Approximating to the Last Bit

Thierry Moreau, Adrian Sampson, Luis Ceze
{moreau, luisceze}@cs.washington.edu, asampson@cornell.edu

WAX 2016
co-located with ASPLOS 2016
April 3rd 2016

What this Talk is About

How many bits in a program are really that important?

1 - AXE: Quality Tuning Framework

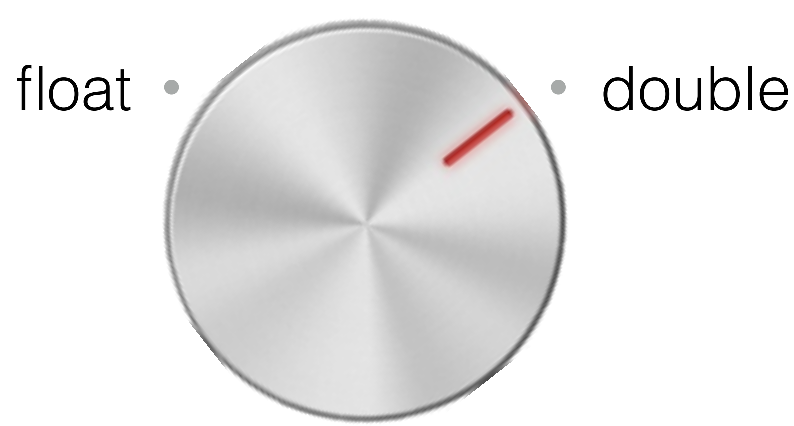
2 - PERFECT Benchmark Study

Precision Tuning

*More **precision** means larger **memory footprint**, more **data movement**, more energy used in **computation***

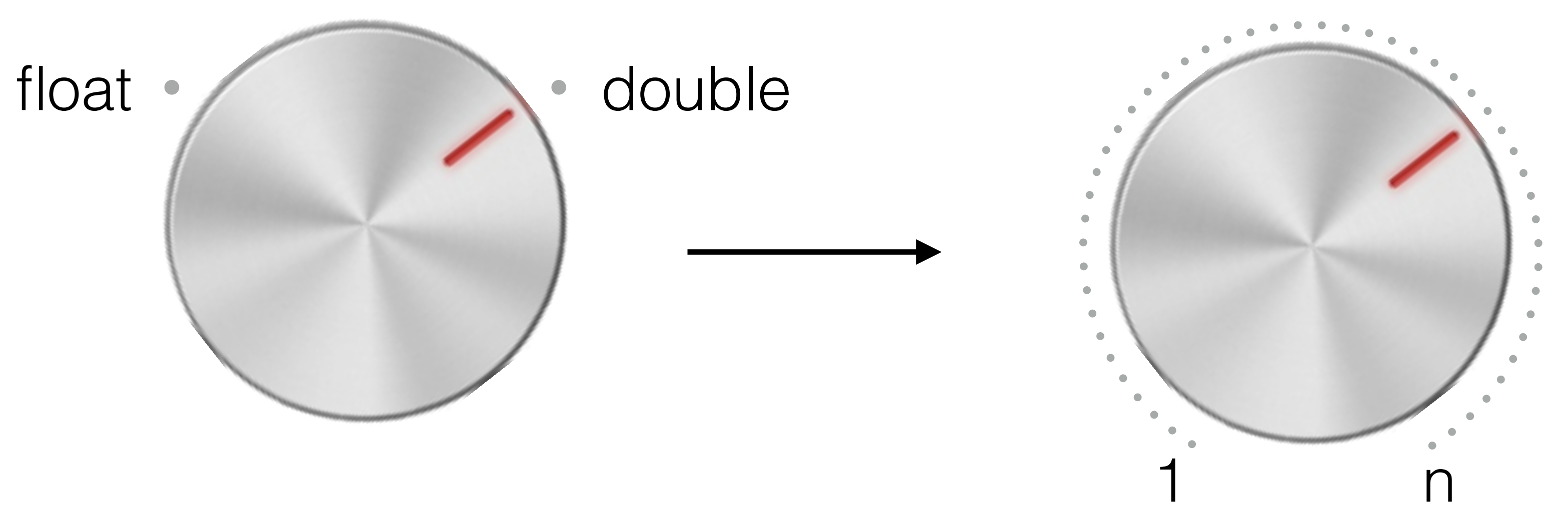
Precision Tuning

*More **precision** means larger **memory footprint**, more **data movement**, more energy used in **computation***



Precision Tuning

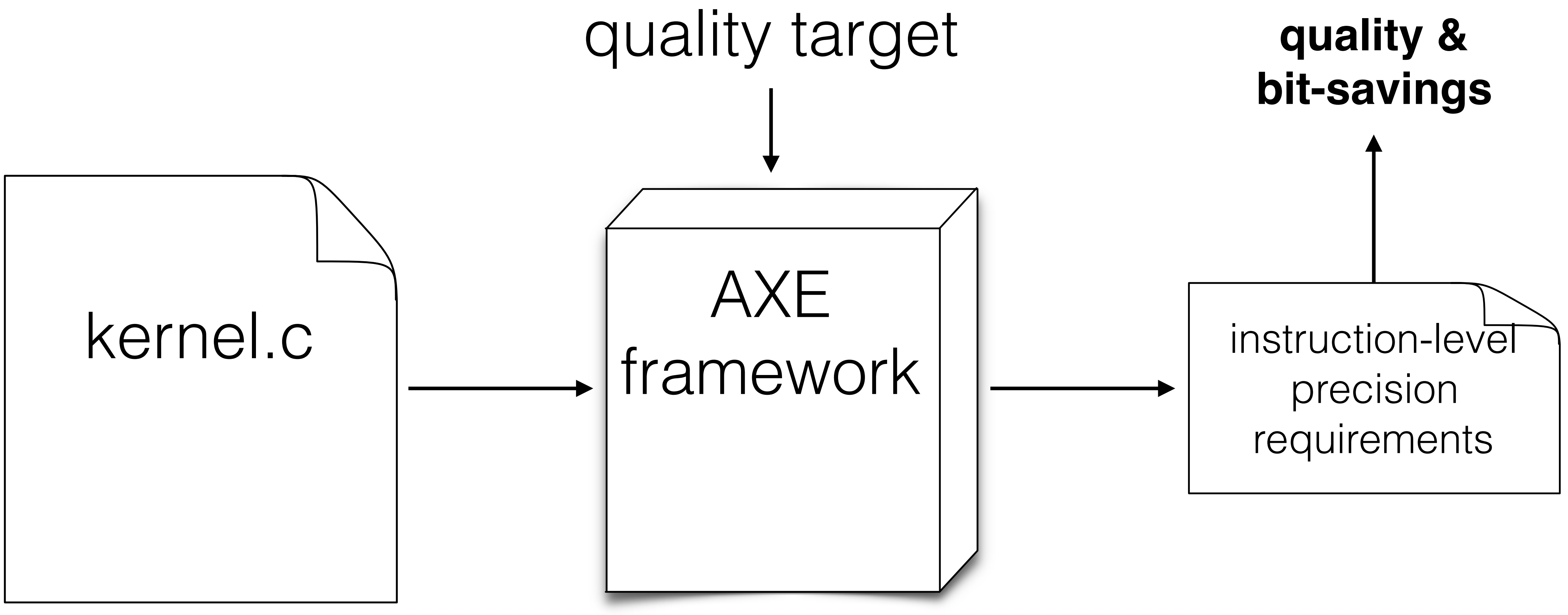
*More **precision** means larger **memory footprint**, more **data movement**, more energy used in **computation***



AXE Precision Tuning Framework

Goal: Maximize Bit-Savings given a Quality Target

AXE Precision Tuning Framework

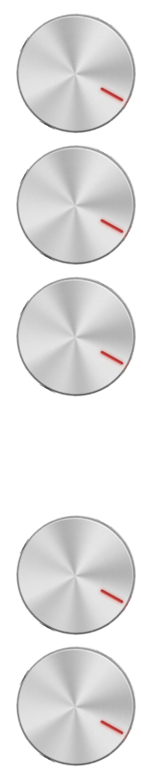


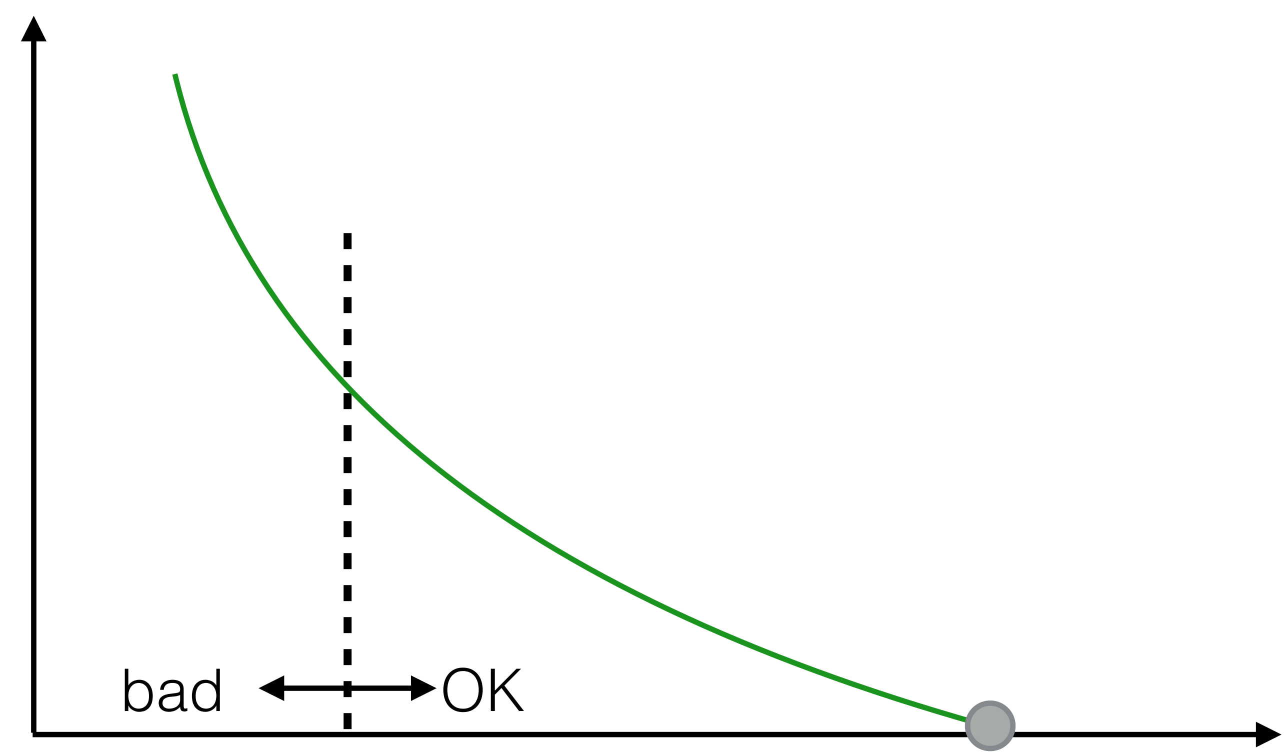
Built on top of **ACCEPT**, the approximate C/C++ compiler
http://accept.rocks

AXE Precision Tuning Framework

Default (no bit-savings)

