

# Dynamic Power Consumption of the Full Posit Processing Unit: Analysis and Experiments

Michele Piccoli ✉ 

Dipartimento di Elettronica,  
Informazione e Bioingegneria (DEIB),  
Polytechnic University of Milano, Italy

William Fornaciari ✉ 

Dipartimento di Elettronica,  
Informazione e Bioingegneria (DEIB),  
Polytechnic University of Milano, Italy

Marco Cococcioni ✉

Dipartimento di Ingegneria dell'Informazione,  
University of Pisa, Italy

Sergio Saponara ✉

Dipartimento di Ingegneria dell'Informazione,  
University of Pisa, Italy

Davide Zoni ✉ 

Dipartimento di Elettronica,  
Informazione e Bioingegneria (DEIB),  
Polytechnic University of Milano, Italy

Giuseppe Massari ✉ 

Dipartimento di Elettronica,  
Informazione e Bioingegneria (DEIB),  
Polytechnic University of Milano, Italy

Federico Rossi ✉

Dipartimento di Ingegneria dell'Informazione,  
University of Pisa, Italy

Emanuele Ruffaldi ✉

MMI spa, Pisa, Italy

---

## Abstract

Since its introduction in 2017, the Posit™ format for representing real numbers has attracted a lot of interest, as an alternative to IEEE 754 floating point representation. Several hardware implementations of arithmetic operations between posit numbers have also been proposed in recent years. In this work, we analyze the dynamic power consumption of the Full Posit Processing Unit (FPPU) recently developed at the University of Pisa. Experimental results show that we can model the dynamic power consumption of the FPPU with an acceptable approximation error from 2.84% (32-bit FPPU) to 7.32% (8-bit FPPU). Furthermore, from the synthesis of the power monitoring unit alongside the FPPU we demonstrate that the additional power module has an area cost that goes from ~ 5% (32-bit FPPU) to ~ 30% (8-bit FPPU) of the total unit area occupation.

**2012 ACM Subject Classification** Hardware → Power estimation and optimization; Hardware → Arithmetic and datapath circuits; Hardware → Reconfigurable logic and FPGAs

**Keywords and phrases** power estimation, computer arithmetic, posit numbers

**Digital Object Identifier** 10.4230/OASICS.PARMA-DITAM.2023.6

**Funding** Work partially supported by H2020 projects EPI-SGA2 (grant no. 101036168, <https://www.european-processor-initiative.eu/>) and TEXTAROSSA (grant no. 956831, <https://textarossa.eu/>).

## 1 Introduction

In the latest years, several representations for real number operations have been proposed by industry and research such as Intel with Flexpoint [17, 19], Google with BFLOAT16 [5], IBM with DLFloat[4], NVIDIA with TensorFloat32 [1], Facebook with logarithmic numbers [16], and Tesla with its configurable floats CFloat8-CFloat16 [2].

Academic research proposed different alternatives to the IEEE 32-bit Floating-point standard, such as [26] or [25]. One of the most promising alternatives to the IEEE 32-bit Floating-point standard is the Posit™ format [14]. Posits proved to be able to match single precision (i.e. IEEE 32-bit floats) accuracy (in machine learning and neural network tasks)



© Michele Piccoli, Davide Zoni, William Fornaciari, Giuseppe Massari, Marco Cococcioni, Federico Rossi, Sergio Saponara, and Emanuele Ruffaldi;  
licensed under Creative Commons License CC-BY 4.0

14th Workshop on Parallel Programming and Run-Time Management Techniques for Many-Core Architectures and 12th Workshop on Design Tools and Architectures for Multicore Embedded Computing Platforms (PARMA-DITAM 2023).

Editors: João Bispo, Henri-Pierre Charles, Stefano Cherubin, and Giuseppe Massari; Article No. 6; pp. 6:1–6:11



OpenAccess Series in Informatics  
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

performance with only 16 bits used for the representation both in our previous works and in independent research [6, 13, 18]. Moreover, with just 8 bits, the overall performances did not degrade critically, as shown in [7, 8].

Several posit-processing hardware architectures have been already proposed.

In [20] a fully functional posit floating point unit was presented alongside a RISC-V posit extension exploiting and overloading the already existent RISC-V IEEE 32-bit float instructions. The authors introduce a posit unit with 32 32-bit posit registers with an additional status register. The final design is a 32-bit posit co-processor that is decoupled from the RISC-V core execution pipeline. The proposed unit reportedly occupies 3507 slice LUTs and 1294 slice registers on an Artix-7-100T Xilinx FPGA running at 100 MHz.

