FeFET-Based Binarized Neural Networks Under Temperature-Dependent Bit Errors

Mikail Yayla, Sebastian Buschjäger, *Student Member, IEEE*, Aniket Gupta, Jian-Jia Chen, *Senior Member, IEEE*, Jörg Henkel, *Fellow, IEEE*, Katharina Morik, *Member, IEEE*, Kuan-Hsun Chen, *Member, IEEE*, and Hussam Amrouch, *Member, IEEE*

Abstract—Ferroelectric FET (FeFET) is a highly promising emerging non-volatile memory (NVM) technology, especially for binarized neural network (BNN) inference on the low-power edge. The reliability of such devices, however, inherently depends on temperature. Hence, changes in temperature during run time manifest themselves as changes in bit error rates. In this work, we reveal the temperature-dependent bit error model of FeFET memories, evaluate its effect on BNN accuracy, and propose countermeasures. We begin on the transistor level and accurately model the impact of temperature on bit error rates of FeFET. This analysis reveals temperature-dependent asymmetric bit error rates. Afterwards, on the application level, we evaluate the impact of the temperature-dependent bit errors on the accuracy of BNNs. Under such bit errors, the BNN accuracy drops to unacceptable levels when no countermeasures are employed. We propose two countermeasures: (1) Training BNNs for bit error tolerance by injecting bit flips into the BNN data, and (2) applying a bit error rate assignment algorithm (BERA) which operates in a layer-wise manner and does not inject bit flips during training. In experiments, the BNNs, to which the countermeasures are applied to, effectively tolerate temperature-dependent bit errors for the entire range of operating temperature.

Index Terms—Non-volatile memory, FeFET, temperature, neural networks, bit error tolerance

1 Introduction

Neural Networks (NNs) are broadly applied in numerous fields. Managing the resource demand of NNs, however, is a challenge. To achieve high accuracy, NNs rely on deep architectures with millions of parameters requiring a large amount of memory. For NN accelerators with on-chip Static random-access memory (SRAM), it has been reported that over 50 percent of the system power is used by the memory, e.g., [1], [2].

Non-volatile memories (NVMs) for machine learning algorithms may achieve energy-efficient and sustainable inference. In particular, neural networks on different types of NVMs have been developed recently, e.g., resistive RAM (RRAM) [3], [4], spin-transfer torque RAM (STT-RAM) or magnetoresistive RAM (MRAM) [5], [6], [7], [8], [9], multi-level charge-trap transistors (CTT) memory [10], and ferroelectric-based memories (FeRAM or FeFET) [11], [12], [13], [14]. Using RRAM (in [3]) and CTT (in [10]) for NN inference systems leads to large factors in energy saving compared to using SRAM.

A common technique to reduce the energy consumption of memories, including NVMs, is voltage scaling. When this is pushed to the extreme, high bit error rates can occur, even up to 5-10 percent. For some memory technologies, such as STT-RAM or RRAM, the bit error rate increases exponentially with respect to the reduction of the supply voltage. Such bit errors can degrade the accuracy of NNs to unacceptable levels if no countermeasures are employed. In the literature, the energy-reliability trade-off has been explored by training NNs to be bit error tolerant when applying voltage scaling, e.g., for RRAM [4] and MRAM or STT-RAM [8], [9]. Another factor that has an impact on the bit error rates is process variation, which has been reported with non-negligible effects on NNs executed on RRAM [7], [15].

Among all existing memory technologies, ferroelectric FET (FeFET) is one of the most promising fully CMOS-compatible NVM technology [16]. Furthermore, FeFET-based memories achieve read and write latencies within 1 ns, which is comparable to SRAM, while using low energy [16]. Another benefit of FeFET is the high density, since FeFET memory cells consist of only one transistor.

In order to use FeFET for NNs, it is necessary to analyze its error model. To the best of our knowledge, the sources of bit errors on FeFET were not explored yet, and hence, there is no FeFET bit error model in the literature. The underlying mechanism in FeFET where information is stored, is the available dipoles within the ferroelectric layer added to the transistor. However, the directions of those dipoles (which

- *Mikail Yayla, Jian-Jia Chen, and Kuan-Hsun Chen are with the Design Automation for Embedded Systems Group, TU Dortmund University, 44227 Dortmund, Germany. E-mail: {mikail.yayla, jian-jia.chen, kuan-hsun.chen}@udo.edu.*
- *Sebastian Buschjäger and Katharina Morik are with the Artificial Intelligence Group, TU Dortmund University, 44227 Dortmund, Germany. E-mail: {sebastian.buschjaeger, katharina.morik}@udo.edu.*
- *Aniket Gupta and Jörg Henkel are with the Chair for Embedded Systems, Karlsruhe Institute of Technology, 76131 Karlsruhe, Germany. E-mail: ganiket919@gmail.com, henkel@kit.edu.*
- *Hussam Amrouch is with the Chair for Semiconductor Test and Reliability (STAR), University of Stuttgart, 70174 Stuttgart, Germany. E-mail: amrouch@iti.uni-stuttgart.de.*

