Toward Multi-Precision, Multi-Format Numerics

David Thien dthien@eng.ucsd.edu
Bill Zorn billzorn@cs.uw.edu
Pavel Panchekha pavpan@cs.utah.edu
Zachary Tatlock ztatlock@cs.uw.edu

Computer Math is Hard

Numerous articles retracted [Altman 99, 03]

Financial regulations [Euro 98]

Market distortions [McCullough 99, Quinn 83]

Computer Math is Hard

Numerous arti

Financial regul

Market distort

How bad is it? No one knows, but it's not getting any better.

-- Bill Kahan (approx)

Demand for New Formats

HPC bandwidth concerns

ML with low precision

Domain specific hardware

Line between algorithm and implementation blurs

Accuracy on a 32-bit Budget

Proposed by John Gustafson

Maximize accuracy with a 32-bit representation

$$\left(\frac{\frac{27}{10} - e}{\pi - (\sqrt{2} + \sqrt{3})} \right)^{67/16}$$

Accuracy on a 32-bit Budget

Precision	Accuracy (decimals)
IEEE 754 binary32	4.37

Accuracy on a 32-bit Budget

Try it with posits

Floating-point has same density of numbers at every order of magnitude

Posts have more numbers around 1

Accuracy on a 32-bit Budget

Precision	Accuracy (decimals)
IEEE 754 binary32	4.37
Posits	7.05

Accuracy on a 32-bit Budget

Able to get better precision with another format

What if we could switch format/precision mid-calculation

How to precisely specify computation

Introducing FPBench 1.2: Multi-Precision Multi-Format Computations

fpbench.org

FPBench 1.2

FPCore: Input format (s-expressions)

Benchmark suite

Tools

- Exporter
- Transformer

FPCore 1.2 Syntax

FPCore 1.2

$$\sqrt{x+1} - \sqrt{x}$$

FPCore 1.2

```
(FPCore (x)
  :name example
  :precision binary64
  (-
    (sqrt (+ x 1))
    (sqrt x)))
```

FPCore 1.2

```
(FPCore (x)
  :name example
  :precision binary64
  (-
    (sqrt (+ x 1))
    (sqrt x)))
```

FPCore 1.2

```
(FPCore (x)
  :name example
  :precision binary64
  (-
    (sqrt (+ x 1))
    (sqrt x)))
```

FPCore 1.2

```
(FPCore (x)
  :name example
  :precision binary64
  (-
    (sqrt (+ x 1))
    (sqrt x)))
```

FPCore 1.2

```
(FPCore (x)
  :name example
  :precision binary64
  (-
    (sqrt (+ x 1))
    (sqrt x)))
```

FPCore 1.2

Previous syntax didn’t allow MPMF computations

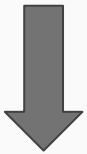
Introduce rounding contexts

FPCore Rounding Contexts

```
(sqrt
  (+ x 1))
```

FPCore Rounding Contexts

```
(sqrt
  (+ x 1))
```



```
(! :precision binary64 (sqrt
  (! :precision binary32 (+ x 1))))
```

FPCore 1.2

```
(FPCore (x)
  :name example
  :precision binary64
  (-
    (sqrt (+ x 1))
    (sqrt x)))
```

FPCore 1.2

```
(FPCore (x)
  :name example
  :precision binary64
  (-
    (! :precision binary32 (sqrt (+ x 1)))
    (! :math-library gnu-libm-2.34 (sqrt x))))
```

FPBench 1.2

Unified way to express MPMF computations

How can we actually run these

Titanic: An MPMF Laboratory

titanic.uwplse.org

Titanic: An MPMF Laboratory

Design and experiment with novel computer arithmetic formats

Python library and online tool

Online tool lets you experiment with FPCores

```
(FPCore
 (a b c)
 :name
 "NMSE p42, positive"
 :cite
 (hamming-1987 herbie-2015)
 :fpbench-domain
 textbook
 :pre
 (and (>=
       (* b b)
       (* 4 (* a c)))
       (!= a 0))
 (/
    (+
       (- b)
       (sqrt (-
              (* b b)
              (* 4 (* a c)))))
    (* 2 a)))
```

