A High-Level Approach for Energy Efficiency Improvement of FPGAs by Voltage Trimming

Mehdi Safarpour, *Student Member, IEEE,* Lei Xun, *Student Member, IEEE,* Geoff V. Merrett, *Senior Member, IEEE,* Olli Silvén, *Senior Member, IEEE*

***Abstract*—Chip manufacturers define voltage margins on top of the "best-case" operational voltage of their chips to ensure reliable functioning in the worst case settings. The margins guarantee correctness of operation, but at the cost of performance and power efficiency. Violating the margins is tempting to save energy, but might lead to timing errors. This paper proposes an algorithmic solution that enables reliable removal of the margins by detecting errors on the fly. In contrast to previous approaches that require special hardware to detect timing errors, the proposed method is fully implementable using high-level synthesis tools without reliance on additional hardware. The approach is demonstrated using a 32×32 matrix-matrix multiplication and a simple multi-layer neural network implemented on two Xilinx ZC702 Field-Programmable Gate Array (FPGA) System-on-Chip (SoC) platforms, showcasing its utility in detecting errors that may originate from different sources of logic circuits, clock tree or memory. Results show that the energy dissipation is halved, while the implementation is clocked at 2.5x faster than specified by the design tool of the vendor.**

***Index Terms*—low power, high level synthesis, matrix multiplier, low voltage, deep neural networks**

I. Introduction

EXPECTED global energy demand of the ICT sector is projected to be 7% of total consumption by 2030. An increasing share of this energy is consumed for Deep Neural Networks training and inference [1] in data centers, or network infrastructure computations of wireless communications.

Thanks to their flexibility and high performance, Field-Programmable Gate Arrays (FPGA) have found their way into high-performance System-On-Chips (SoC) as accelerators [2]. In particular, matrix arithmetic forms the core operations in telecommunication algorithms, and FPGAs are attractive in fast-tracked development of their implementations using High-Level Synthesis (HLS) tools [1], [3]. Gate level programmability of FPGAs and recent support of HLS based EDA tools enables approaching ASIC level performance with far smaller investment times. However, the energy efficiency gap between ASIC designs and FPGA designs still remains considerable [1], [4].

The energy consumption in digital circuits consists of dynamic and static power dissipation, which are due to transistor switching activity and transistor leakage current in off-state, respectively. Dynamic power has a quadratic, and static power has a linear relation to the supply voltage (V_{dd}), so operating at a reduced voltage lowers total energy consumption [5]. Unfortunately, scaling down the voltage leads to longer propagation delays and, to avoid timing errors, the clock rate and hence performance should also be reduced [6]. The challenge with voltage reduction schemes is the uncertainty in choosing the optimal, or close to optimal, operation point: the minimum voltage and maximum clock frequency without sacrificing either reliability or performance [6]. Process, Voltage and Temperature (PVT) variations result in modeling uncertainties for timing analysis, in particular at reduced voltages [6]. The impact of PVT variations can amount to even a 100x difference in gate delay between the slowest and fastest process corners [7].

Aiming to maximize the yield, manufacturers define margins (or "guardbands") on top of the "best-case" supply voltages to guarantee correct functionality for all manufactured chips [5]. The presence of large voltage margins, up to 30%, has been demonstrated for commercial CPUs and GPUs [8]. Similar studies have been conducted on voltage scaling of Block RAMs of FPGAs, demonstrating up to 60% energy saving [8], [9]. However, computational errors in logic fabric, e.g., Look-Up-Tables and DSP units, are difficult to tackle, while being the largest energy sink [9].

In this paper, a low-overhead error detection method is proposed to enable reliable reduction of operational voltage of FPGAs. The approach detects virtually all errors in matrix-matrix multiplications, regardless of whether they originate from the logic fabric or memory.

In contrast to the state of the art methods [6], [10], the implementation is simply carried out using only High Level Synthesis (HLS) tools without any intervention in vendor provided EDA tools or the generated netlist.

II. Prior Approaches

One of the earliest approaches to set the voltage adaptively is the employment of logic delay measurement circuits [6]. The idea is to measure the circuit's critical-path delay to set the voltage to the optimum. The longest delay path of the circuit is imitated by, e.g., a chain of inverters, a ring oscillator, or a Linear-Feedback Shift Register (LFSR) [9]. However, voltage tuning with this approach has restrictions, as local temperature and process variations, and circuit aging and cross-coupled noise, cannot be faithfully modeled [6], hence necessitating a voltage headroom.