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

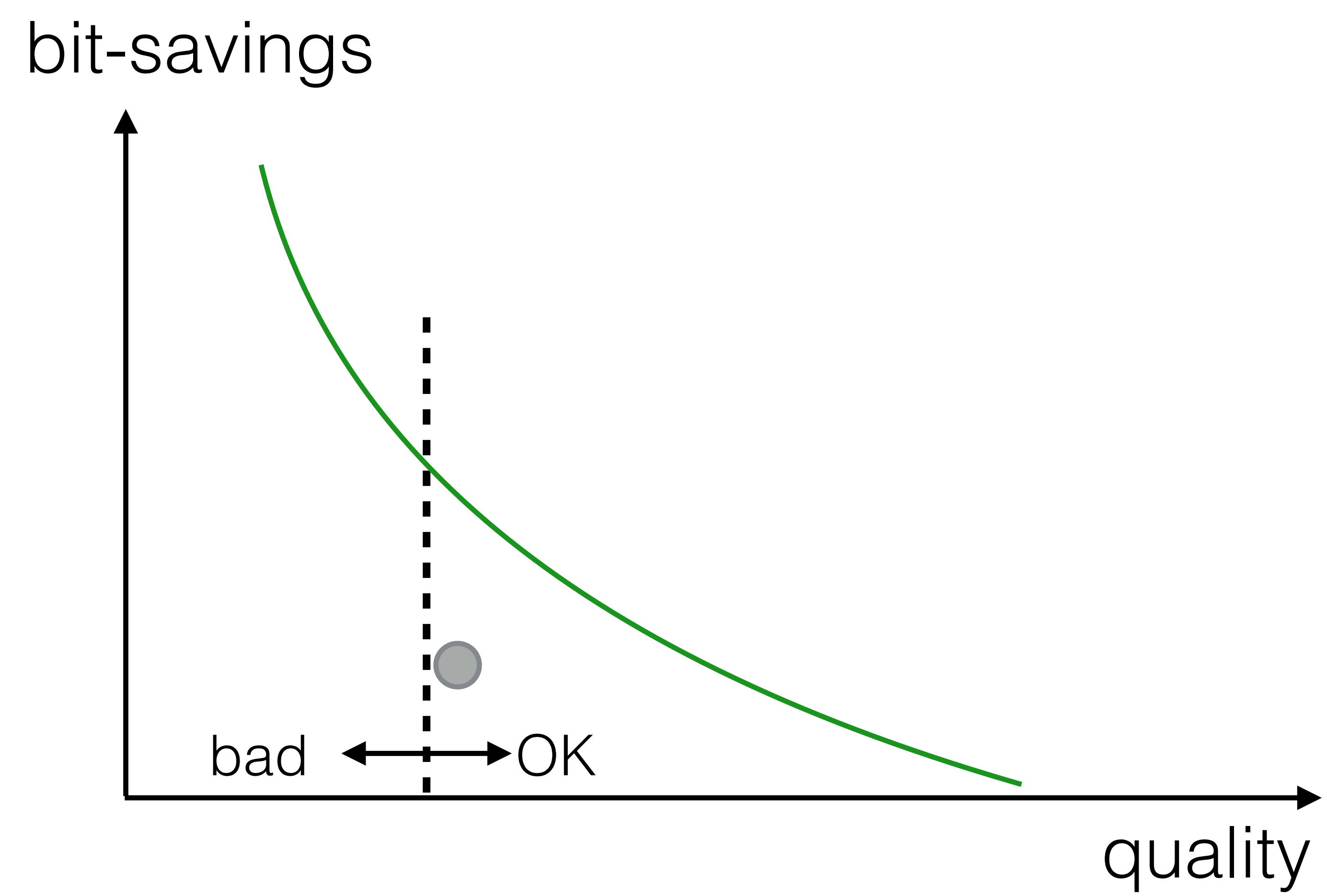




AXE Precision Tuning Framework

Coarse-Grained
Precision Reduction

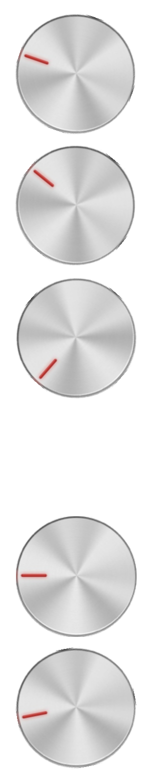
```
instruction 0
instruction 1
instruction 2
...
instruction n-1
instruction n
```