Titanic Evaluator browse benchmarks

Multiple Precision, Multiple Format Evaluator

Enter options as properties in the FPCore

FPCore arguments:

```
1 5 3
```

Evaluate FPCore permalink

a = 1.0 b = 5.0 c = 3.0

```
-0.69722436226812
```

```
1  (FPCore
2   (a b c)
3   :name
4   "NMSE p42, positive"
5   :cite
6   (hamming-1987 herbie-2015)
7   :fpbench-domain
8   textbook
9   :pre
10  (and (>=
11        (* b b)
12        (* 4 (* a c)))
13        (!= a 0))
14  (/
15     (+
16        (- b)
17        (sqrt (-
18                 (* b b)
19                 (* 4 (* a c)))))
20     (* 2 a)))
```

Titanic Evaluator browse benchmarks

Multiple Precision, Multiple Format Evaluator ▾

Enter options as properties in the FPCore

FPCore arguments:

```
1 5 3
```

Evaluate FPCore permalink

a = 1.0 b = 5.0 c = 3.0

```
-0.69722436226812
```

```
(FPCore
 (a b c)
 :name
 "NMSE p42, positive"
 :cite
 (hamming-1987 herbie-2015)
 :fpbench-domain
 textbook
 :pre
 (and (>=
       (* b b)
       (* 4 (* a c)))
      (!= a 0))
 (/
    (+
       (- b)
       (sqrt (-
              (* b b)
              (* 4 (* a c)))))
    (* 2 a)))
```

Titanic Evaluator [browse benchmarks]

Multiple Precision, Multiple Format Evaluator

Enter options as properties in the FPCore

FPCore arguments:

```
1 5 3
```

Evaluate FPCore [permalink]

```
a = 1.0  b = 5.0  c = 3.0
-0.69722436226812
```

```
1  (FPCore
2   (a b c)
3   :name
4   "NMSE p42, positive"
5   :cite
6   (hamming-1987 herbie-2015)
7   :fpbench-domain
8   textbook
9   :pre
10  (and (>=
11        (* b b)
12        (* 4 (* a c)))
13        (!= a 0))
14  (/
15     (+
16        (- b)
17        (sqrt (-
18                 (* b b)
19                 (* 4 (* a c)))))
20     (* 2 a)))
```

Titanic Evaluator browse benchmarks

Multiple Precision, Multiple Format Evaluator ▾

Enter options as properties in the FPCore

FPCore arguments:

```
1 5 3
```

Evaluate FPCore permalink

a = 1.0 b = 5.0 c = 3.0

```
-0.69722436226812
```

```
(FPCore
 (a b c)
 :name
 "NMSE p42, positive"
 :cite
 (hamming-1987 herbie-2015)
 :fpbench-domain
 textbook
 :pre
 (and (>=
       (* b b)
       (* 4 (* a c)))
      (!= a 0))
 (/
    (+
       (- b)
       (sqrt (-
              (* b b)
              (* 4 (* a c)))))
    (* 2 a)))
```

Titanic Evaluator [browse benchmarks]

Multiple Precision, Multiple Format Evaluator ▾

Enter options as properties in the FPCore

FPCore arguments:

```
1 5 3
```

Evaluate FPCore [permalink]

```
a = 1.0  b = 5.0  c = 3.0
-0.69722436226812
```

```
(FPCore
 (a b c)
 :name
 "NMSE p42, positive"
 :cite
 (hamming-1987 herbie-2015)
 :fpbench-domain
 textbook
 :pre
 (and (>=
       (* b b)
       (* 4 (* a c)))
       (!= a 0))
 (/
    (+
       (- b)
       (sqrt (-
              (* b b)
              (* 4 (* a c)))))
    (* 2 a)))
```

