Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Performance, Energy, and Accuracy of Non-Standard Computer Number Formats

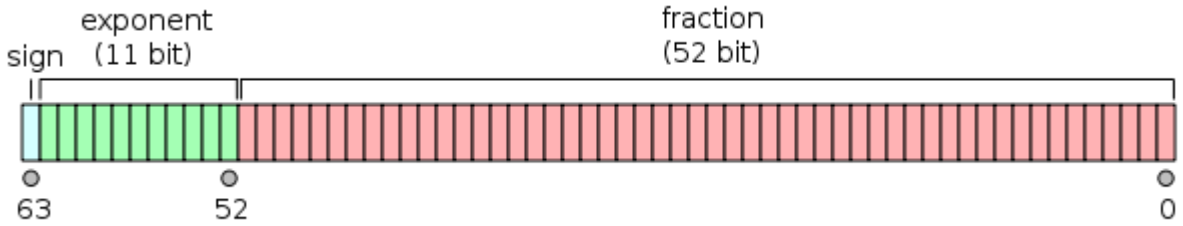
Marc Casas

SIAM Conference on Computational Science and Engineering (CSE23)

01/03/2023

The IEEE 754 Standard

- The IEEE Standard for Floating-Point Arithmetic (IEEE 754) is a technical standard for floating-point arithmetic established in 1985. It defines several floating-point formats, rounding rules, operations, and exceptions.
- The standard defines five basic formats in terms of numeric base and number of bits.
  - Three binary floating-point basic formats (encoded with 32, 64 or 128 bits).
  - Two decimal floating-point basic formats (encoded with 64 or 128 bits).

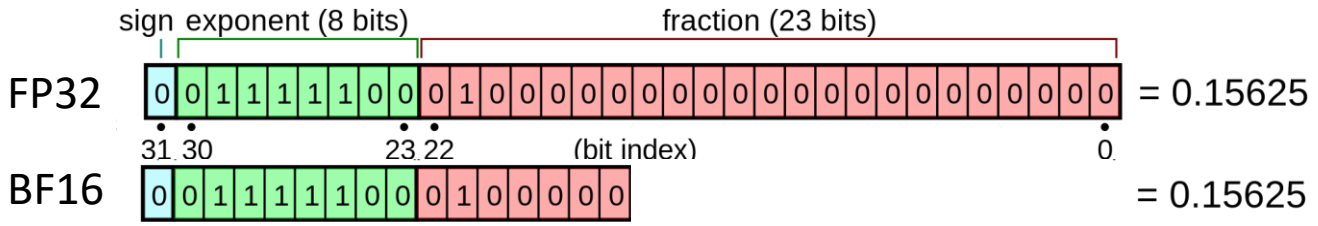


- Besides these formats, the IEEE 754 standard defines several rounding rules, operations, and exceptions.

William Kahan
Turing Award 1989

The BrainFloat16 Format

- To reduce DNN training costs, the BrainFloat16 (BF16) numerical data type is gaining popularity as it offers compelling precision and it is simple to convert from/to Floating Point-32 (FP32).



- Area of Fused Multiply-Add (FMA) hardware units grows quadratically with respect to the number of mantissa bits.

- BrainFloat16 is being adopted by major industrial organizations like Intel, Google, Arm, and others.

The Tapered Floating-Point Format

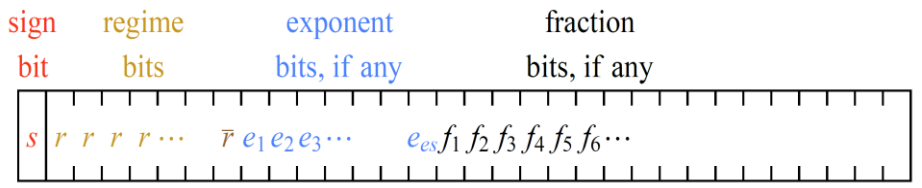
- Tapered Floating-Point (TFP) format is similar to standard floating-point, but with variable-sized entries for the significand and exponent instead of fixed-length entries.
- The idea of TFP was first proposed by Robert Morris in 1971.
- A TFP number is composed of 3 fields:
  - Exponent field: Contains es bits plus the exponent sign.
  - Fraction Field: Contains fr bits plus the fraction sign.
  - G Field: Contains g bits specifying the size of the es field.
- The total number of bits N is constant and $N = es + fr + g + 2$
- The maximum of bits devoted to represent the exponent is $2^g - 1$
- TFP allows the definition of computer number formats with more accuracy than IEEE 754 for certain intervals and less accuracy for other intervals.

The Posits Format

- The Posit format is an evolution of the tapered floating-point format.
- Posit was proposed by John Gustafson in 2017.
- Each particular posit format is defined by a tuple $<N, es>$
  - total number of bits of the format representation N.
  - maximum number of exponent bits es.
- Posit numbers are similar to classical floating-point scheme, except for a scaling factor that is defined by
  - Some regime bits.
  - The es parameter.

$$X = \underbrace{useed^{k}}_{\text{Scaling Factor}} \cdot \underbrace{(-1)^{Sign} \cdot 2^{e} \cdot (1 + fraction)}_{\text{Similar to floating point}}$$

$$\rightarrow \begin{cases} useed = 2^{2^{es}} \\ k \text{ is defined from the regime bits} \end{cases}$$

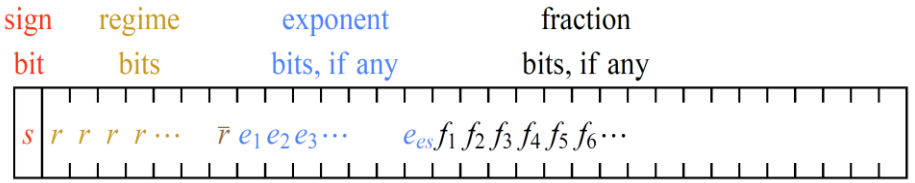


Taken from "Posit Arithmetic" by John Gustafson

The Posits Format

$$X = \underbrace{useed^{k}}_{\text{Scaling Factor}} \cdot \underbrace{(-1)^{Sign} \cdot 2^{e} \cdot (1 + fraction)}_{\text{Similar to floating point}}$$

$$\begin{cases} useed = 2^{2^{es}} \\ k \text{ is defined from the regime bits} \end{cases}$$

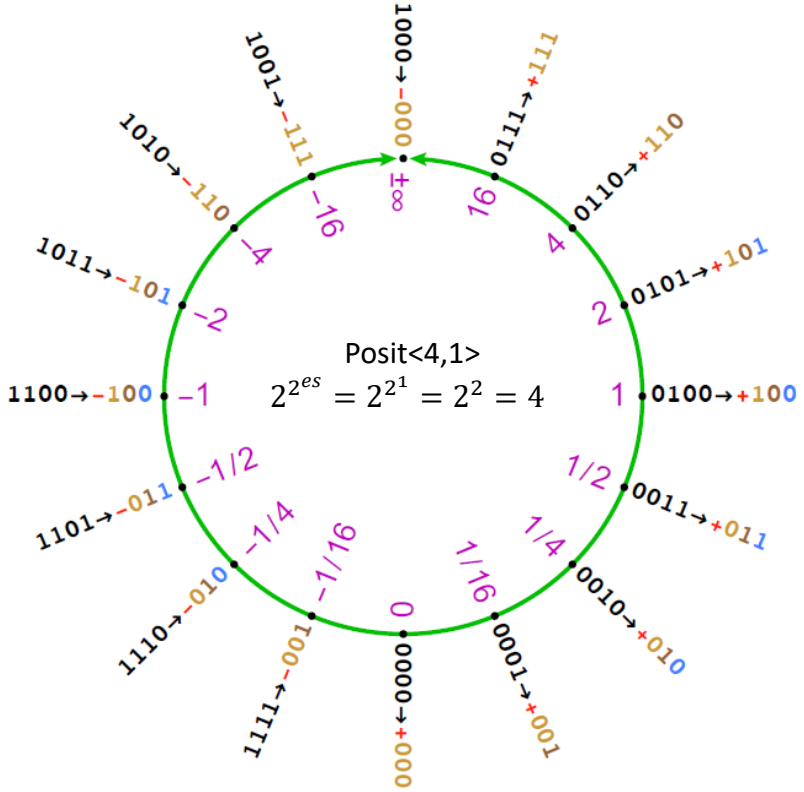


- The number of identical regime bits *r* determines k, a positive or negative power of $2^{2^{es}}$.

Binary	0000	0001	001x	01xx	10xx	110x	1110	1111
Numerical meaning, k	−4	−3	−2	−1	0	1	2	3

- Posit has two special values:
  - If all bits are **0**, the number represented is *zero*.
  - If only the first bit is **1** and the rest are **0**, the value is $\pm\infty$.
- Posit format allows to represent values close to 0 with high precision.