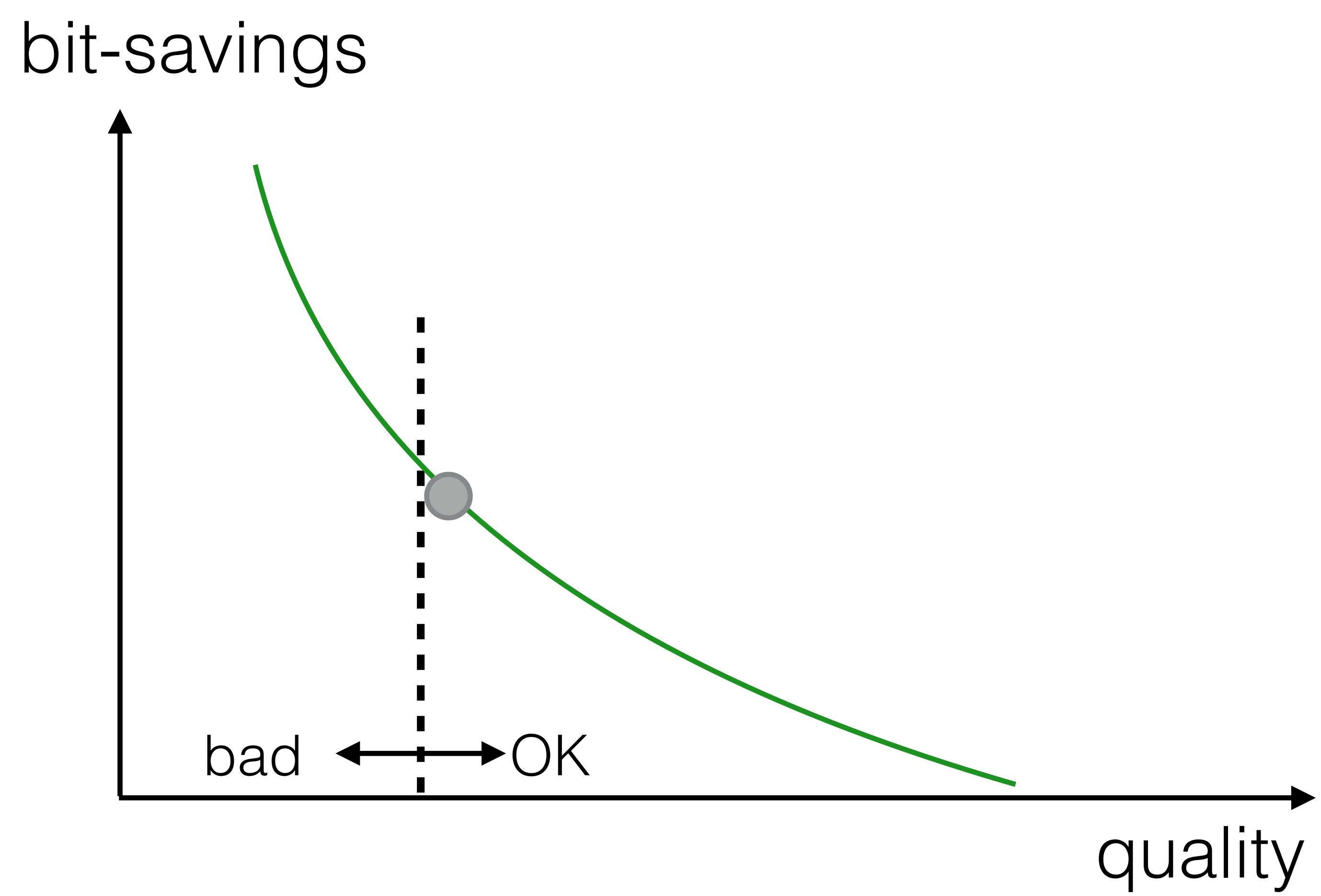


AXE Precision Tuning Framework

Fine-Grained Precision Reduction

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

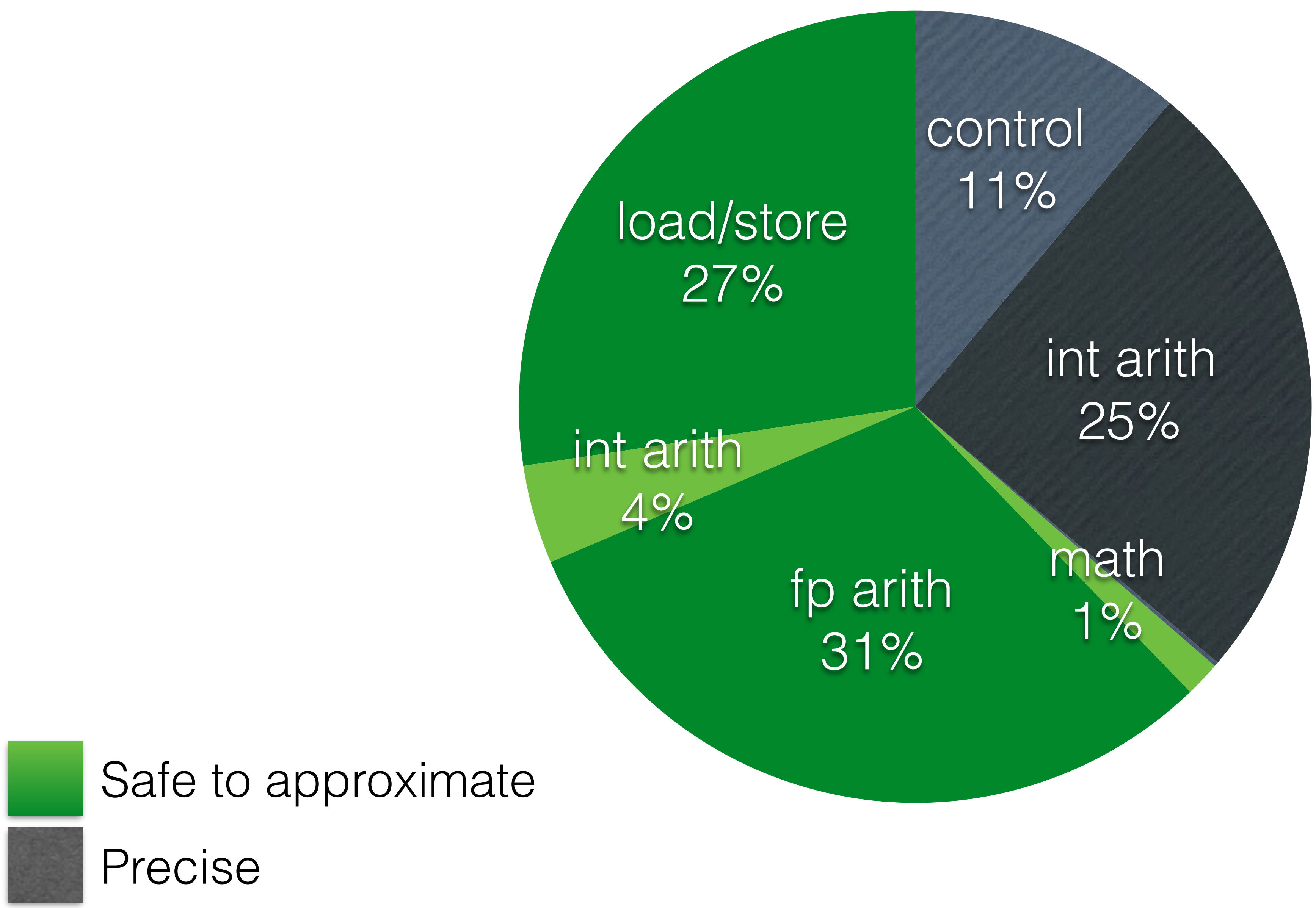




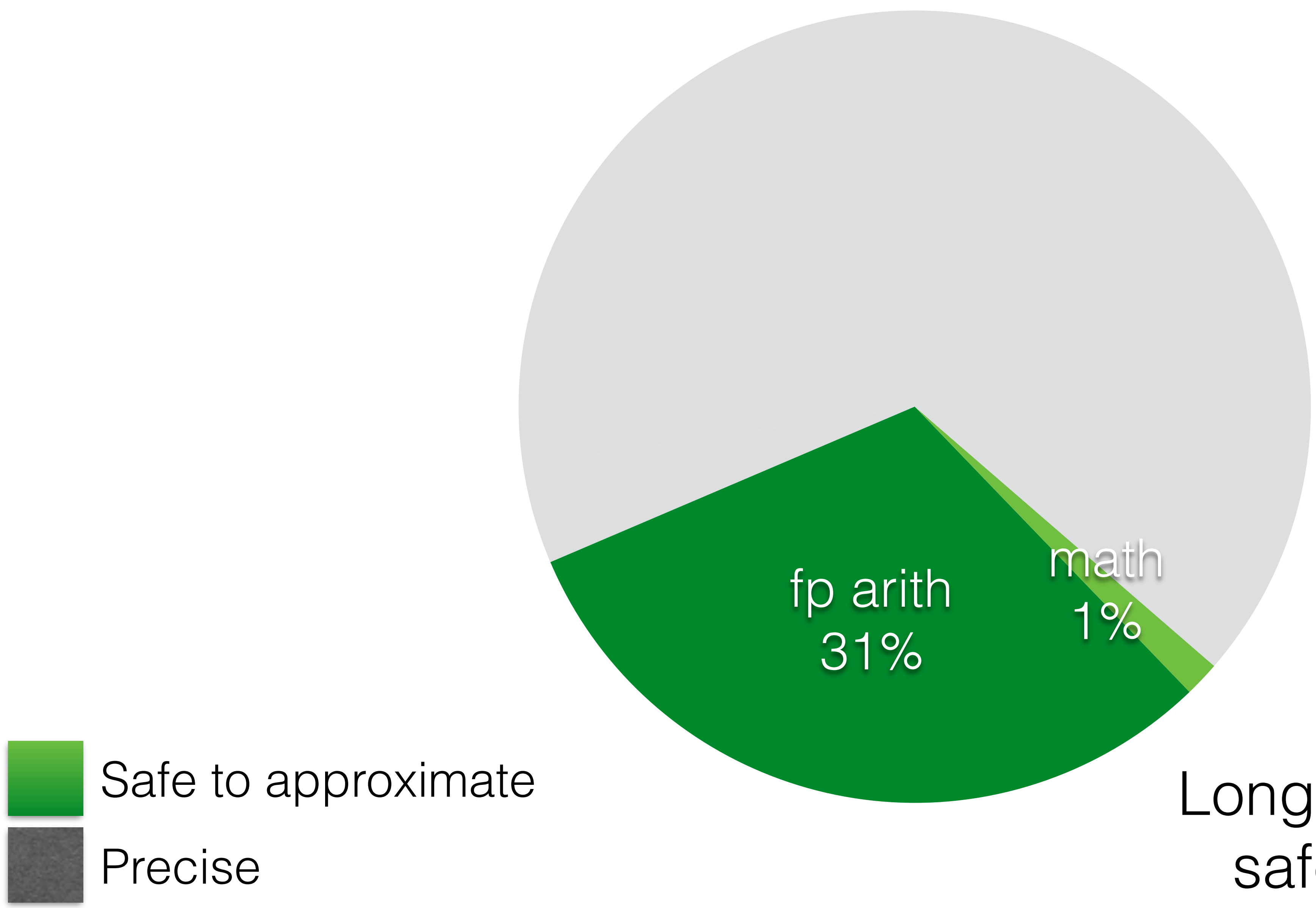
PERFECT Benchmark Suite

Application Domain	Kernels	Metric
PERFECT Application 1	Discrete Wavelet	**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	

1 - PERFECT Dynamic Instruction Mix



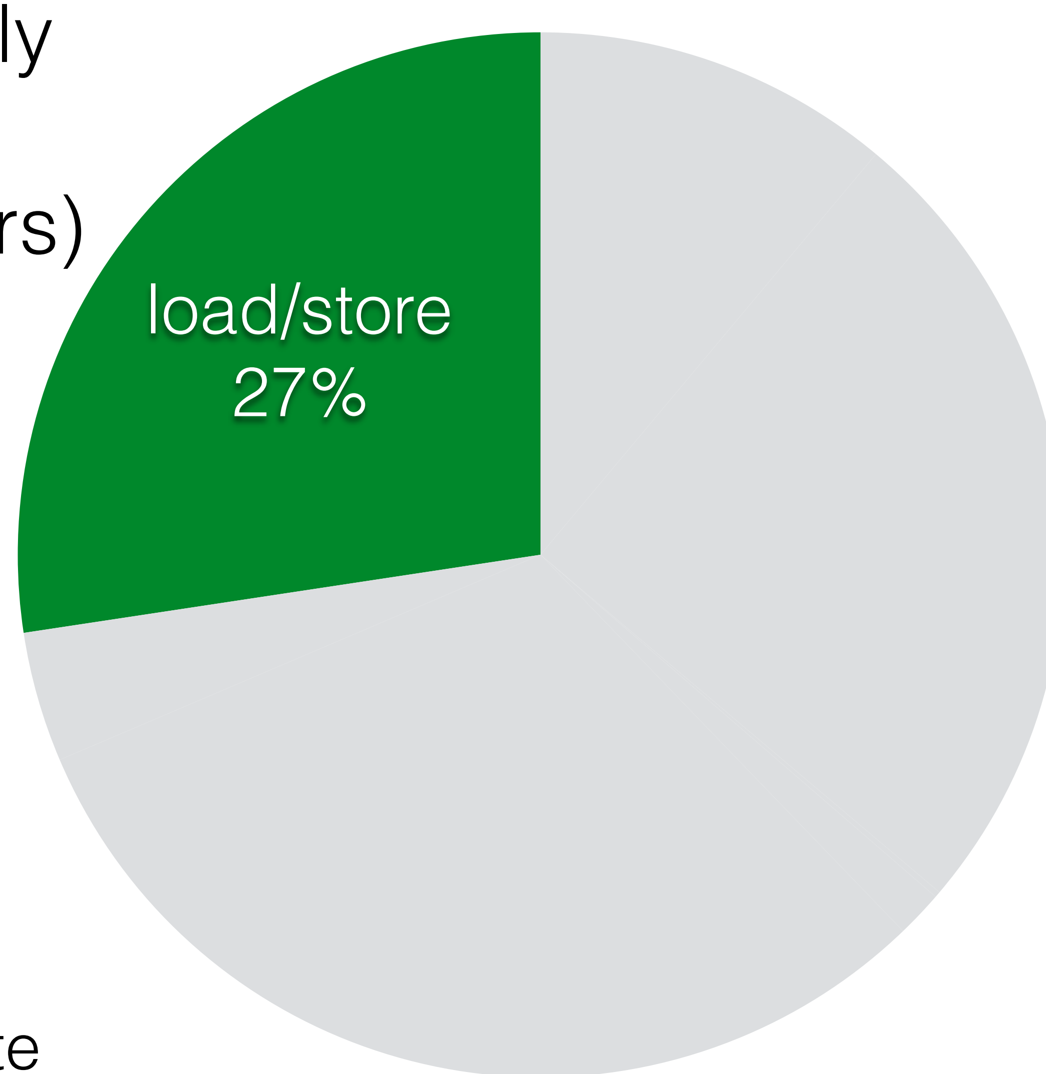
1 - PERFECT Dynamic Instruction Mix



Long latency ops are all safe to approximate

1 - PERFECT Dynamic Instruction Mix

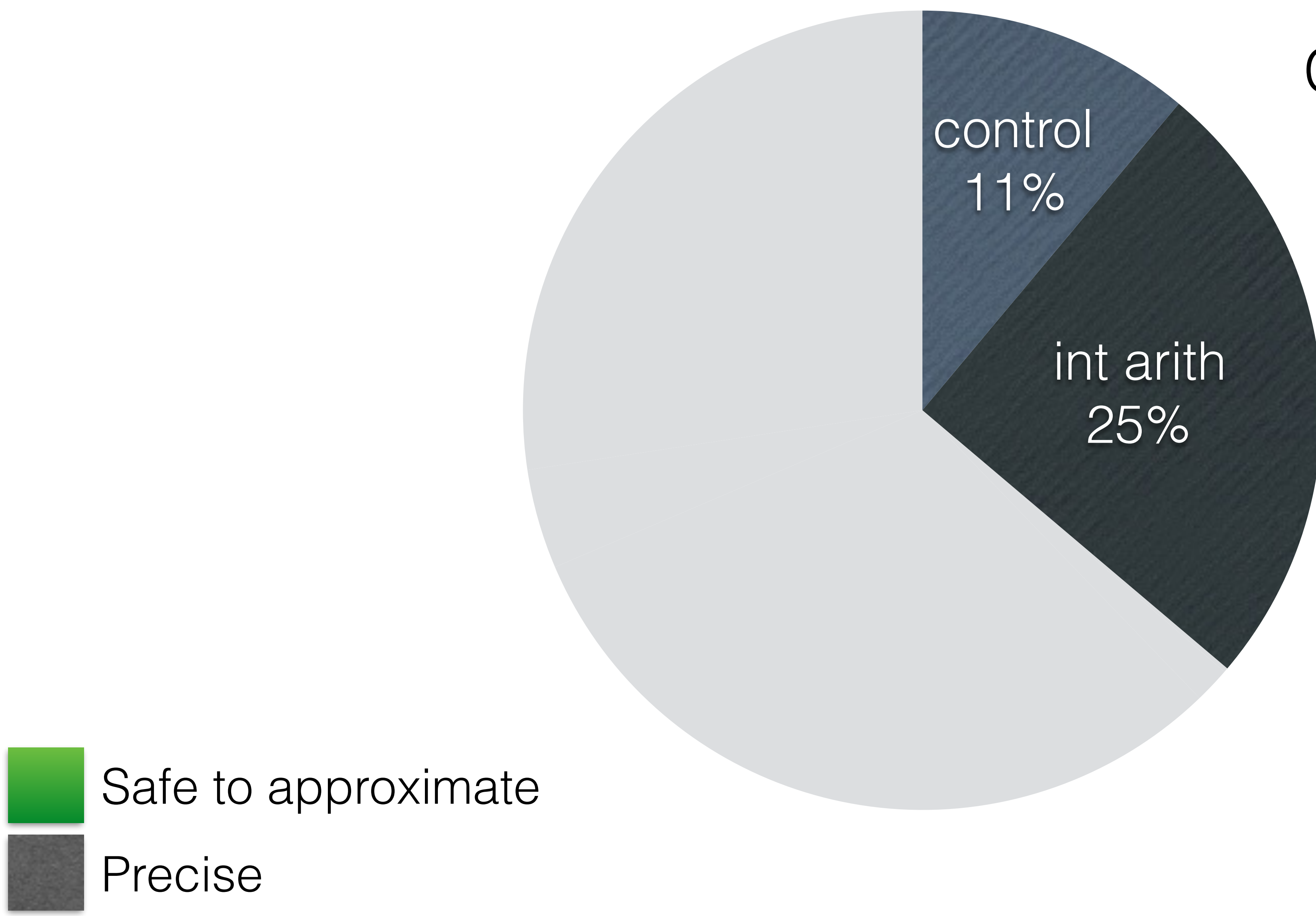
Memory ops are mostly safe to approximate (mostly data vs. pointers)