Titanic Evaluator browse benchmarks

Multiple Precision, Multiple Format Evaluator

Enter options as properties in the FPCore

FPCore arguments:

```
1 5 3
```

Evaluate FPCore permalink

a = 1.0 b = 5.0 c = 3.0

```
-0.69722436226812
```

Accuracy on a 32-bit Budget

Proposed by John Gustafson

Maximize accuracy with a 32-bit representation

$$\left(\frac{\frac{27}{10} - e}{\pi - (\sqrt{2} + \sqrt{3})} \right)^{67/16}$$

Accuracy on a 32-bit Budget

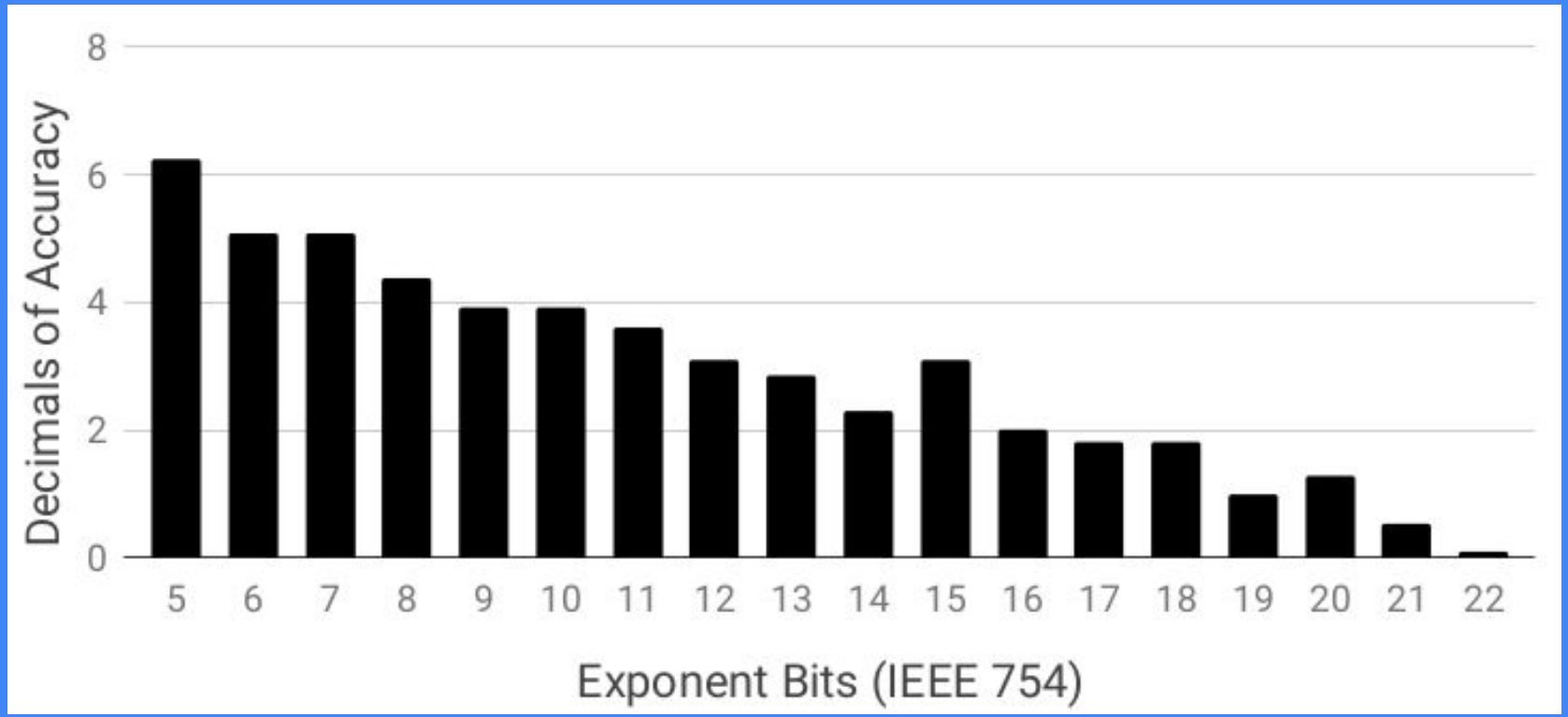
```
(FPCore (x y)
  :name "Accuracy on a 32-bit budget"
  :pre (and (>= x 0) (>= y 0))
  (pow
    (/
      (- (/ 27 10) E)
      (- PI (+ (sqrt x) (sqrt y))))
    (/ 67 16)))
```