Manuscript received 9 Sept. 2020; revised 5 May 2021; accepted 2 Aug. 2021. Date of publication 13 Aug. 2021; date of current version 9 June 2022.
(Corresponding authors: Mikail Yayla and Hussam Amrouch.)
Recommended for acceptance by R. Marculescu.
Digital Object Identifier no. 10.1109/TC.2021.3104736

determines the stored state, i.e., either logic 0 or logic 1) are sensitive to temperature in which fluctuations in temperature can lead to flipping the direction of dipoles. This manifests itself as changes in the induced ferroelectricity and, hence, sensing circuits will later erroneously read the stored value, i.e., a bit flip will occur during reading. Process variation (either within the domains of the ferroelectric material or within the underlying transistor itself), on the other hand, manifests itself as changes in electrical characteristics of the FeFET device such as the threshold voltage, ON current, OFF current, etc. This leads, similar to temperature effects, to fluctuation in the sensing current during read operations and hence bit flips can occur. This raises the following key question: **(Q1)** *What is the bit error model of FeFET?*

In this work, we cover both types of bit flip sources, i.e., run-time errors induced by temperature effects and design-time induced by process variation. In addition to the above observation of bit errors on FeFET due to temperature and process variations, our model in Section 2 also shows that the effect of temperature manifests itself as highly *asymmetric* bit error rates. This means that the probability of a bit flip from logical '1' to '0' is different from logical '0' to '1'.

In the literature, the effects of asymmetric bit error rates on NNs have not been assessed and no countermeasures against them have been investigated yet. For these reasons we explore the following questions: **(Q2)** *Do the FeFET asymmetric bit errors cause significant accuracy drop in NNs?* **(Q3)** *How can we exploit the asymmetry of the FeFET bit error model to achieve tolerance against FeFET bit errors?*

Our Contributions. Answering these questions, we focus on binarized neural networks (BNNs), which are highly resilient to bit errors [17]. Specifically, we make the following contributions:

- We answer **Q1** in Section 2 revealing the critical impact of temperature on the reliability of FeFET-based memories by a temperature-dependent asymmetric bit error model that we acquired from precise simulations. Our physics-based model extracts the error rate based on realistic FeFET devices in which the underlying FET transistor is calibrated with Intel 14nm FinFET measurement data and the added ferroelectric layer is also calibrated with measurements from fabricated ferroelectric capacitor. The effects of process variation as well as temperature are accurately modeled using commercial Technology-CAD (TCAD) tool flows from Synopsys. For accurate modeling, all required physics models including quantum physics are included.
- We answer **Q2** by a series of evaluations on different BNNs. Our experimental results show that the accuracy degradation of BNNs can amount to 35 percent when no bit error tolerance treatment or countermeasures are employed. When the existing bit flip training method is applied without taking the asymmetry into account, the accuracy degradation can be up to 10 percent, even when the probability of a bit flip from logical '0' to '1' is 1.090 percent and that of logical '1' to '0' is 2.198 percent.
- We exploit the asymmetry of the FeFET bit error rates (**Q3**) by a bit error rate assignment algorithm (BERA) which operates in a layer-wise manner. BERA takes an accuracy drop, which is estimated per layer, as an input and assigns the bit error rates in a layer-wise manner such that the accuracy drop is minimized. In contrast to the existing methods for bit flip training, which employ bit flip injection, BERA does not require any bit flip injection during training.

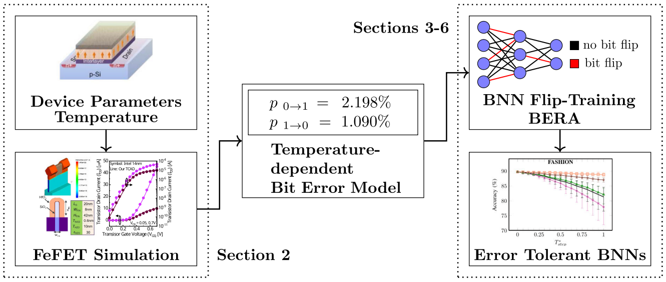


Fig. 1. The general overview of our work. Errors due to temperature effects, stemming from underlying FeFET devices, are modeled and then injected during the BNN training. This provides robust BNNs against run-time temperature fluctuations.