In [15] a benchmark platform for alternative real number arithmetic was designed, including posits. They introduced two components: i) Melodica, a complete posit unit implementing several arithmetics, quires and fused multiply-add operations; ii) Clarinet, a RISC-V core with Melodica support. The authors leveraged the custom op-code space in RISC-V to add custom instructions, as well as a custom C compiler toolchain. Furthermore, they added a new set of posit registers with parametric posit size.

In this paper, we will characterize a standalone and pipelined Full Posit Processing Unit developed at the University of Pisa [11]. The characterization of the unit will be performed using a run-time power estimation methodology [21]. The choice is motivated by the fact that HPC systems have always been subjected by thermal limitations [3]. To operate in an efficiently and reliable way, heat-dissipation and thermal management techniques must be taken into account. To achieve these results, [24] and [23] propose an energy-constrained controller for hardware accelerators and multi-cores CPUs, while [12] implements a resource-constrained methodology. The idea of a complete power identification flow comes from [22], where a model has been instrumented on an OpenRisc 1000 compliant CPU.

We adopt [21], since it involves the measurement of several metrics for different design configurations and boards: i) resource utilization and area, ii) timing properties and maximum frequency iii) dynamic power and switching activity characterization.

Hereafter we state the paper organization: in Section 2 we present the posit format and the architecture of the Full Posit Processing Unit. In Section 3 the power identification flow is described, while in Section 4 the experimental results are shown and commented on. Finally, in Section 5 we draw the conclusions and discuss possible future works.

## 2 The Posit Format and the Full Posit Processing Unit

A posit number [14] is represented by an integer in 2's complement encoding. The format can be configured in the number of bits  $nbits$  and the number of exponent bits  $esbits$ . The format can have at most 4 fields:

- Sign field  $s$ : 1 bit;
- Regime field: variable length, composed by a sequence of identical bits stopped by a bit of the opposite value
- Exponent field: variable length, at most  $esbits$  bits;
- Fraction field: variable length

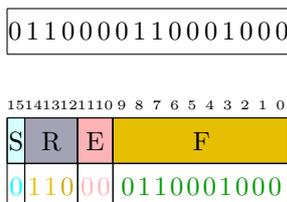
Let us consider a posit  $(nbits, esbits)$ , represented in 2's complement signed integer  $P$  and let  $e$  and  $f$  (on  $F$  bits) be the real values represented by exponent and fraction fields. The real number  $r$  represented by  $X$  encoding is:

$$r = \begin{cases} 0, & \text{if } X = 0 \\ \text{NaN}, & \text{if } X = 2^{(nbits-1)} \\ (-1)^s \cdot useed^k \cdot 2^e \cdot (1 + \frac{f}{F}), & \text{otherwise} \end{cases}$$

Where  $used = 2^{2^{esbits}}$ . The regime value  $k$  is computed from the regime length  $l$ :

$$k = \begin{cases} -l, & \text{if } b = 0 \\ l - 1, & \text{otherwise} \end{cases}$$

Where  $b$  is the value of the single bit of the identical bits in the regime. An example of Posit number is shown in Figure 1.



■ **Figure 1** An example of Posit configuration with  $nbits=16$  and  $esbits=2$ . The associated real value to the shown Posit is:  $+1 \cdot 16^1 \cdot 2^0 \cdot (1 + 392/1024) = 22.125$ . The value of  $used$  is  $2^{2^2} = 16$ , since  $esbit = 2$  is assumed in this case.

## 2.1 Full Posit Processing Unit (FPPU)

In a previous work [9] we have presented a light Posit Processing Unit, called  $PPU^{light}$ . It was an arithmetic unit able to convert from float to posit and vice-versa, integrated within a RISC-V CPU. Then we have implemented a pipelined full posit processing unit, called FPPU [11], which natively supports all the four arithmetic operations between posits, other than comparison and conversion operations. In this work, we aim to analyze the dynamic power consumption of the FPPU, by using modeling and verification tools presented in [21]. Figure 2 shows the FPPU hardware component in its principal internal components. The module has 5 inputs:

- Posit A,B: the two posit operands
- Op: operation code (e.g. ADD, SUB, MUL, DIV)
- clk: clock reference
- valid\_in: states whether FPPU inputs are ready