Accuracy on a 32-bit Budget

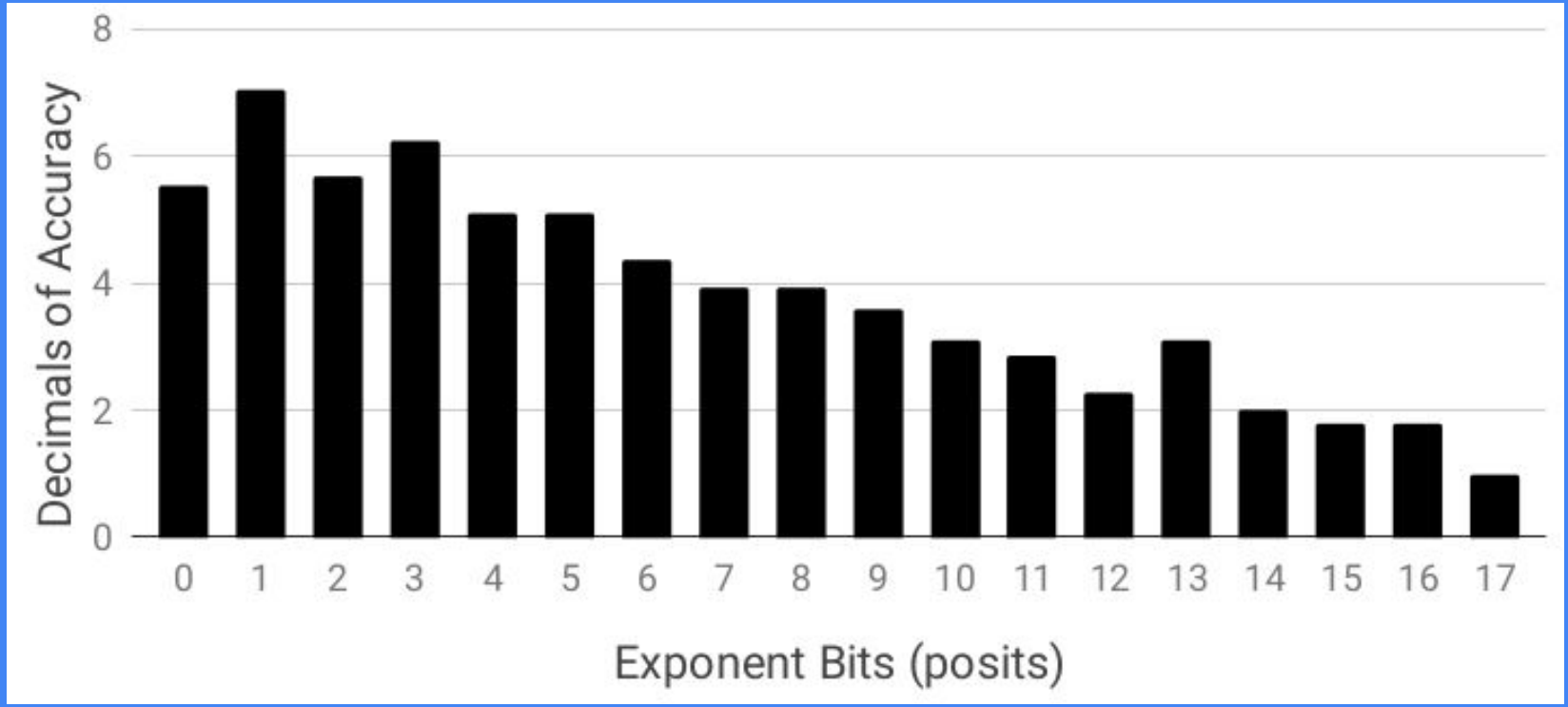
Titanic allows us to carry out MPMF computations

Also easy way to test new formats

Accuracy on a 32-bit Budget



Accuracy on a 32-bit Budget



Accuracy on a 32-bit Budget

Precision	Accuracy (decimals)
IEEE 754 binary32	4.37
32-bit floating-point 5-bit exponent	6.24
Posits	7.05

Accuracy on a 32-bit Budget

Get increased accuracy with slightly different floating-point representation

What if we can mix precisions

Accuracy on a 32-bit Budget

```
(FPCore (x y)
  :name "Accuracy on a 32-bit budget"
  :pre (and (>= x 0) (>= y 0))
  (pow
    (/
      (- (/ 27 10) E)
      (- PI (+ (sqrt x) (sqrt y))))
    (/ 67 16)))
```

Example: Accuracy on a 32-bit Budget

```
(FPCore (x y)
  :name "Accuracy on a 32-bit budget"
  :pre (and (>= x 0) (>= y 0))
  (! :precision A (pow                                    ;A
    ( ! :precision B (/                                   ;B
      (! :precision C (- (/ 27 10) E))                    ;C
      (! :precision D (- PI (+ (sqrt x) (sqrt y))))))     ;D
    (! :precision E (/ 67 16)))))                         ;E
```

Accuracy on a 32-bit Budget

Representation	A	B	C	D	E	Constants	Accuracy (decimals)
IEEE 754	8	8	8	8	8	8	4.37
Uniform-IEEE	5	5	5	5	5	5	6.24
Uniform-posit	1	1	1	1	1	1	7.05
Mixed-IEEE	5	8	3	2	4	2	**7.40**
Mixed-posit	0	6	0	0	0	1	7.38

MPMF Tools

MPMF can increase accuracy of computations

Tooling for MPMF presents its own challenges

Techniques for one precision might not work in another

Herbie 1.2: MPMF

herbie.uwplse.org

MPMF Herbie

Herbie is a tool originally designed to find and fix floating-point computations

Extended to support posits

Easily extensible specification for new formats

MPMF Herbie

Add 16-bit posits to Herbie

- No full libm
- Experimental number system
- Large precision accumulator (mixed-precisions?)