Safe to approximate

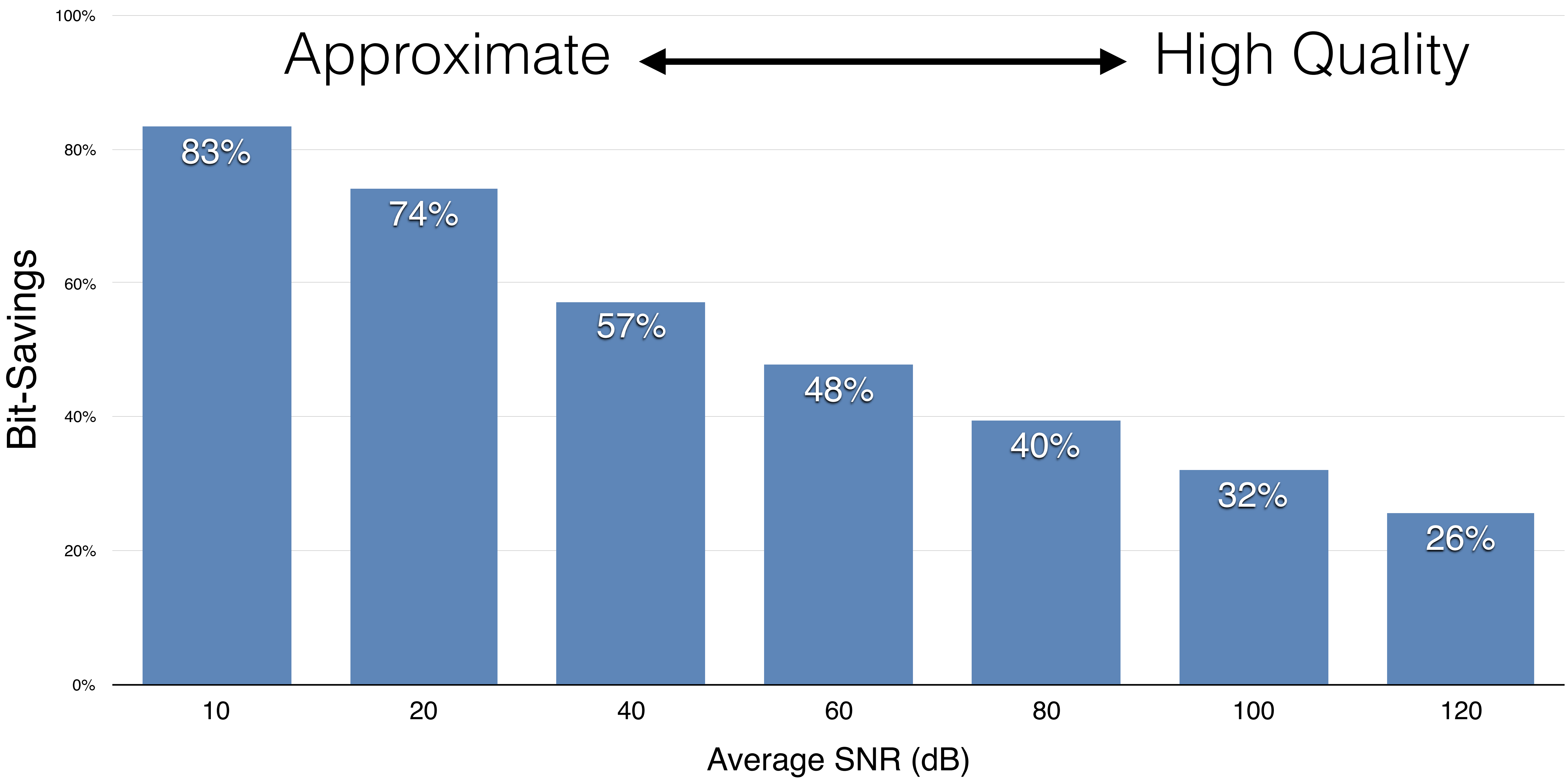
Precise

1 - PERFECT Dynamic Instruction Mix

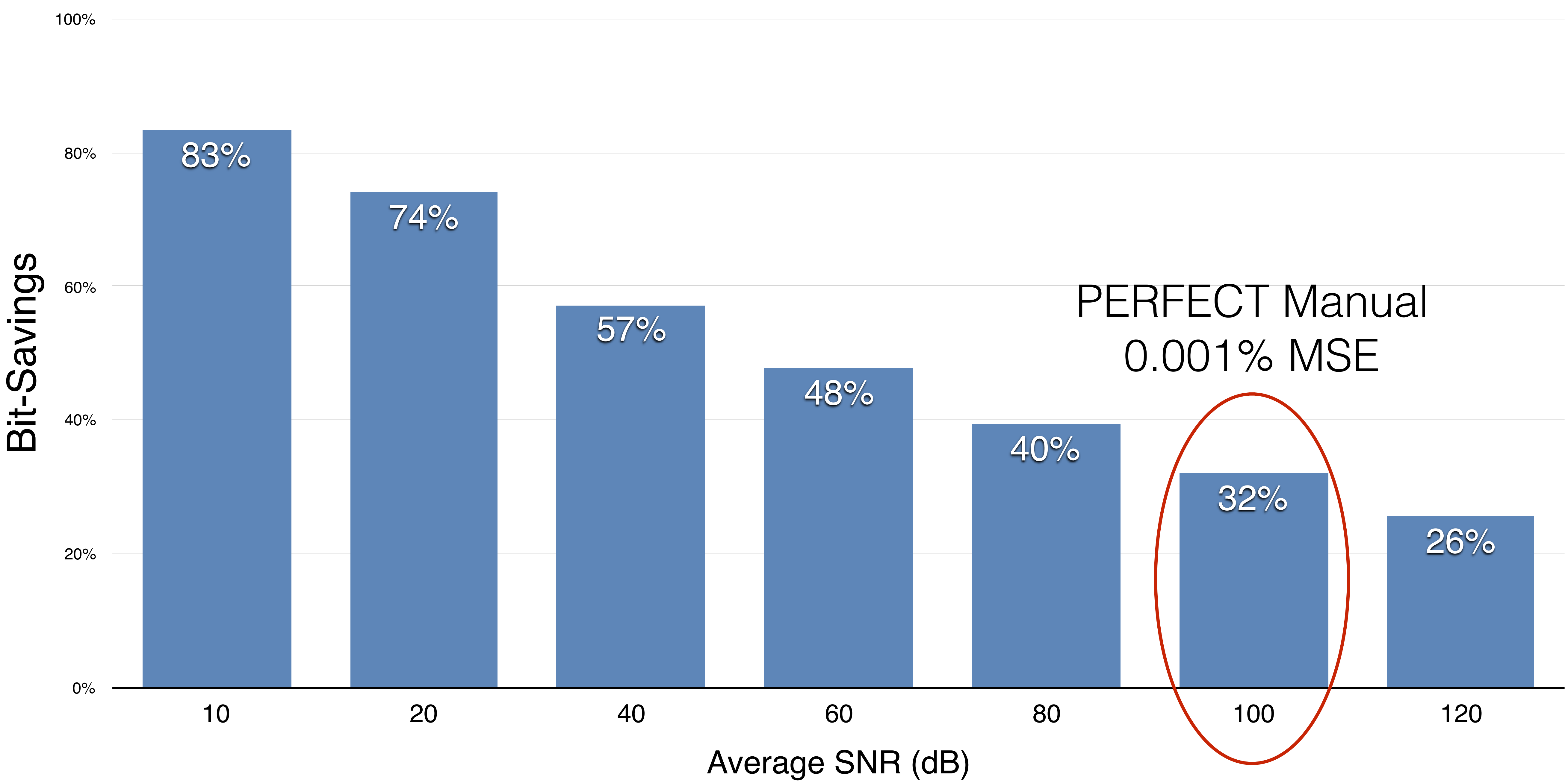


Control and address computation must remain precise

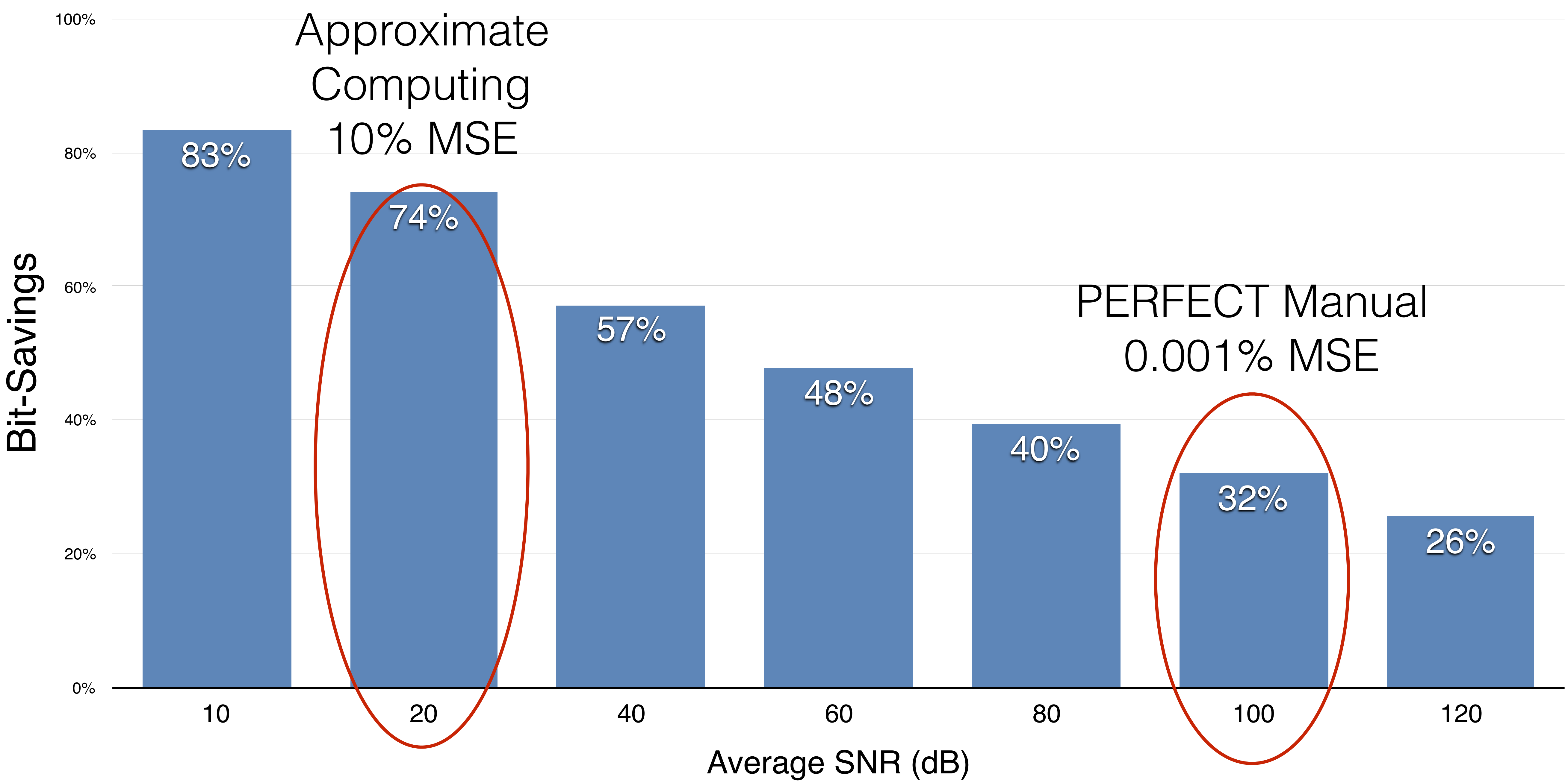
2 - Bit-Savings over Approximate Instructions



2 - Bit-Savings over Approximate Instructions



2 - Bit-Savings over Approximate Instructions



Future Architectural Challenges

Mechanisms to translate bit-savings into energy savings?

New data types/representations?

ISA extensions?

Thank You!

