MANCHESTER 1824
The University of Manchester

Energy-accuracy trade-offs in floating-point arithmetic architectures

Mantas Mikaitis

SpiNNaker team meeting, 18th December 2018

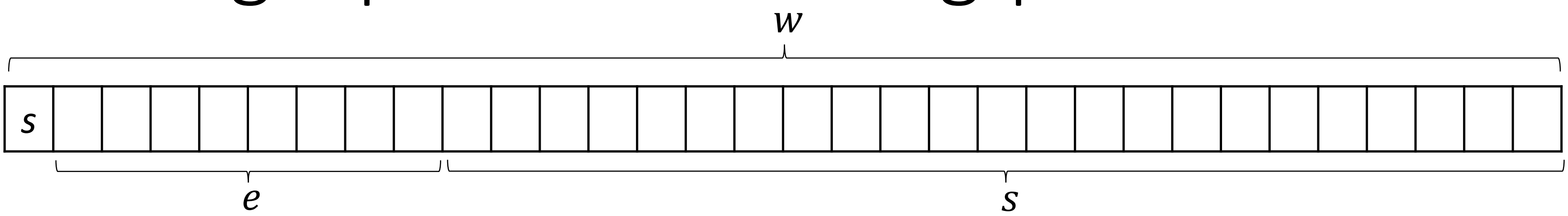
Contents

- Motivation for floating-point format
- On accuracy of floating-point numbers
- Accuracy requirements in the standards
- Results

Motivation for floating-point arithmetic

- SpiNNaker2 computing node ARM M4F has an FPU containing: **ABS, ADD, SUB, CVT, DIV, MUL, MLA, MLS, FMA, SQRT**.
- Larger range of representable values – avoid under/overflow in complex neuron model equations (AdExp, Hodgkin-Huxley).
- Very accurate nearer to 0.0 – the gap between two neighbouring values is getting smaller and smaller as it is relative to the exponent.

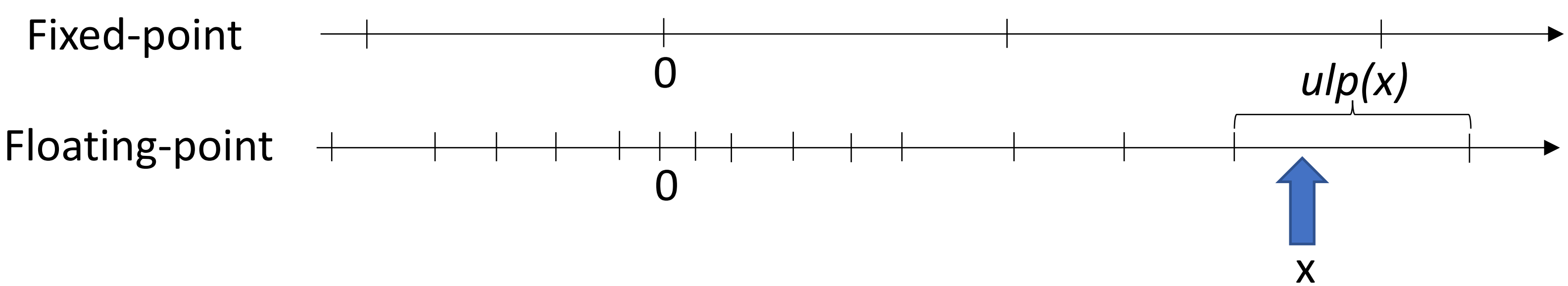
Single precision floating-point format



- e – biased exponent bits
- s - significand
- Implicit 1 at MSB of the significant

- $\epsilon = 2^{-23}$(Machine epsilon)
- Smallest positive value is $2^{-126} \approx 1.17 \times 10^{-38}$.
- Largest value is $(2 - 2^{-23}) \times 2^{127} \approx 3.4 \times 10^{38}$.