Fig. 1 illustrates the overview of this work. In Section 2, we present our FeFET model and the experiments with which we extract the temperature-dependent bit error rates. In Section 3, we introduce the system model, i.e., the BNNs and the data stored in NVM. In Section 4, we present the BNN execution in which less layers are susceptible to bit errors. In Section 5 we present the methods we use to protect the BNNs from FeFET bit errors.

The experiment setup for bit error tolerance training of the BNNs is detailed in Section 6. In Section 7, we first assess the impact of the FeFET bit errors on BNN accuracy when no countermeasures are employed, and then we evaluate our proposed countermeasures. We provide a literature review in Section 8 and conclude the work in Section 9.

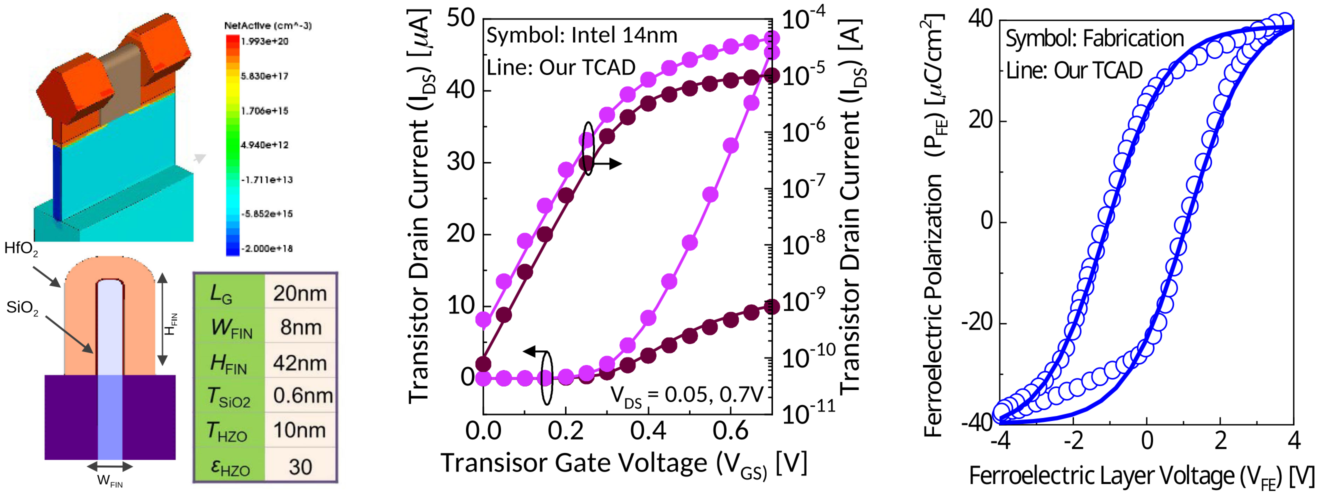
2 FeFET-Based NVM: From Device Calibration to Error Modeling

This work is the first to study bit error rates in FeFET-based NVM memory. Let us start with an overview of FeFET technology in Section 2.1 and explain the FeFET devices used in this paper in Section 2.2, before we provide a bit error model for FeFET devices in Section 2.3.

2.1 Overview of FeFET Technology

The discovery of ferroelectricity in hafnium oxide-based materials in 2012 has paved the way for Ferroelectric Field-Effective Transistor (FeFET) to become compatible with the existing CMOS fabrication process [18], [19]. Thereafter, many prototypes from both academia and industry have successfully demonstrated the ability of turning conventional FET transistors into Non-Volatile Memory (NVM) devices by integrating a ferroelectric layer inside the transistor gate stack. This allowed to integrate, for the first time, NVM devices along side logic transistors within the same silicon die, unlike other existing NVM technologies that still face challenges when it comes to CMOS compatibility.

Besides the compatibility of FeFET technology with the current CMOS fabrication process, which is an indispensable condition for any emerging technology to become reality, FeFET



(a) 14nm FinFET calibration with Intel measurement data

(b) 14nm FinFET calibration with Intel measurement data

(c) Ferroelectric capacitor (FeCAP) calibration with fabrication data

Fig. 2. Ferroelectric FET (FeFET)-based NVM device calibration [20].

technology provides the highest density, which is essential for on-chip memories, because every FeFET-based memory cell consists of only a single transistor [16].