Approximating to the Last Bit

Thierry Moreau, Luis Ceze, Adrian Sampson
{moreau, luisceze}@cs.washington.edu, asampson@cornell.edu

WAX 2016
co-located with ASPLOS 2016
April 3rd 2016

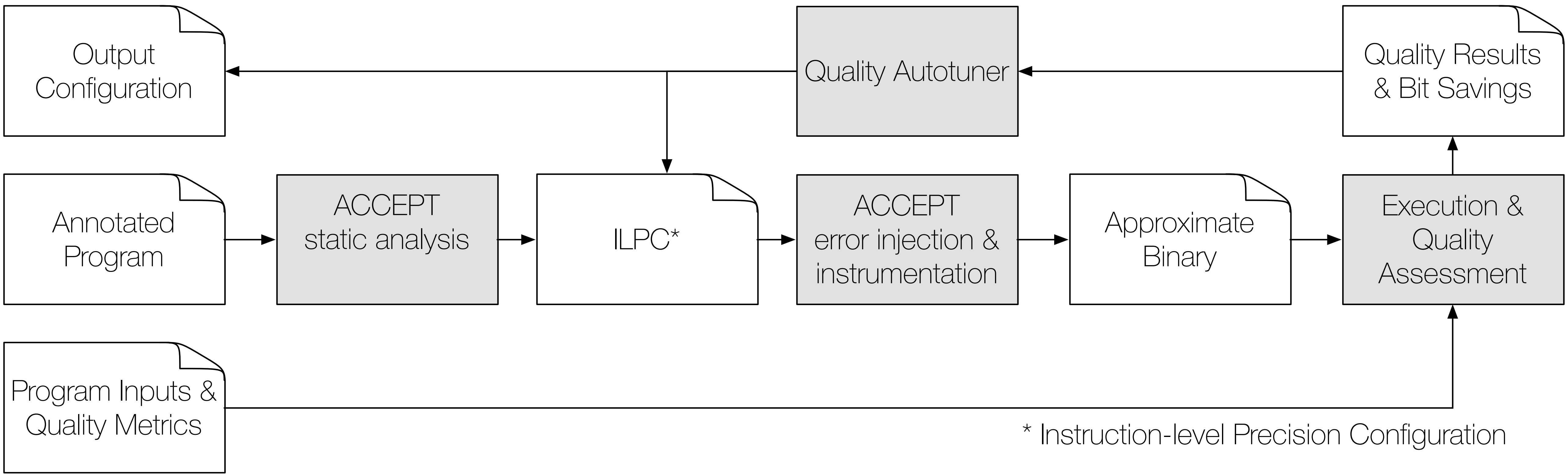
Backup Slides

Bit Savings

Explore the opportunity for precision reduction in a *hardware-agnostic* way

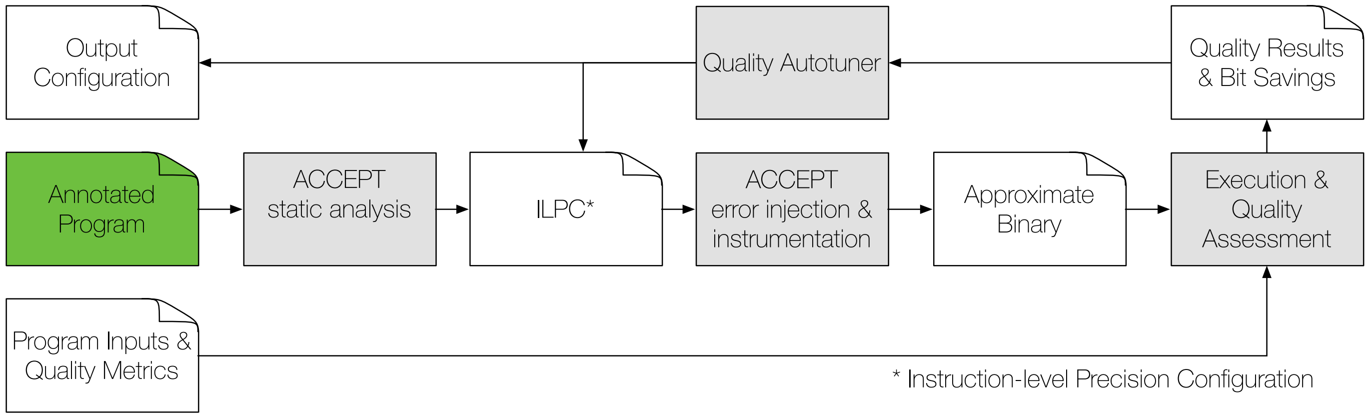
$$BitSavings = \sum_{insn_{static}} \frac{(precision_{ref} - precision_{approx})}{precision_{ref}} \times \frac{execs}{execs_{total}}$$

Framework Overview



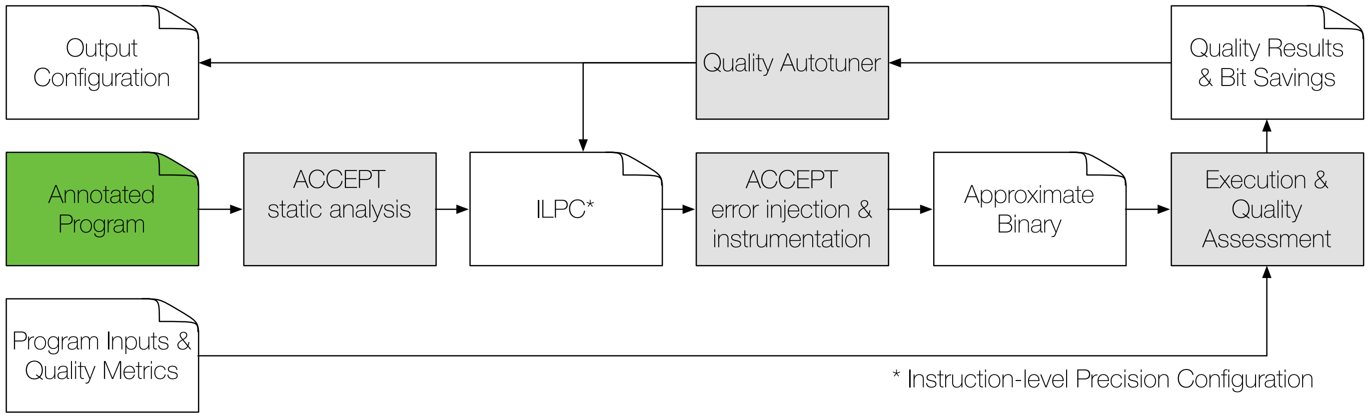
Built on top of **ACCEPT**, the approximate C/C++ compiler
http://accept.rocks

Program Annotation




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

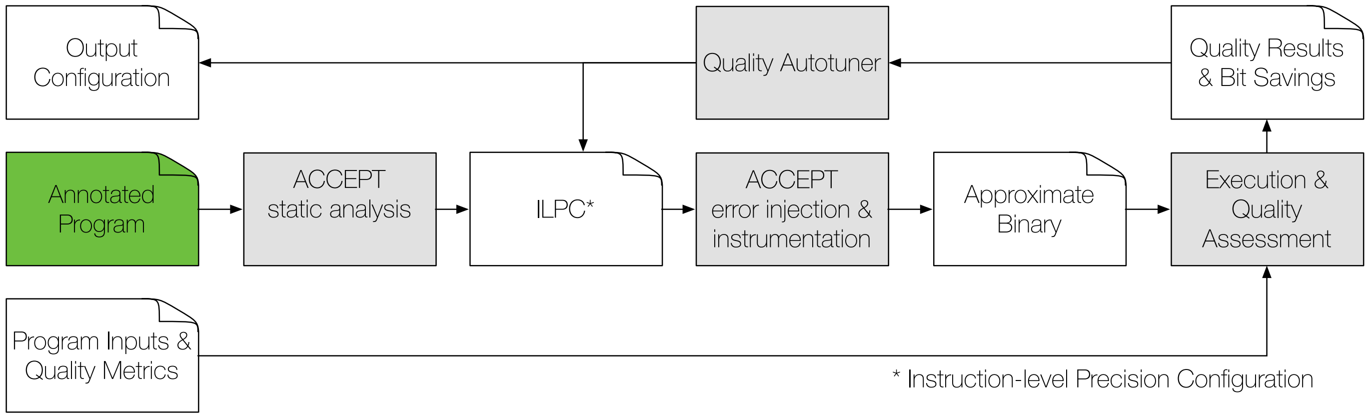
Program Annotation




```
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

Program Annotation



tips on annotating programs faster

```
typedef float flt
typedef int pix
```

↓

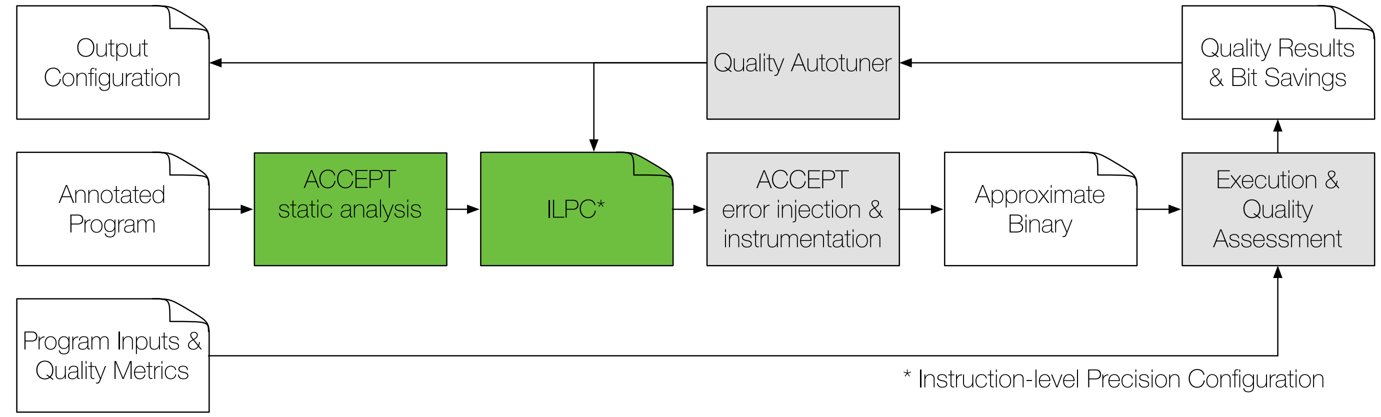
```
typedef APPROX float flt
typedef APPROX int pix
```

Takeways:

Annotating **data** is intuitive (~10 mins to annotate a kernel)

Variables used to index arrays cannot be safely approximated

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
  }
 }
}
```