- Decimal value is decoded as: $(-1)^{s} \times 2^{e-127} \times (1 + s \times 2^{-23})$
- Definition of ulp(x) [Kahan, Muller05]: *Given some value x of infinite precision, ulp is a gap between the two neighbouring floating-point numbers, even if x is one of them.*

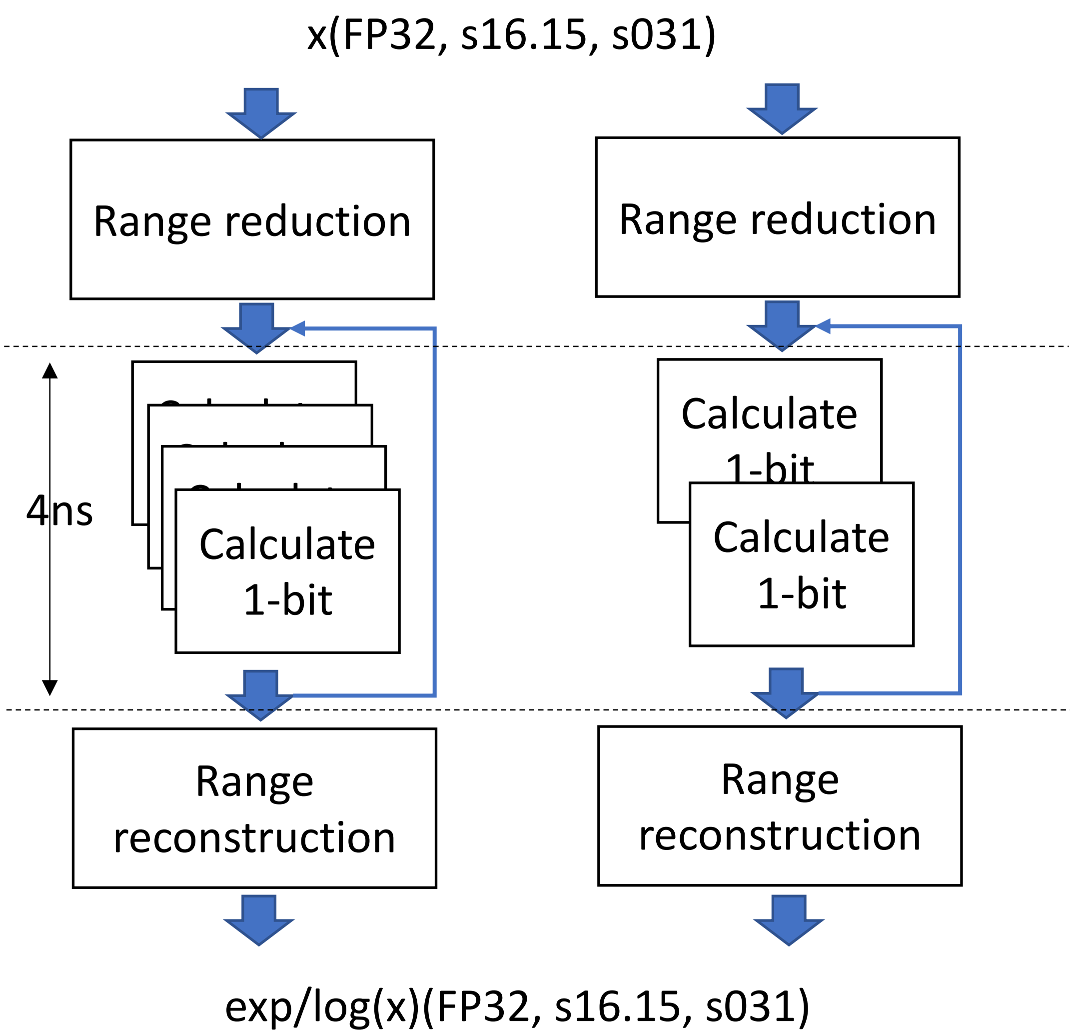


Accuracy requirements in the standards

- IEEE 754-2008 floating-point standard specifies that all functions should return a ***correctly rounded result:*** *"every operation shall be performed as if it first produced an intermediate result correct to infinite precision and with unbounded range, and then rounded that result"*
- This means that the result is rounded to nearest and is **within 0.5ulp from the mathematically exact result**.
- Another useful standard is **OpenCL parallel programming standard**. It specifies that both exp and log functions should have accuracy $\leq 3ulps$ in **desktop profile**, $\leq 8192ulps$ for *half_exp() and half_log()* and $\leq 4ulps$ in **embedded profile. Flush subnormals to 0.**

Standard	**exp()**	**log()**
IEEE 754-2008	0.5ulp	0.5ulp
OpenCL	3 (desktop) 4 (embedded) 8192 (half) ulps.	3 (desktop) 4 (embedded) 8192 (half) ulps.