FeFET, in principle, is similar to conventional FET transistors used in all existing CMOS technologies nowadays with only one exception regarding to the gate-stack. In FeFET devices, a thick (10nm) layer of a ferroelectric material is additionally deposited on top of the oxide layer. The presence of ferroelectricity creates a hysteresis behavior in which the electrical property of the underlying transistor considerably shifts based on the previously-applied gate voltage. When a positive write voltage (+4V) is applied to the transistor gate, the polarization direction within the ferroelectric material is switched in a certain direction that supports the transistor channel formulation. Hence, a relatively high drain current (in the order of micro Ampere) will be later provided by the transistor after the write voltage is ceased. The drain current in this case is typically called low-V_{th} curve because it has a very low threshold voltage (V_{th}). On the other hand, when a negative write voltage (-4V) is applied, the polarization direction within the ferroelectric material is oppositely switched, which strongly resists the transistor channel formulation. Hence, a very low drain current (in the order of nano Ampere) will be later provided by the transistor after the voltage is ceased. The drain current in this cases is typically called high-V_{th} curve because it has a very high threshold voltage (V_{th}).

The very large difference in the current provided by the transistor, depending on the dipoles direction within the ferroelectric material, allows the differentiation between stored logic '1' (when *positive* write voltage was previously applied, i.e., low-V_{th} curve case) and stored logic '0' (when negative write voltage was *negative* write voltage was previously applied, i.e. high-V_{th} curve case).

2.2 Our Calibrated 14nm FeFinFET Device and Measurements

We first implemented a 14nm n-type FinFET device, as shown in Fig. 2a, using Synopsys Technology CAD (TCAD) tool flow [21], which is the standard commercial tool to simulate device fabrication of semiconductor technologies. Afterwards, we calibrated the built device to reproduce data measurements for production-quality 14nm FinFET device from Intel [22]. As demonstrated in Fig. 2b, the electrical characteristics of our built FinFET match very well the 14nm Intel measurement data. Afterwards, we deposited a 10nm ferroelectric layer ($Hf_{0.5}Zr_{0.5}O_2$) on top of the oxide layer.

Fig. 2b demonstrates how the hysteresis-loop of polarization versus voltage (which captures the nonvolatile property in FeFET devices) matches very well the measurement data from a fabricated ferroelectric capacitor [23].

We adopt the calibrated FeFET device presented above to build a bit error model. We investigate the impact that temperature increase has on the behavior of FeFET-based NVM. In practice, when the temperature rises, the key characteristics of the ferroelectric layer (i.e. remanent polarization P_r and coercive field E_c) degrade which considerably reduces the available noise margin as shown in Fig. 3. At higher temperatures the distance between the two current curves, i.e. low-V_{th} and high-V_{th} curves, (which represents the two logic states) becomes narrower. Hence, during reading operations, the sensing circuit might not be able to

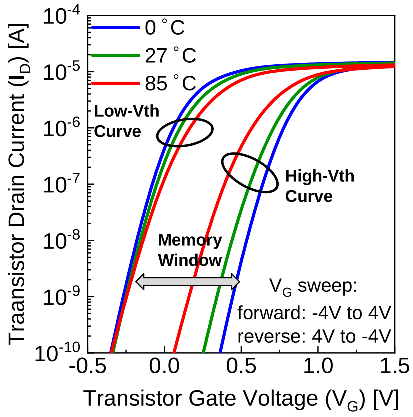


Fig. 3. Temperature impact on the memory window of FeFET. A higher temperature impact more considerably the high-V_{th} curve that represents logic '0' than low-V_{th} curve that represents logic '1' [20].

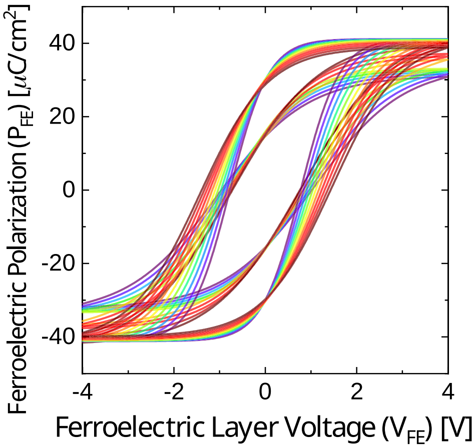


Fig. 4. Impact of intrinsic variations in the ferroelectric layer on the induced hysteresis loop. Process variation together with temperature effects cause errors due to read operations of FeFET-based NVM.

correctly distinguish between logic ‘0’ and ‘1’ because the current level associated with each logic state becomes very close to each other or even overlapping - especially when process variation effects come into play. Note that temperature effects have been measured from fabricated devices and then modeled inside our built TCAD-based FeFET. Please refer to [24], [25] for further details on the device calibration and measurements.

To calculate probability of error during read operations, we perform accurate TCAD Monte-Carlo (MC) simulations. As can be seen in Fig. 4, process variation effects degrade the quality of ferroelectric layer. From the performed MC simulations, the probability of error when a logic ‘1’ is erroneously read while logic ‘0’ was previously stored p_{01}, and when probability of error when a logic ‘0’ is erroneously read while logic ‘1’ was previously stored p_{10} are calculated for different temperatures. Those calculated error-rates are then employed at the system level to investigate the ultimate impact on the inference accuracy.

