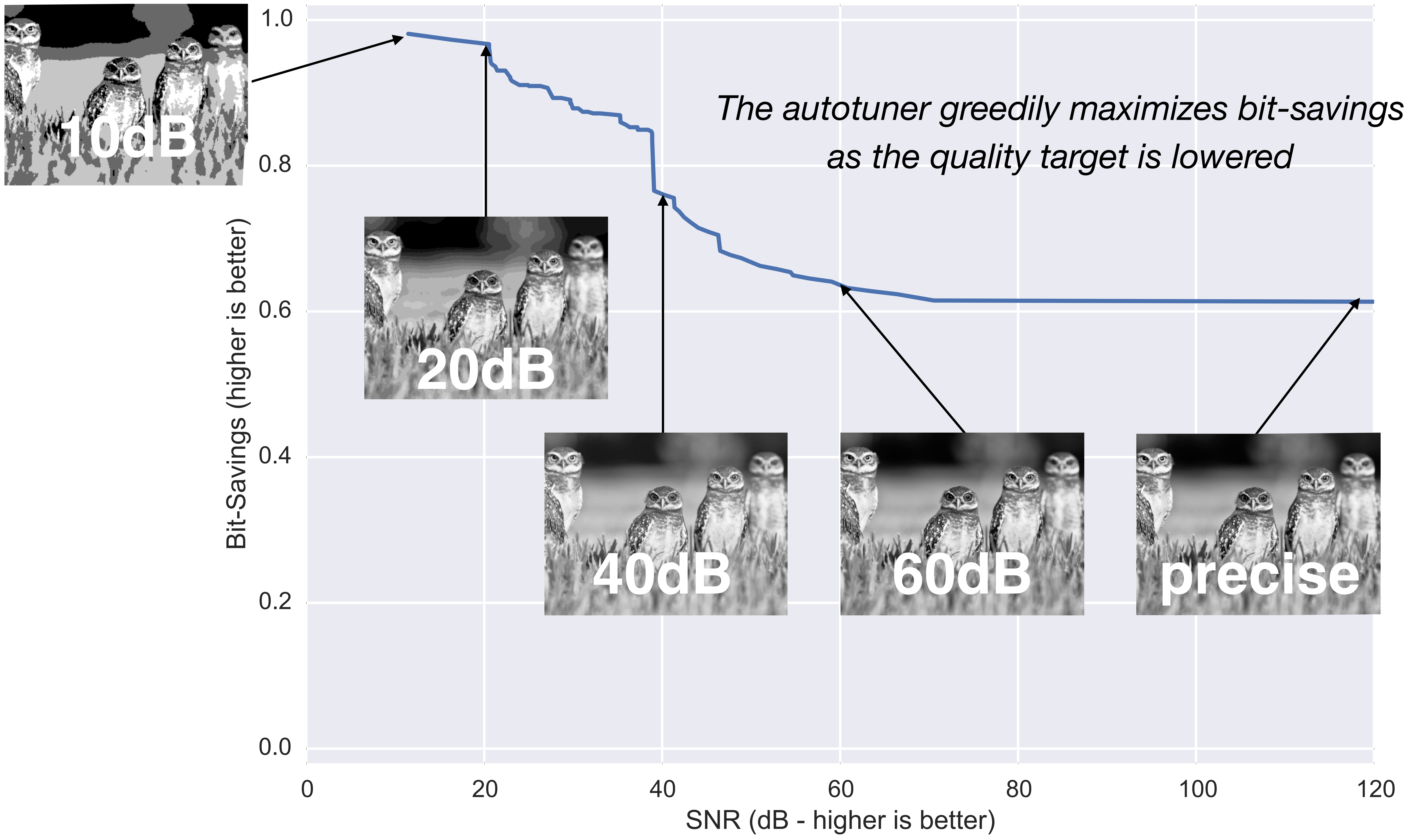


Finds solution in $O(m^2n)$ worst case where m is the number of static safe-to-approximate instructions and n are the levels of precision for all instructions