Accelerator proposed for SpiNNaker2



- Two versions: optimized for performance and optimized for leakage.
- Internal representation is fixed-point s3.35 – early choice for maximizing s16.15 accuracy.
- Range reduction/reconstruction stages 1-2 clock cycles.
- 2 cycles for bus operation.
- For full accuracy, version on the left requires 13 cycles and the version on the right 21 cycles.
- **13→21 cycles reduces leakage 50% and area 26%.**

Range reduction: exp()

Function: $\exp(x) \in (0, MAX_{FLOAT}]\ with\ x \in [\sim -88(-104\ for\ denormal\ output), \sim 88]$.

Algorithm 1:

1. Convert x to fixed-point.
2. Find n such that $x' = x - n \times log(2)$ and $x' \in [-1.242, 0.869]$ (Take $n = \left\lfloor x \cdot \frac{369}{256} \right\rfloor$).
3. Calculate $\exp(x') \in [0.29, 2.38]$.
4. (Range reconstruction) $\exp(x) = \exp(x' + n \times \log(2)) = \exp(x') \times 2^n$.

In step 4, we first normalize $\exp(x')$(using CLZ), cut off the MSB and treat that as a significand. We also add n+CLZ to get the exponent of the floating point representation.

NOTE: We never have to keep very large numbers like MAX_{FLOAT} in fixed-point.

Range reduction: log()

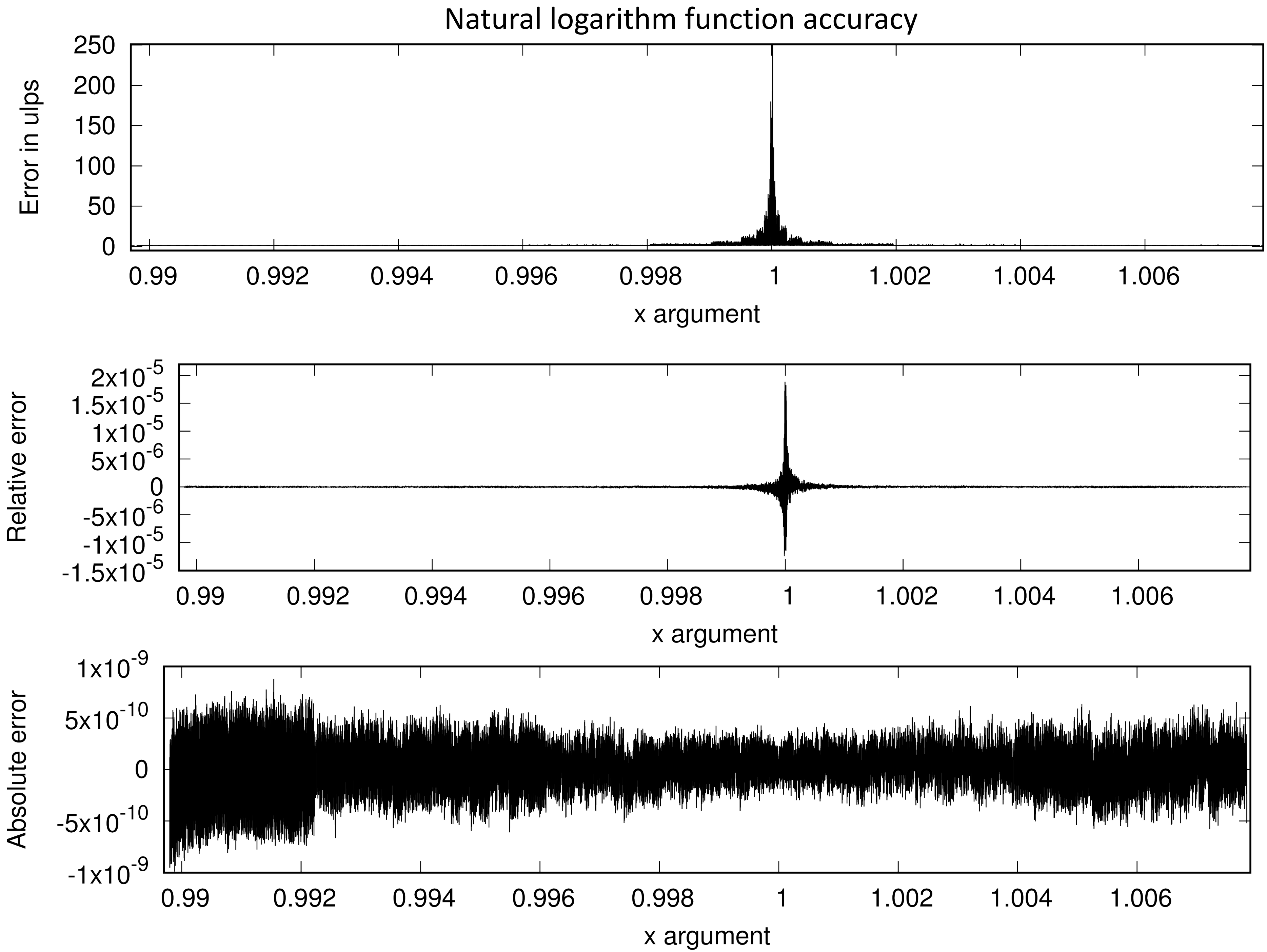
Function: $\log(x) \in (\sim -88(-104\ for\ denormal\ input), 88]\ with\ x \in (0, MAX_{FLOAT}]$.