This work was supported by 6G Flagship research programme under Academy of Finland Grant 318927.

Mehdi Safarpour and Olli Silvén are with the Center for Machine Vision and Signal Analysis, University of Oulu, Oulu, Finland (e-mail: firstname.lastname@oulu.fi).

Lei Xun and Geoff V. Merrett are with the School of Electronics and Computer Science, University of Southampton, UK. (e-mail: lx2u16@soton.ac.uk, gvm@ecs.soton.ac.uk).

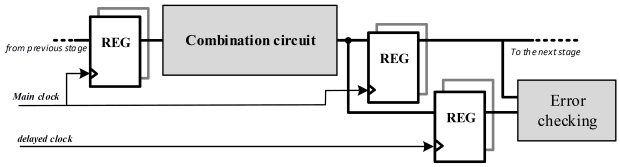

Figure 1. General concept of Timing Error Detection [6].

A more complex circuit level approach is to detect timing errors in-situ by adding Timing Error Detection (TED) circuits to the identified critical paths of circuitry. As depicted in Fig. 1, an extra register is added at the tail of the combinatorial logic path to sample the output with a slightly delayed clock. In the case of a late arriving signal, a conflict between the original sample and the delayed sample reveals the timing error [6]. In reduced voltage operation, TED circuits help in optimizing the operating point to maximise energy efficiency [6]. However, TED circuits cannot be utilized with hard FPGA blocks such as BRAM or DSP units [9]. Furthermore, TED schemes add non-trivial circuit complexity and power consumption overheads, which scale up with the size of the design. Using the approach presented in this paper, the share of the extra logic is reduced as the design grows. Finally, the TED schemes require significant design effort to integrate them into FPGA designs.

An apparently straightforward approach is to determine the voltage-frequency dependencies of the circuit at every power-up. Such a calibration-based approach, described in [9], utilizes the reconfigurability of the FPGA. Running "offline calibration" before the actual application is configured on the FPGA was shown to give 40% energy savings or, alternatively, a 25% performance improvement. While the optimization scheme captures the parameters of the FPGA, calibrations are temperature, design and chip-specific. The approach is probably most suitable for sustained data center computing, as the energy needed in the start-up is not included in the savings estimate. Furthermore, phenomena such as the Inverse Temperature Dependence might render the "offline calibrations" inaccurate. Then errors could be generated by factors that remain hidden from the calibrations.

Concerning error tolerance, [11] demonstrates that Neural Networks can operate with reduced voltage. However, without error feedback, the voltage down-scaling cannot be stopped before system failure. In this paper, we propose a simple effective error feedback mechanism that enables reliable reduced voltage operation without harming functionality. Compared to earlier schemes, it is straightforward to implement. For our demonstration design, common high-level languages (C/C++) were used.

III. Proposed Solution

The focus of the following is on the energy savings achievable through voltage reductions. Matrix multiplication has been selected as an example case due to it being the core of key operations for wireless communications and deep neural networks. The matrix multiplier design used on the FPGA is a run-of-the-mill scheme generated using a vendor provided HLS tool.

A. Algorithm Based Fault Tolerance

The Algorithm Based Fault Tolerance (ABFT) technique proposed by Huang and Abraham [12], briefly described below, is the foundation of the solution that we present. Starting from the matrix $A_{N\times N}$, a row checksum matrix $A^r_{N\times(N+1)}$ is defined as follows:

$$A^r = \begin{bmatrix} A & Ae^T \end{bmatrix} \tag{1}$$

where e is $e_N = [1, 1, ..., 1]$. This means the nth element of the vector Ae is the sum of all elements in the nth row of the matrix A, i.e., $a^r_{n,(N+1)} = \sum_i a_{n,i}$. Similarly, we can define the *column checksum*, A^c, and *full checksum*, A^f matrices as in equations (2) and (3), respectively [12].

$$A^c = \begin{bmatrix} A \\ eA \end{bmatrix} \tag{2}$$

$$A^f = \begin{bmatrix} A & Ae^T \\ eA & eAe^T \end{bmatrix} \tag{3}$$

The checksum property is preserved in many linear algebra operations, including matrix multiplication, dot product, transpose, etc., as well as in more sophisticated signal processing algorithms such Fast Fourier Transform (FFT) and multi-dimensional convolutions. In the case of matrix multiplication the checksum property is illustrated in Fig. 2. It can be exploited to detect errors in the result matrix by comparing the checksum vector against the sums of elements of the corresponding rows of the matrix. As the analysis by Huang and Abraham showed [12], the ABFT checksum technique provides high error detection coverage.