Details on Our TCAD-Based Modeling of FeFET. As earlier mentioned, in our work the entire Ferroelectric FinFET (FeFET) device is fully implemented and modeled in Technology CAD (TCAD) framework (Synopsys Sentaurus). Concisely, we model and calibrate a 14nm FinFET device using TCAD, and then the high-κ layer (1.7nm) in the gate stack is replaced with a 10nm ferroelectric layer ($Hf_{0.5}Zr_{0.5}O_2$) located on top of the SiO_2 layer. It is noteworthy that our device modeling is not just a combination of a Ferroelectric capacitor model and a FinFET model but, in fact, the complete ferroelectric FinFET device is fully modeled in TCAD using a meta-ferroelectric-insulator-semiconductor (MFIS) stacks. For accurate modeling, the density gradient model is used to account for the quantum confinement effects and, additionally, the thin-layer mobility model is used to account for the scaled fin dimensions. The crystal orientation effects on the channel mobility, channel strain-related mobility improvement, and ballistic mobility to account for the quasi-ballistic carrier transport in such nano-scaled dimensions have also been considered. The S/D doping, sub-fin doping, gate metal work function, S/D series resistance, low-field mobility parameters and high-field saturation parameters were carefully calibrated to reproduce the measurement data of 14nm FinFET. A Preisach model is applied in TCAD to describe the ferroelectric polarization. In this model, the FE film is considered to be composed of multiple independent switching domains. Each domain charge voltage relationship can be described as a classical rectangular hysteresis loop. The coercive field of each loop follows a certain distribution. As a result, the total ferroelectric behavior is obtained by integrating all the domain responses. This model has been widely applied to describe large ferroelectric films characteristics.

Temperature Effects Modeling. We first extracted the main characteristics of ferroelectric capacitor, i.e., remnant polarization (P_r), saturation polarisation (P_s), and coercive field (E_c), under different operating temperatures. Then, we model the temperature dependency of P_r, P_s, and E_c inside our calibrated TCAD-based FeFET. Note that existing models in Synopsys TCAD only account for temperature effect in the underlying transistor (FET) and do not account for temperature degradation in ferroelectric layer. Therefore, we had to build the model of temperature dependency based on our measurement data inside the TCAD framework.

To Model the Effect of Process Variation. We consider variation in both underlying transistor as well as the added ferroelectric layer. In the underlying transistor, we have considered each of the work function variation, random dopant fluctuation RDF, interface trap, variations in the thickness of the ferroelectric layer and interfacial layer (SiO_2). In the ferroelectric layer, we have considered the variation in all ferroelectric parameters (P_s, P_r, and E_c). Note that, similar to temperature, existing models in Synopsys TCAD only account for variation effects in the underlying transistor (FinFET) and do not account for any variation in the ferroelectric layer. Therefore, we have run Monte-Carlo simulations for our ferroelectric capacitor, calibrated against fabrication, in order to capture and extract how variation in the ferroelectric parameters (P_r, P_s, and E_c) impact the ferroelectric capacitor.

Probability of Error Extraction. After integrating the temperature and variation effects inside our calibrated TCAD models, we perform Monte-Carlo simulations for the entire FeFET device. This provides us with the complete I_D-V_G hysteresis loops. Then, for a certain read voltage, we extract the probability of error in which a high V_{th} curve is wrongly classified as a low V_{th} curve and vice versa. In other words, we calculate the probability that a stored logic ‘0’ is read as logic ‘1’ (i.e., $P_{error}(0 \rightarrow 1)$) and a stored logic ‘1’ is wrongly read as logic ‘0’ (i.e., $P_{error}(1 \rightarrow 0)$). Note that for different read voltage values the noise margin, i.e., the available separation between the two states, high V_{th} and low V_{th} curves, is different. Therefore, the probability of errors ($P_{error}(1 \rightarrow 0)$ and $P_{error}(0 \rightarrow 1)$ varies with read voltage as reported

Discussion on the Impact of Defects. The impact of interface traps are not considered in this work. Such traps and other types of generated defects are critical and can degrade the memory window during the projected lifetime. Recent studies such as [26], [27] aimed at investigating and modeling the impact of interface and oxide traps on the memory window at both device and system levels, respectively. In addition, charge trapping and detrapping at Si-SiO_2 interface plays an important role in degrading the memory window and it has shown that controlling the charge trapping and the resulting imprinting is necessary in order to ensure the reliability of the FeFET [28]. Recently, it has been shown in [29] that the non-perfect screening of the polarization charges may lead to a residual electric field. This counteracts the ferroelectric polarization and results in depolarization field that may degrade the memory window of FeFET devices.