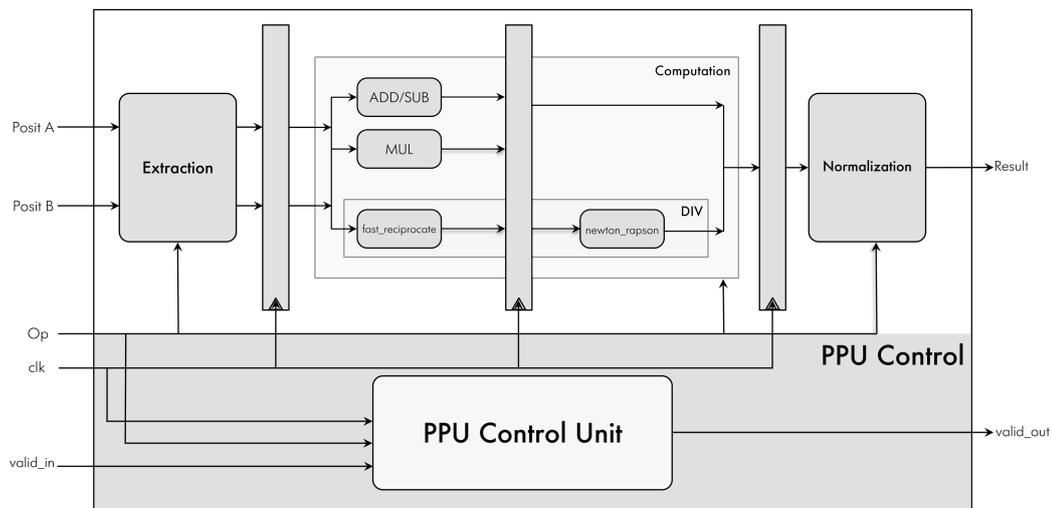
The output *Result* is the posit resulting from the operation, while *valid\_out* states whether the FPPU output is valid. The unit has 4 pipeline stages to reduce the overall latency in terms of maximum clock period constraint; splitting the unit into 4 stages allowed us to increase the clock frequency without incurring timing violations with the registers.

## 3 Dynamic Power Modeling

To analyze dynamic power consumption we adopt the approach proposed in [21], which consists of a three-stage power identification flow (see fig. 3). Starting from the synthesized netlist, the design is simulated by executing different benchmarks, each one selected to stress specific parts of the architecture. During the simulation, the required information is represented by two file types:

- Switching Activity Interchange Format (SAIF): a report which encapsulates all the switching activity information provided by the simulator;
- Value Change Dump (VCD): a file containing all the values assumed by the signals during the simulation.

## 6:4 Dynamic Power FPPU



■ **Figure 2** Full Posit Processing Unit (FPPU) with 4-stage pipeline.

■ **Table 1** FPGA resources utilization for different FPPU cores. All the cores have a conversion with binary32 enabled. The various posit configurations are noted as PXXEYY, where XX denotes *nbits* and YY denotes *esbits*.

| Part        | Posit | LUTs | (%)   | Registers | (%)  |
|-------------|-------|------|-------|-----------|------|
| Artix7-2L   | P16E0 | 1249 | 15.61 | 16000     | 2.19 |
|             | P16E1 | 1410 | 17.63 | 16000     | 2.27 |
|             | P16E2 | 1412 | 17.65 | 16000     | 2.28 |
|             | P8E0  | 453  | 5.66  | 16000     | 1.42 |
|             | P8E1  | 444  | 5.55  | 16000     | 1.49 |
|             | P8E2  | 449  | 5.61  | 16000     | 1.53 |
| Kintex-7    | P16E0 | 1249 | 3.05  | 82000     | 0.43 |
|             | P16E1 | 1410 | 3.44  | 82000     | 0.44 |
|             | P16E2 | 1412 | 3.44  | 82000     | 0.45 |
|             | P8E0  | 453  | 1.10  | 82000     | 0.28 |
|             | P8E1  | 444  | 1.08  | 82000     | 0.29 |
|             | P8E2  | 449  | 1.10  | 82000     | 0.30 |
| Spartan-7   | P16E0 | 1319 | 35.17 | 7500      | 4.85 |
|             | P16E1 | 1480 | 39.47 | 7500      | 5.03 |
|             | P16E2 | 1475 | 39.33 | 7500      | 5.03 |
|             | P8E0  | 453  | 12.08 | 7500      | 3.03 |
|             | P8E1  | 444  | 11.84 | 7500      | 3.17 |
|             | P8E2  | 449  | 11.97 | 7500      | 3.27 |
| Artix7-100T | P16E0 | 1249 | 1.97  | 350       | 0.28 |
|             | P16E1 | 1410 | 2.22  | 363       | 0.29 |
|             | P16E2 | 1412 | 2.23  | 365       | 0.29 |
|             | P8E0  | 453  | 0.71  | 227       | 0.18 |
|             | P8E1  | 444  | 0.70  | 238       | 0.19 |
|             | P8E2  | 449  | 0.71  | 245       | 0.19 |