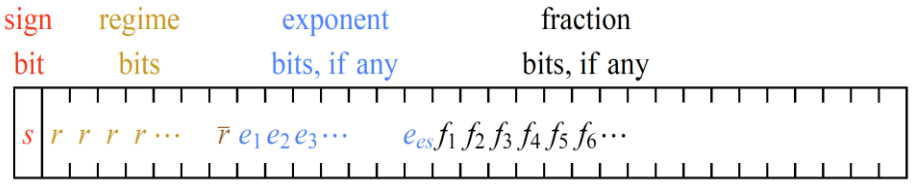
Taken from "Posit Arithmetic" by John Gustafson



The Posits Format

$$X = \underbrace{useed^{k}}_{\text{Scaling Factor}} \cdot \underbrace{(-1)^{Sign} \cdot 2^{e} \cdot (1 + fraction)}_{\text{Similar to floating point}}$$

$$\begin{cases} useed = 2^{2^{es}} \\ k \text{ is defined from the regime bits} \end{cases}$$

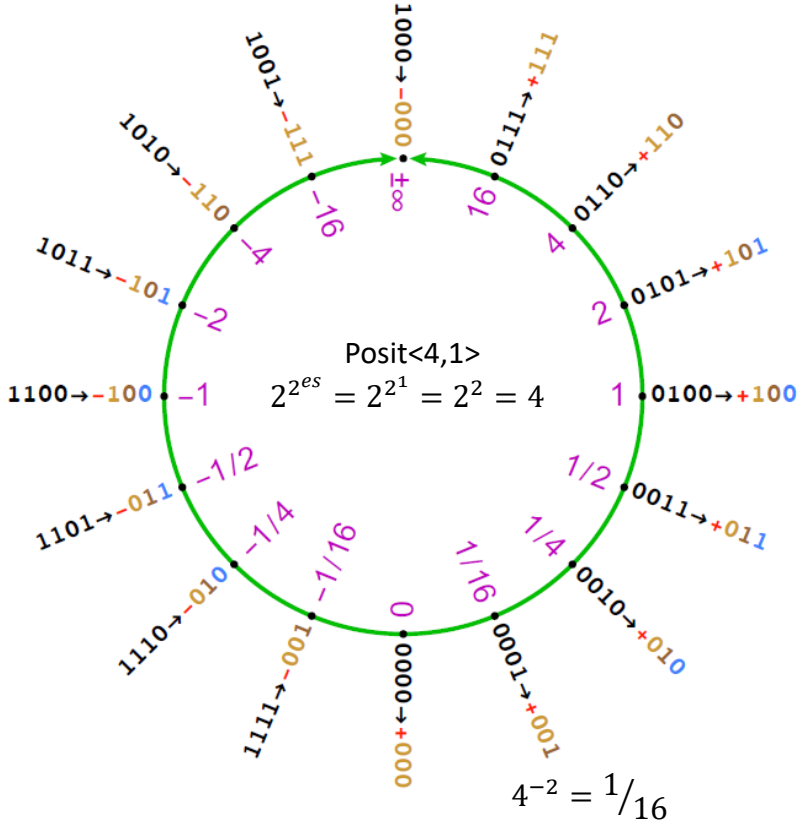


- The number of identical regime bits *r* determines k, a positive or negative power of $2^{2^{es}}$.

Binary	0000	0001	001x	01xx	10xx	110x	1110	1111
Numerical meaning, k	−4	−3	−2	−1	0	1	2	3

- Posit has two special values:
  - If all bits are **0**, the number represented is *zero*.
  - If only the first bit is **1** and the rest are **0**, the value is $\pm\infty$.
- Posit format allows to represent values close to 0 with high precision.

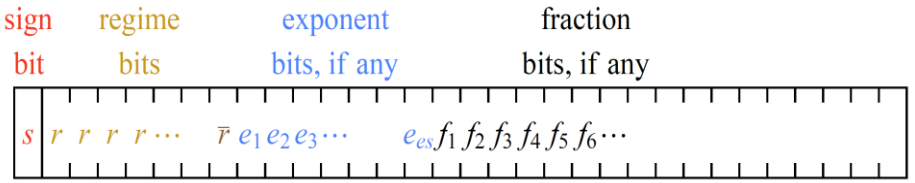
Taken from "Posit Arithmetic" by John Gustafson



The Posits Format

$$X = \underbrace{useed^{k}}_{\text{Scaling Factor}} \cdot \underbrace{(-1)^{Sign} \cdot 2^{e} \cdot (1 + fraction)}_{\text{Similar to floating point}}$$

$$\begin{cases} useed = 2^{2^{es}} \\ k \text{ is defined from the regime bits} \end{cases}$$

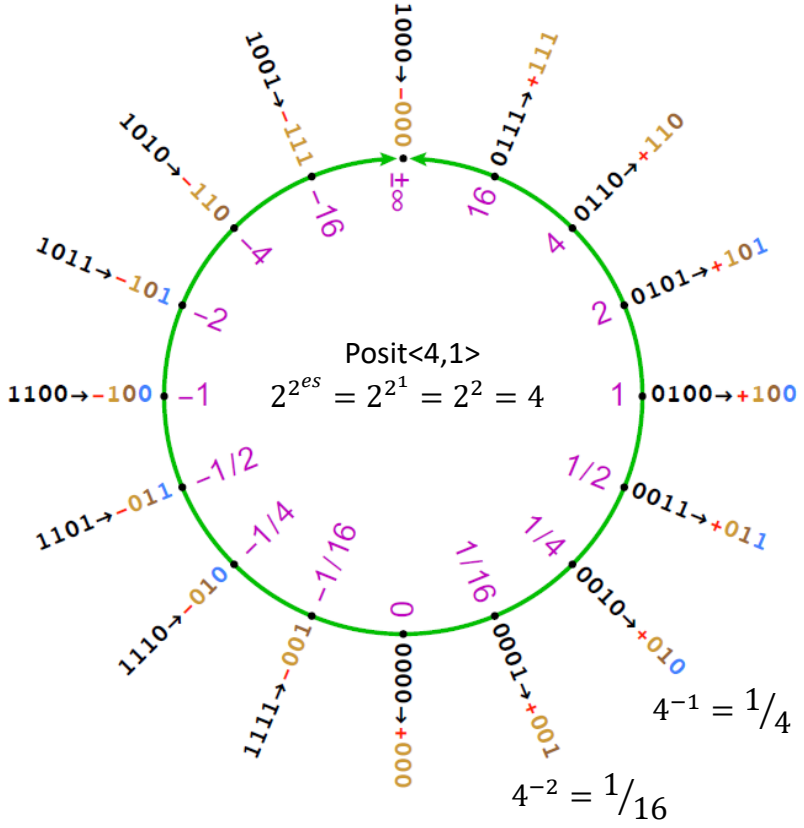


- The number of identical regime bits *r* determines k, a positive or negative power of $2^{2^{es}}$.

Binary	0000	0001	001x	01xx	10xx	110x	1110	1111
Numerical meaning, k	−4	−3	−2	−1	0	1	2	3

- Posit has two special values:
  - If all bits are **0**, the number represented is *zero*.
  - If only the first bit is **1** and the rest are **0**, the value is $\pm\infty$.
- Posit format allows to represent values close to 0 with high precision.

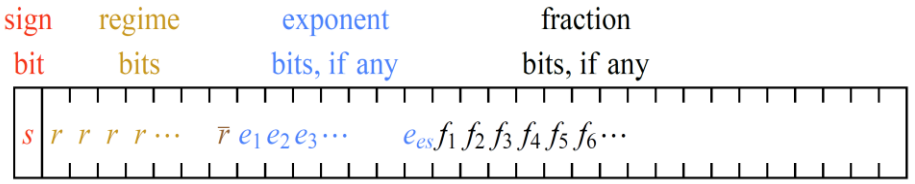
Taken from "Posit Arithmetic" by John Gustafson



The Posits Format

$$X = \underbrace{useed^{k}}_{\text{Scaling Factor}} \cdot \underbrace{(-1)^{Sign} \cdot 2^{e} \cdot (1 + fraction)}_{\text{Similar to floating point}}$$

$$\begin{cases} useed = 2^{2^{es}} \\ k \text{ is defined from the regime bits} \end{cases}$$

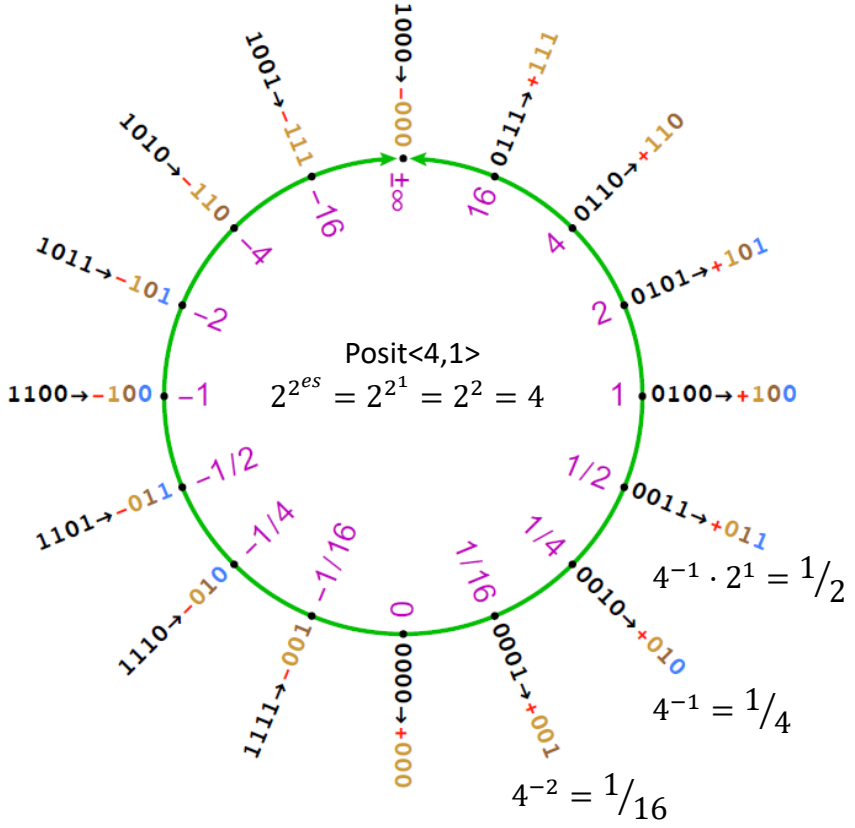


- The number of identical regime bits *r* determines k, a positive or negative power of $2^{2^{es}}$.