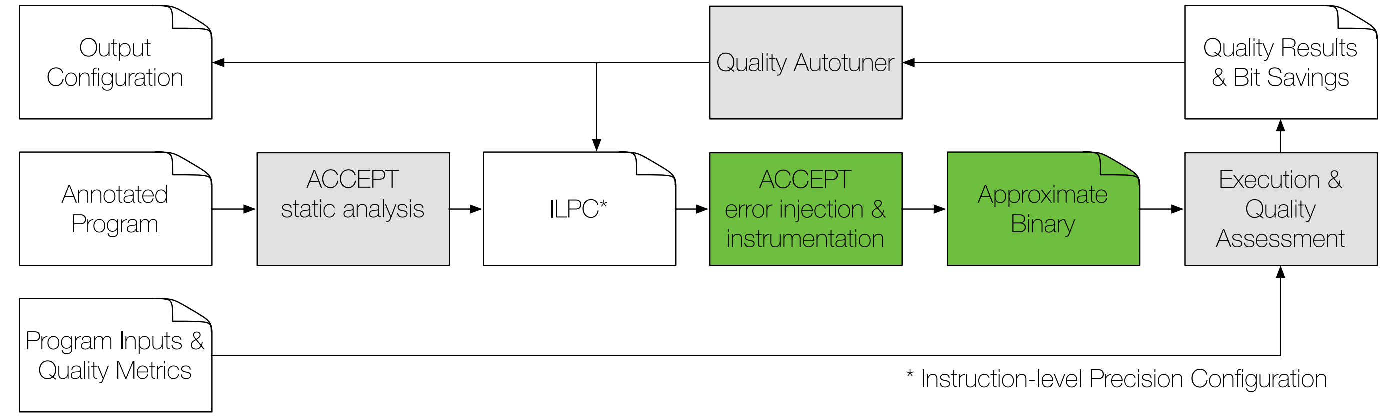
ACCEPT →

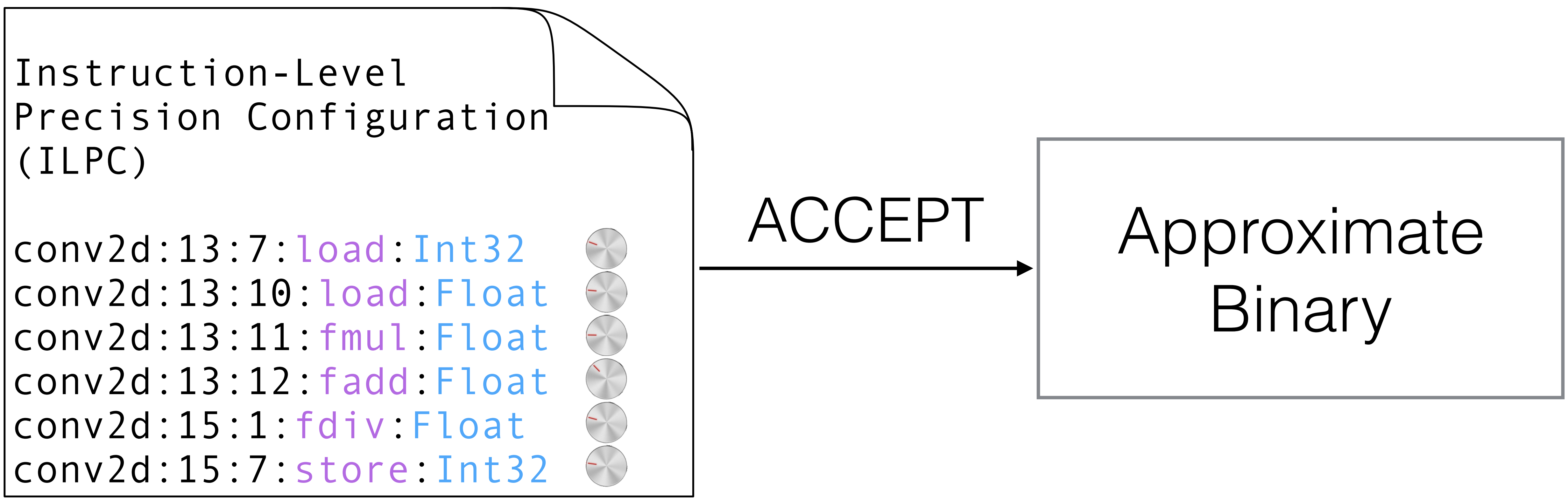
```
Instruction-Level
Precision Configuration
(ILPC)

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 identified safe-to-approximate instructions from data annotations using flow analysis

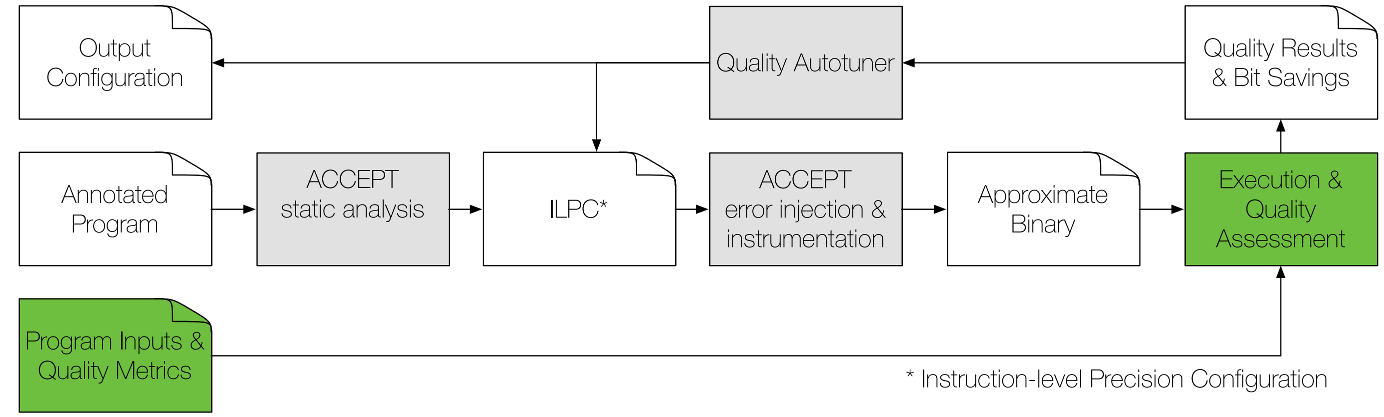
Error Injection

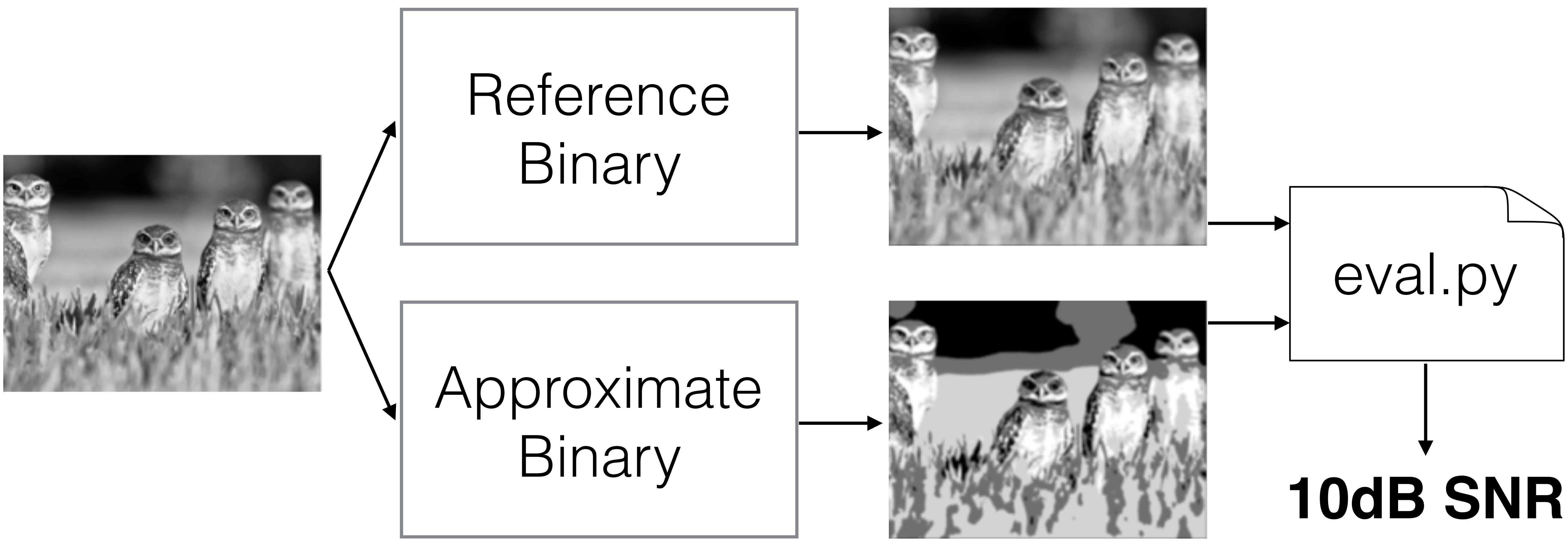




Each instruction in the ILCP acts as a quality knob that the autotuner can use to maximize bit-savings

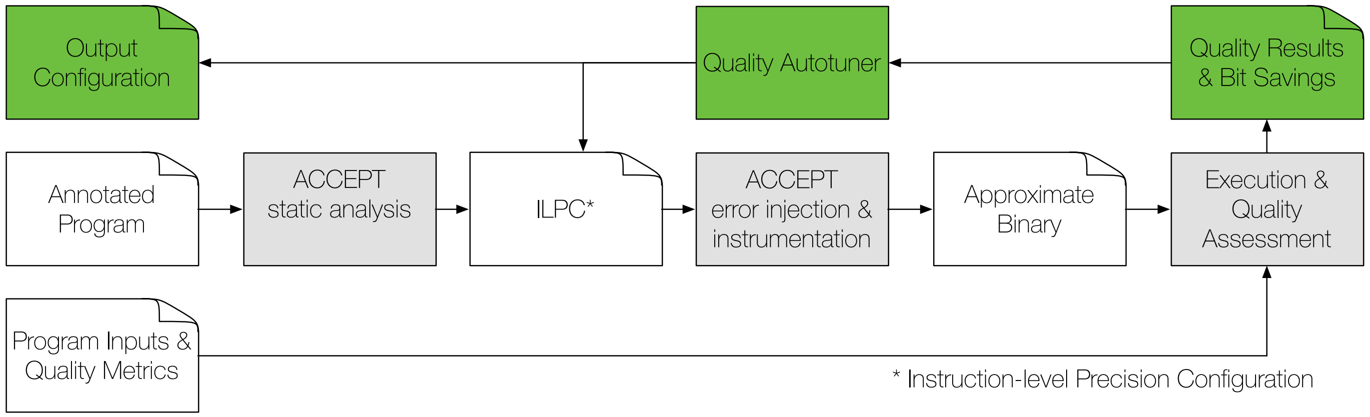
Quality Assessment



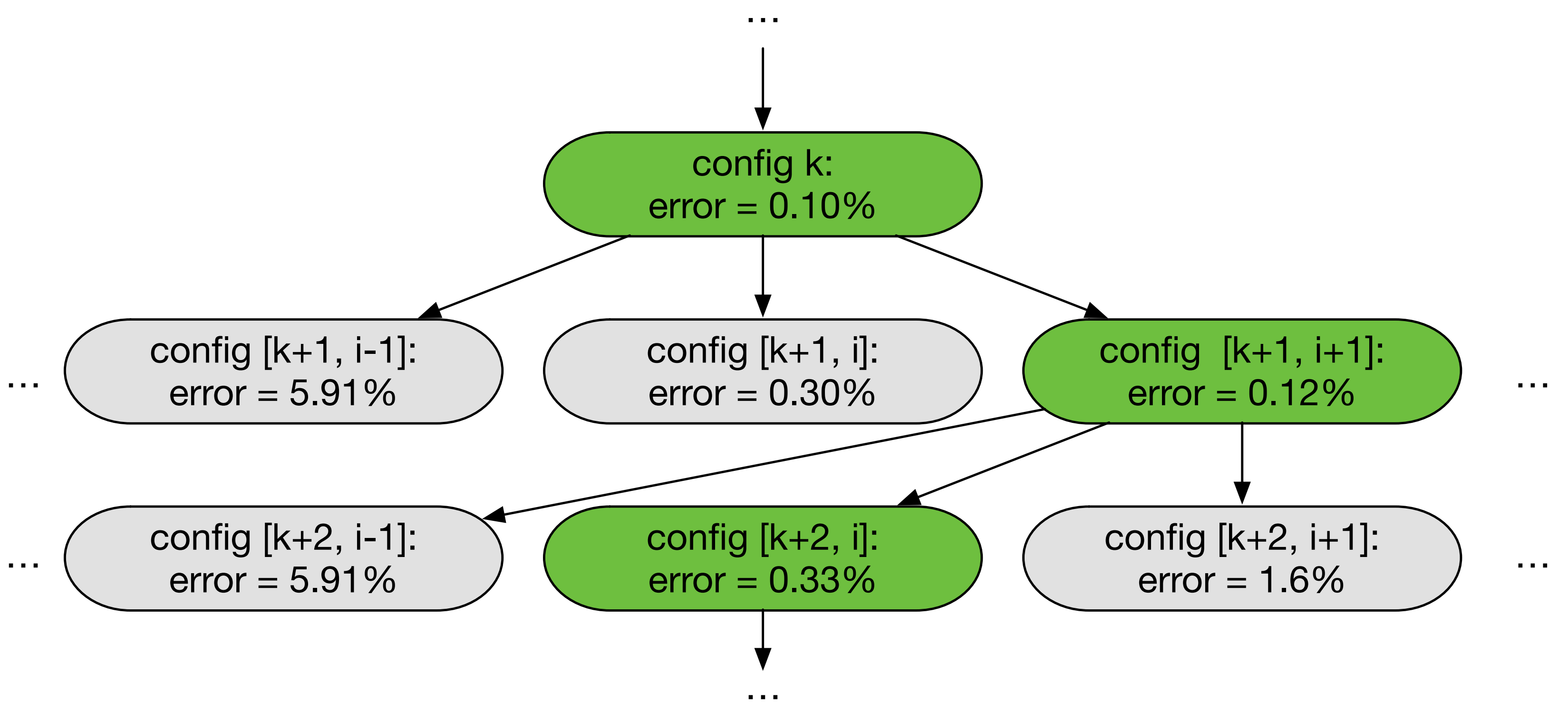


The programmer provides a quality assessment script to evaluate quality on the program output

Autotuner

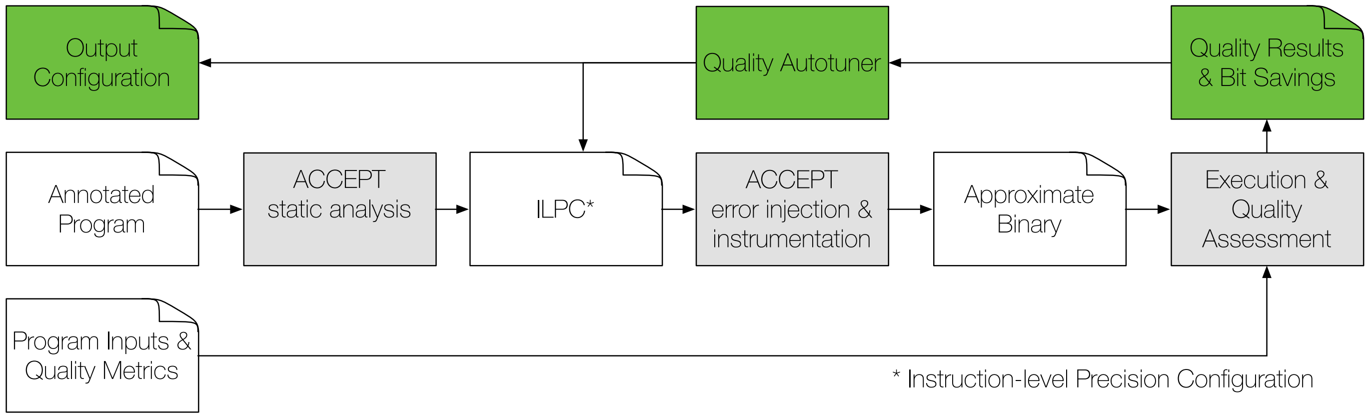


Greedy iterative algorithm: reduces precision requirement of the instruction that impacts quality the least

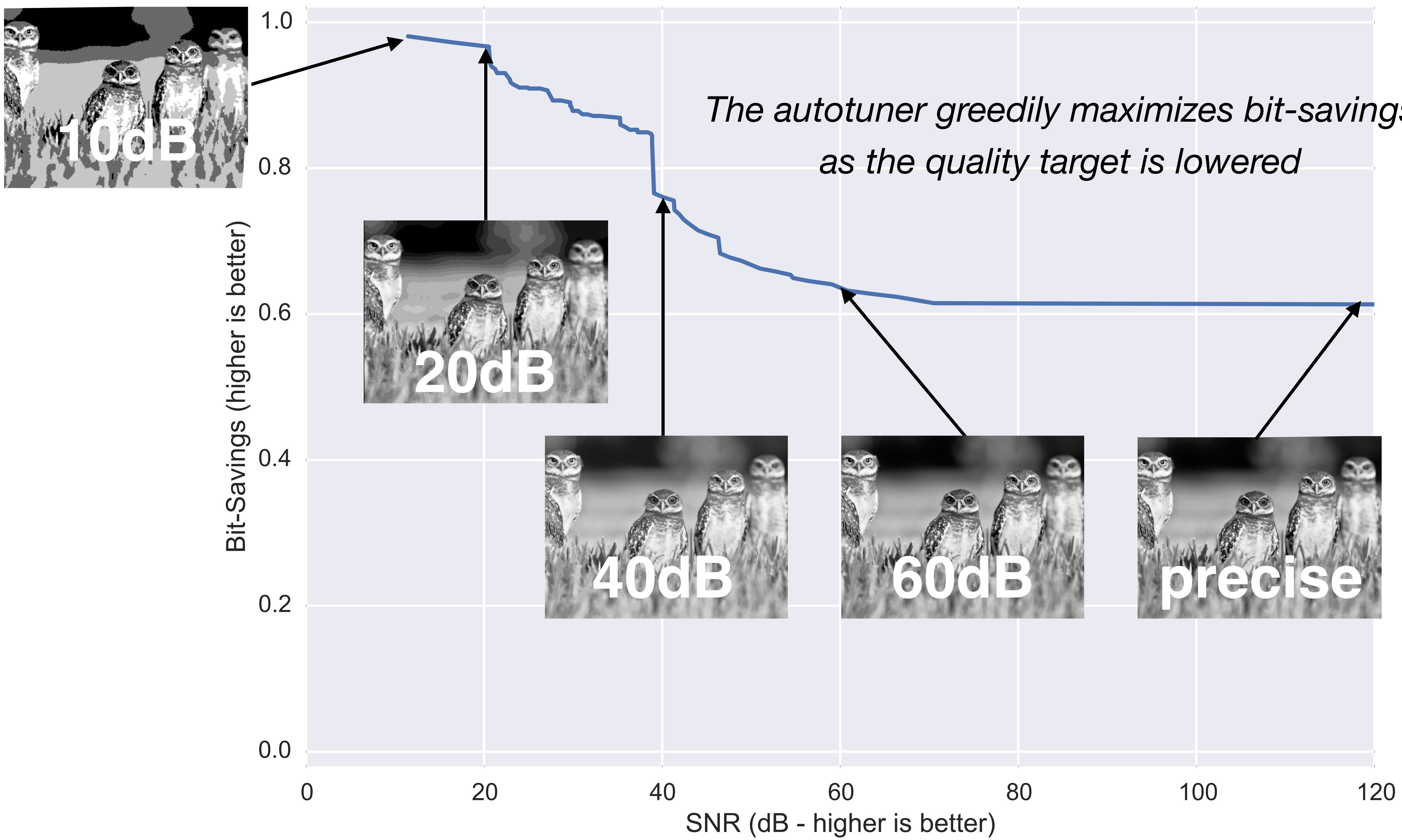