■ **Table 2** Timing summary of different FPPU cores with maximum theoretically achievable frequency.

| Part        | Posit | Min clock period (ns) | Max frequency (MHz) |
|-------------|-------|-----------------------|---------------------|
| Artix7-2L   | P16E0 | 22.608                | 44.232              |
|             | P16E1 | 22.162                | 45.122              |
|             | P16E2 | 22.039                | 45.374              |
|             | P8E0  | 15.186                | 65.850              |
|             | P8E1  | 14.715                | 68.847              |
|             | P8E2  | 14.525                | 67.958              |
| Kintex-7    | P16E0 | 12.878                | 77.652              |
|             | P16E1 | 12.605                | 79.955              |
|             | P16E2 | 12.507                | 79.334              |
|             | P8E0  | 8.589                 | 116.428             |
|             | P8E1  | 8.256                 | 121.595             |
|             | P8E2  | 8.224                 | 121.124             |
| Spartan-7   | P16E0 | 17.727                | 56.526              |
|             | P16E1 | 17.691                | 56.411              |
|             | P16E2 | 17.691                | 56.526              |
|             | P8E0  | 12.372                | 80.828              |
|             | P8E1  | 12.049                | 83.313              |
|             | P8E2  | 12.003                | 82.994              |
| Artix7-100T | P16E0 | 22.457                | 44,529              |
|             | P16E1 | 22.181                | 45,083              |
|             | P16E2 | 21.972                | 45.512              |
|             | P8E0  | 15.028                | 66.542              |
|             | P8E1  | 14.576                | 68.605              |
|             | P8E2  | 14.596                | 68.511              |

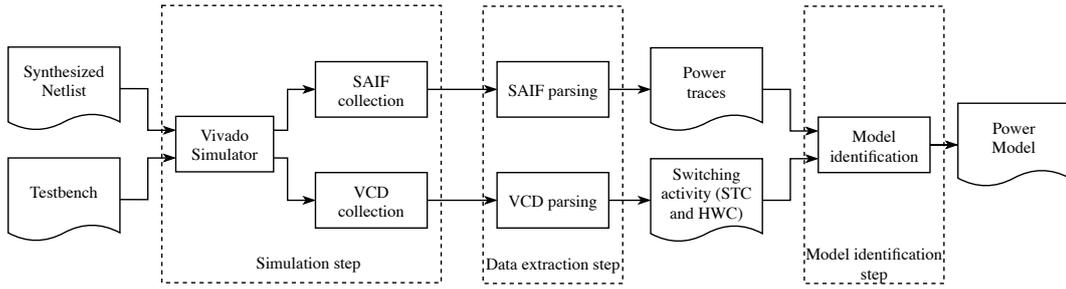
SAIF and VCDs are extracted and then parsed, giving us power consumption and signal-switching activity. In particular, two metrics are adopted for measuring the switching activity:

- Hamming Weight Count (HWC): used for data signals, represents the actual number of bits that change their state;
- Single Toggle Count (STC): used for control signals, represents the number of times that the signal changes, regardless of its number of bits.

This distinction is driven by design rules and the purpose of the signals. Usually, in data signals, the number of changing bits is strongly correlated with the power consumption variation, while instead, the toggle of the control signals indicates a change in the hardware operation being executed. This change, and so the power consumption, is correlated more to the toggling rate rather than the actual number of bits, hence the choice.

In the third step, the power model is identified employing a linear regression, where the input matrix is composed of the switching activity of the signals and the observation is the collected power consumption.