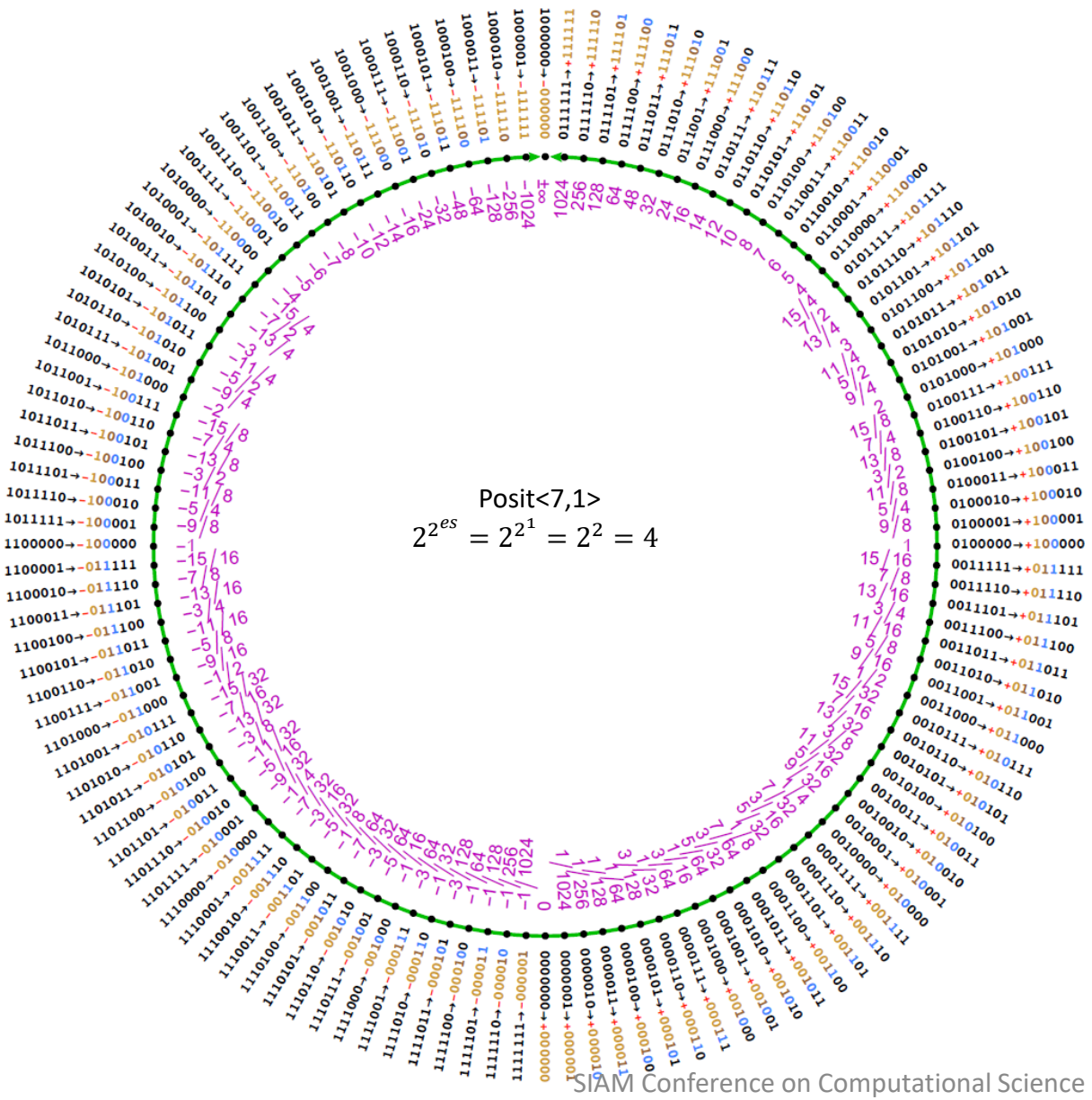
Binary	0000	0001	001x	01xx	10xx	110x	1110	1111
Numerical meaning, k	−4	−3	−2	−1	0	1	2	3

- Posit has two special values:
  - If all bits are **0**, the number represented is *zero*.
  - If only the first bit is **1** and the rest are **0**, the value is $\pm\infty$.
- Posit format allows to represent values close to 0 with high precision.

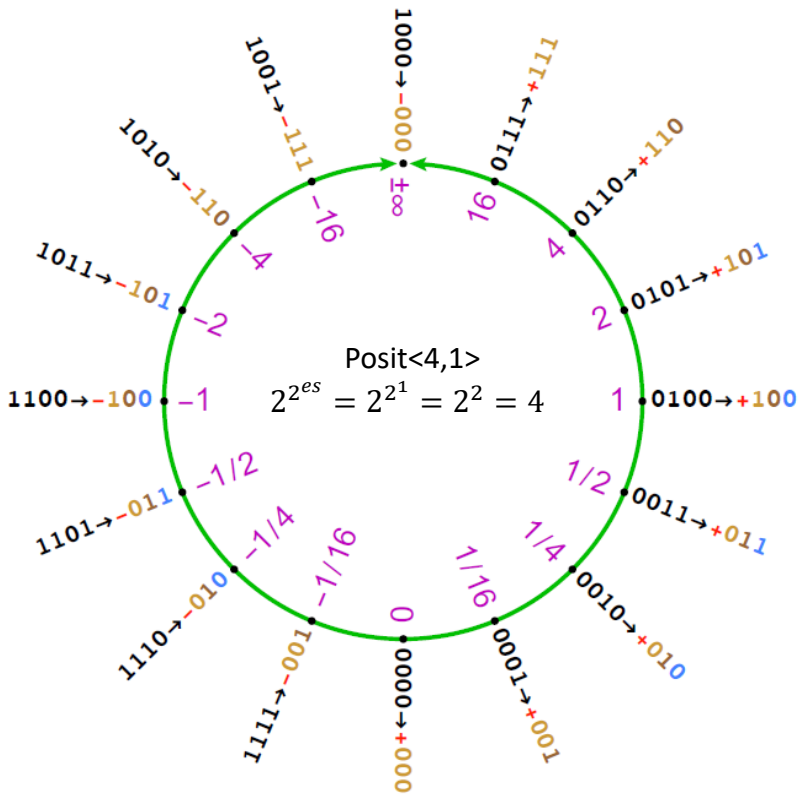
Taken from "Posit Arithmetic" by John Gustafson



The Posits Format

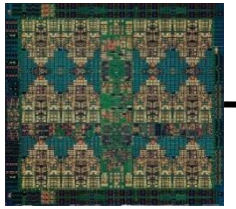


Taken from “Posit Arithmetic” by John Gustafson



Analysis of Floating-Point Number Formats

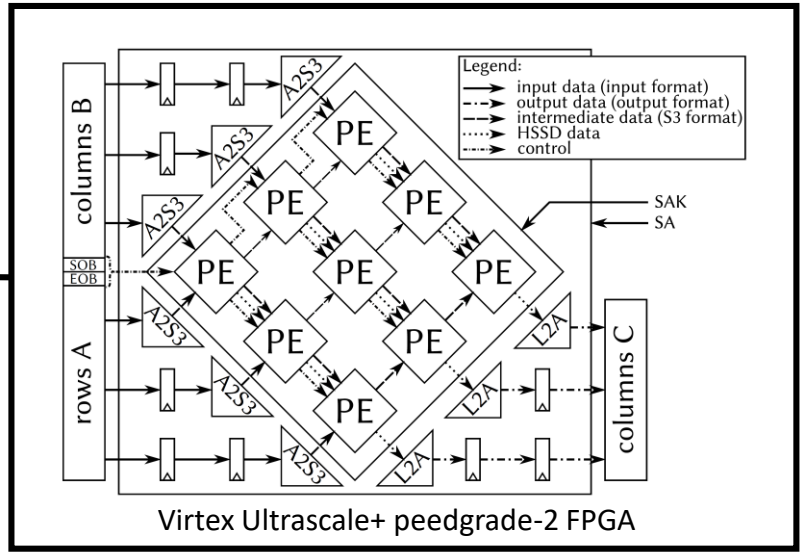
- We consider one of the most important kernels in the deep learning area, the General Matrix Multiplication (GEMM).
  - GEMM is defined as $C = \alpha \cdot A \cdot B + \beta \cdot C$ where A, B, C are matrices and α, β are scalar values.
  - GEMM is a fundamental building block for many operations in neural networks, for example fully-connected layers, recurrent layers such as RNNs, LSTMs or GRUs, and convolutional layers.
- We implement a wide set of floating-point number formats in a highly-reconfigurable FPGA-based GEMM accelerator composed of several Processing Elements (PE).
  - Each processing element contains logic to perform a Fused Multiply-Accumulate (FMA) operation on three registers:
    - $R_C = R_A \cdot R_B + R_C$
  - We send input data from a POWER9 chip host to the FPGA-based accelerator.
  - We get the GEMM output from the FPGA.
  - Many other features!