Finds solution in $O(m^2n)$ worst case where m is the number of static safe-to-approximate instructions and n are the levels of precision for all instructions

Autotuner



Output Configuration
Quality Autotuner
Quality Results & Bit Savings
Annotated Program
ACCEPT static analysis
ILPC*
ACCEPT error injection & instrumentation
Approximate Binary
Execution & Quality Assessment
Program Inputs & Quality Metrics
* Instruction-level Precision Configuration



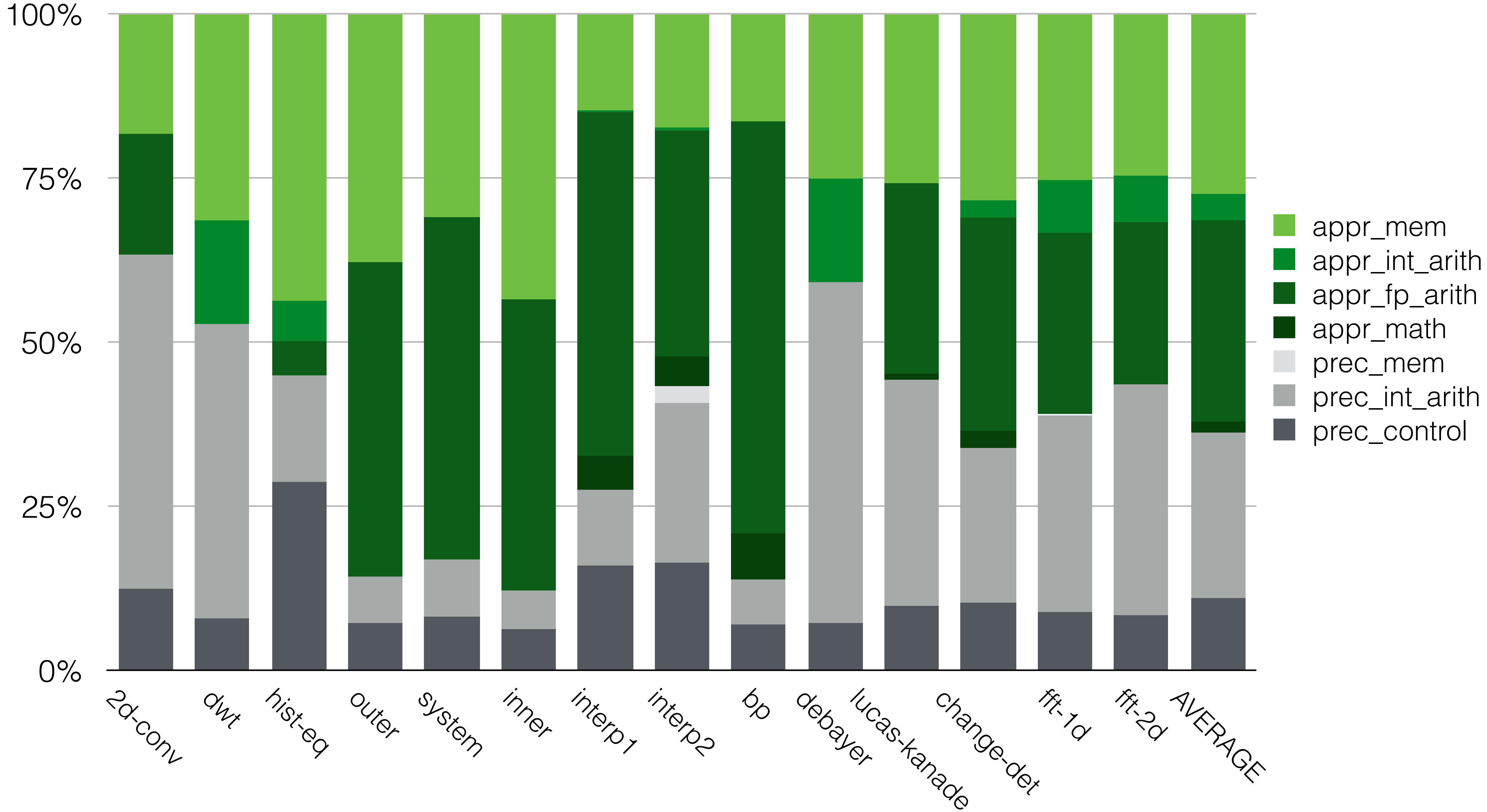
The autotuner greedily maximizes bit-savings as the quality target is lowered
10dB
20dB
40dB
60dB
precise
Bit-Savings (higher is better)
1.0
0.8
0.6
0.4
0.2
0.0
0
20
40
60
80
100
120
SNR (dB - higher is better)

Precision “Guarantees”

Currently **empirically derived** and **input dependent**

Future work would extend on the current infrastructure to assimilate data dependence information in order to derive **formal error guarantees**

1 - PERFECT Dynamic Instruction Mix



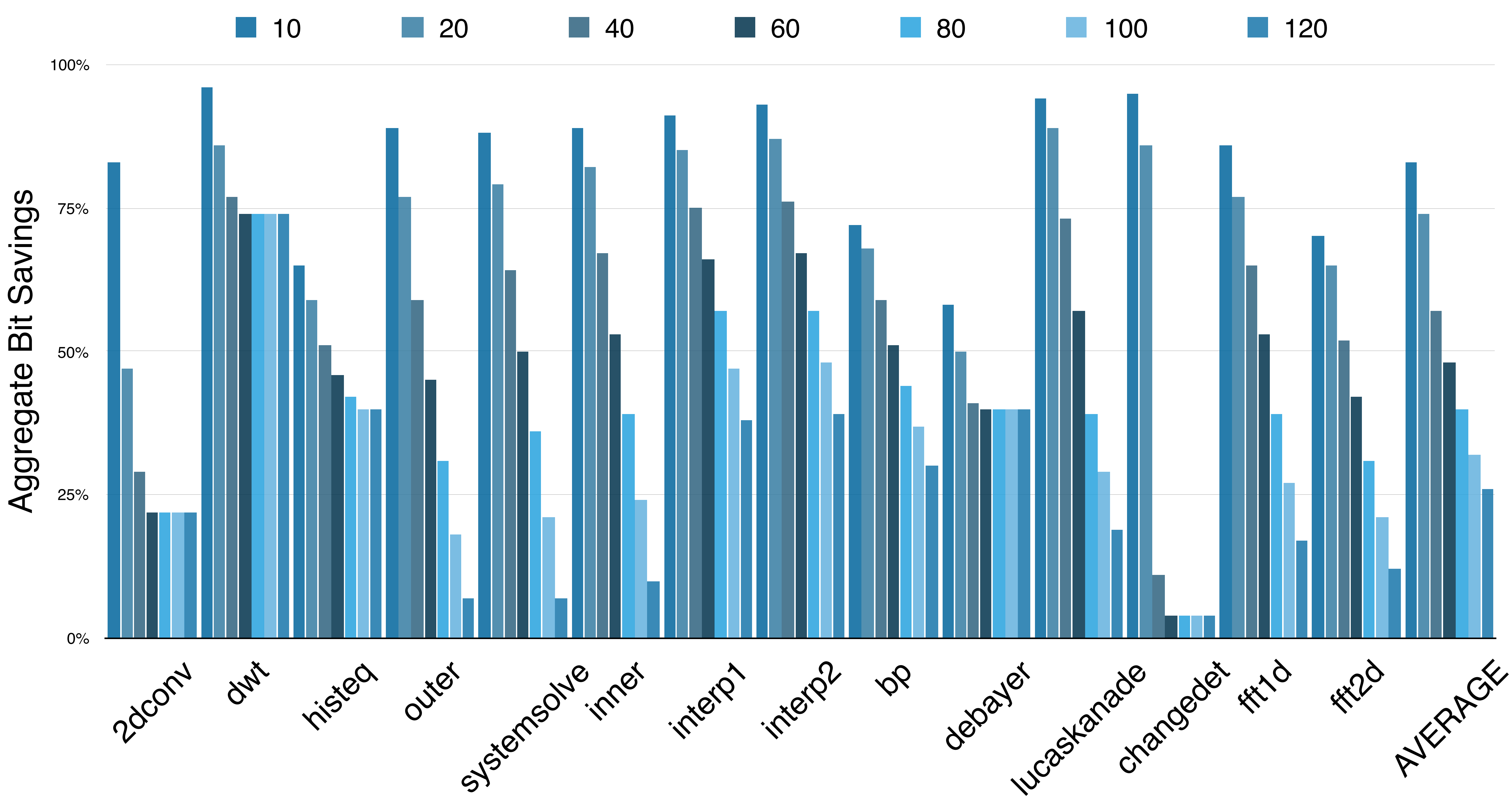
PERFECT Benchmark Suite

