Tunable Floating Point for High Quality Audio Systems: The Sound of Numbers

Cardarilli, G. C.; Di Nunzio, L. ; Fazzolari, R.; La Cesa, R. ; Nannarelli, Alberto; Re, Marco

Published in:
Proceedings of 57th Annual Asilomar Conference on Signals, Systems, and Computers

Link to article, DOI:
10.1109/IEEECONF59524.2023.10476741

Publication date:
2024

Document Version
Peer reviewed version

Tunable Floating Point for High Quality Audio Systems: The Sound of Numbers

G. C. Cardarilli, L. Di Nunzio, R. Fazzolari, R. La Cesa, A. Nannarelli,[(1)] *and M. Re*

Department of Electronics, University of Rome Tor Vergata, Rome, Italy
[(1)]DTU Compute, Technical University of Denmark, Kongens Lyngby, Denmark

***Abstract*—The purpose of this work is to show the advantages of implementing digital signal processing for high quality audio applications in custom floating-point. We consider the trade-offs dynamic range vs. precision (i.e., quantization) by comparing standard floating-point (namely, *binary32*) to custom floating-point. Moreover, by resorting to Tunable Floating-Point (TFP) hardware units, we can adjust the dynamic range and the precision in different parts of the algorithms to explore several alternatives. Results show that 16-bit floating-point formats can give a good compromise between quality and energy efficiency.**

I. Introduction

In this work, we investigate the effects of the finite precision arithmetic representation for different floating-point representations in high quality audio applications.

Although the use of single or double precision floating-point (FP) in audio applications is well established, it is interesting to evaluate the impact of the recently introduced reduced-storage FP formats.

Machine Learning (ML) algorithms require a large number of operations on huge amounts of data. To limit the memory bandwidth and the energy footprint, the ML computation is progressively migrating from the traditional double and single precision FP to "ad-hoc" formats with reduced storage.

For example, in place of using single precision FP, *binary32* in the IEEE 754 standard [1], requiring 32 bits of storage, several ML applications resort to *binary16* (half precision) or *BFloat16* [2], requiring only 16 bits of storage.

In this work, we assess the use of 32/24/16 and even 8-bit storage formats for audio applications in terms of audio quality and energy efficiency. We take advantage of the Tunable Floating-Point (TFP) representation, introduced in [3], to simulate audio applications on actual variable-precision hardware units.

The experiments, are carried out both in the time and frequency domains. Moreover, to validate the analytical results, we performed a psychoacoustic test for selected audio samples done by 45 volunteers.

The results of the analysis and the test indicate that the audio quality in 16-bit storage formats is as good as in larger storage formats (namely, 32 and 24 bits) at lower power dissipation.

In contrast, acoustic tests of applications implemented in 8-bit storage formats are affected by crackling noise, and the overall audio quality is poor.

II. Binary Floating-Point Formats

The floating-point representation of a real number x is

$$x = (-1)^{S_X} \cdot M_X \cdot b^{E_X} \qquad x \in \mathcal{R}$$

where S_X is the sign, M_X is the significand or mantissa (represented by m bits), b is the base ($b = 2$ in the following), and E_X is the exponent (represented by e bits). The representation in the IEEE 745 standard [1] has significand normalized $1.0 \leq M_X < 2.0$ and biased exponent: *bias*$=2^{e-1}-1$.

The dynamic range of the representation is the ratio between the largest and the smallest (non-zero and positive) number [4]. The dynamic range for binary floating-point (BFP) is

$$DR_{BFP} = (2^m - 1) \cdot 2^{2^e - 1} \ . \tag{1}$$

For example, for *binary32* (32-bit storage with m=24, e=8) the dynamic range is

$$DR_{b32} = (2^{24} - 1)2^{2^8 - 1} \approx 9.7 \times 10^{83} \ .$$

In comparison, the dynamic range of the 32-bit fixed-point (FXP) representation is much smaller

$$DR_{FXP} = 2^{32} - 1 \approx 4.3 \times 10^9 \ .$$

In signal processing, the dynamic range is sometimes expressed in *deciBell* (*dB*). According to [5], the dynamic range, in *dB*, for n-bit fixed-point numbers is

$$DR_{FXP}(dB) = 6.02 \cdot n + 1.76 \ dB \simeq 6 \cdot n \ dB \ .$$

while for floating-point, the empiric expression is

$$DR_{BFP}(dB) = 6 \cdot (2^e - 1) \ dB \ . \tag{2}$$

The precision of a floating-point format is the weight of the *"unit in the last position" (ulp)* in the significand. Considering the normalization, the fraction bits of the significand are f=$m-1$, and consequently, the precision is expressed as $ulp = 2^{-f}$.