Once obtained the final model, this can be injected into the monitored design through a simple piece of logic, as mentioned in [21]. This additional hardware is composed of the identified counters (STC and HWC) plus an adder, implementing the equation 1, where:



■ **Figure 3** View of the dynamic power modeling flow.

- $\hat{p}_t$  is the estimated power at time sample  $t$ ;
- $c_i$  is the  $i$ -th HWC coefficient;
- $c_j$  is the  $j$ -th STC coefficient;
- $S_{t,*}$  are the classified signals,  $i$  for HWC and  $j$  for STC, at time sample  $t$ .

$$\hat{p}_t = \sum_{i \in HWC} c_i * S_{t,i} + \sum_{j \in STC} c_j * S_{t,j} \tag{1}$$

The model tells us that the power at time sample  $t$  is given by the contribution of the classified signals (HWC or STC) at time sample  $t$ , conveniently multiplied by the estimated coefficient.

Note that it is also possible to constrain the identification step both on used resources and exploration depth. The first constraint limits the number of available resources (LUT and FF) and thus performance counters size, while the second one sets a maximum level in the design hierarchy, where the identification will stop.

## 4 Experimental Results

To assess our model, we tested four benchmarks (the basic arithmetic operations) in random order, adding also no-op periods to instruct it on operative and idle states, on different configurations of the FPPU. Below, in Table 3, area and timing configurations are reported for each FPPU configuration, the target FPGA is an Artix7-100T (part xc7a100tcs324-1).

■ **Table 3** Synthesis results targeting an Artix7-100T.

| Synthesis results   |                           |                           |
|---------------------|---------------------------|---------------------------|
| Posit configuration | Synthesis frequency (MHz) | Used resources (LUT + FF) |
| P32E2               | 25.00                     | 4049 + 520                |
| P32E1               | 25.00                     | 3669 + 513                |
| P32E0               | 25.00                     | 3523 + 509                |
| P16E2               | 40.00                     | 1817 + 378                |
| P16E1               | 40.00                     | 1785 + 367                |
| P16E0               | 40.00                     | 1500 + 238                |
| P8E2                | 50.00                     | 750 + 259                 |
| P8E1                | 50.00                     | 734 + 250                 |
| P8E0                | 50.00                     | 718 + 238                 |

After the benchmarks have been preprocessed correctly, they are randomly shuffled and fed into the identification flow. Note that the dataset is split into train and test sets, one of the traditional methods.

To evaluate model quality we adopt the RMSE metric, used also in [21]. RMSE is defined in equation 2, where  $E$  is the mean,  $\hat{p}$  and  $p$  are, respectively, the estimated and actual power.

$$RMSE = \sqrt{E((\hat{p} - p)^2)} \quad (2)$$

Then the RMSE has been normalized w.r.t. the difference between the maximum and minimum values of  $p$ , see equation 3. This choice tries to give more context to the error measurement, taking into account the computing peak and rest values, quantified respectively in  $max(p)$  and  $min(p)$ .

$$RMSE_{\%} = RMSE / (max(p) - min(p)) \quad (3)$$

In Table 4, for each FPPU, we report the RMSE and the estimated performance counters area w.r.t. the design total.

■ **Table 4** Identification results targeting an Artix7-100T.

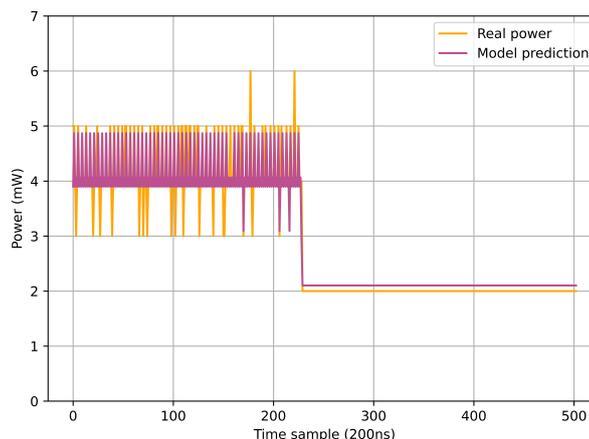
| Model identification results |           |                     |          |
|------------------------------|-----------|---------------------|----------|
| Posit configuration          | RMSE (mW) | RMSE Normalized (%) | Area (%) |
| P32E2                        | 0.426     | 2.84                | 5.01     |
| P32E1                        | 0.461     | 3.33                | 16.53    |
| P32E0                        | 0.429     | 3.06                | 4.26     |
| P16E2                        | 0.223     | 3.72                | 7.06     |
| P16E1                        | 0.333     | 5.56                | 5.34     |
| P16E0                        | 0.284     | 4.75                | 4.33     |
| P8E2                         | 0.284     | 7.10                | 33.13    |
| P8E1                         | 0.279     | 6.99                | 19.32    |
| P8E0                         | 0.293     | 7.32                | 29.66    |

One can notice that the FPPU 8 presents an error and area degradation w.r.t. the other two configurations, this happens since the unit has a very low power consumption, on average between 3 and 5 mW when computing, thus the metric is more sensitive to outliers in this small range.