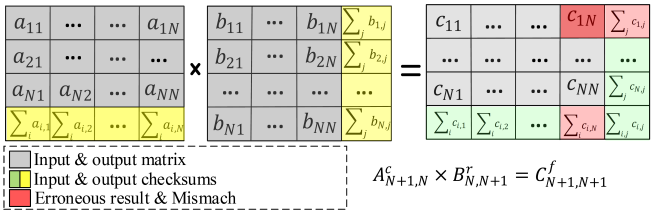

Figure 2. The mismatch in the checksum reveals the existence and location of errors.

B. Reduced Voltage Matrix Multiplication with ABFT

We leverage ABFT in reduced voltage matrix multiplication to detect errors and to adjust the operation point accordingly. Checksum augmentations are added to matrices before computations through the FPGA. The matrix multiplier in the FPGA is considered a black-box, while error checking can be an external add-on implementation powered from a different voltage rail. Both correct and incorrect error detections, true and false positives, respectively, result in the adjustment of the operating point. However, the latter ones drive the system further from the energy optimum. The silent errors, false negatives, propagate to the output without being detected and may cause erroneous actions. The checking and voltage control can be done either using hardware and/or software. The voltage is reduced until the first error appears.

IV. Experimentation

To detect possible inter-chip variations, two Xilinx Zynq-XC7Z020 FPGA SoCs on evaluation boards were employed to demonstrate the approach [2]. A 32×32 single-precision floating-point matrix-matrix multiplier was synthesized on the Programmable Logic (PL) side of the SoC while the Processing System (PS) was harnessed for monitoring and communication. The clock of the PL was set to 250 MHz. Reduction of guardband voltages was performed using the on-board UCD7242 voltage regulators that control the voltage-rails supplying the subsystems. Three of the voltage-rails power the PL: VCCINT feeds the internal circuitry (LUTs, DSPs, etc.), VCCBRAM powers the Block-RAMs, and VCCAUX the auxiliary parts. Each rail can be adjusted separately through Power Management BUS (PMBUS) commands transmitted via the serial I2C bus connected to either the host PC or the PS [2]. The setup is depicted in Fig. 3.

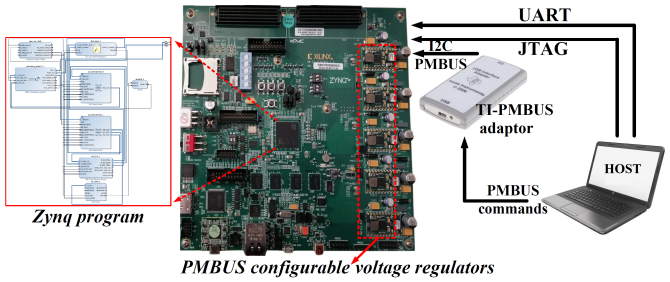

Figure 3. Setup for scaling FPGA supply voltages.

Furthermore, a Fully Connected Neural Network (FC-NN) [13] with ABFT in matrix multiplications on the largest layers was synthesized on the FPGA. The checksum augmentation of the inputs and the inspection of the results were both done by circuit logic on the fly. The neural network was fed with inputs from the test dataset.

In our implementation, matrix A was augmented with checksums, hence computing C^c as $\begin{bmatrix} C \\ eAB \end{bmatrix} = \begin{bmatrix} A \\ eA \end{bmatrix} \times B$. The checksum vector of the matrix A, i.e., eA, is computed on the fly by accumulation of the rows of matrix A as they are read and multiply the columns of matrix B. Similarly, the results of each row-column multiplication of A and B are accumulated in a row-wise manner to form vector eC. Once all the rows of A have been consumed, vector eA becomes available and multiplies B. The PS receives the computed eC, eAB and C at the same time, and compares the vectors eC and eAB to detect errors, verifying the results. In the experiments, once the onset of errors was reported by the ABFT mechanism, the host controller increased the voltage gradually until the errors disappeared. In practical applications, the frequency drop would come first. For each operating point, at least 100 000 multiplications with random values were carried out. The checksum violation was detected by inspecting the output checksum property at the PS side.