POWER9 chip host

15.75GB/s

PCIe link

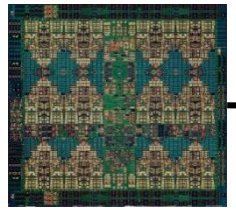


Virtex Ultrascale+ peedgrade-2 FPGA

L. Ledoux and M. Casas, "A Generator of Numerically-Tailored and High-Throughput Accelerators for Batched GEMMs,"
2022 IEEE 30th Annual International Symposium on Field-Programmable Custom Computing Machines (FCCM)

We consider 20 different formats and 2 accumulators

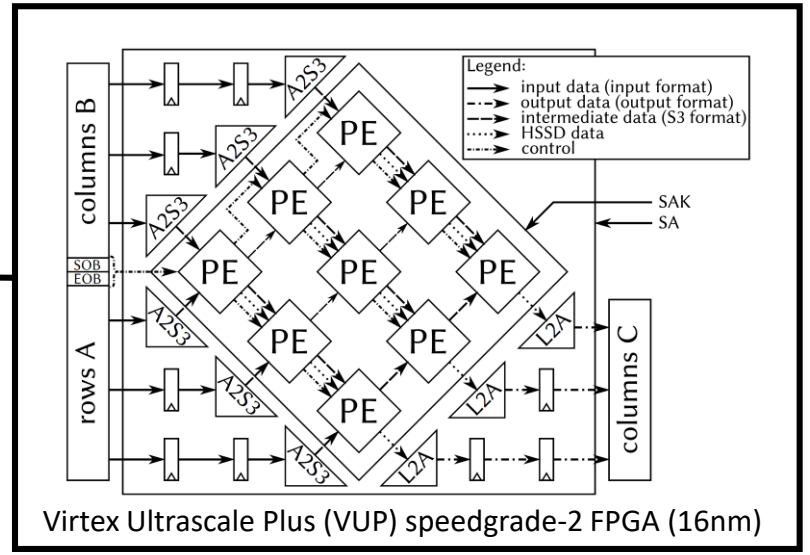
- We consider the IEEE 754, Bfloat16, TFP, and Posit formats.
  - IEEE 754: we consider the 8-, 16-, 32-, and 64-bit standard widths.
  - Bfloat16: we consider the only bitwidth it allows, 16-bit.
  - TFP: we consider 8-, 16-, 32-, and 64-bit widths.
  - For Posit we consider:
    - two 8-bit encodings: <8,0> and <8,1>,
    - three 16-bit encodings: <16,0>, <16,1>, <16,2>,
    - three 32-bit encodings: <32,0>, <32,1>, <32,2>,
    - three 64-bit encodings: <64,0>, <64,1>, <64,2>.



POWER9 chip host

15.75GB/s

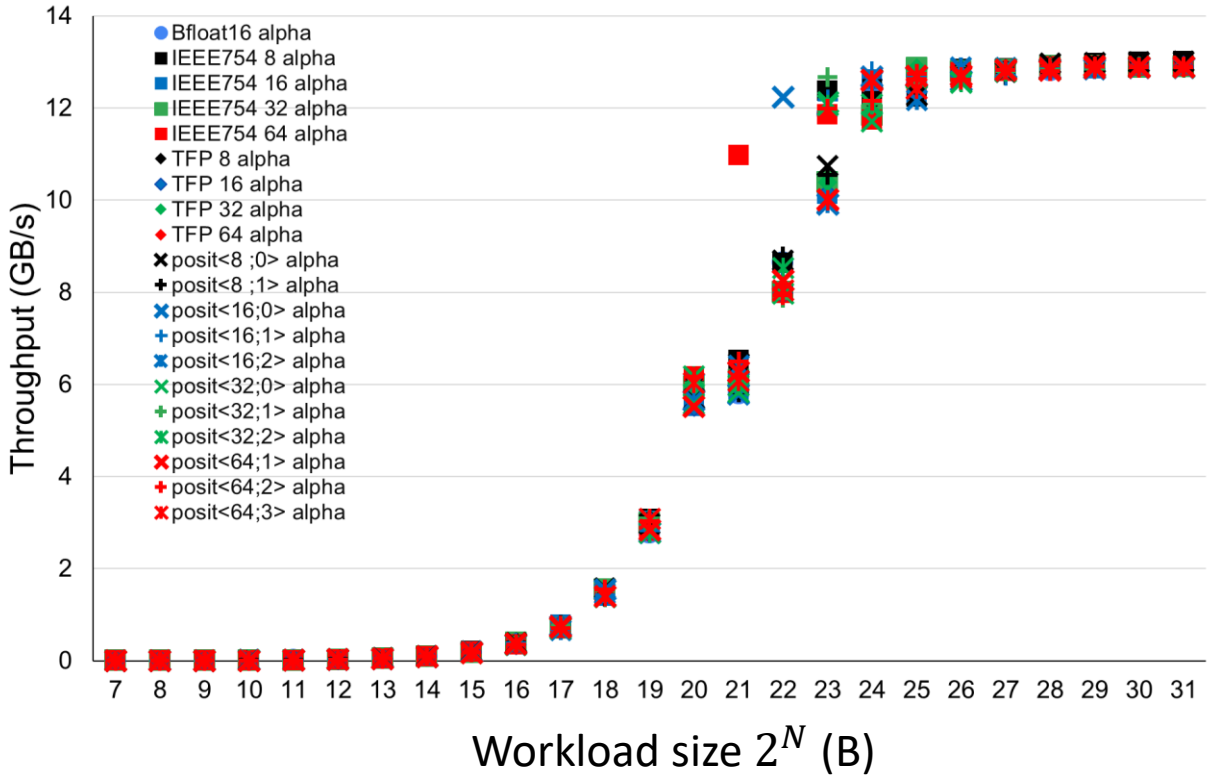
PCIe Link

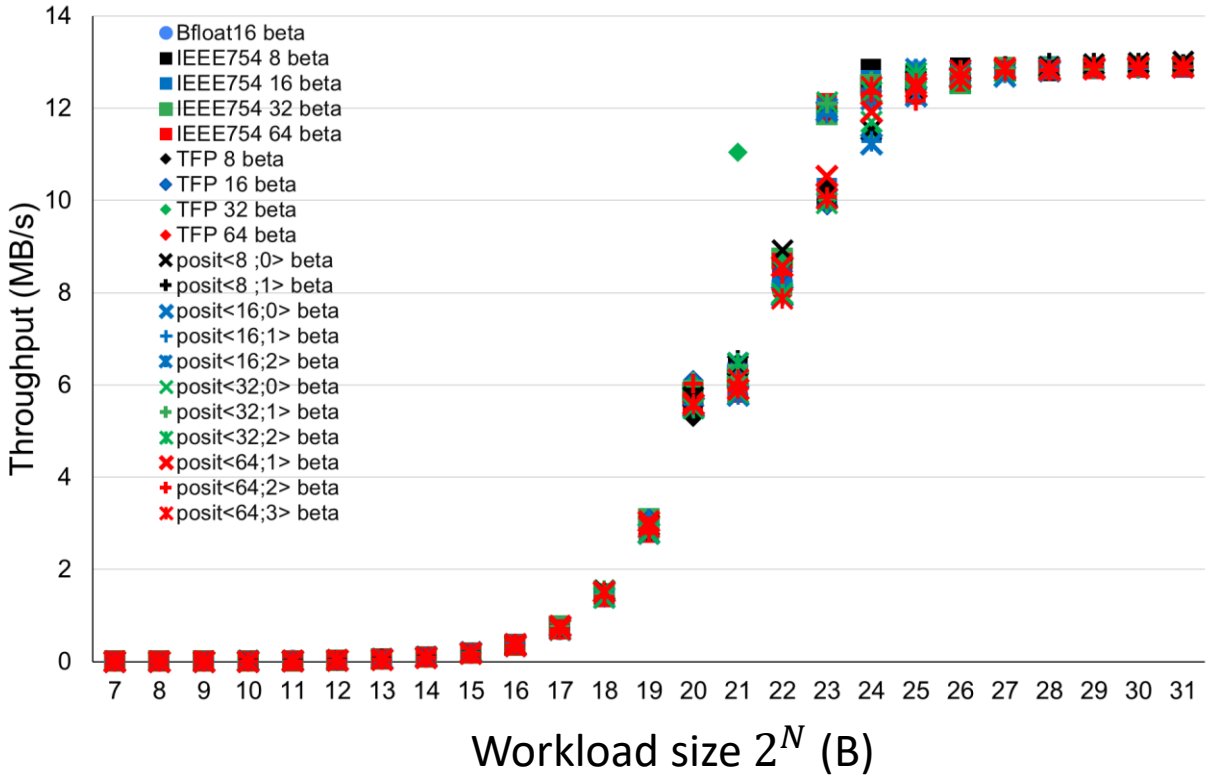


Virtex Ultrascale Plus (VUP) speedgrade-2 FPGA (16nm)