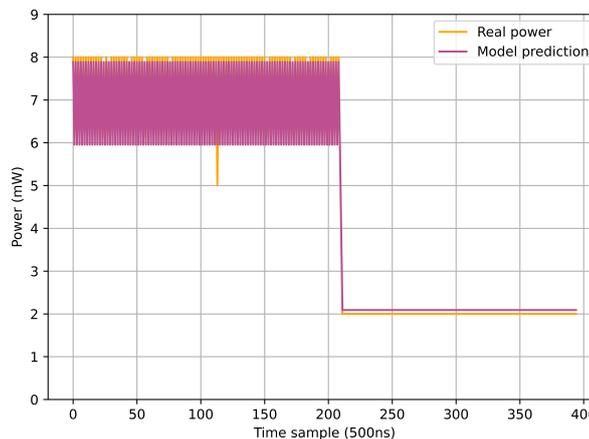
Regarding the higher area ratio, the estimated performance counters size is similar to the other solutions, while the FPPU 8 area decreases, thus increasing the ratio.

To provide a complete overview of the identified model, we report the plots of the ones obtained by the FPPUs with exponent  $esbits=2$ , since they are the most efficient in deep neural networks applications, as [10] reports.

Figure 4 shows the prediction on the FPPU operation ADD. The model in Figure 5 has been tested on the operation SUB instead. Finally, in Figure 6 the prediction on operation DIV is reported. Note that the number of plotted samples is reduced to provide a more detailed view of the two plots.



■ **Figure 4** Prediction of the model trained on an FPPU with  $nbits=8$  and  $esbits=2$ . Around time sample 220 the unit goes into an idle state.



■ **Figure 5** Prediction of the model trained on an FPPU with  $nbits=16$  and  $esbits=2$ . Around time sample 210 the unit goes into an idle state.

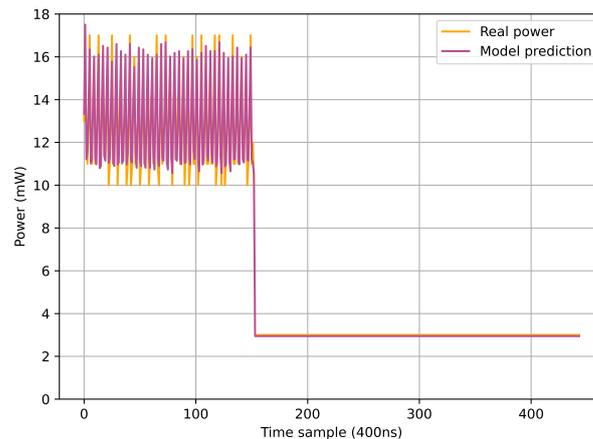
## 5 Conclusions and future works

In this paper, we first reviewed a recently designed and synthesized configurable Posit Processing Unit, previously developed by the authors of this study.

Then we modeled for the first time its dynamic power consumption and we reported the resulting figures from the power model component synthesized in FPGA. For this second part, we employed the framework presented in [21]. The results show an acceptable error for each of the proposed FPPUs and a light area impact, proving the feasibility of a possible performance counters implementation along the FPPU.

As a future work, we plan to perform a comparison between our FPPU and a traditional FPU [26]. This would highlight the pros and cons of the Posit approach in hardware design and, in general, with intensive computing workloads.

Another possibility involves the integration in a real case CPU, to evaluate the performances while executing standard CPU benchmarks or while training deep neural networks.



■ **Figure 6** Prediction of the model trained on an FPPU with  $nbits=32$  and  $esbits=2$ . Around time sample 160 the unit goes into an idle state.