[*] Precimonious, Rubio-Gonzalez et al., SC'13

5. Autotuning Algorithm



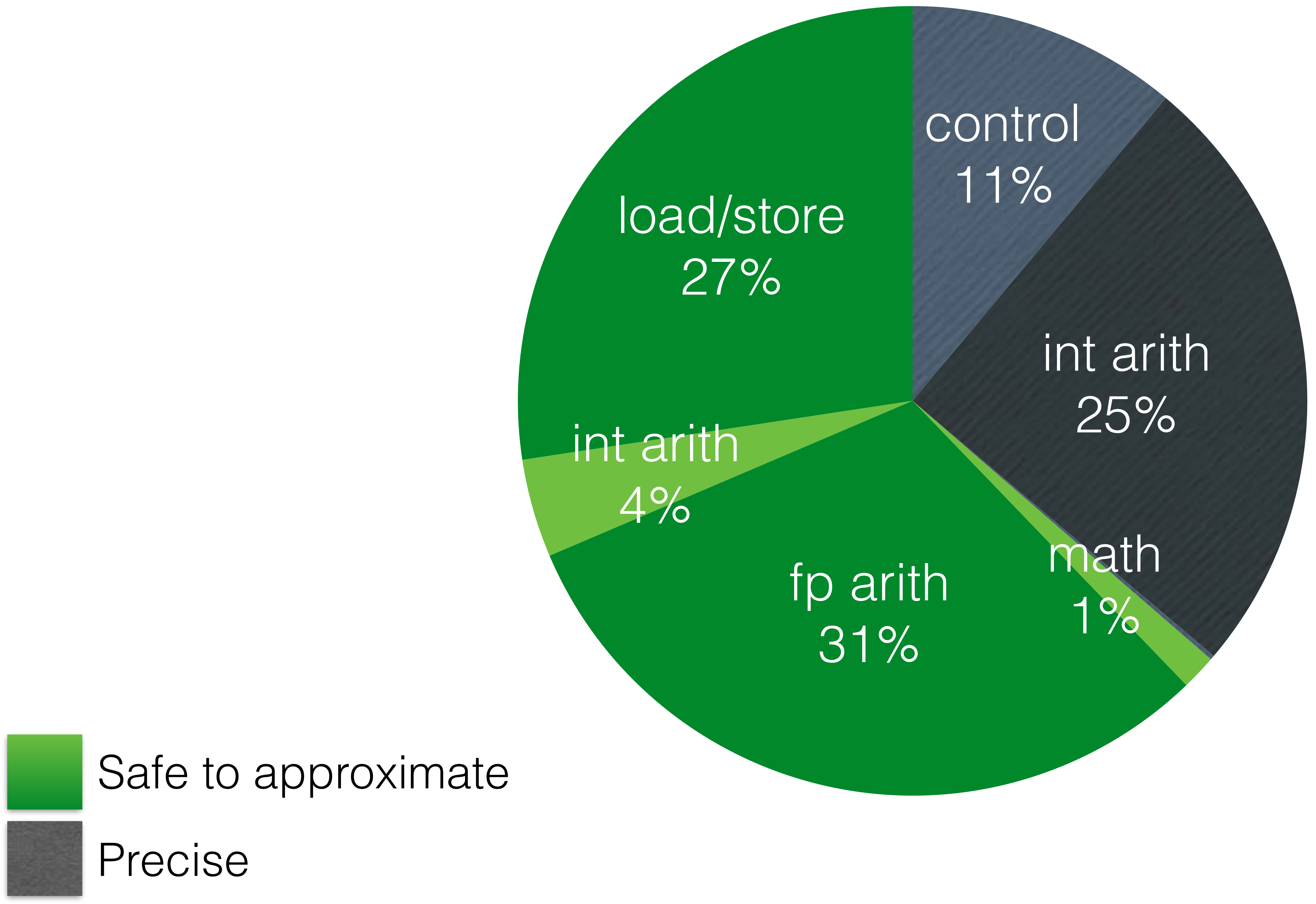


PERFECT Application Study

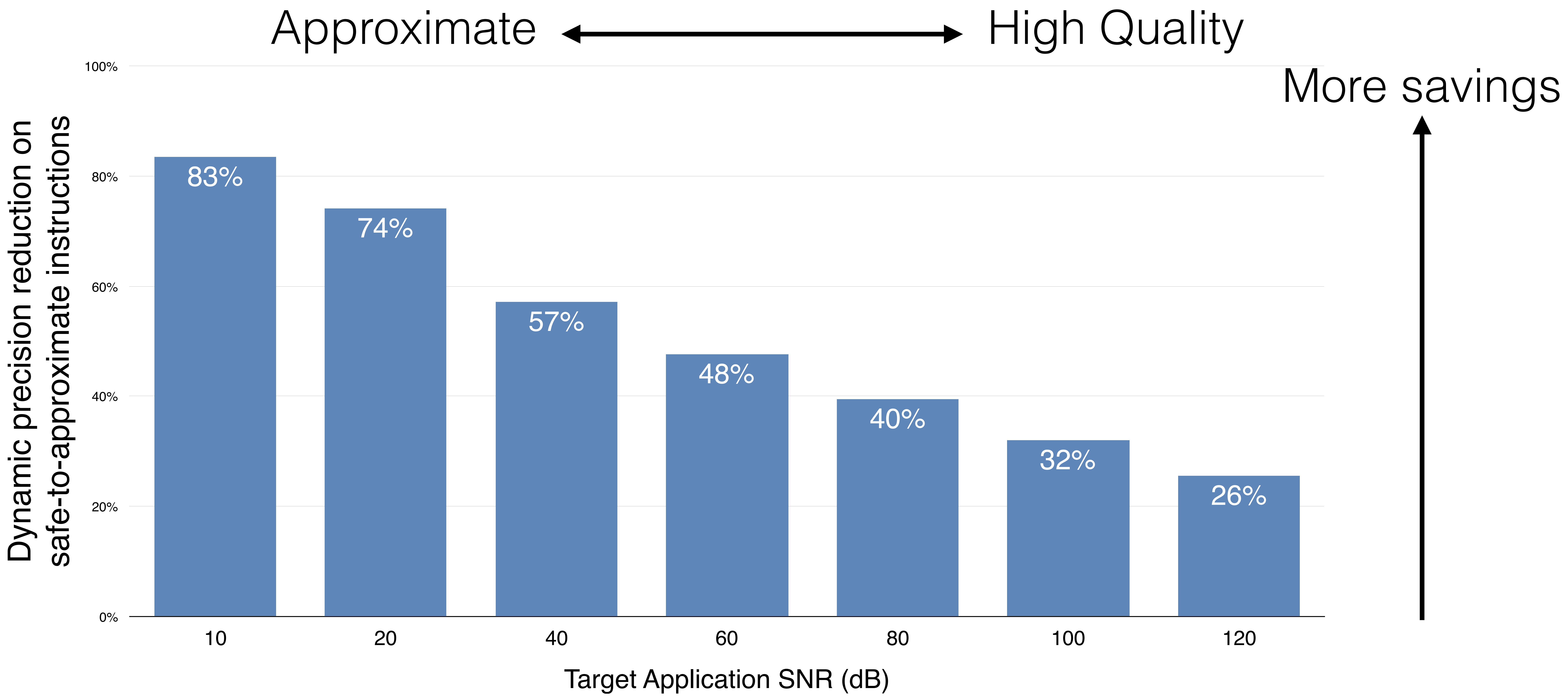
Application Domain	Kernels	Metric
PERFECT Application 1	Discrete Wavelet Transform	**Signal to Noise Ratio (SNR)** [120dB to 10dB] (0.0001% to 31.6% MSE)
	2D Convolution	
	Histogram Equalization	
Space Time Adaptive Processing	Outer Product	
	System Solve	
	Inner Product	
Synthetic Aperture Radar	Interpolation 1	
	Interpolation 2	
	Back Projection	
Wide Area Motion Imaging	Debayer	
	Image Registration	
	Change Detection	
Required Kernels	FFT 1D	
	FFT 2D	

Opportunity of Approximations

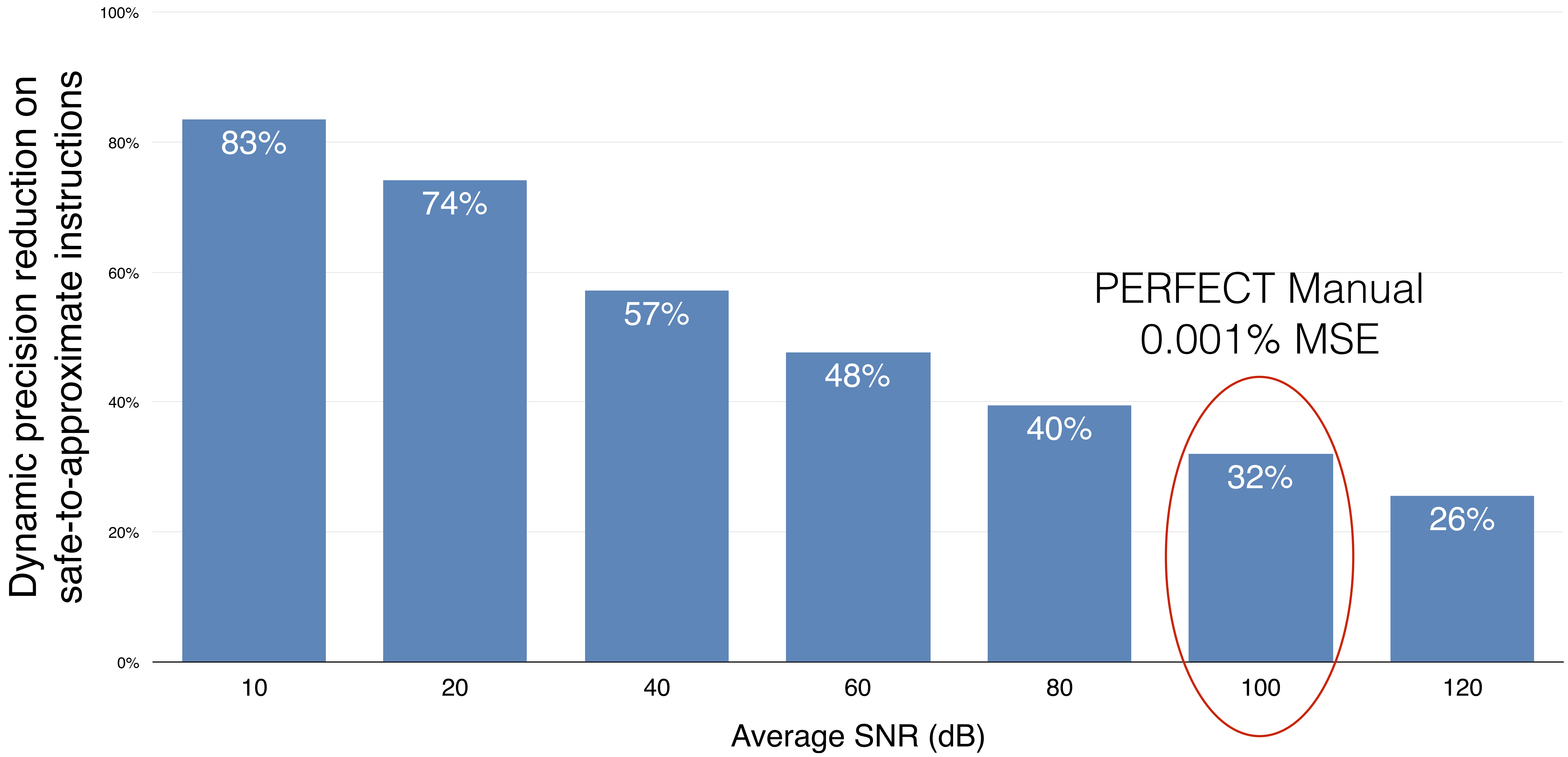
QAPPA Analyzes PERFECT Dynamic Instruction Mix