For our analysis, we consider some popular floating-point formats:

- **binary32**, or single precision, 24-bit significand (fraction bits f=$m-1$=23), 8-bit exponent. We consider in the following the representation for *binary32 "equivalent"* to the real number. In other words, we assume the representation error for *binary32* to be zero.

TABLE I
DYNAMIC RANGE AND PRECISION FOR THE SELECTED FORMATS.

Format	m	e	dynamic range (1)	dynamic range (2)	precision ulp
binary32	24	8	9.7×10^{83}	1530 dB	2^{-23}
PXR24	16	8	3.79×10^{81}	1530 dB	2^{-15}
binary16	11	5	4.3×10^{12}	186 dB	2^{-10}
BFloat16	8	8	1.48×10^{79}	1530 dB	2^{-7}
FP8m4e4	4	4	4.92×10^{5}	90 dB	2^{-3}
FP8m3e5	3	5	1.50×10^{10}	186 dB	2^{-2}
FXP 32-bit	32	0	4.3×10^{9}	192 dB	2^{-31*}

*Assuming 1 integer and 31 fractional bits (1.31)

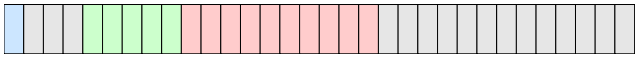

Fig. 1. Alignment of TFP32 numbers in 32-bit words: *binary16* (f=10, e=5) case. Biasad exponents (green) are sign-extended to 8 bits. Significands (red) are padded with zeros beyond bit of weight 2^{-f}.

- **PXR24** a 24-bit storage format, introduced by Pixar [6], with 16-bit significand (f=15), and 8-bit exponent.
- **binary16**, or half precision, 11-bit significand (fraction bits f=10), 5-bit exponent. The storage is 16 bits for *binary16*.
- **BFloat16**, Google's Brain-FP [2] a 16-bit storage format with 8-bit significand (f=7), and 8-bit exponent.

We also consider two 8-bit storage formats used by Nvidia [7] and Tesla [8] in their deep learning applications:

- **FP8m4e4** with m=4 (f=3) and e=4.
- **FP8m3e5** with m=3 (f=2) and e=5.

We summarize the dynamic range and precision for the above formats in Table I. In the table, we also included the 32-bit fixed-point (FXP) format for comparison purposes.

III. TUNABLE FLOATING-POINT

The Tunable Floating-Point (TFP) representation, introduced in [3], is a format in which the precision and the dynamic range of operands and results can be "tuned" by adjusting the number of bits for the significand and the exponent in the floating-point representation. For example, by tuning the precision of a given algorithm to the minimum precision achieving an acceptable target error, we can make the computation more power efficient.

The TFP representation is normalized to have the conversions compatible with the IEEE 754 standard. Moreover, TFP supports several rounding modes, including *roundTiesToEven*, the default mode in IEEE 754.

The basic format, TFP32, reserves 32 bits for storage and can handle significand precision from 24 to 3 bits, and exponent from 8 to 4 bits. The maximum precision and range (m=24, e=8) is that of *binary32*.

The TFP32 format include all the formats listed in Table I. The significands are rightward aligned to the decimal point (fractional numbers), while the exponents are leftward aligned to zero (integer numbers) as illustrated in Fig. 1 for *binary16*.