Application Domain	Kernels	Metric
PERFECT Application 1	Discrete Wavelet	**SNR** [120dB to 10dB]
	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	

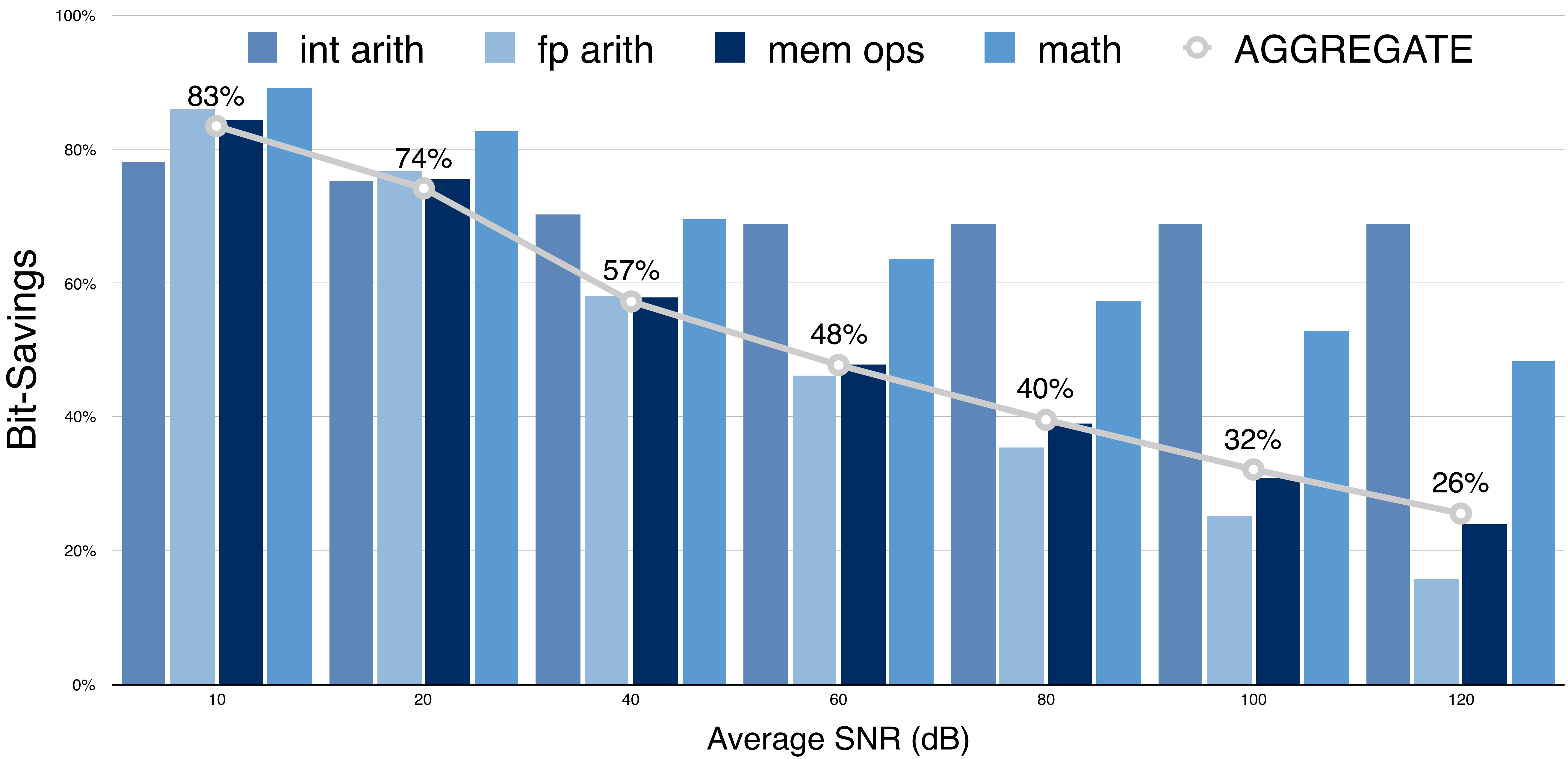
$$10\log_{10}\left(\frac{\sum_{k=1}^{N}|r_k|^2}{\sum_{k=1}^{N}|r_k - a_k|^2}\right)$$

N: number of output elements
r_k: reference value of element k
t_k: approximate value of element k

2 - Bit-Savings over Approximate Instructions



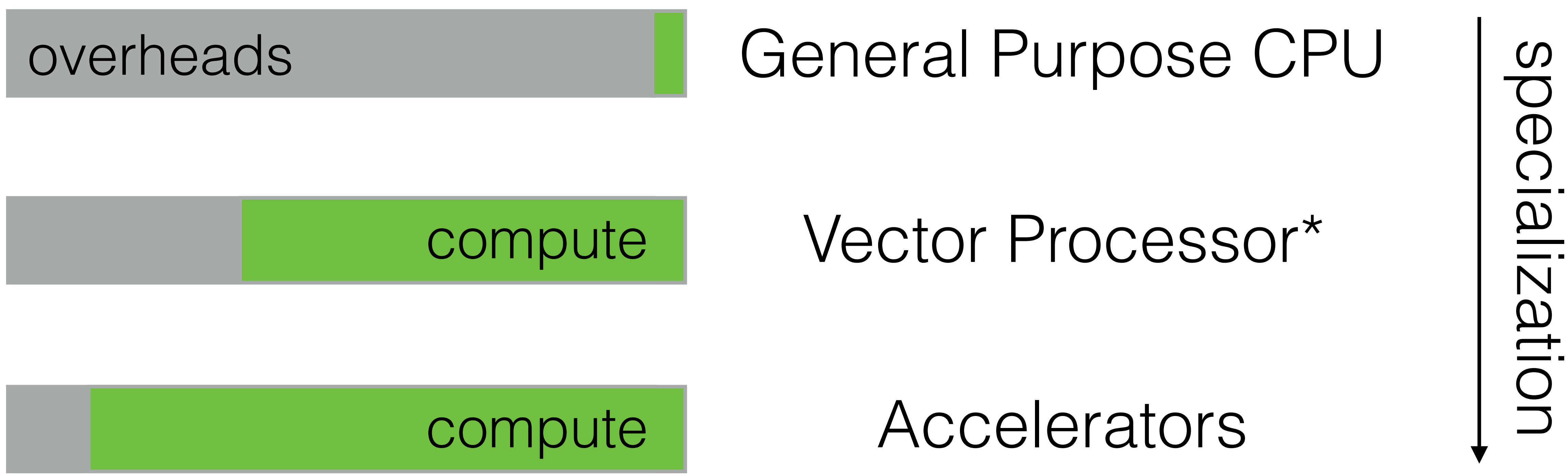
2 - Bit-Savings over Approximate Instructions



You don't need a lot of bits to obtain an acceptable output!

Architectural Target

Core Energy Breakdown



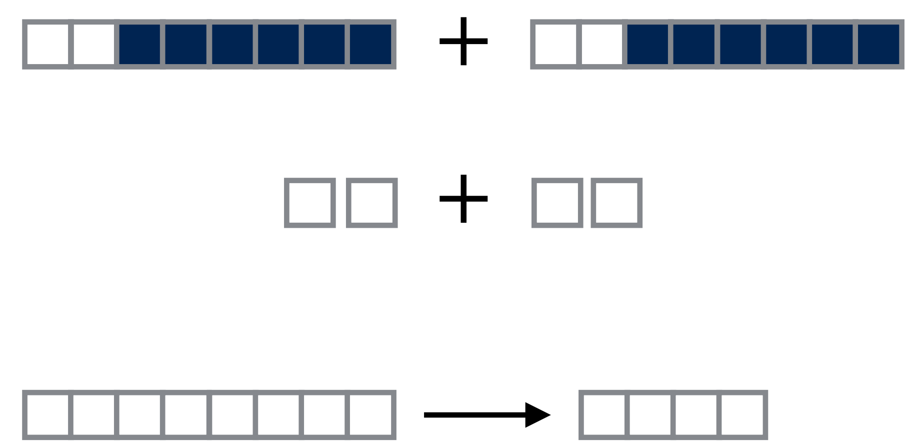
the smaller the overheads, the larger the potential gains

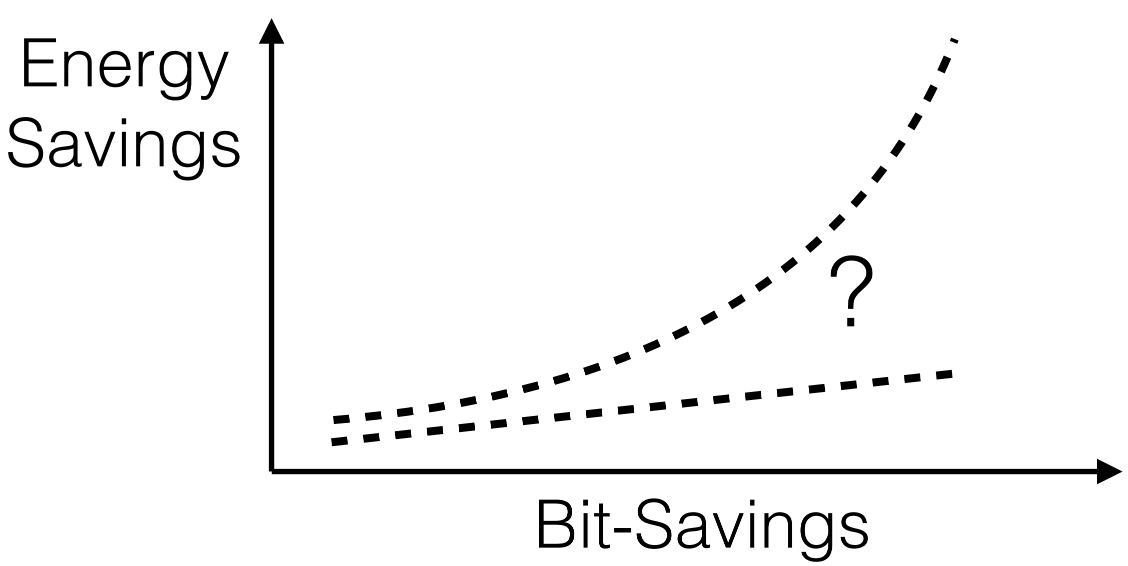
* [Quora, Venkataramani et al., MICRO2013]

Precision Scaling

Mechanisms for precision scalability:

- Fine-grained ALU power gating*
- Bit-sliced ALU units
- Lossy Compression





* [Quora, Venkataramani et al., MICRO2013]