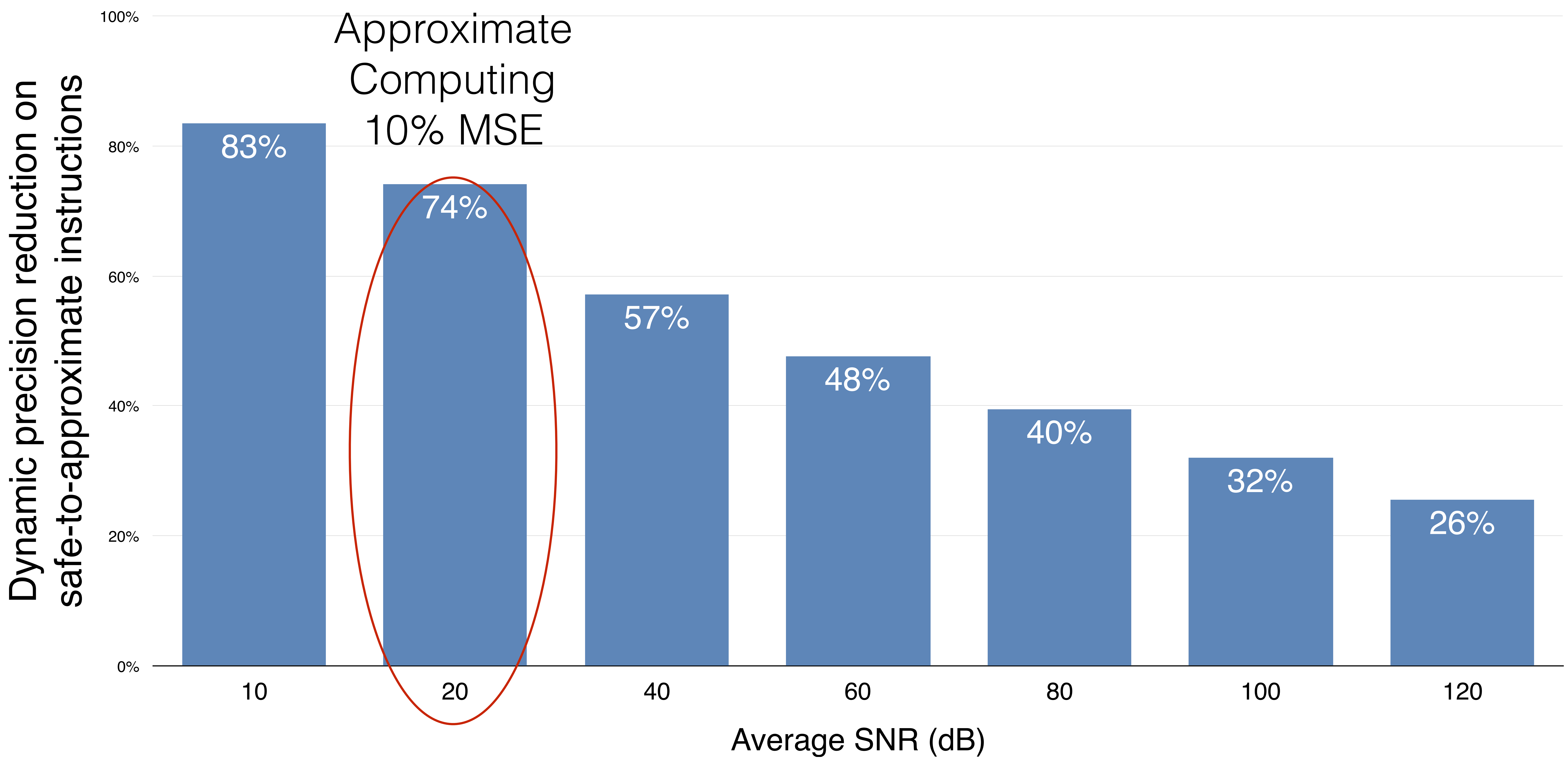
Average Precision Reduction Achieved Across PERFECT Kernels



Average Precision Reduction Achieved Across PERFECT Kernels



Average Precision Reduction Achieved Across PERFECT Kernels



Talk Overview

1. How much precision is needed at different stages of a program?

QAPPA - Precision Autotuner

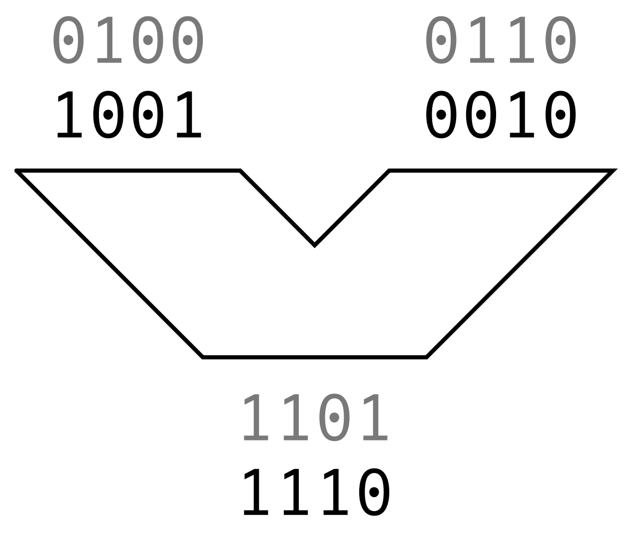
2. How much energy can be saved (upper bound)?

Case Study of Precision Scaling Hardware Mechanisms

3. How does this inform approximate computing research?

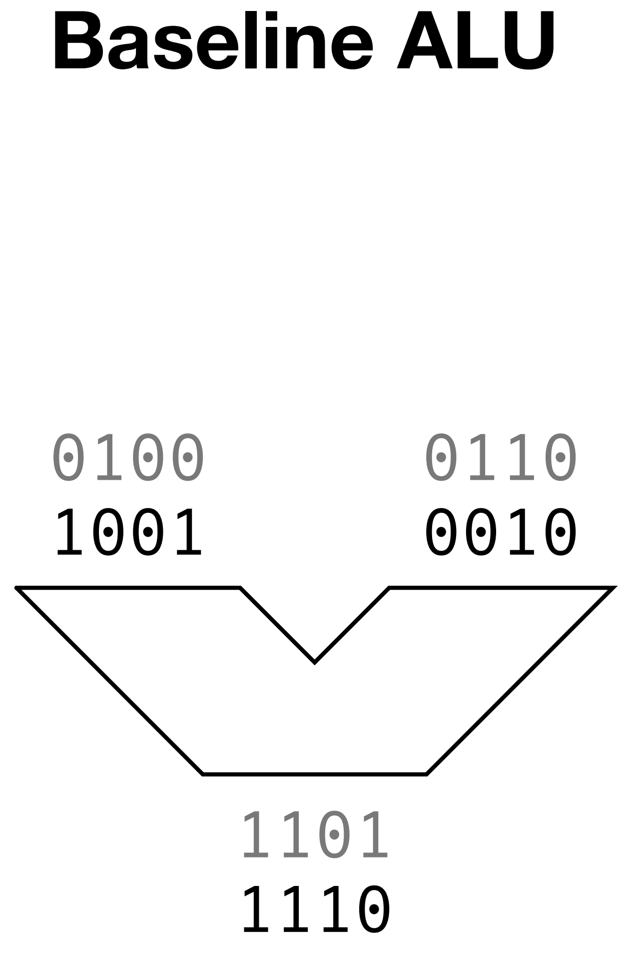
Translating Precision Reduction into Energy Savings (Compute)