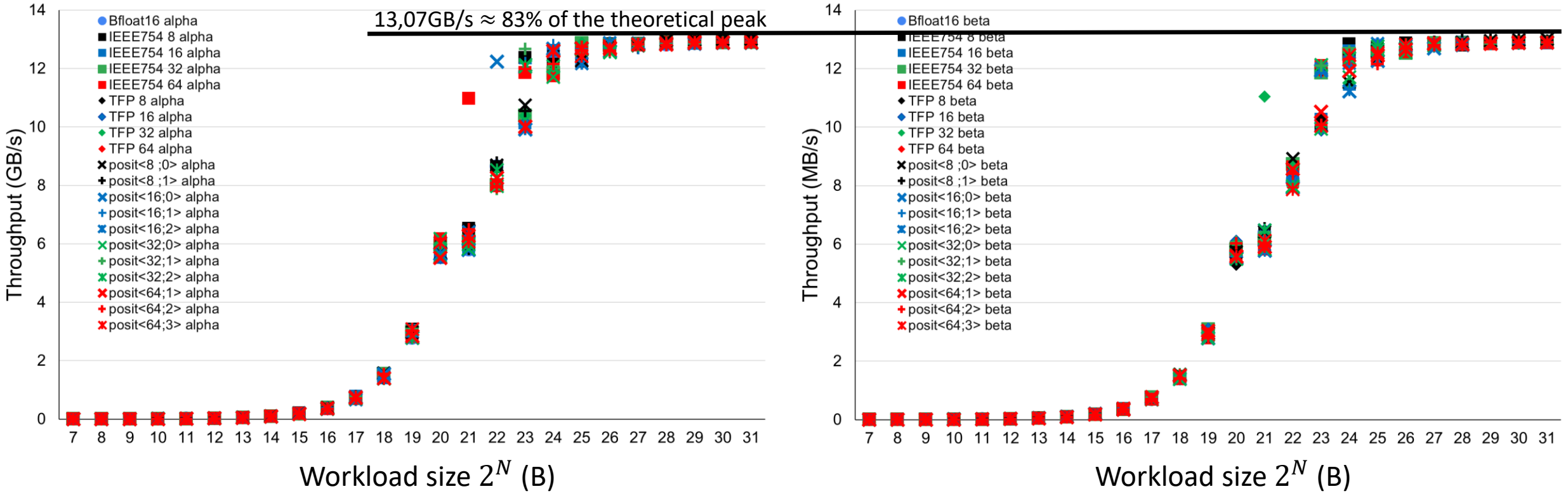
- We consider 20 different number formats!
- We consider two different versions of the FMA accumulation $R_C = R_A \cdot R_B + R_C$.
  - The size of R_C is two times the bitwidth the computer number format (α accumulator).
  - The size of R_C is long enough to fit the result of the $R_A \cdot R_B$ multiplication exactly (β accumulator).
- Input data is randomly generated using a random variable following a U[-2,2] distribution.