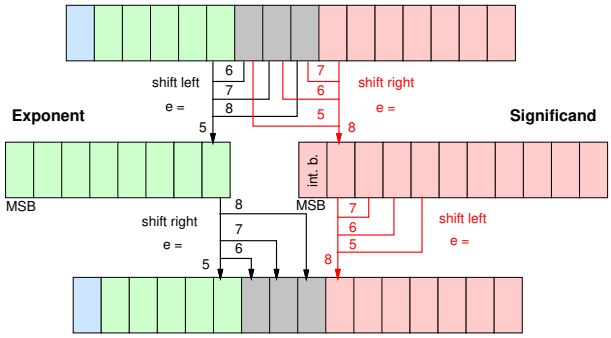

Fig. 2. TFP16 unpacking and re-packing in 16-bit storage format. Only LSB of exponent (black) and MSB of fraction (red) are shown in figure [9].

The 16-bit storage format, TFP16, introduced in [9], comprises four formats: *binary16*, IBM's *DLFloat16* [10] (m=10, e=6), *TFPm9e7* (m=9, e=7), and *BFloat16*. Differently than TFP32, in TFP16 the 16-bit operands are expanded (unpacked) in the processing units and re-packed before storage, as shown in Fig. 2.

IV. THE EXPERIMENTS

In this section, we describe the experiments carried out to explore the influence of the different representation formats in the audio domain.

We start by characterizing a sinusoidal signal converted to the floating-point formats in Table I, and compute the Total Harmonic Distortion (THD). We consider both the quantization of the sinusoidal signal and the impact of arithmetic operators, such as adders and multipliers, on the THD.

As a second experiment, we compute the quantization error introduced by the different FP representations on the time-domain response of a typical filter used in audio processing.

The third experiment is a psychoacoustic test performed on actual audio tracks, generated for the different FP formats, to compare the representations auditorily as well.

A. Total Harmonic Distortion (THD)

In the first experiment, we estimate the THD introduced by the different floating-point representation formats in Table I.

The THD evaluation is performed by converting a sine wave, with frequency of 900 Hz and sample frequency of 48 kHz, to the various formats and analyzing the resulting spectra [11].

The spectra of the selected floating-point formats are shown in Fig. 3 together with the computation of the THD introduced by the different floating-point representations. The spectra are obtained by using the **thd()** Matlab function that returns the THD in *dBc*[1] of the sinusoidal signal. The THD is computed from the fundamental frequency and the first five harmonics using a modified periodogram with the same length of the

[1] *dBc*: deciBells relative to the carrier.