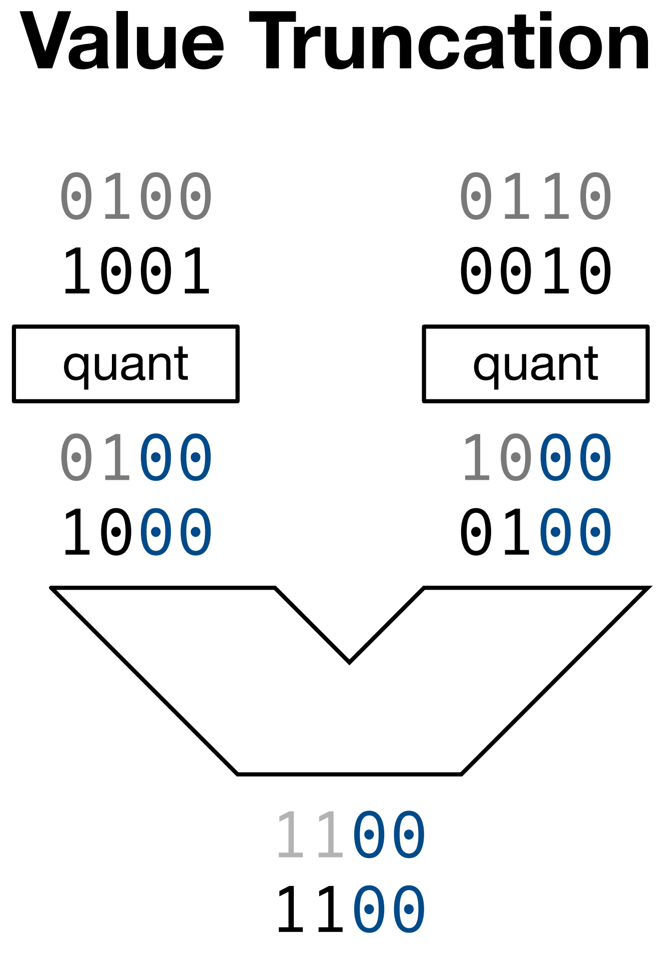
Baseline ALU



No savings

Translating Precision Reduction into Energy Savings (Compute)



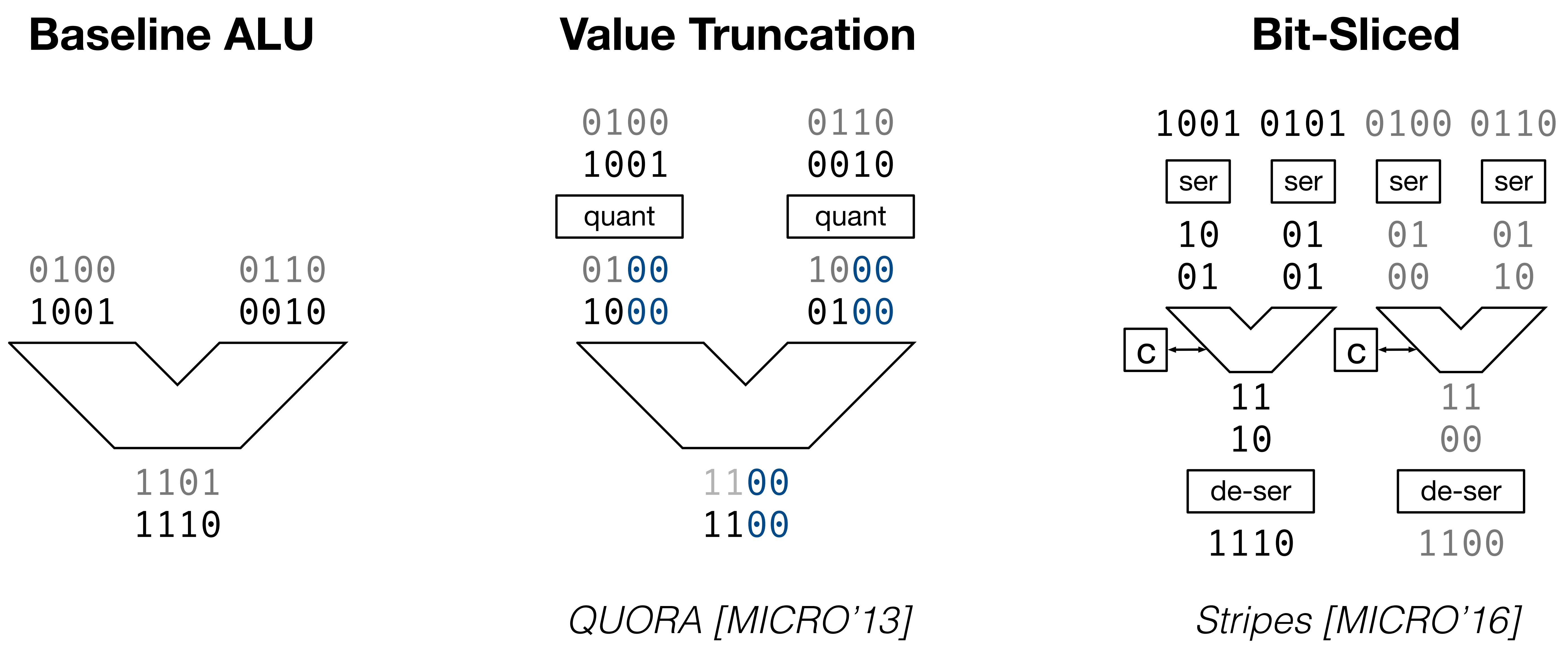


QUORA [MICRO'13]

No savings

Less Power

Translating Precision Reduction into Energy Savings (Compute)