Physical Link Throughput in GB/s

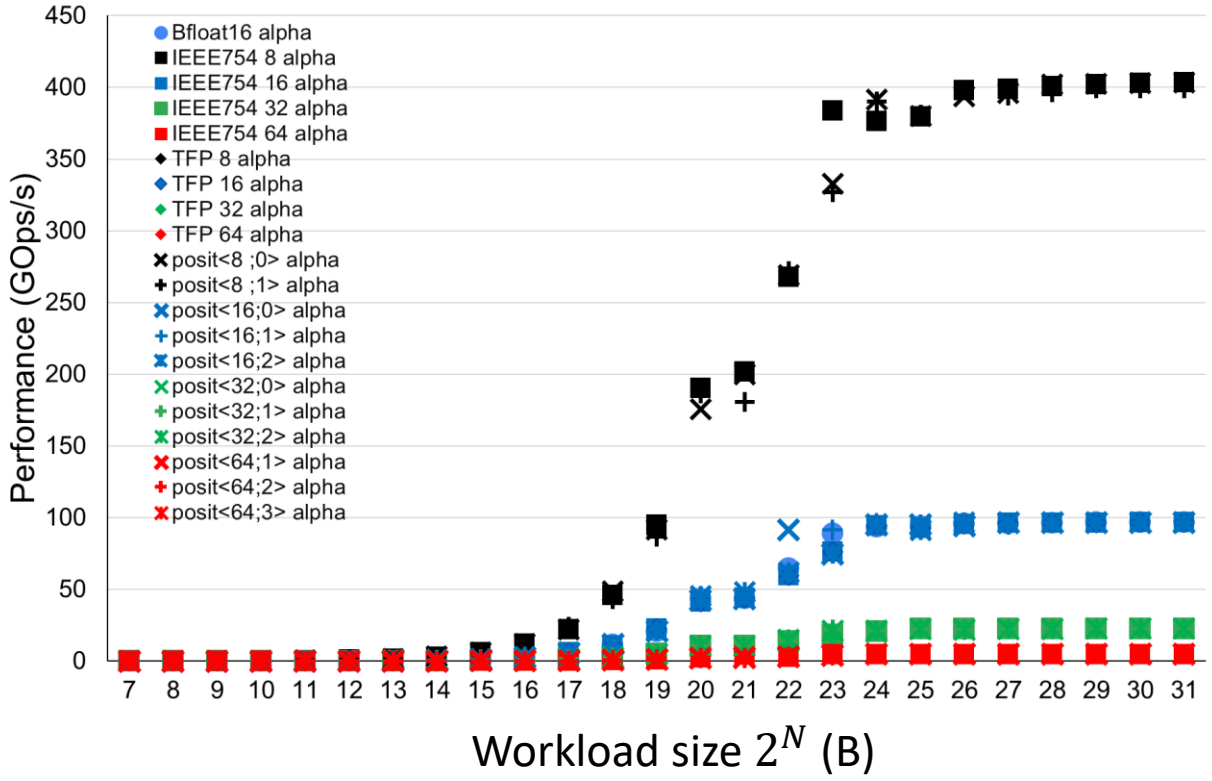


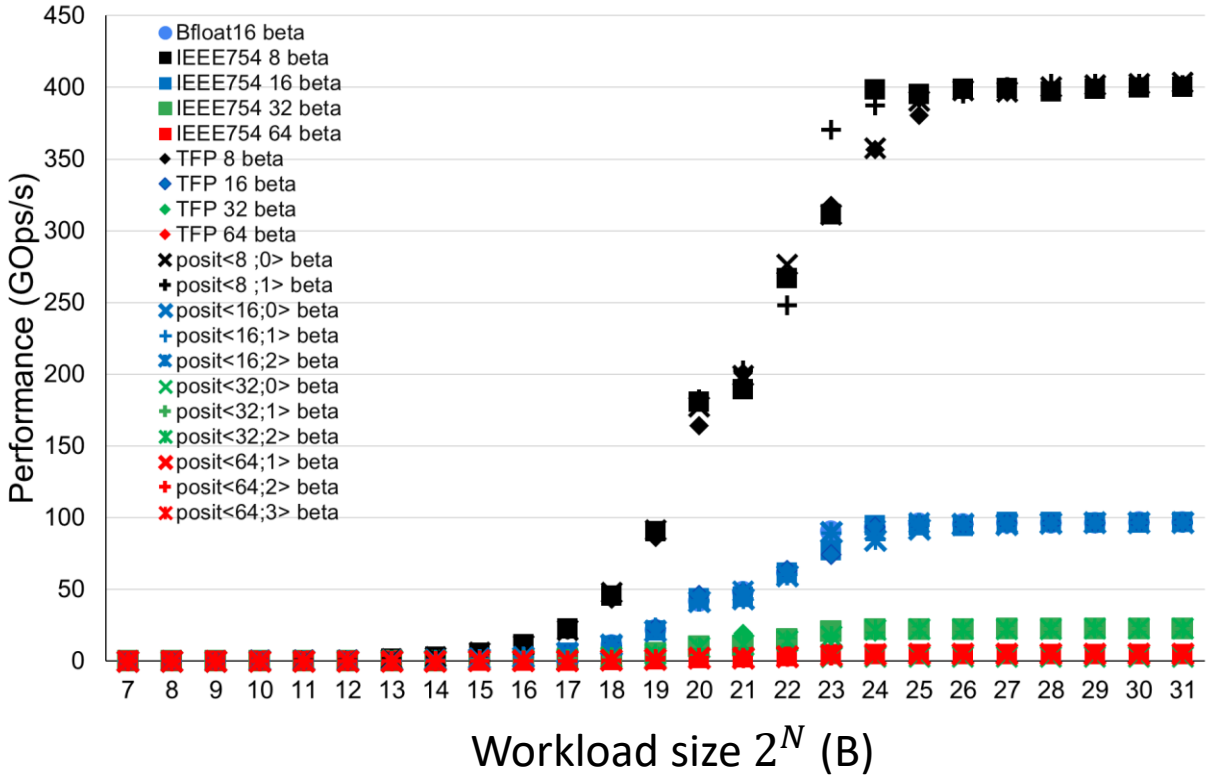


Physical Link Throughput in GB/s

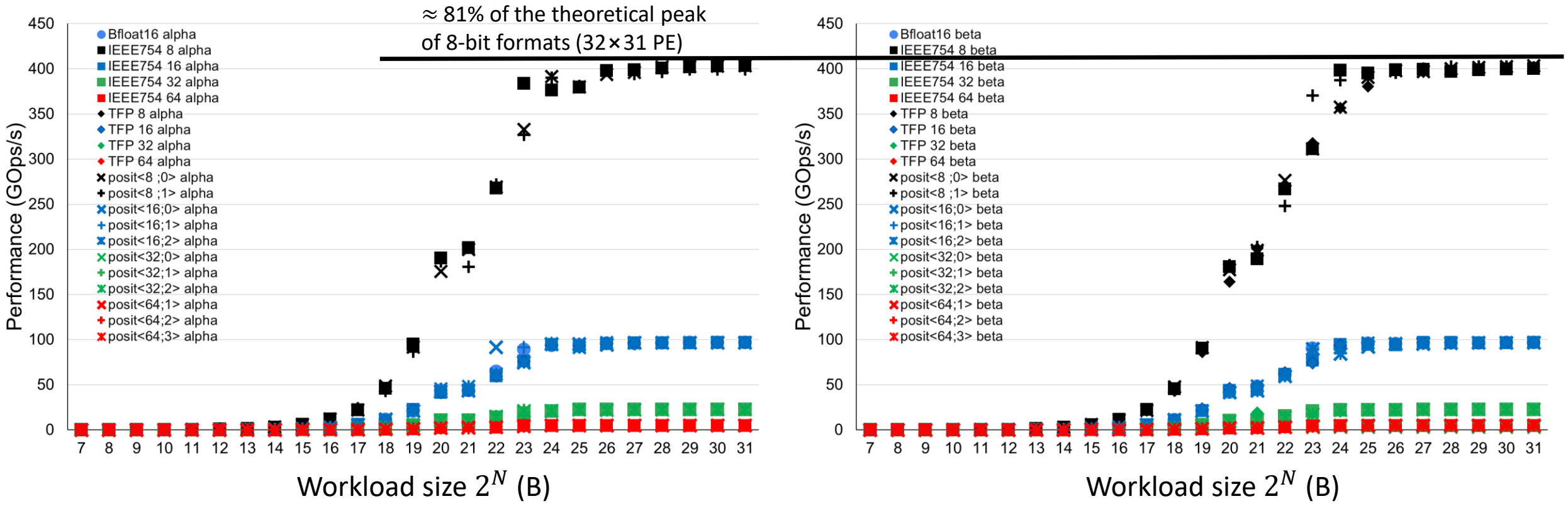


Performance in GigaOperations per second (GOps/s)

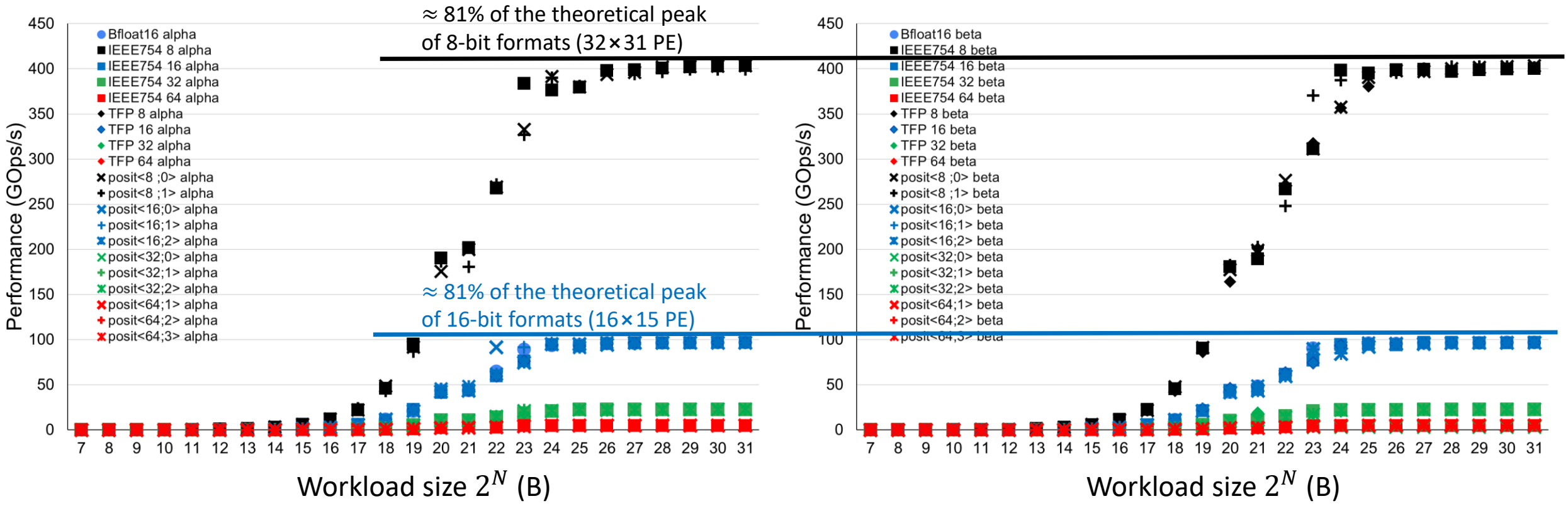




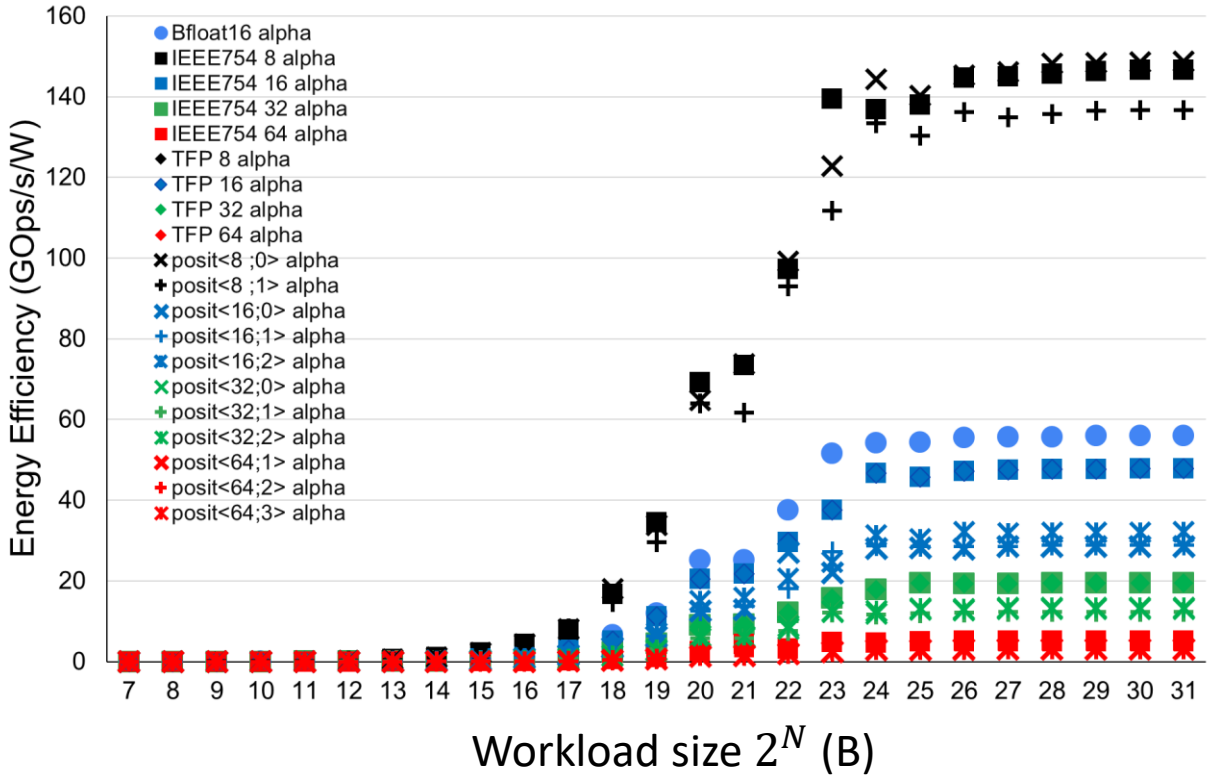
Performance in GigaOperations per second (GOps/s)

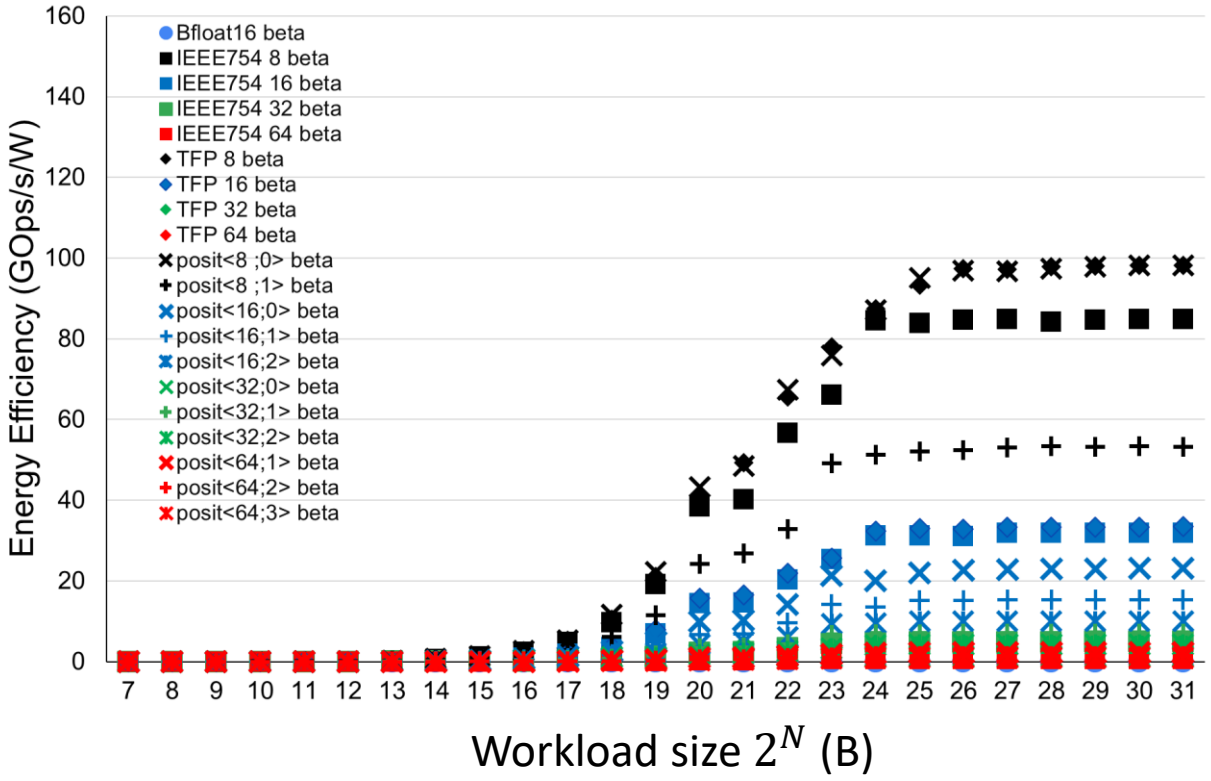


Performance in GigaOperations per second (GOps/s)