Algorithm 2:

1. Find n such that $x' = \frac{x}{2^n}$ and $x' \in [1,2]$ (If x is normalized, $x' = 1.significand$).
2. Calculate $\log(x') \in [0, \sim 0.69]$.
3. (Range reconstruction) $\log(x) = \log(x' \cdot 2^n) = \log(x') + n \cdot \log(2)$.
4. Normalize the output and construct the exponent to convert to floating-point.

NOTE: When x is very close to 1, e.g. $x = 1 - \epsilon$, the logarithm will be computed as $\log(2 - 2\epsilon) - \log(2)$, resulting in **catastrophic cancellation** issue.

Solution: Before step 2, if the significand ($1.significand$) is close to 2, divide it by 2 and add 1 to the exponent. This way we make the significand, $x' \in [0.75, 1.5]$.


Natural logarithm function accuracy
Error in ulps
250
200
150
100
50
0
0.99
0.992
0.994
0.996
0.998
1
1.002
1.004
1.006
x argument
Relative error
2x10^-5
1.5x10^-5
1x10^-5
5x10^-6
0
-5x10^-6
-1x10^-5
-1.5x10^-5
0.99
0.992
0.994
0.996
0.998
1
1.002
1.004
1.006
x argument
Absolute error
1x10^-9
5x10^-10
0
-5x10^-10
-1x10^-9
0.99
0.992
0.994
0.996
0.998
1
1.002
1.004
1.006
x argument

Related work on accelerators in single-precision float

<table>
<tr><th>Work</th><th>Speed (MHz)</th><th>exp() latency (ns, cc)</th><th>log() latency(ns, cc)</th><th>Accuracy</th><th>Pipelined</th><th>Platform</th></tr>
<tr><td>Detrey et al. 2005</td><td>100</td><td>-</td><td>61, 7</td><td>1ulp</td><td>Y</td><td>FPGA</td></tr>
<tr><td>Detrey et al. 2007</td><td>100</td><td>123, 13</td><td>88, 9</td><td>1ulp</td><td>N</td><td>FPGA</td></tr>
<tr><td>Dinechin et al. 2010</td><td>~213</td><td>76, 16</td><td>-</td><td>1ulp</td><td>Y</td><td>FPGA</td></tr>
<tr><td>Langhammer& Pasca, 2017</td><td>~480</td><td>65, 31</td><td>52, 25</td><td>3ulp</td><td>Y</td><td>FPGA</td></tr>
<tr><td>This work (performance)</td><td rowspan="2">250</td><td colspan="2">52, 13</td><td rowspan="2">1ulp(exc. log for x≈1)</td><td rowspan="2">N</td><td rowspan="2">22nm</td></tr>
<tr><td>This work (leakage)</td><td colspan="2">84, 21</td></tr>
</table>

Summary

- Accuracy and energy results of the floating-point accelerator are presented.
- Two options are available: optimized for performance or optimized for leakage.
- Next steps would be to integrate the analysis into the whole SpiNNaker2 chip energy outlook and see which version of the accelerator is better – is it actually worth reducing the speed of functions for some leakage reduction?
- Is floating-point needed and worth paying for with leakage/area in general?

Questions?

(extra slide)Iteration unit architecture