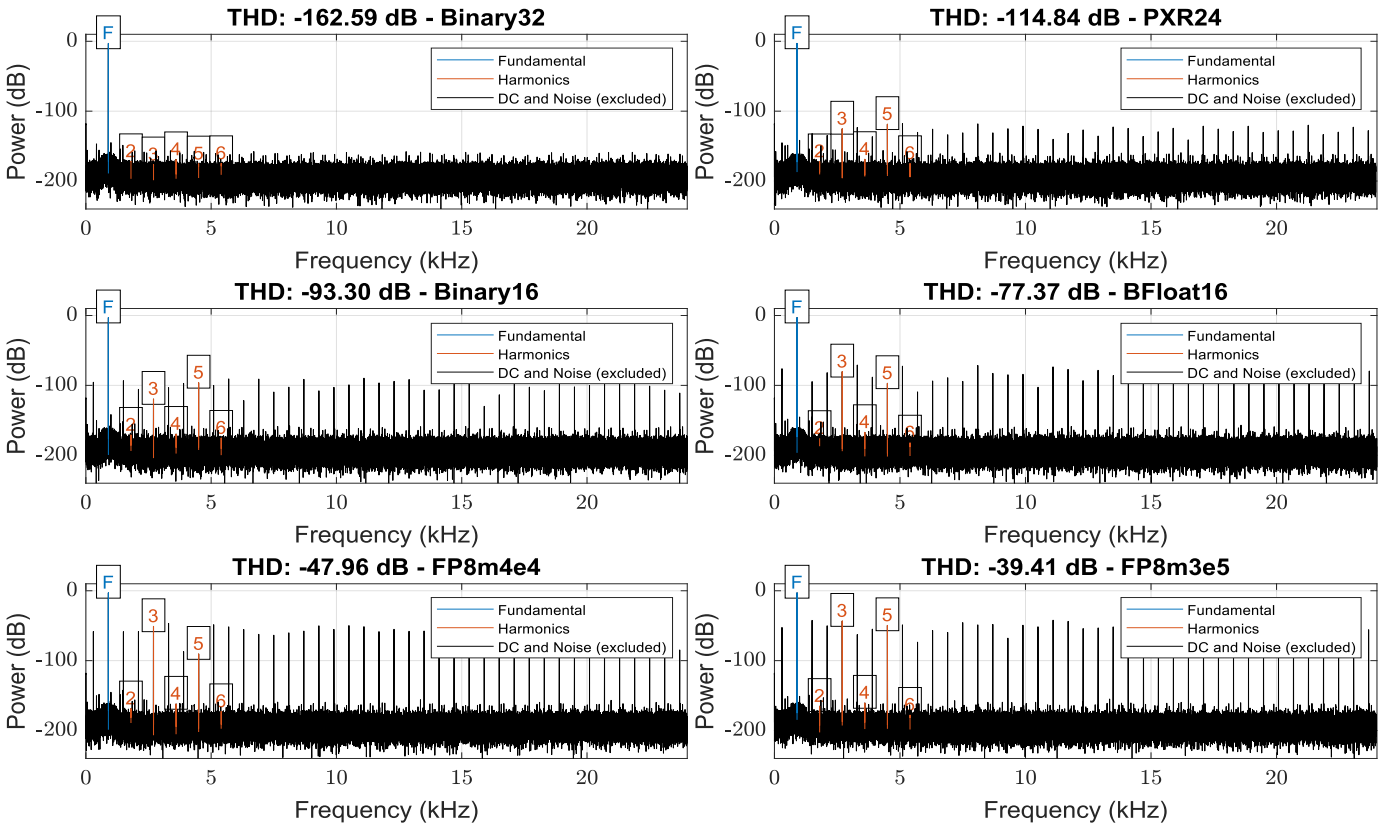

Fig. 3. THD (from top-left) for *binary32*, *PXR24*, *binary16*, *BFloat16*, *FP8m4e4*, and *FP8m3e5*.

input signal. The modified periodogram uses a Kaiser window with $\beta = 38$ [12].

Fig. 3 displays that the different FP formats introduce even- and odd-order harmonics associated to the format of the representation. The distortion increases as the number of storage bits decreases and the precision of the representation drops.

To examine the impact of arithmetic operators (adders and multipliers) on the THD, we conducted experiments using a bi-quadratic IIR filter, a commonly utilized component in audio applications.

Specifically, the filter is a second-order Butterworth IIR filter, digitally implemented in the direct form of Type 1, with cut-off frequency $F_{c\text{-}off} = 10\ kHz$ and quality factor $Q = \frac{1}{\sqrt{2}}$.

We processed the sinusoidal signal through the IIR biquad filter for the different formats, and generated the THDs. The resulting spectra resemble those in Fig. 3, but the THD values worsen due to the impact of the arithmetic operators. The THD comparison for the two cases (wave only, and filtered wave), is depicted in Fig. 4.

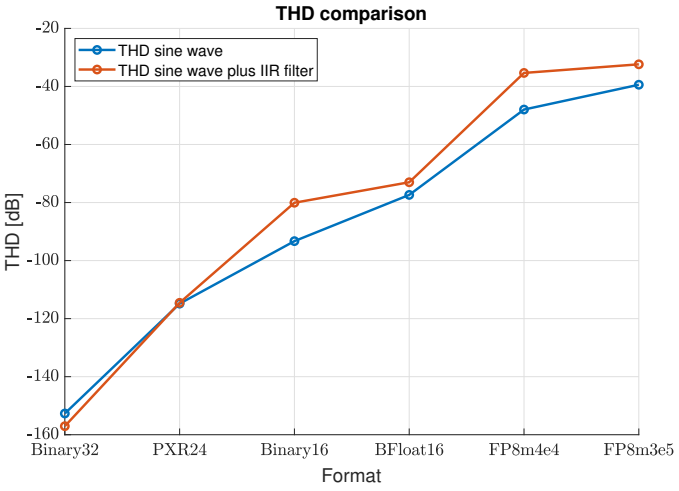

Fig. 4. Comparison of THDs. Sinusoidal signal only, and IIR filtered signal.

B. Audio Track Played through an IIR Filter

An audio track (about 9 s. duration), sampled at 48 KHz, is filtered by the IIR biquad filter specified in Sec. IV-A.