The focus of the experimentation was the detection of hardware related errors at each voltage-temperature operation point. Therefore, to avoid false positives from floating point mantissa round-off errors, the matrices adhered to the error avoidance restrictions laid out by Dutt and Assaad [14]. The detections by the ABFT were compared against the actual errors captured by element-wise comparison of the result and the pre-computed matrices.

ABFT Overheads: Employing ABFT for a $N \times N$ matrix multiplication increases the number of arithmetic operations from $(2N^3 - N^2)$ to $(2N^3 + 5N^2 + 4N)$, while checking for errors inspecting the row checksums requires additional N^2 summations and N comparison. This totals in $(2N^3 + 6N^2 + 5N)$ operations. The overheads from ABFT for large matrices met in neural network implementations, e.g., 128×128 elements is only 2%. For storage, the overhead is simply an extra row and/or column.

V. Results

The setup was used to investigate the potential of ABFT in operating the FPGA at reduced voltages. The important voltage rails of the FPGA were investigated separately and together.

A. Voltage Reduction of Internal Logic Circuits

The internal logic circuitry of the FPGA section of the SoC dominates the power dissipation. Fig. 4 shows the power consumption of the section when the operating voltage is reduced towards $0.70V$ in $0.01V$ steps (from right to left in the figure). Down-scaling the supply voltage levels beyond the crashing points of the FPGA, i.e., 710mV, 570mV and 1100mV for VCCINT, VCCBRAM and VCCAUX, respectively, causes the FPGA to stop operating. The clocking for PL was set to 250MHz exploiting all the timing margins at the crashing voltage. Reduction from the default 1V to 0.77V saves half of the power dissipation without incurring any errors.

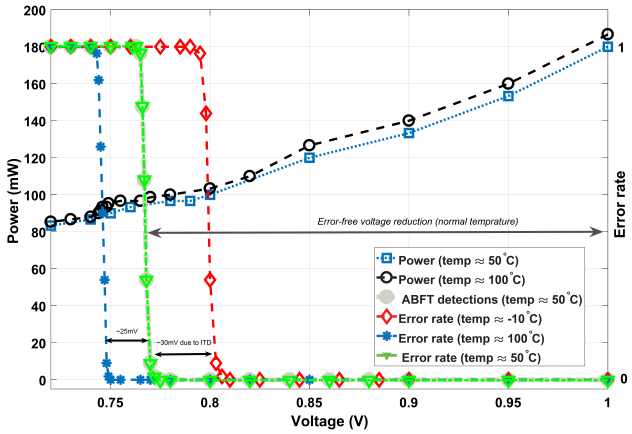

Figure 4. Error rate and power consumption through a VCCINT voltage-rail.

Different temperature points of operation were explored as well, using one of the evaluation boards, and plotted in Fig. 4. This was done by warming the chip up to slightly over 100°C and cooling it down to -10°C. The temperature was monitored with the internal temperature sensor of the Zynq device [2]. Due to the Inverse Temperature Dependence phenomena [1], the device can operate at even higher frequencies with a

[1] https://www.youtube.com/watch?v=5klGBAmTmdA

reduced voltage when the chip is at a higher temperature, as shown in Fig. 4 [7]. The same trend was observed for $-10°C$ but the error on-set voltage was measured to be around 0.806mV, i.e. 30mV and 55mV higher than that of 55°C and 100°C, respectively. The ABFT approach can manage such temperature dependency and exploit the performance opportunity.

Silent errors here are defined as the existing errors in matrix elements that "neutralize" each other when the checksum is inspected. The ABFT scheme has a non-zero silent error rate [12]. However, in our experiments with randomly generated matrices, no silent errors were detected. The inputs in the experiments were *i.i.d.* Gaussian random matrices, but the potential dependence of the silent error rate on the matrix type and distribution of element values has been recognized. We surmise that substantially prolonged experimentation with different matrices may expose a non-zero, but probably negligible silent error rate, as the analysis by Huang and Abraham indicates [12].