No savings *Less Power* *Higher Throughput*

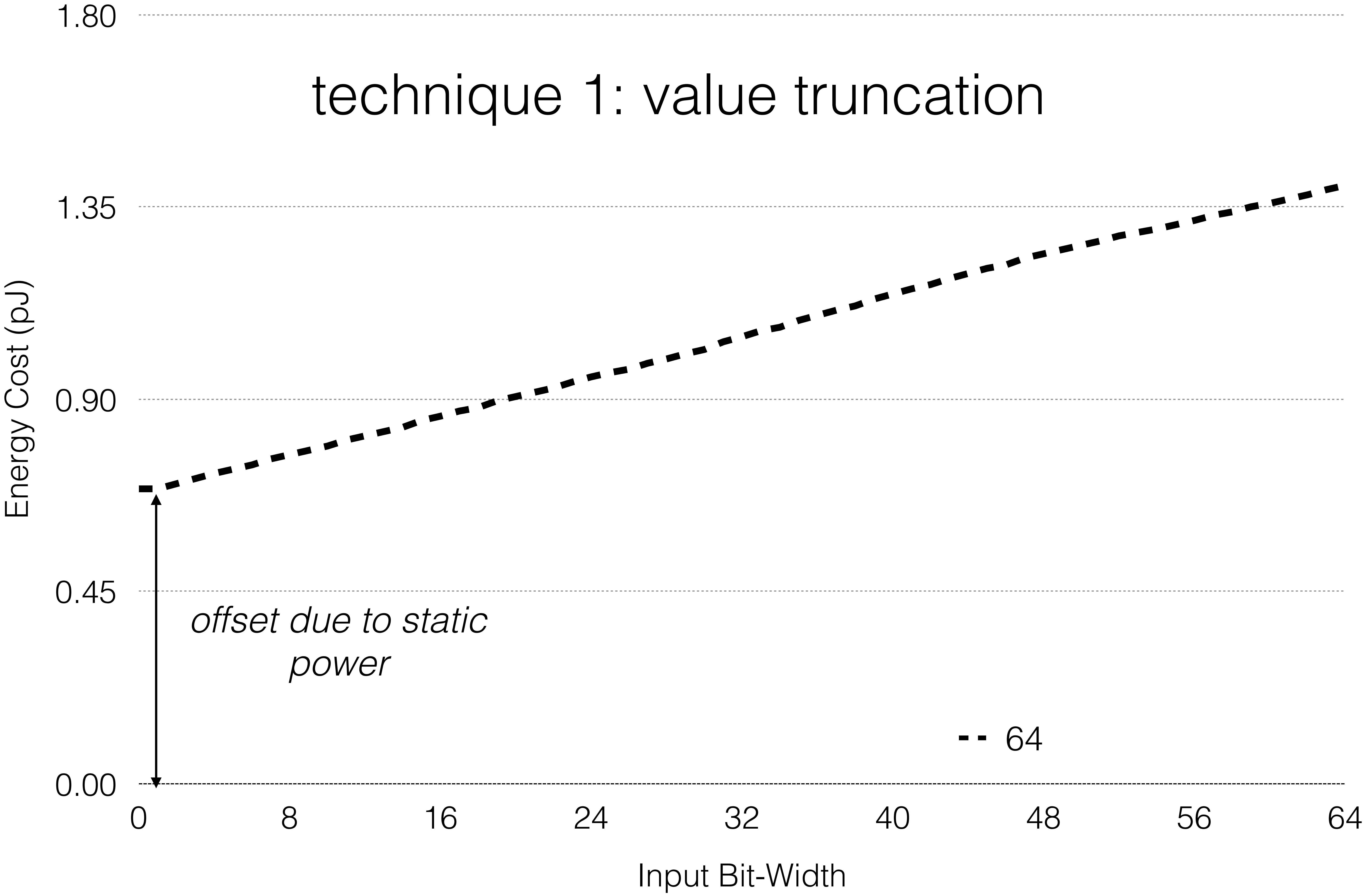
Case Study: Precision Scaled Adder

Goal: Design an precision scalable adder that can elegantly trade lower precision for energy savings

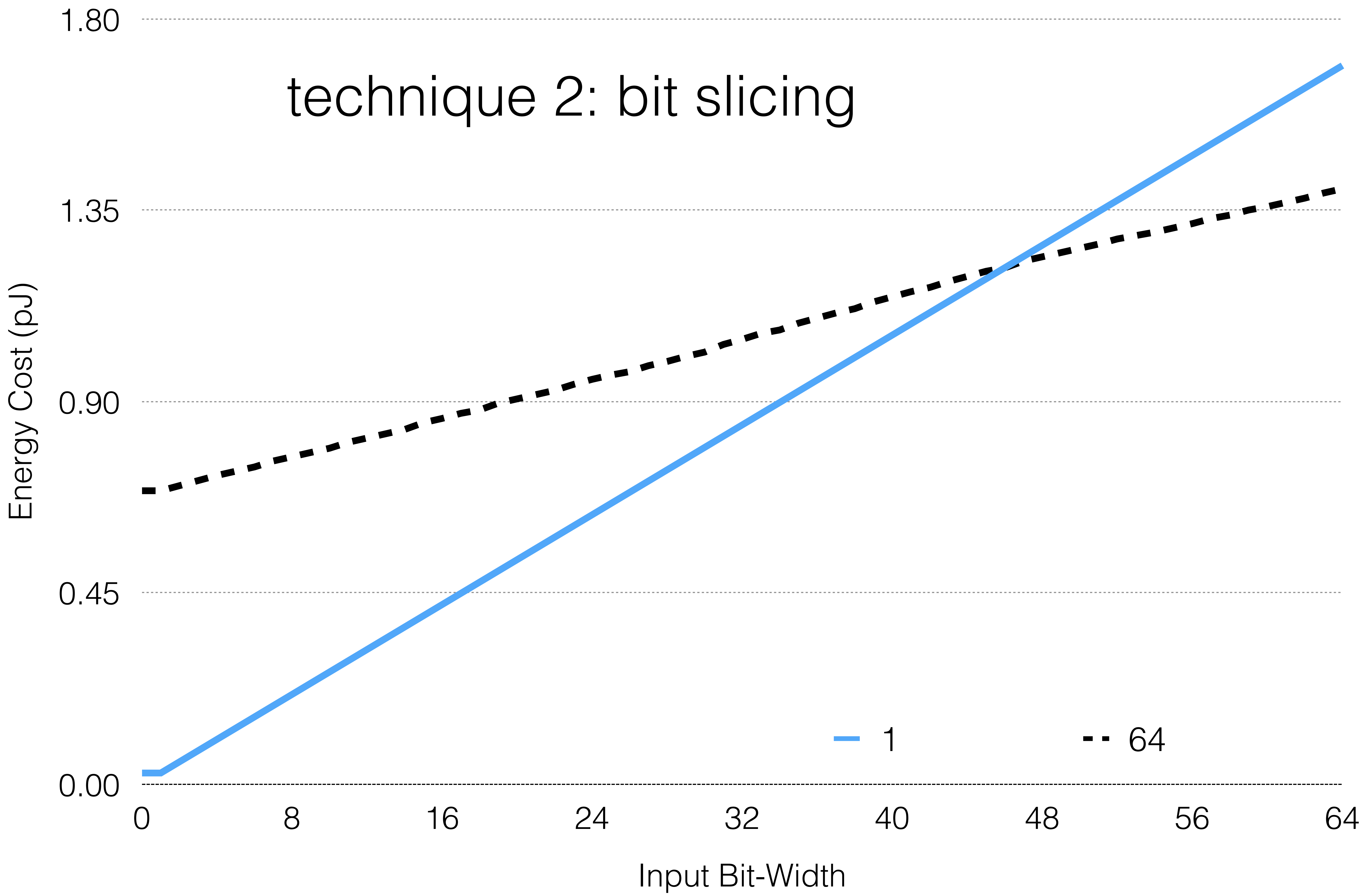
Exploration: Combine value truncation and bit slicing techniques, and vary the slice width in increments of powers of 2