The audio track is processed by the filter (in Matlab) and the following traces/info are extracted:

- A trace of the operations executed in the filter: a sequence of multiplications and additions.
- A trace file with TFP test vectors to be used for the VHDL simulations of the hardware units in Sec. V.
- Statistics on the errors with respect to the *binary32* representation.

The error estimate is carried out by reading the *binary32*

TABLE II
ERRORS (MAX. AND AVERAGE) WITH RESPECT TO *binary32* FOR OTHER FP-FORMATS.

Format	m	e	ϵ_{max}	ϵ_{ave}
binary32	24	8	-	-
PXR24	16	8	$1.41 \times 10^{-5} < 2^{-16}$	$6.93 \times 10^{-7} < 2^{-20}$
binary16	11	5	$2.98 \times 10^{-4} < 2^{-11}$	$2.23 \times 10^{-5} < 2^{-15}$
BFloat16	8	8	$3.02 \times 10^{-3} < 2^{-8}$	$2.18 \times 10^{-4} < 2^{-12}$
FP8m4e4	4	4	$3.91 \times 10^{-2} < 2^{-4}$	$3.43 \times 10^{-3} < 2^{-8}$
FP8m3e5	3	5	$3.91 \times 10^{-2} < 2^{-4}$	$2.97 \times 10^{-3} < 2^{-8}$

TABLE III
RESULTS OF THE MUSHRA PSYCHOACOUSTIC TEST (45 PARTICIPANTS).

Format	Basic Audio Quality			
	ROCK	FEVER	SONNO	Average
binary32	87.45	94.48	91.69	91.21
PXR24	89.14	92.59	90.64	90.79
binary16	86.52	93.24	90.74	90.16
BFloat16	89.29	86.48	85.43	87.06
FP8m4e4	24.17	19.64	13.62	19.14
FP8m3e5	26.83	20.17	12.17	19.72

reference trace, and by reading a second trace with one of the other formats-Under-Test (fUT). By calling z_{b32} the result of the specific operation executed in *binary32*, and z_{fUT} the result executed in another format, we compute the error (absolute value) as: $\epsilon = |z_{b32} - z_{fUT}|$.

Analyzing the whole trace, we determine the maximum error ϵ_{max}, and the average error ϵ_{ave}.

The maximum and average errors for the different FP formats with respect to *binary32* are reported in Table II.

By examining Table II, we can notice that the maximum error corresponds to $\frac{ulp}{2}$, i.e., the weight of bit in position 2^{-m}. For the *FP8m4e4* format, the average error is worse than for *FP8m3e5* because, due to the limited dynamic range of the representation, about 18% of operations produce an underflow and the results are flushed to 0 (ϵ_{ave} increases).

C. Psychoacoustics Tests for Several Audio Tracks

The error rates in Table II show that when precision and dynamic range decrease the degradation becomes large from an analytical viewpoint. However, it is important to know what level of degradation is audible by humans.

For this reason, we set up a MUSHRA (Multiple Stimuli with Hidden Reference and Anchor) listening test [13]. The test consists in rating the Basic Audio Quality (BAQ) of audio samples produced in different conditions. The BAQ is a single and global attribute that is used to judge any and all detected differences between the reference audio sample and samples with modified conditions.

The listener is presented three different audio fragments: "ROCK", "FEVER" and "SONNO". For each fragment, the reference audio sample is provided along with the samples in the different TFP formats. The listener must provide a BAQ score on a scale from 0 to 100 ("excellent"). The test is available on-line at *https://mushra.inglacesa.it/* [14], and its results are reported in Table III. In the table, the BAQ score is averaged for all listeners (45 participants) for the single fragment, and averaged for the three fragments (last column).

The BAQ scores in Table III indicate that the quality of the audio is in the "excellent" range for the 32/24/16-bit formats (BAQ above 85). In contrast, for 8-bit storage formats, the scores between 12 and 27 indicate poor audio quality (e.g., presence of crackling noise).

V. TUNABLE FLOATING-POINT UNITS

Hardware support for floating-point arithmetic is normally more expensive than support for fixed-point. However, if the extra costs are not in excess, floating-point may be a good trade-off for flexibility and to adapt the best format to different parts of the algorithm.