2.3 Our FeFET Bit Error Model

As shown in Fig. 3 temperature impacts on the memory window of FeFET. The memory window is defined as the distance between the low-V_{th} and curves high-V_{th} curves (i.e. I_D-V_G curves) which represents logic ‘1’ and logic ‘0’, respectively. Importantly, temperature increase has a more considerable impact on the high-V_{th} curve than the low-V_{th} curve. As can be noticed in Fig. 3, at a higher temperature high-V_{th} moves towards the left side, whereas, low-V_{th} curve remains almost unaffected. Hence, the memory window becomes smaller and the resiliency of FeFET-based NVM becomes smaller. Hence, the likelihood of errors during read operations becomes larger. However, because temperature impacts *asymmetrically* the low-V_{th} curve and high-V_{th} curves, errors in logic ‘1’ and ‘0’ will occur *asymmetrically*. In practice, the bit error rate of p_{10} is smaller than the bit error rate p_{01} because temperature impacts logic ‘0’ higher than logic ‘1’. As explained, the read operation in FeFET is performed by applied a gate voltage to the transistor and then sensing the provided drain current (I_D). Based on the applied gate voltage (V_G), the probability of error will be different. As can be noticed in Fig. 3, based on the selected V_G, the temperature-induced shift in the high-V_{th} curve is different and hence reading at different V_G can result in different probability of errors.

As shown in Fig. 4, intrinsic variations within the ferroelectric layer strongly impact the induced hysteresis loop. This, in turn, seriously reduces the resiliency of FeFET-based NVM devices to noise and increases the probability of error, as well. Both temperature effects and process variation effects together degrade the reliability of FeFET-based NVM devices during run-time and hence errors during read operations occur. Those errors will *asymmetrically* impact the FeFET-based NVM devices and the probability of flipping logic ‘0’ and logic ‘1’ is different.

Since the probability of bit error depends on the applied read voltage (i.e. gate voltage V_G). Therefore, we estimate the flip probabilities (p_{01}, p_{10}) at different read voltages 0.1V and 0.25V. The bit error rates of the FeFET bit error model at 85°C are $(p_{01}, p_{10}) = (2.198\%, 1.090\%)$ for the case of using read voltage 0.1V and $(p_{01}, p_{10}) = (2.098\%, 0.190\%)$ for using read voltage 0.25V.

In Fig. 5, we summarize the relation between temperature and P_r (remnant polarization), P_s (saturation polarisation), and E_c (coercive field), which capture the key properties of the ferroelectric material. The three parameters degrade linearly with temperature increase. Therefore, we use the value $T_{step} \in \{0, 1, \ldots, 16\}$ for describing the magnitude of temperature.

Based on the above model, the bit error rate (BER) at temperature T for any $0°\text{C} \leq T \leq T_{peak} = 85°\text{C}$ is $BER(T) = \frac{T}{T_{peak}} \cdot (p_{01}, p_{10})$, where p_{01} and p_{10} are defined at 85°C above.

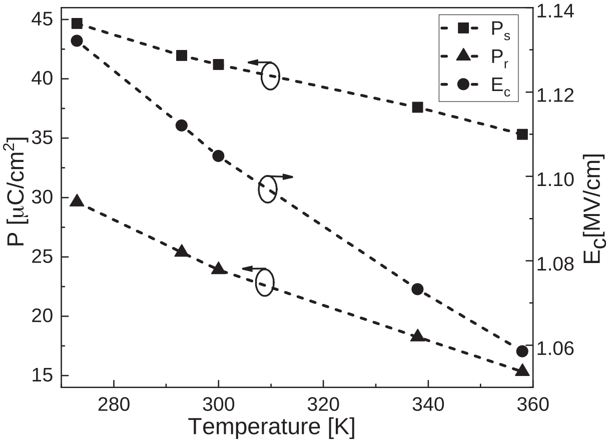


Fig. 5. Relation between P_r (remnant polarization), P_s (saturation polarisation), and E_c (coercive field) over temperature. These three parameters capture the key properties of the ferroelectric material.

It is noteworthy that Fig. 5 has two different Y axes (right and left). The curve of coercive field (E_C) belongs to the right Y-axis. Therefore, the coercive field degrades by temperature by merely 0.08MV/cm (from around 1.14MV/cm to 1.06MV/cm), which is not very significant. The range of temperature-induced degradation is aligned with presented measurement data in literature such as [30], [31]. In this work, we have relied on our previous work where we studied the impact of temperature on FeFET devices [20]. The impact of temperature increase on the P_r (remnant polarization), P_s (saturation polarisation), and E_c (coercive field) have been extracted from measured Q_{FE}-V_{FE} hysteresis loop at different temperatures [30].