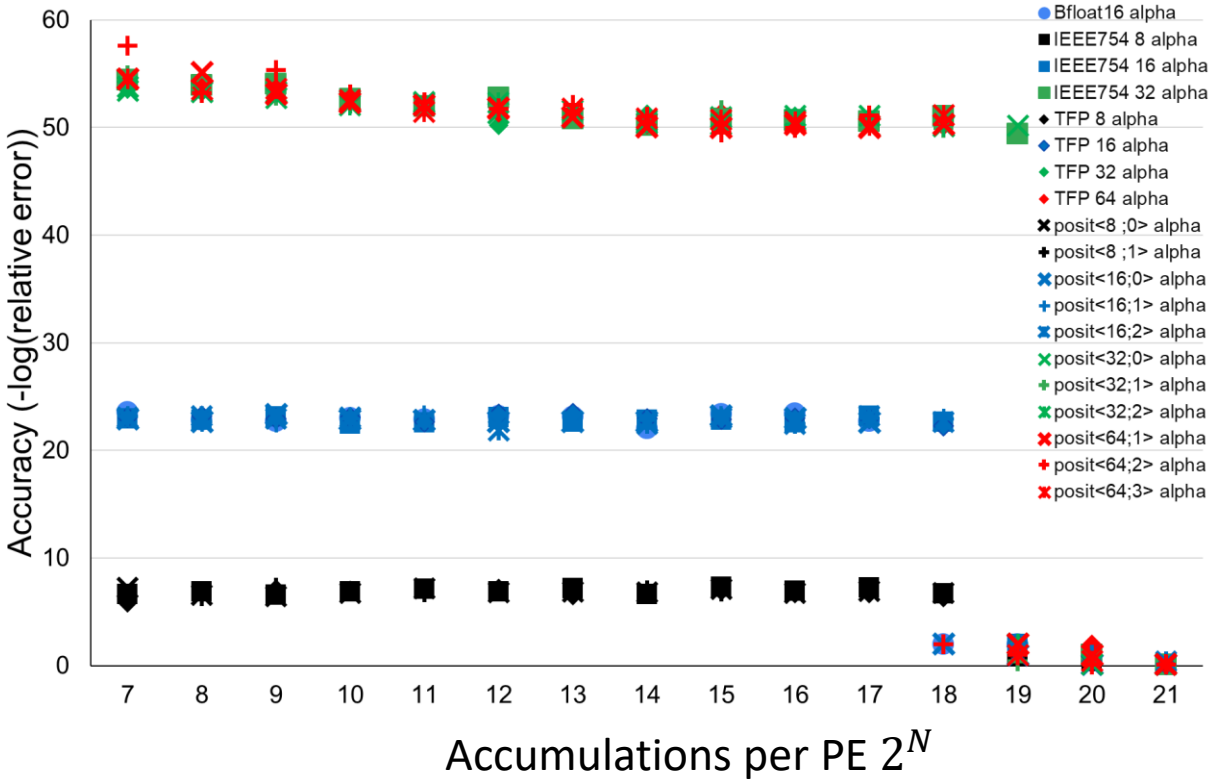
Energy Efficiency in GOps/s per Watt (GOps/s/W)

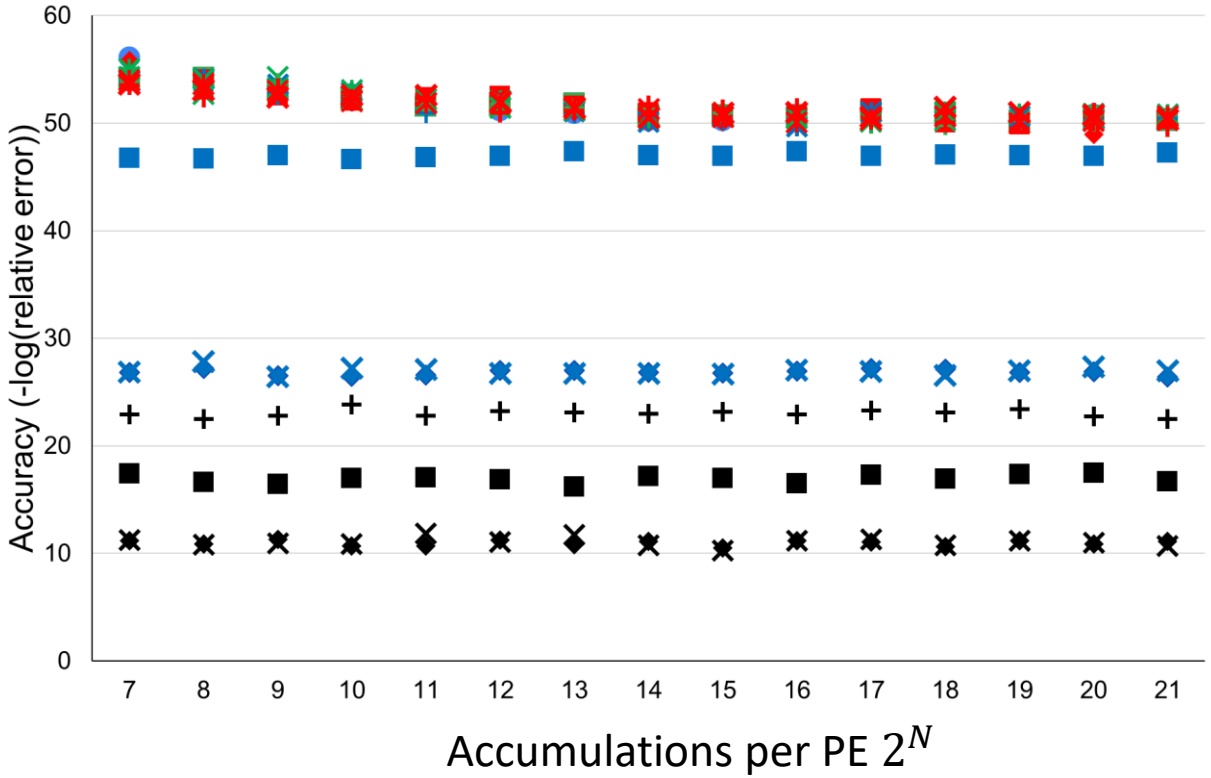




Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Accuracy with respect to IEEE 754 64-bits