B. Voltage Reductions of BRAMs

In the previous experiments, errors in BRAM and communications could not be discriminated from computational errors. When the voltage is reduced, the data might be stored or retrieved incorrectly. Therefore, a study on both the power and error impacts of BRAM voltage reductions is necessary.

We employed ABFT as the error detection mechanism for the memory instead of ECCs [15], adjusting the operating voltage and frequency based on error detection. The input matrices augmented with checksums were stored in the BRAM. The recalculation of the checksums, and verifying them against the stored ones was done by the PS when reading the matrices back from the BRAM. The BRAM voltage was reduced from the default 1V towards 0.5V. In total, 91% of the BRAM capacity was utilized. Power dissipation was reduced from ≈32mW at 1.0v to ≈17mW at 0.58V, where errors start appearing, as shown in Fig. 5. This experiment demonstrated ABFT in detecting BRAM errors independently from the PL. With large matrices ABFT incurs low memory overheads, while it suffers from higher computational cost in comparison to ECCs that support error correction to some extent. However, ECCs do not enable the detecting of computation errors.

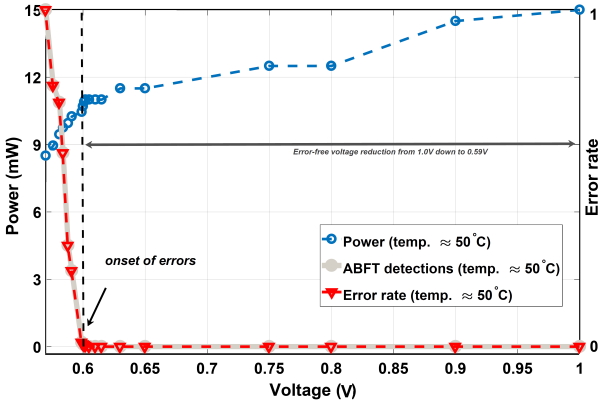

Figure 5. BBRAM error rate and power consumption versus voltage.

C. Voltage Reductions of Auxiliary Circuits

Auxiliary circuitry for PL, including the mixed-mode clock manager, amplifiers, buffer circuits of IO, etc. are powered from the VCCAUX voltage-rail. The experiment in this section is to ensure that overheads from auxiliaries do not end up canceling gains from the main FPGA components at reduced voltages. Their voltage dependency was investigated to identify the threshold for system crashing. The power dissipation and error rate are shown in Fig. 6 against voltage. A minimal margin exists between the onset-of-errors voltage and system crash, which advises against aggressive voltage scaling of auxiliary components, unless done with extreme care.

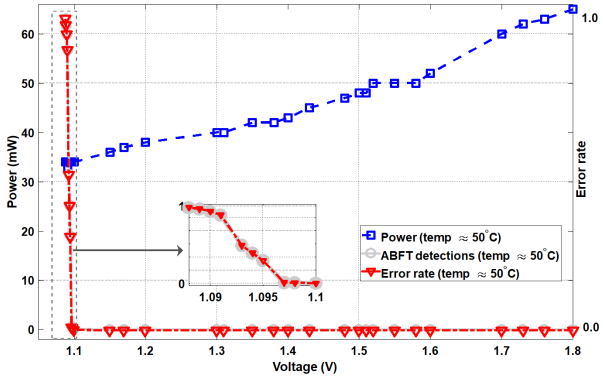

Figure 6. VCCAUX error rate and power consumption versus voltage.

D. Total Energy Optimization

Based on the above results, the voltages of the FPGA SoC should be controlled in a manner that the errors appear first in computations and BRAM to avoid an unrecoverable crash. Aiming for the lowest achievable energy dissipation, the voltages of all three domains of interest were reduced one after the other using 1% steps from the default voltages. When errors appeared, the most recent change was backtracked, occasionally by several steps before recovery, and then stepping forward to the detected threshold. The results of this process are shown in Fig. 7.