How does Herbie Work

Specify input computation

Establish ground truth

Rewrite with equivalence classes and taylor series

Partition into regimes

Herbie Interface

Interface requires specification of

- Casts between format and bigfloat
- Casts between format and ordinals
- Special values (e.g. NaN)
- Operators

Optionally, users can also specify additional rewrite rules

Herbie Experiment

Wanted to test how well existing rewrite rules work for other formats

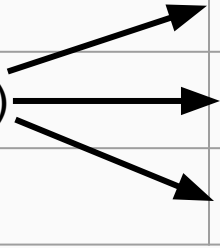
Compare Herbie’s output for several different precisions

Herbie Format Adaptation

Real
F(x)

Herbie Format Adaptation

Real	Impl
	$f_A(x)$
F(x)	$f_B(x)$
	$f_C(x)$



Abstract syntax to computer program

Herbie Format Adaptation

Real	Impl	Impl
	$f_A(x)$ →	$f_A'(x)$
$F(x)$ →	$f_B(x)$ →	$f_B'(x)$
	$f_C(x)$ →	$f_C'(x)$

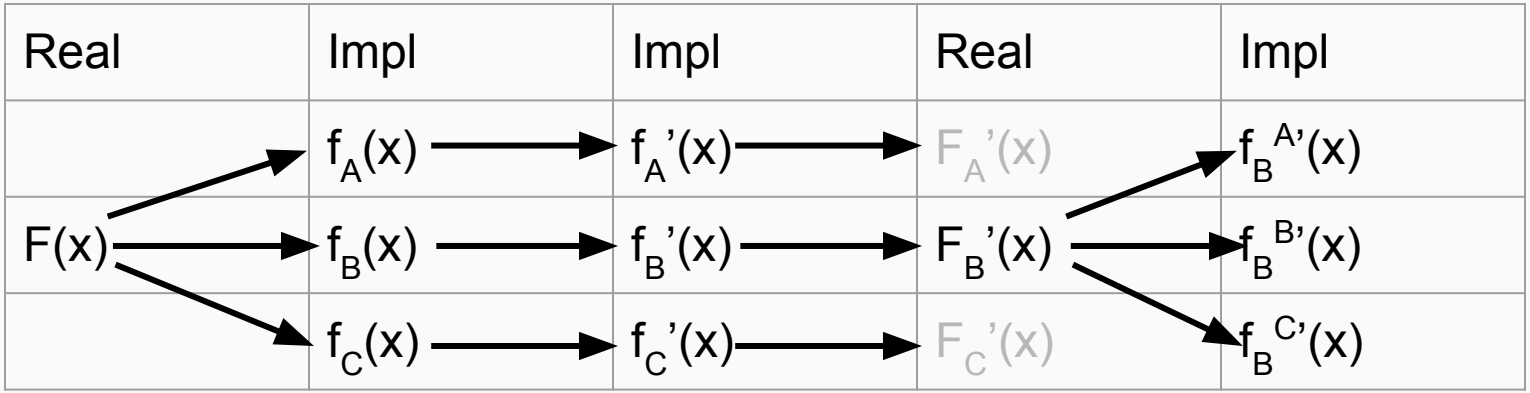
Run herbie

Herbie Format Adaptation

Real	Impl	Impl	Real
	$f_A(x)$ →	$f_A'(x)$ →	$F_A'(x)$
$F(x)$ →	$f_B(x)$ →	$f_B'(x)$ →	$F_B'(x)$
	$f_C(x)$ →	$f_C'(x)$ →	$F_C'(x)$