3 System Model and Problem Definition

In this section, we provide the system model and the problem definition studied in this paper based on the bit error model presented in Section 2.

3.1 Binarized Neural Networks

Binarized Neural Networks (BNNs) are a relatively novel type of BNNs that were first proposed in 2016 in [32]. Since then, BNNs have gained large popularity in application cases where memory and inference latency are key issues and a small trade-off in accuracy is acceptable. In the following, we summarize the benefits of BNNs in four points.

(1) The key reason behind selecting BNN as a case study in our work, is the ability to perform training with errors and achieve error-tolerance neural network in BNN unlike in traditional deep neural networks (DNNs). Traditional neural networks use floating-point (e.g., 32 bits) or integer values (e.g., 8 bits) to represent the NN parameters (i.e., weights, activations, inputs, etc.). In such a case, the position of the occurred error (i.e., the bit flip in the value) does matter a lot. Specifically, in floating-point NNs, one bit error in one weight can cause the prediction of the NN to become useless (detailed in [33]). This typically occurs when a bit flip in the exponent of the floating point representation occurs leading to an error with an unacceptable magnitude. On the other hand, in BNNs, a flip of one bit in a binary weight or binary input causes a change of the computation result by merely 1 (with binarization to $\{0, 1\}$). Additionally, the output of

every neuron in the hidden layers is binarized, which has a saturating effect. (2) BNNs are very resource-efficient and hardware friendly. In BNNs, the memory needed to store the parameters and also the communication overhead is significantly reduced, because the floating-point or integer values are replaced with binary values. (3) With binary weights, the costly MAC operations are performed with simple XNOR and bitcount (often called popcount) circuits. Due to the simpler hardware operations, the inference latency is significantly reduced as a result. (4) BNNs synergize outstandingly with NVM. Traditional SRAM memory is typically used as on-chip memory, which suffers from high leakage power and large area footprint (6 transistors to store a single bit, comapred to merely 1 transistor in FeFET). Therefore, using a non-volatile memory such as ones based on FeFET for BNNs will considerably reduces the overall inference energy. Concisely, inefficient SRAM memories are replaced with efficient non-volatile FeFET memories.

Inference. To describe the inference of BNNs, we first start with regular NNs, In the general case, each layer in a convolutional neural network (NN) computes a convolution between the weights (parameters) and the input. Subseqently, the result of the convolution is passed through an activation function. The output of the activation function is then passed as the new input to the next layer. All layers are executed until the NN returns the final outputs. We refer to the entire computation as forward pass. We denote the convolution between weights and inputs, and the subsequent activation function on the result by $\sigma(\sum_i W_i^l X^{l-1})$, where σ is the activation function, W_i^l the weights of the ith filter in layer l, and X^{l-1} the input. In regular NNs, the weights, and activations are floating point values. In binarized neural networks (BNNs), the weights and activations are in $\{-1,+1\}$, which reduces the resource demand in memory [17]. Furthermore, the convolution and activation can be computed by

$$2 \cdot \mathit{POPCOUNT}(\mathit{XNOR}(W_i^l, X^{l-1})) - \#bits > T,$$

where *POPCOUNT* accumulates the number of bits set, $\#bits$ is the number of bits in the XNOR operands, and T is a learnable threshold parameter, the comparison against which produces an activation value in $\{-1,+1\}$ [17], [34]. Bit error tolerance in BNNs can be achieved by injecting bit flips during training [4]. Since BNNs can be trained for bit error tolerance, it is possible to use memories with bit errors like FeFET. Because of the resource efficiency and bit error tolerance of BNNs, the combination with FeFET is a highly promising design choice for low-power inference on future edge devices.

Training. For regular floating point NNs, stochastic gradient descent (SGD) is employed with mini-batches. The training data is described with $\mathcal{D} = \{(x_1,y_1),\ldots,(x_I,y_I)\}$ with $x_i \in \mathcal{X}$ as the inputs, $y_i \in \mathcal{Y}$ as the labels, and $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}$ as the loss function. We write the weight tensors of a layer as $W = (W^1,\ldots,W^L)$ and $f_W(x)$ as the output of the NN with weights W. The objective is to find a solution for the optimization problem

$$\arg\min_W \frac{1}{I} \sum_{(x,y)\in\mathcal{D}} \ell(f_W(x), y),$$

using a mini-batch SGD, by computing the gradient $\nabla_W \ell$ using backpropagation. In the training process of BNNs, the floating point weights are stored as well, so that the gradient updates can be applied to the weights. Only in the forward pass the weight and activations are binarized deterministically. This method of training BNNs was proposed by Hubara *et al.* [17]. If achieving high bit error tolerance is another objective besides accuracy, then bit flips need to be injected in the BNN data during the forward pass [4].