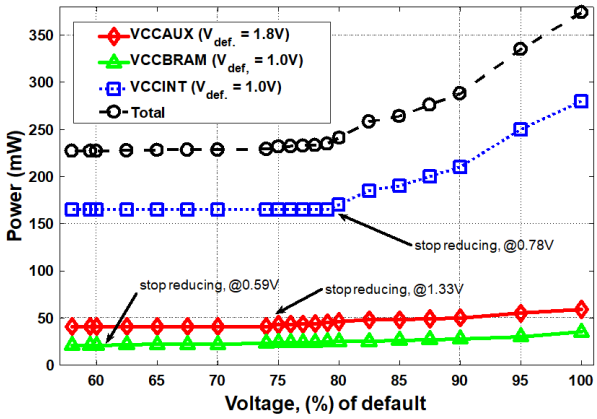

Figure 7. Impact of reducing each voltage rail from its default.

The voltage reductions of the VCCINT that powers the PL section generated the first errors. While VCCBRAM and

VCCAUX could be set at a lower voltage, VCCINT could not be dropped below 0.77V. The total power consumption was cut from ≈380mW to ≈220mW. It can be seen in Fig. 7, that in this experiment, the error threshold voltage for auxiliary circuits was at around a 200mV higher level than when VCCAUX was reduced independently from VCCINT and VCCBRAM, see Fig. 6.

The power measurements do not include the small power dissipation in the ARM processor, used for control and running the ABFT error checking of the matrix multiplication results on the SoC platform. Checking all the results in the experiments required 3 MCycles out of 660 MCycles/s of the processor, corresponding to approximately 2.5mW. Experiments on the second board revealed similar results, except for negligible differences in error on-set voltage. The utilization report shows $\approx 7\%$ of FPGA resources being used for ABFT checksum computations at a 32×32 matrix size. With larger matrices, the resource utilization drops significantly, since the overhead ratio scales with $O(1/N)$.

By operating at reduced voltages our run-of-the-mill design for the effective matrix size 31×31 achieves a power efficiency of 6.2 GFLOPS/W with ABFT augmentation. That is close to highly optimized designs [1].

Fully connected neural network with ABFT: As shown by Fig. 8, the ABFT error detections appear slightly earlier than the point where the output of the NN differs from the golden result. The difference is probably due to the fault-tolerance of the neural network. As ABFT in this case acts as an early warning, it can be used to control the operating point. Zhao et al. [16] integrated ABFT in convolution operations of DNNs and showed it incurred up to only 8% overheads.

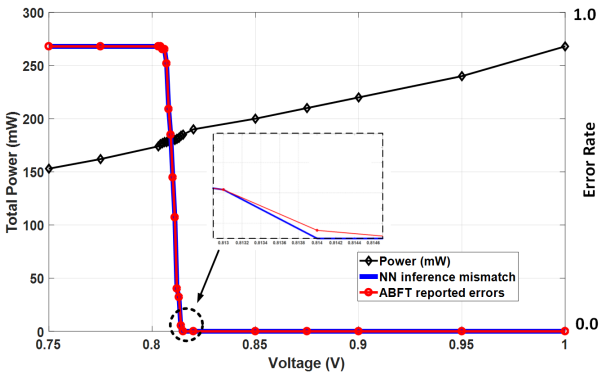

Figure 8. Voltage scaling impact on an ABFT augmented NN.

VI. Discussion

To the best knowledge of the authors, this is the first contribution in which a low overhead algorithmic error detection technique was employed to realize a low-voltage processing solution. The proposed scheme can be used with any matrix multiplier design. Implementations of other algorithms can be foreseen to benefit from matching ABFT schemes. Designing chips to operate at optimum near-threshold points is notoriously difficult. The demonstrated ABFT-based error feedback mechanism, is an enabling approach that suits extreme low-power applications where occasional errors can be tolerated. We envisage ASIC designs utilizing our scheme.

VII. Summary

The objective of the current contribution is to present and showcase safe voltage reduction of matrix accelerators using ABFT. The results demonstrate that the energy efficiency of FPGA based matrix multiplication can be improved substantially by using HLS tools and eliminating conservative voltage margins. The utility of ABFT as a low-cost feedback mechanism was shown in controlling the voltages and detection of errors originating from the internal logic, BRAM, and auxiliary circuits [†].