Convert new program to abstract program

Herbie Format Adaptation

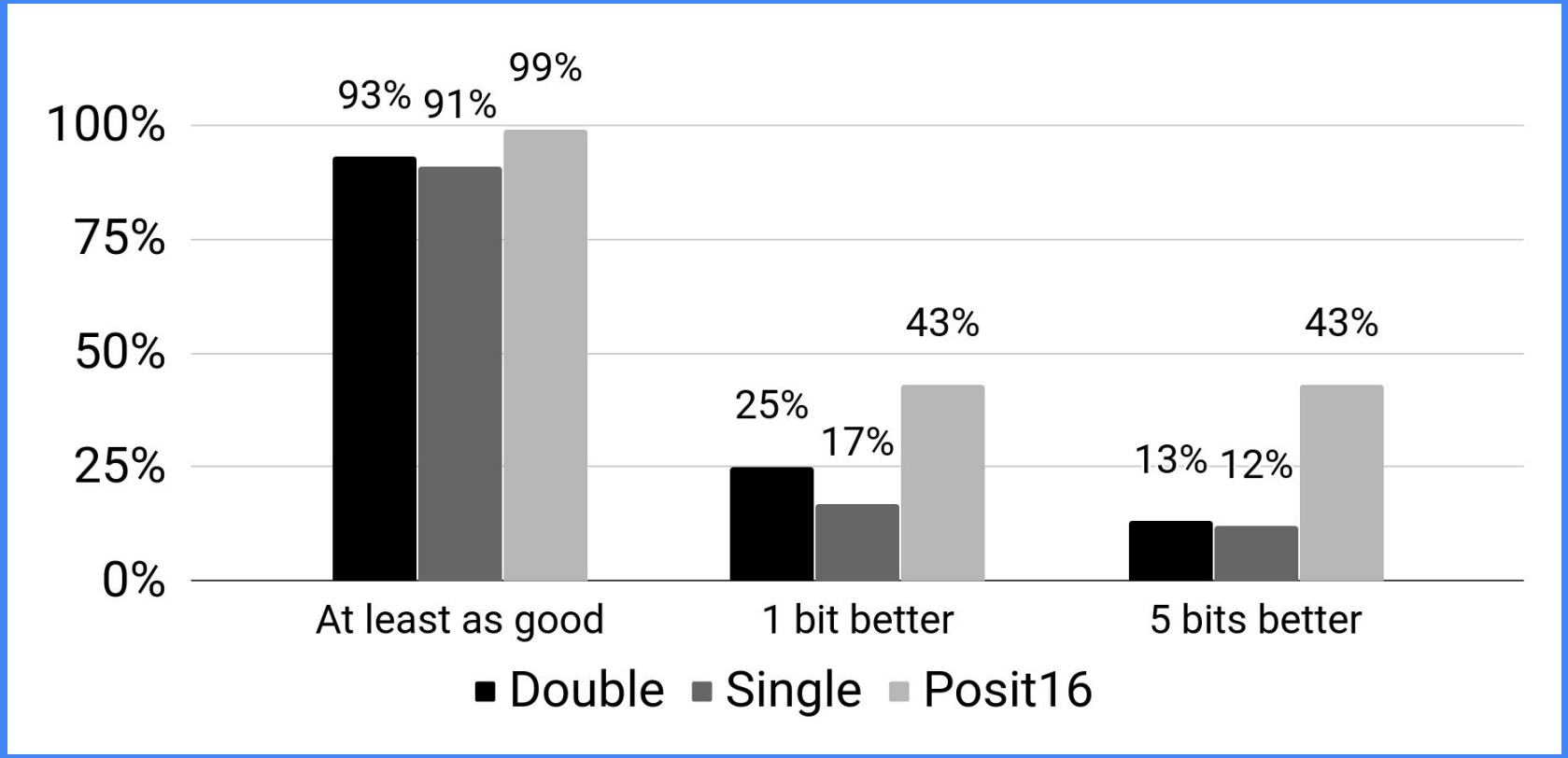


Convert real program to all precisions

Herbie Format Adaptation

The best program for each precision/format should be created when Herbie optimizes for that precision/format

Herbie should take advantage of special features of a given precision/format


100%
75%
50%
25%
0%
93%
91%
99%
25%
17%
43%
13%
12%
43%
At least as good
1 bit better
5 bits better
Double
Single
Posit16

What Now

Call to Action

Looking beyond MPMF

- Already started work on FPBench 1.3
- Figure out how to do tensors

Call to Action

Support MPMF in your tools

- A few tools that support the older FPBench 1.0 standard
- New standard is meant to provide all the expressibility you need

Call to Action

Join the FPBench community

- Send us your benchmarks
- Use the FPBench tools (and file issues/PRs if you see something to improve)
- Try out our FPCore format and play around with Titanic

Questions