Methodology: Post-place-and-route prime-time power analysis on 65nm TSMC library

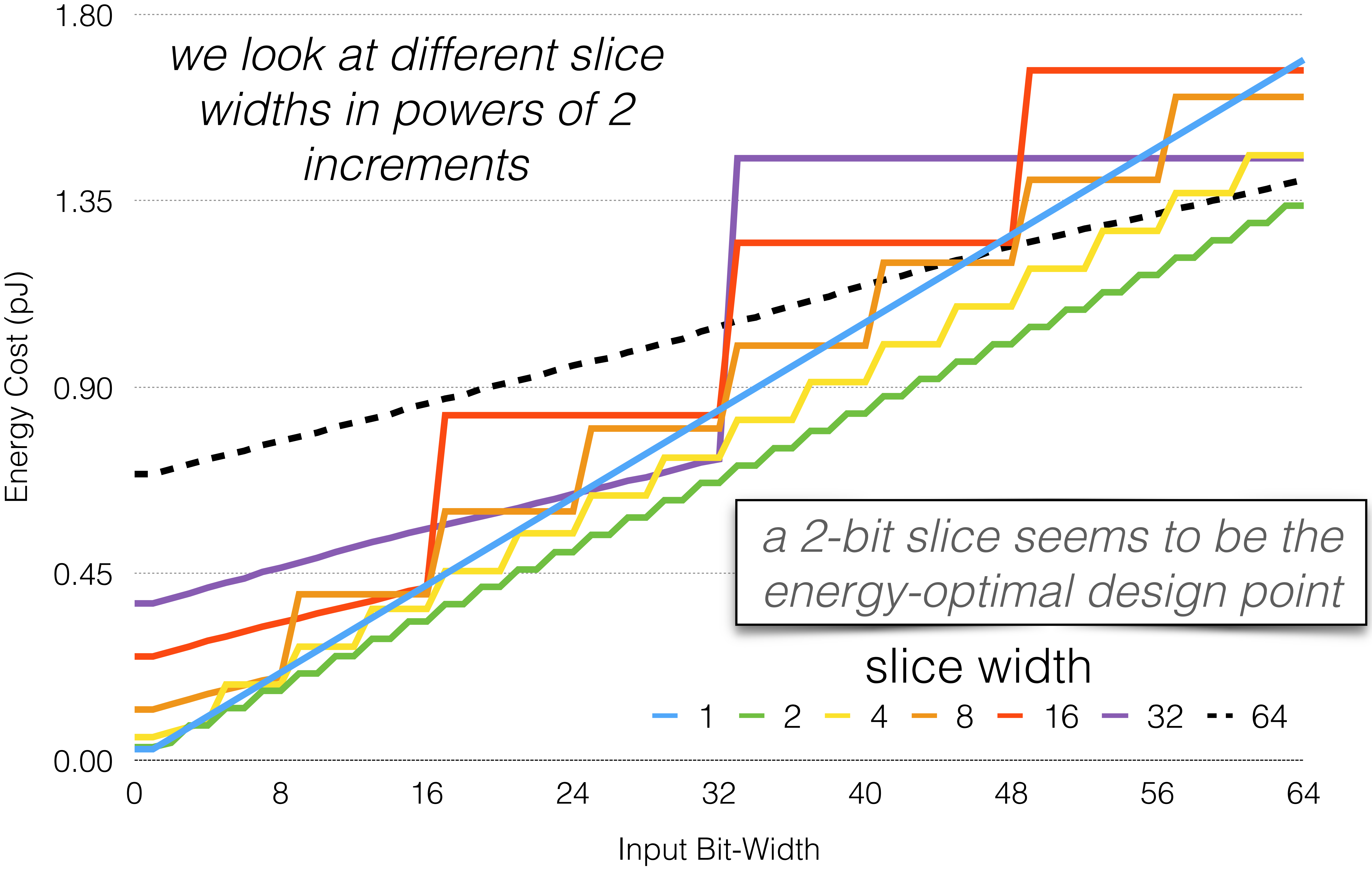
Precision Scaled Adder Study



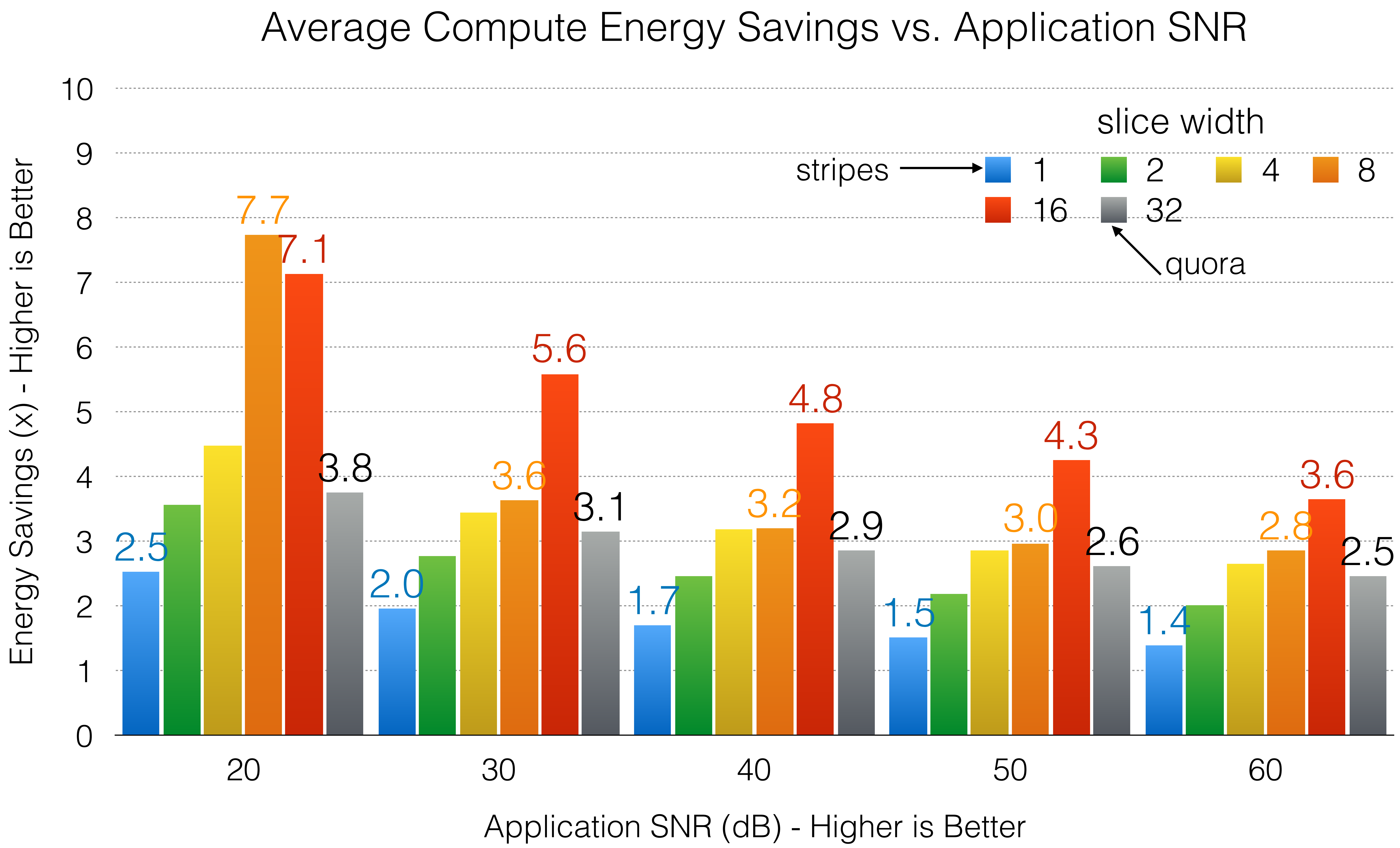
Precision Scaled Adder Study



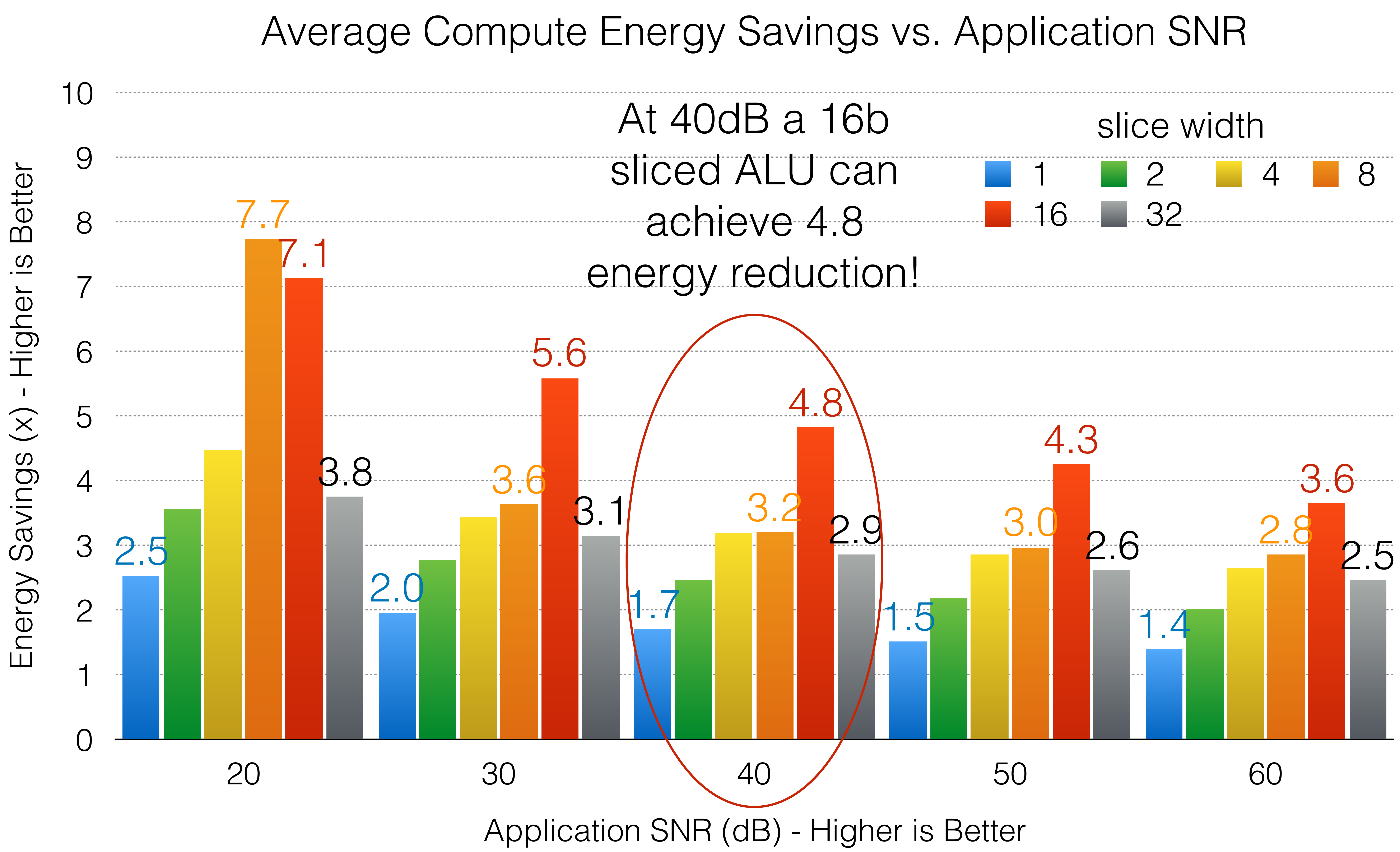
Case Study: Precision-Scaled Adder



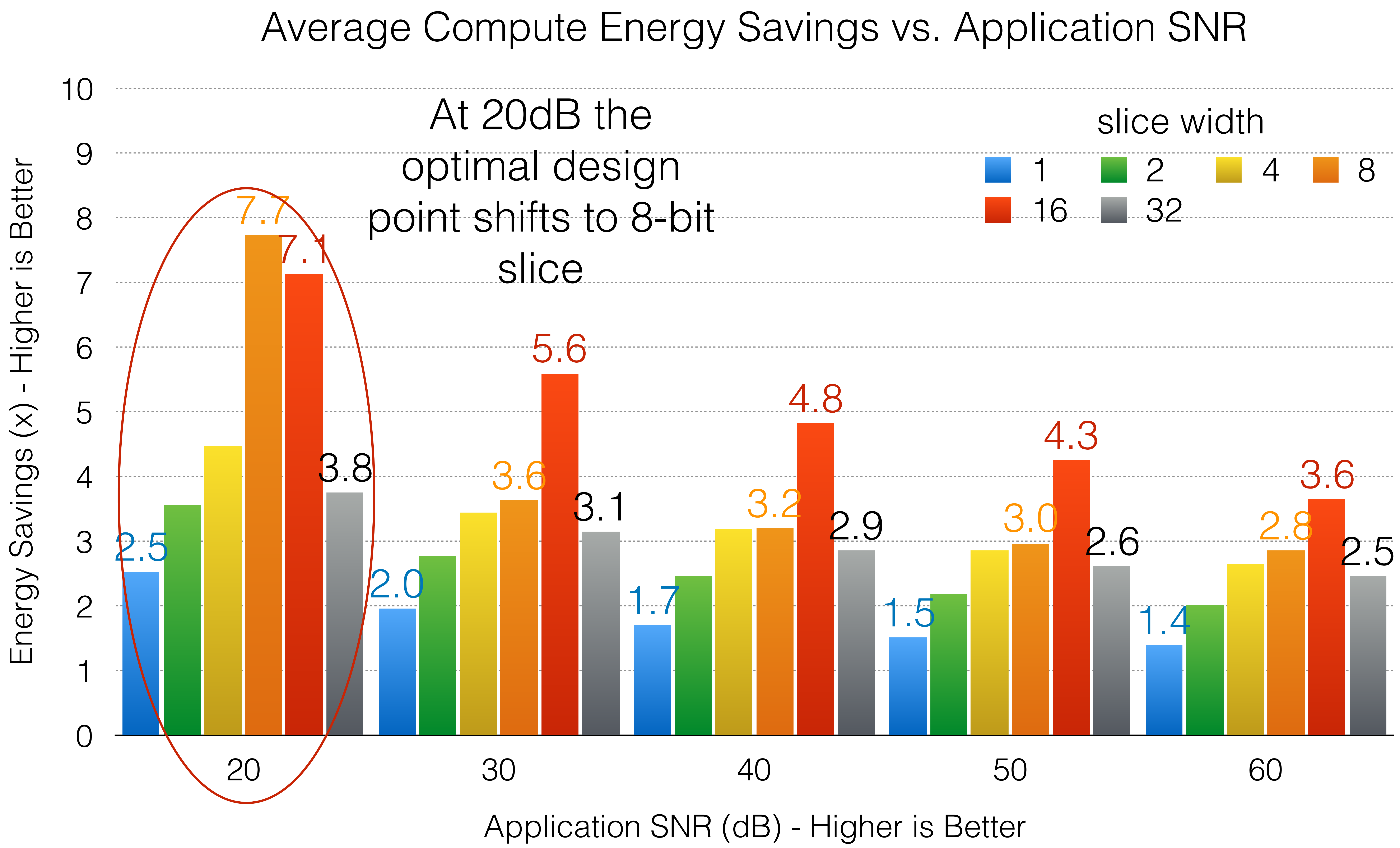
PERFECT Study: Compute Energy Savings



PERFECT Study: Compute Energy Savings



PERFECT Study: Compute Energy Savings



Talk Overview

1. How much precision is needed at different stages of a program?

QAPPA - Precision Autotuner

2. How much energy can be saved (upper bound)?

Case Study of Precision Scaling Hardware Mechanisms

3. How does this inform approximate computing research?

Comparative Study of Approximation Techniques

Comparative Study

Many papers on approximate computing state:
"Our technique provided n times speedup at x% error"