## References

- 1 Nvidia TensorFlow32. <https://blogs.nvidia.com/blog/2020/05/14/tensorfloat-32-precision-format/>.
- 2 Tesla Dojo Technology: a Guide to Tesla’s Configurable Floating Point Formats & Arithmetic. [https://tesla-cdn.thron.com/static/SBY4B9\\_tesla-dojotechnology\\_OPNZOM.pdf](https://tesla-cdn.thron.com/static/SBY4B9_tesla-dojotechnology_OPNZOM.pdf), 2022.
- 3 Giovanni Agosta, Marco Aldinucci, Carlos Alvarez, Roberto Ammendola, Yasir Arfat, Olivier Beaumont, Massimo Bernaschi, Andrea Biagioni, Tommaso Boccali, Berenger Bramas, Carlo Brandolese, Barbara Cantalupo, Mauro Carrozzo, Daniele Cattaneo, Alessandro Celestini, Massimo Celino, Iacopo Colonnelli, Paolo Cretaro, Pasqua D’Ambra, Marco Danelutto, Roberto Esposito, Lionel Eyraud-Dubois, Antonio Filgueras, William Fornaciari, Ottorino Frezza, Andrea Galimberti, Francesco Giacomini, Brice Goglin, Daniele Gregori, Abdou Guermouche, Francesco Iannone, Michal Kulczewski, Francesca Lo Cicero, Alessandro Lonardo, Alberto R. Martinelli, Michele Martinelli, Xavier Martorell, Giuseppe Massari, Simone Montangero, Gianluca Mittone, Raymond Namyst, Ariel Oleksiak, Paolo Palazzari, Pier Stanislaw Paolucci, Federico Reghenzani, Cristian Rossi, Sergio Saponara, Francesco Simula, Federico Terraneo, Samuel Thibault, Massimo Torquati, Matteo Turisini, Piero Vicini, Miquel Vidal, Davide Zoni, and Giuseppe Zummo. Towards extreme scale technologies and accelerators for eurohpc hw/sw supercomputing applications for exascale: The textarossa approach. *Microprocessors and Microsystems*, 95:104679, 2022. doi:10.1016/j.micpro.2022.104679.
- 4 A. Agrawal, S. M. Mueller, B. M. Fleischer, X. Sun, N. Wang, J. Choi, and K. Gopalakrishnan. DLFLOAT: A 16-b floating point format designed for deep learning training and inference. In *2019 IEEE 26th Symp. on Computer Arithmetic (ARITH’19)*, pages 92–95, 2019. doi:10.1109/ARITH.2019.00023.
- 5 N. Burgess, J. Milanovic, N. Stephens, K. Monachopoulos, and D. Mansell. Bfloat16 processing for neural networks. In *2019 IEEE 26th Symp. on Computer Arithmetic (ARITH’19)*, pages 88–91, 2019. doi:10.1109/ARITH.2019.00022.
- 6 Z. Carmichael, H. F. Langroudi, C. Khazanov, J. Lillie, J. L. Gustafson, and D. Kudithipudi. Deep positron: A deep neural network using the posit number system. In *2019 Design, Automation Test in Europe Conference Exhibition (DATE)*, pages 1421–1426, 2019.