For example, we evaluated the implementation in a 45 nm CMOS library of standard cells of a fixed-point 32x32-bit multiplier (FXP-32), similar to the multiplier in the ADI SHARC DSP [15], and the implementation of a *binary32* multiplier (FP32-mul). The results show that the FP32-mul is 15% slower than the FXP-32 multiplier, due to the extra latency for normalization and rounding. However, the area of the FP32-mul is about 30% smaller than the FXP-32, and also the FP32-mul power consumption is reduced by about one third.

Next, we implement a TFP32 and a TFP16 unit to estimate the power dissipation to play music in the different TFP formats. The units are implemented in a 45 nm CMOS library of standard cells by using commercial synthesis tools.

A. TFP32 Unit

The unit to process TFP32 formats is composed of:

- a TFP32 multiplier, similar to the one described in [3];
- a TFP32 add/subtract unit, similar to the one in [16].

Both sub-units are pipelined into two stages at a clock rate of 667 MHz (T_{clk}=1.5 ns, latency is 3 ns). The implementation metrics are given in Table IV.

B. TFP16 Unit

The unit to process TFP16 formats is composed of:

- a TFP16 multiply/divide unit [9];
- a TFP16 add/subtract unit.

As for the case of TFP32, both sub-units are 2-stage pipelined, and the metrics are given in Table IV.

C. Power Estimate

The power dissipation is estimated by simulating the IIR filtering application, based on the trace-generated VHDL patterns of Sec. IV-B, for the different formats in the TFP32 and TFP16 units. The results of the average power dissiapation (P_{ave} at 100 MHz), together with the average errors, ϵ_{ave} of Table II, are reported in Table V and depicted in Fig. 5.

TABLE IV
HARDWARE IMPLEMENTATION OF TFP UNITS.

	Units	stages	T_{clk} [ns]	Area unit⋆	Area total⋆	ratio
TFP32	ADD	2	1.5	6,080	16,330	1.00
	MUL	2		10,250		
TFP16	ADD	2	1.5	1,960	5,980	0.37
	MUL/DIV	2		4,020		

⋆Area is given in $[\mu m^2]$. Area NAND-2 $\simeq 1.06 \mu m^2$.

TABLE V
AVERAGE POWER DISSIPATION FOR DIFFERENT TFP FORMATS AND UNITS.

TFP32 Unit

Format	m	e	ϵ_{ave}	P_{ave} $[mW]$	P_{ave} Ratio
binary32	24	8	–	0.999	1.00
PXR24	16	8	$< 2^{-20}$	0.754	0.75
binary16	11	5	$< 2^{-15}$	0.613	0.61
BFloat16	8	8	$< 2^{-12}$	0.514	0.51
FP8 formats	4	4	$< 2^{-8}$	0.390	0.39
	3	5	$< 2^{-8}$	0.382	0.38

TFP16 Unit

Format	m	e	ϵ_{ave}	P_{ave} $[mW]$	P_{ave} Ratio⋆
binary16	11	5	$< 2^{-15}$	0.434	0.71
BFloat16	8	8	$< 2^{-12}$	0.368	0.72

⋆ Ratio **TFP16/TFP32** for same format.

The results show that, if the distortion is tolerable (as in the case of 16-bit formats), the power savings are sizable compared to *binary32*. Moreover, by resorting to the smaller-datapath TFP16 unit, the power is further reduced, without any error penalty, by about 30% with respect to the TFP32 unit.

VI. CONCLUSIONS

In this work, we present an analysis on how the floating-point representation of signals can impact the performance of audio systems in terms of harmonic distortion, error rates, and energy efficiency.

The combination of analytical results and psychoacoustic tests carried out on audio samples indicate that FP 16-bit formats are particularly attractive for audio applications because they offer the best distortion-energy efficiency trade-off.

Moreover, the TFP16 unit is about 30% more power efficient than the TFP32 unit when executing operations on 16-bit formats.

The 32/24-bit formats, although the error is lower, do not offer better audio quality than 16-bit formats, and they consume significantly more power.

In contrast, 8-bit storage formats result in low quality audio and they seem not suitable for high quality audio.