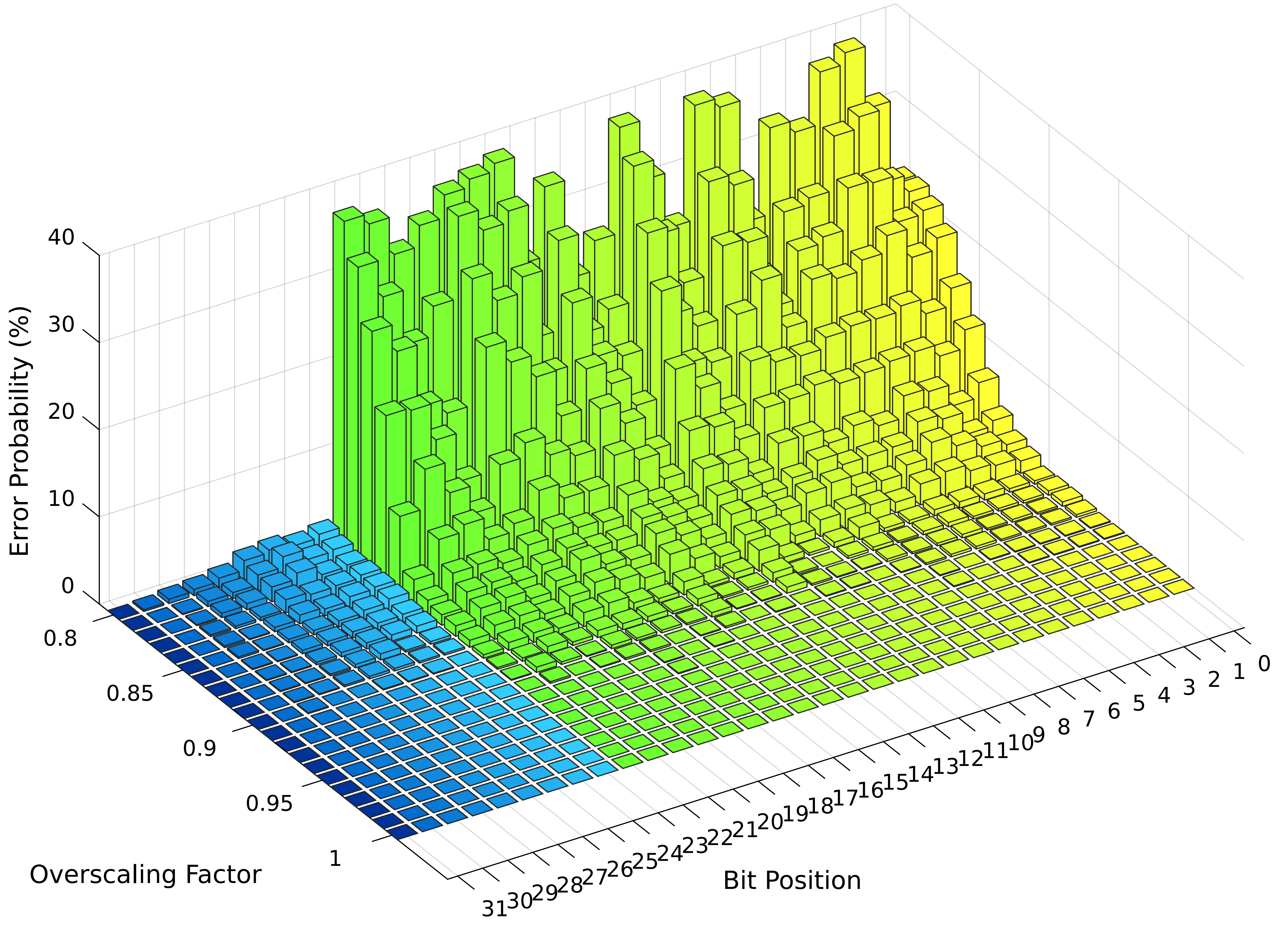
Problem: This give us a data point but doesn't quite say much about the merits of the technique at trading accuracy for efficiency

Solution: Use QAPPA to produce quick comparison results to assess effectiveness of technique

Comparative Study - Voltage Overscaling

<u>Methodology (1/2)</u>: Spice simulation of ALU/FPU design under different voltage overscaling factors.

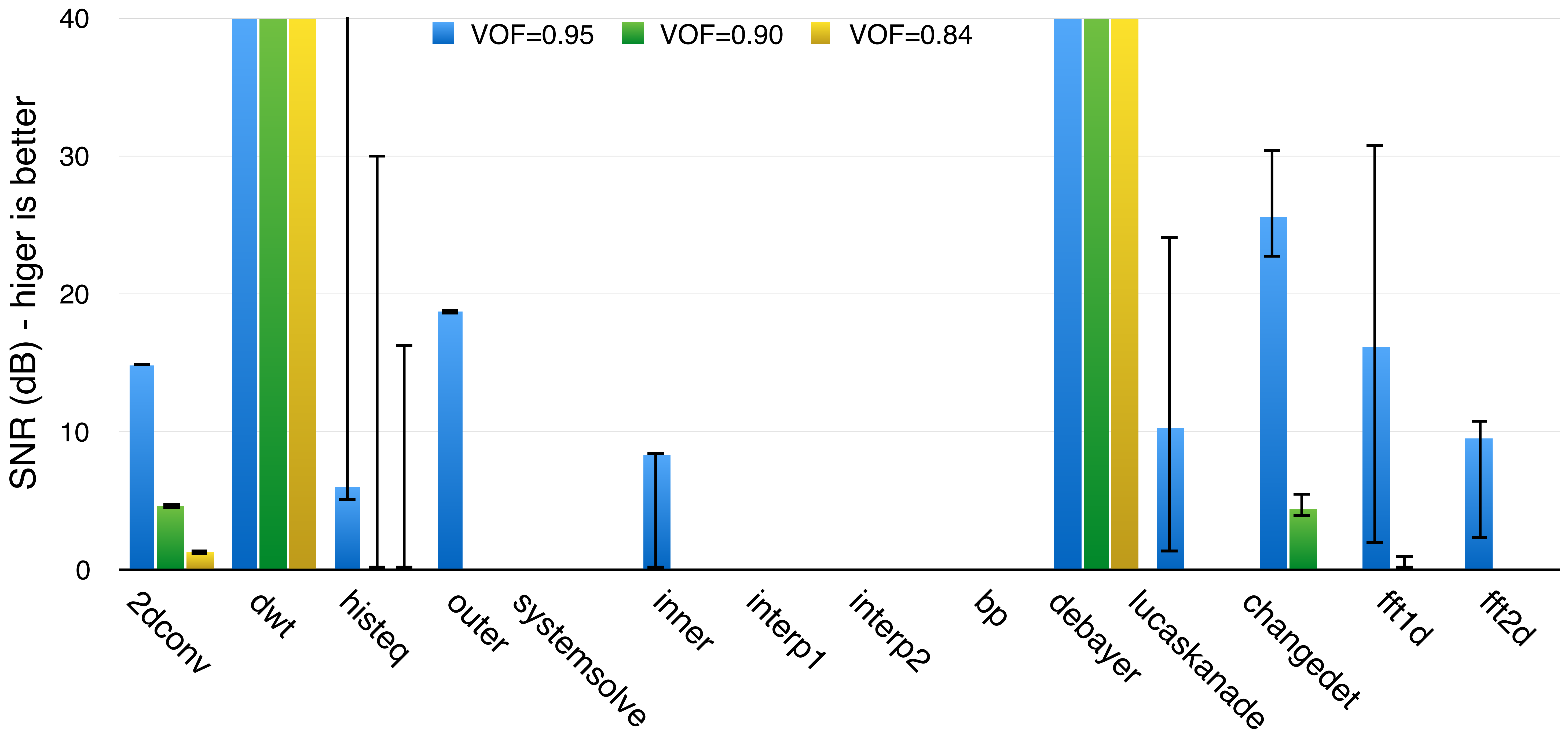
fp adder example



Comparative Study - Voltage Overscaling

Methodology (2/2): Then we feed the error model into QAPPA's error injection framework to assess application error.

Results: Precision scaling always produces better quality/efficiency



Future Directions in Architecture/CAD

Precision Scaling Architectures: Need to see more precision-scaled accelerators for more applications of the likes of Quora[MICRO'13], Stripes[MICRO'16]

CAD tools with Quality Awareness: Need to see more tools that can leverage quantization, especially in the FPGA community, of the likes of AHLS[DATE'17]

Conclusion

1. How much precision is needed at different stages of a program?

QAPPA - Precision Autotuner

2. How much energy can be saved (upper bound)?

Case Study of Precision Scaling Hardware Mechanisms

3. How does this inform approximate computing research?

Comparative Study of Approximation Techniques

Thank you!

Exploiting Quality-Efficiency Tradeoffs with Arbitrary Quantization

Special Session - CODES+ISSS

Thierry Moreau, Felipe Augusto, Patrick Howe
Armin Alaghi, Luis Ceze