- 7 M. Cococcioni, F. Rossi, E. Ruffaldi, S. Saponara, and B. Dupont de Dinechin. Novel arithmetics in deep neural networks signal processing for autonomous driving: Challenges and opportunities. *IEEE Signal Processing Magazine*, 38(1):97–110, 2021. URL: [10.1109/MSP.2020.2988436](https://doi.org/10.1109/MSP.2020.2988436).
- 8 Marco Cococcioni, Federico Rossi, Emanuele Ruffaldi, and Sergio Saponara. Fast approximations of activation functions in deep neural networks when using posit arithmetic. *Sensors*, 20(5), 2020. URL: <https://www.mdpi.com/1424-8220/20/5/1515>.
- 9 Marco Cococcioni, Federico Rossi, Emanuele Ruffaldi, and Sergio Saponara. A lightweight posit processing unit for risc-v processors in deep neural network applications. *IEEE Transactions on Emerging Topics in Computing*, pages 1–1, 2021. doi:10.1109/TETC.2021.3120538.
- 10 Marco Cococcioni, Federico Rossi, Emanuele Ruffaldi, and Sergio Saponara. Small reals representations for deep learning at the edge: A comparison. In John Gustafson and Vassil Dimitrov, editors, *Next Generation Arithmetic*, pages 117–133, Cham, 2022. Springer International Publishing.
- 11 Marco Cococcioni, Federico Rossi, Emanuele Ruffaldi, Sergio Saponara, and Francesco Urbani. Fppu: Design and implementation of a pipelined full posit processing unit. *submitted*, 2022.
- 12 Luca Cremona, William Fornaciari, and Davide Zoni. Automatic identification and hardware implementation of a resource-constrained power model for embedded systems. *Sustainable Computing: Informatics and Systems*, 29:100467, 2021. doi:10.1016/j.suscom.2020.100467.
- 13 Seyed Hamed Fatemi Langroudi, Zachariah Carmichael, John Gustafson, and Dhireesha Kudithipudi. Positnn framework: Tapered precision deep learning inference for the edge. In *2019 IEEE Space Computing Conference (SCC)*, pages 53–59, July 2019. doi:10.1109/SpaceComp.2019.00011.
- 14 John L Gustafson and Isaac T Yonemoto. Beating floating point at its own game: Posit arithmetic. *Supercomputing Frontiers and Innovations*, 4(2):71–86, 2017.
- 15 Riya Jain, Niraj Sharma, Farhad Merchant, Sachin Patkar, and Rainer Leupers. CLARINET: A RISC-V based framework for posit arithmetic empiricism. *CoRR*, abs/2006.00364, 2020. arXiv:2006.00364.
- 16 Jeff Johnson. Rethinking floating point for deep learning. *CoRR*, abs/1811.01721, 2018. arXiv:1811.01721.
- 17 Urs Köster, Tristan Webb, Xin Wang, Marcel Nassar, Arjun K Bansal, William Constable, Oguz Elibol, Scott Gray, Stewart Hall, Luke Hornof, et al. Flexpoint: An adaptive numerical format for efficient training of deep neural networks. In *In Proc. of the 31st Conference on Neural Information Processing Systems (NIPS'17)*, pages 1742–1752, 2017.
- 18 J. Lu, C. Fang, M. Xu, J. Lin, and Z. Wang. Evaluations on deep neural networks training using posit number system. *IEEE Transactions on Computers*, pages 1–1, 2020.
- 19 V. Popescu, M. Nassar, X. Wang, E. Tumer, and T. Webb. Flexpoint: Predictive numerics for deep learning. In *In Proc. of the 25th IEEE Symp. on Computer Arithmetic (ARITH'18)*, pages 1–4, 2018. doi:10.1109/ARITH.2018.8464801.
- 20 Sugandha Tiwari, Neel Gala, Chester Rebeiro, and V. Kamakoti. PERI: A posit enabled RISC-V core. *CoRR*, abs/1908.01466, 2019. arXiv:1908.01466.
- 21 Davide Zoni, Luca Cremona, Alessandro Cilaro, Mirko Gagliardi, and William Fornaciari. Powertap: All-digital power meter modeling for run-time power monitoring. *Microprocessors and Microsystems*, 63:128–139, 2018. doi:10.1016/j.micpro.2018.07.007.
- 22 Davide Zoni, Luca Cremona, and William Fornaciari. Powerprobe: Run-time power modeling through automatic rtl instrumentation. In *2018 Design, Automation & Test in Europe Conference & Exhibition (DATE)*, pages 743–748, 2018. doi:10.23919/DATE.2018.8342106.
- 23 Davide Zoni, Luca Cremona, and William Fornaciari. All-digital control-theoretic scheme to optimize energy budget and allocation in multi-cores. *IEEE Transactions on Computers*, 69(5):706–721, 2020. doi:10.1109/TC.2019.2963859.

- 24 Davide Zoni, Luca Cremona, and William Fornaciari. All-digital energy-constrained controller for general-purpose accelerators and cpus. *IEEE Embedded Systems Letters*, 12(1):17–20, 2020. doi:10.1109/LES.2019.2914136.
- 25 Davide Zoni and Andrea Galimberti. Cost-effective fixed-point hardware support for risc-v embedded systems. *Journal of Systems Architecture*, 126:102476, 2022. doi:10.1016/j.sysarc.2022.102476.
- 26 Davide Zoni, Andrea Galimberti, and William Fornaciari. An fpu design template to optimize the accuracy-efficiency-area trade-off. *Sustainable Computing: Informatics and Systems*, 29:100450, 2021. doi:10.1016/j.suscom.2020.100450.