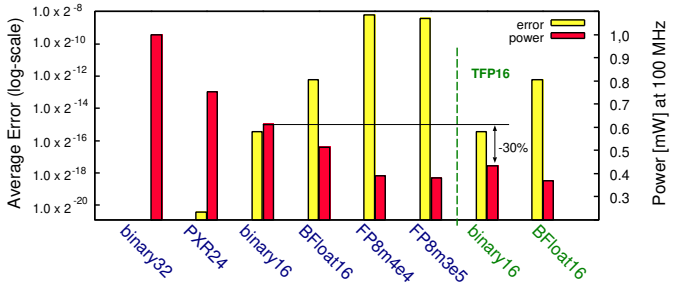

Fig. 5. Average error and average power dissipation in TFP32 (blue) and TFP16 (green) units for 1 sec. audio sample.

ACKNOWLEDGEMENTS

We would like to thank Valerio Santosuosso, website creation expert, for helping in creating the MUSHRA test website.

REFERENCES

[1] "IEEE Standard for Floating-Point Arithmetic," *IEEE Std 754-2019 (Revision of IEEE 754-2008)*, pp. 1–84, Jul. 2019.

[2] N. P. Jouppi *et al.*, "In-Datacenter Performance Analysis of a Tensor Processing Unit," in *ACM/IEEE 44th Annual International Symposium on Computer Architecture (ISCA)*, 2017, pp. 1–12.

[3] A. Nannarelli, "Tunable Floating-Point for Energy Efficient Accelerators," in *25th IEEE Symposium on Computer Arithmetic (ARITH)*, Jun. 2018, pp. 29–36.

[4] M. Ercegovac and T. Lang, *Digital Arithmetic*. Morgan Kaufmann Publishers, 2004.

[5] J. Tomarakos. "Relationship of Data Word Size to Dynamic Range and Signal Quality in Digital Audio Processing Applications". Analog Devices Inc., Technical Articles. [Online]. Available: https://www.analog.com/en/technical-articles/relationship-data-word-size-dynamic-range.html

[6] Wikipedia. (2021) BFloat16 Floating-Point Format. [Online]. Available: https://en.wikipedia.org/wiki/Bfloat16_floating-point_format

[7] nVIDIA. "NVIDIA H100 Tensor Core GPU Architecture". Whitepaper. [Online]. Available: https://resources.nvidia.com/en-us-tensor-core/gtc22-whitepaper-hopper

[8] Tesla Motors. "Tesla Dojo Technology". Whitepaper. [Online]. Available: https://digitalassets.tesla.com/tesla-contents/image/upload/tesla-dojo-technology

[9] A. Nannarelli, "Variable Precision 16-Bit Floating-Point Vector Unit for Embedded Processors," in *27th IEEE Symposium on Computer Arithmetic (ARITH)*, Jun. 2020, pp. 96–102.

[10] A. Agrawal *et al.*, "DLFloat: A 16-b Floating Point Format Designed for Deep Learning Training and Inference," in *26th IEEE Symposium on Computer Arithmetic*, Jun. 2019, pp. 92–95.

[11] S. Temme, "Audio distortion measurements," *Application Note, Bruel & Kjar*, 1992.

[12] Mathworks. (Rev. 2022a) Matlab Documentation THD Function. [Online]. Available: https://www.mathworks.com/help/signal/ref/thd.html

[13] M. Schoeffler, F.-R. Stöter, B. Edler, and J. Herre, "Towards the next generation of web-based experiments: A case study assessing basic audio quality following the ITU-R recommendation BS. 1534 (MUSHRA)," in *1st Web Audio Conference*, 2015, pp. 1–6.

[14] R. La Cesa. (2023) "Tunable Floating Point (TFP) representation webMUSHRA test". Electronic Appendix. [Online]. Available: https://mushra.inglacesa.it/

[15] Analog Devices Inc. SHARC Audio Processors/SoCs. [Online]. Available: https://www.analog.com/en/product-category/sharc-audio-processors-socs.html

[16] A. Nannarelli, "Tunable Floating-Point Adder," *IEEE Transactions on Computers*, vol. 68, no. 10, pp. 1553–1560, Oct. 2019.