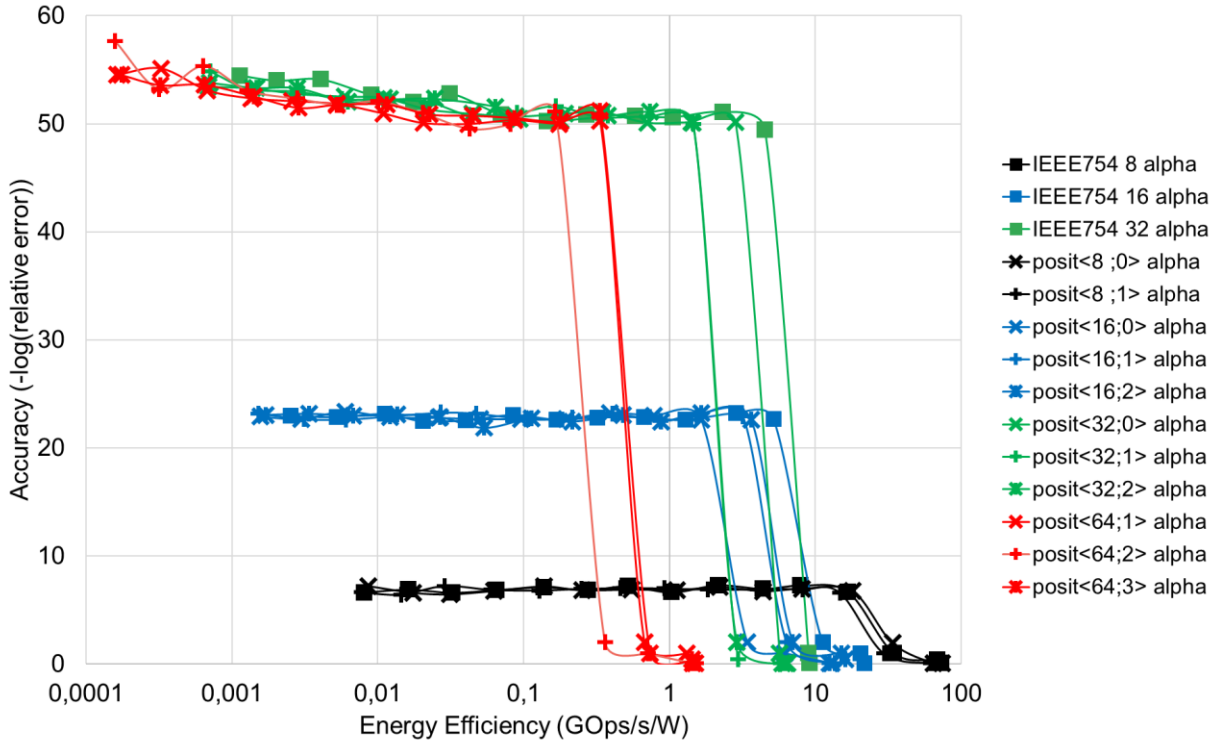


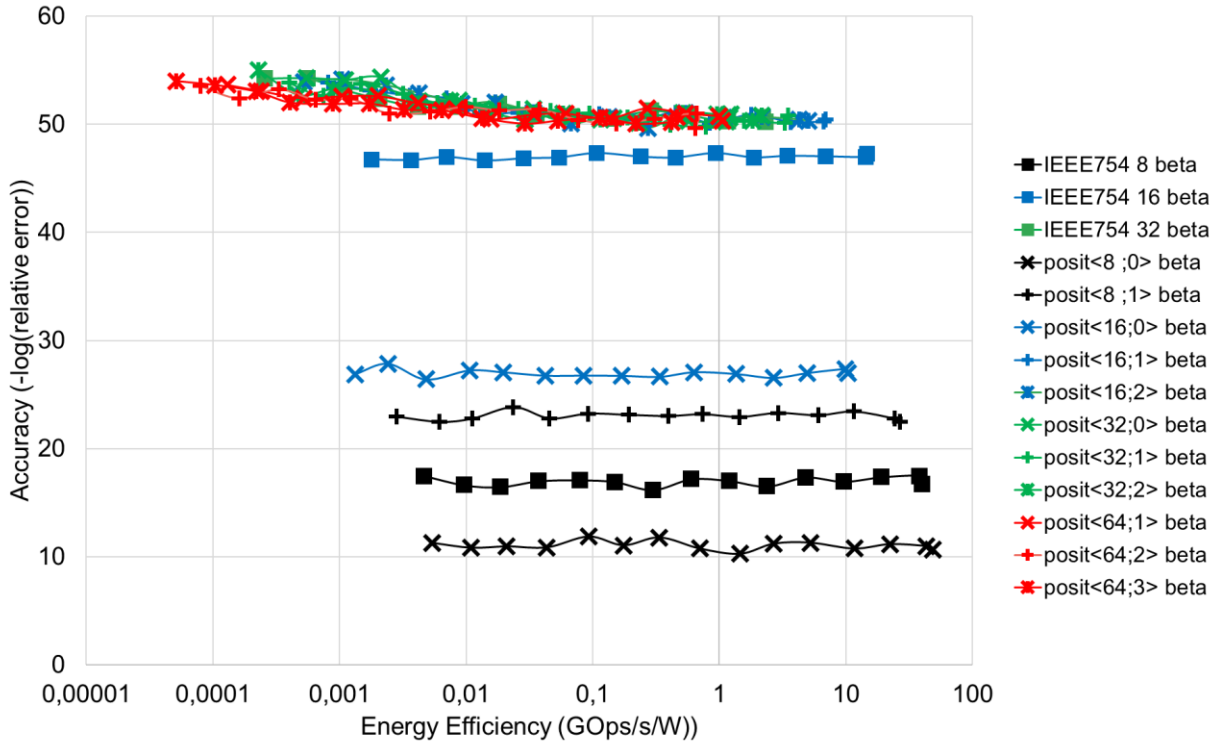


Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Energy Efficiency versus Accuracy

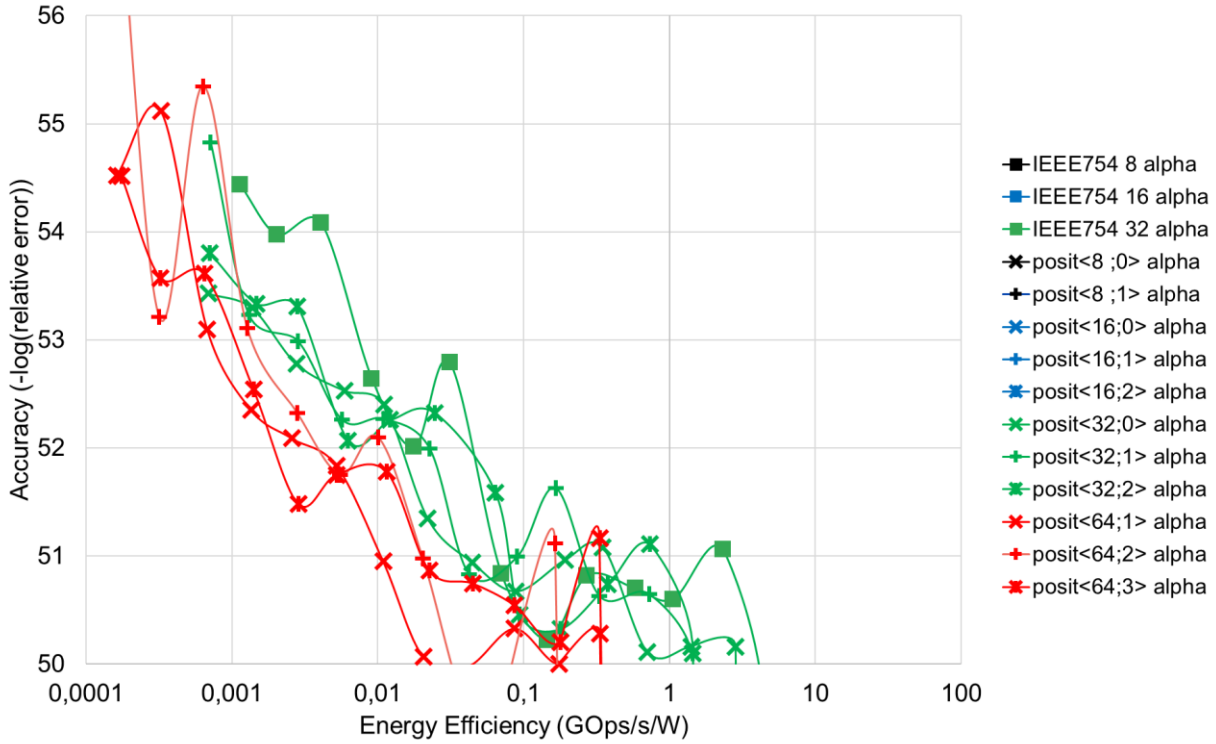
- We compare the IEEE 754 standard with Posits in terms of energy efficiency and accuracy.

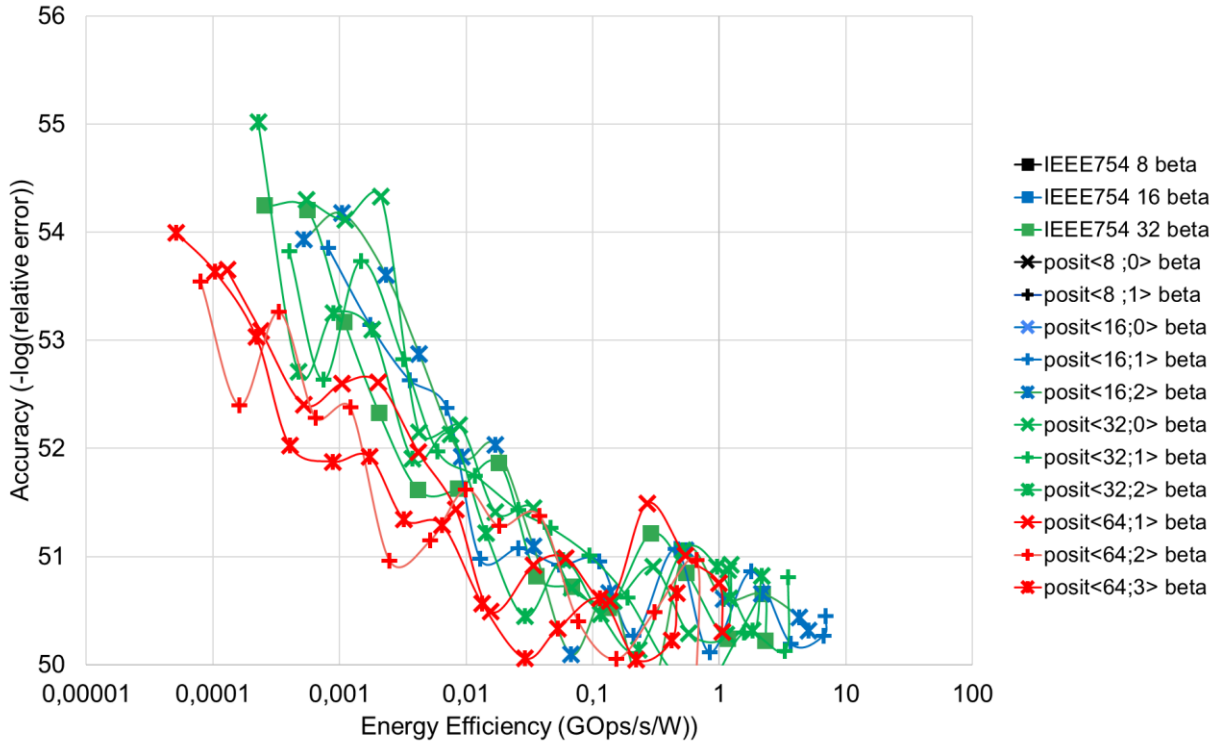




Energy Efficiency versus Accuracy

- We compare the IEEE 754 standard with Posits in terms of energy efficiency and accuracy.





Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Thanks

marc.casas@bsc.es