References

[1] A. Ahmad and M. A. Pasha, "Optimizing Hardware Accelerated General Matrix-Matrix Multiplication for CNNs on FPGAs," *IEEE Transactions on Circuits and Systems II: Express Briefs*, 2020.

[2] Xilinx Inc., "7000 All Programmable SoC ZC702 Evaluation Kit," Xilinx Inc., Tech. Rep., 2015. [Online]. Available: http://www.xilinx.com/products/boards-and-kits/ek-z7-zc706-g.html

[3] M. Wu, B. Yin, G. Wang, C. Dick, J. R. Cavallaro, and C. Studer, "Large-scale mimo detection for 3gpp lte: Algorithms and fpga implementations," *IEEE Journal of Selected Topics in Signal Processing*, vol. 8, no. 5, pp. 916–929, 2014.

[4] J. Nunez-Yanez, "Energy proportional neural network inference with adaptive voltage and frequency scaling," *IEEE Transactions on Computers*, vol. 68, no. 5, pp. 676–687, 2018.

[5] P. Koutsovasilis, C. Antonopoulos, N. Bellas, S. Lalis, G. Papadimitriou, A. Chatzidimitriou, and D. Gizopoulos, "The impact of cpu voltage margins on power-constrained execution," *IEEE Transactions on Sustainable Computing*, pp. 1–1, 2020.

[6] R. G. Dreslinski, M. Wieckowski, D. Blaauw, D. Sylvester, and T. Mudge, "Near-threshold computing: Reclaiming moore's law through energy efficient integrated circuits," *Proceedings of the IEEE*, vol. 98, no. 2, pp. 253–266, 2010.

[7] M. Safarpour, R. Inanlou, and O. Silvén, "Algorithm Level Error Detection in Low Voltage Systolic Array," *IEEE Transactions on Circuits and Systems II: Express Briefs*, pp. 1–1, 2021.

[8] G. Papadimitriou, M. Kaliorakis, A. Chatzidimitriou, C. Magdalinos, and D. Gizopoulos, "Voltage margins identification on commercial x86-64 multicore microprocessors," in *2017 IEEE 23rd International Symposium on On-Line Testing and Robust System Design (IOLTS)*. IEEE, 2017, pp. 51–56.

[9] I. Ahmed, L. L. Shen, and V. Betz, "Becoming more tolerant: Designing fpgas for variable supply voltage," in *2019 29th International Conference on Field Programmable Logic and Applications (FPL)*. IEEE, 2019, pp. 1–8.

[10] I. Ahmed, S. Zhao, O. Trescases, and V. Betz, "Measure twice and cut once: Robust dynamic voltage scaling for fpgas," in *2016 26th International Conference on Field Programmable Logic and Applications (FPL)*. IEEE, 2016, pp. 1–11.

[11] W. Jiang, H. Yu, J. Zhang, J. Wu, S. Luo, and Y. Ha, "Optimizing energy efficiency of cnn-based object detection with dynamic voltage and frequency scaling," *Journal of Semiconductors*, vol. 41, no. 2, p. 022406, 2020.

[12] K.-H. Huang and J. A. Abraham, "Algorithm-based fault tolerance for matrix operations," *IEEE transactions on computers*, vol. 100, no. 6, pp. 518–528, 1984.

[13] R. Novickis, D. J. Justs, K. Ozols, and M. Greitāns, "An approach of feed-forward neural network throughput-optimized implementation in fpga," *Electronics*, vol. 9, no. 12, p. 2193, 2020.

[14] S. Dutt and F. T. Assaad, "Mantissa-preserving operations and robust algorithm based fault tolerance for matrix computations," *IEEE transactions on computers*, vol. 45, no. 4, pp. 408–424, 1996.

[15] D. Shin, J. Park, J. Park, S. Paul, and S. Bhunia, "Adaptive ecc for tailored protection of nanoscale memory," *IEEE Design & Test*, vol. 34, no. 6, pp. 84–93, 2016.

[16] K. Zhao, S. Di, S. Li, X. Liang, Y. Zhai, J. Chen, K. Ouyang, F. Cappello, and Z. Chen, "Algorithm-based fault tolerance for convolutional neural networks," *IEEE Transactions on Parallel and Distributed Systems*, 2020.

[†]A patent application based on this research has been filed by the University of Oulu early 2021