BNN Building Blocks. We focus here on binarized convolutional NNs (CNNs) which perform object recognition and use the following layers: convolutional, maxpool, batch normalization, and fully connected. In the convolutional (C) layer, a 2D convolution of the input and the filters is computed, where the weights are binarized. We use a filter size of 3×3 for every neuron in a C layer. If we use for example 64 neurons in a layer, we write C64. In the maxpool (MP) layer, in this study a window size 2×2 is used. The MP layer always choses the largest value in this window, and by this downsamples the input. The batch normalization (BN) layer in BNNs is used to stabilize the training process [34]. For forward propagation the BN layer in BNNs can be calculated by thresholding the input. The thresholds of the BN layer can be quantized to signed integers. In the fully connected layer (FC) every neuron in layer l is connected with every neuron in layer $l+1$. When for example 2048 neurons are used, we denote FC2048.

3.2 System Model

In this work, we focus on studying the impact of FeFET-induced bit errors (due to reliability reductions caused by temperature effects as described in Section 2.1) on the accuracy of BNNs and then investigating the existing tradeoffs between read energy and reliability of FeFET devices. Because read operations occur much more frequently than write operations in NNs, reducing the read voltage provides a considerable energy saving. In practice, we assume that FeFET memory devices are being reliably written (i.e., receiving a sufficiently large voltage to flip all ferroelectric domains during writing) and later during read operations the bit errors due to temperature occur. The intensity of the bit error depends on the applied read voltage. The lower the read voltage, the larger the bit error.

For these reasons, we assume that bit errors only occur during the read processes of the parameters and inputs (i.e., input images and activations, which are the inputs to the convolutional layers) in BNN inference. To clarify that, we present the considered system model with in Fig. 6. The assumed system consists of reliable traditional off-chip memory (e.g., DRAM) and unreliable emerging on-chip FeFET memory. To perform computations of one layer, the CPU initiates the retrieval of the weights from the off-chip DRAM to the on-chip FeFET memory. Then, the values are sent to the processing elements (PEs), for executing the operations of BNNs in parallel (i.e., XNOR, popcount, accumulation, and thresholding, etc.). The results of the computations, which are the activations, are then written back to the on-chip FeFET memory in order to be later used in the computations for the subsequent layer.

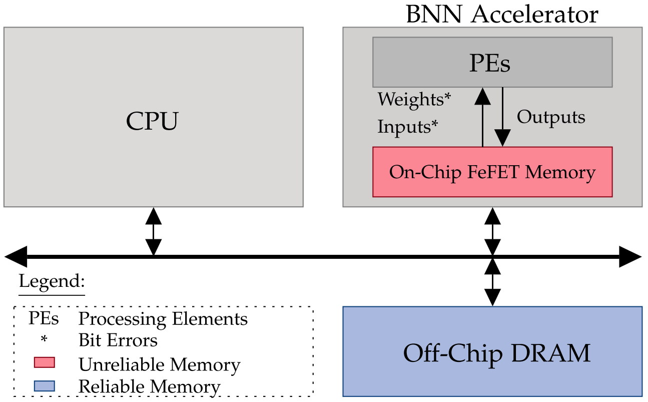


Fig. 6. System model with unreliable on-chip FeFET memory and reliable off-chip DRAM.

Data and instructions for other operations, which are related to the control of the inference (e.g., from the operating system to provide a run-time environment to initiate the inference) are not stored in the FeFET memory, but are stored in a reliable memory (e.g., off-chip DRAM).

Please note that our methods of training the BNN in the presence of bit-errors to obtain a more resilient BNN that exhibit less accuracy loss during inference are not limited to only having unreliable on-chip memory. Our methods are general and can be analogously applied despite the actual origin of the underlying bit error (e.g., having unreliable off-chip memory for storing BNN parameters of weights and inputs instead of the unreliable on-chip memory).

Please also note that NVM using ferroelectric transistors is an emerging memory, and commercial processors that employ such technology are not yet publicly available. Hence, we do not use real FeFET memory for running the experiments. The way we model the usage of FeFET is by applying the corresponding bit error model during training and inference on conventional servers with GPUs. In this work, our focus is on the effect of FeFET bit errors on BNN accuracy, when on-chip FeFET memory would be used for storing the weights, inputs, and activations. For evaluating the bit error tolerance of our built BNNs with respect to the FeFET-induced errors, the application of the bit error model is sufficient and the real FeFET memory does not need to be used.

3.3 Problem Definition

In this work we use FeFET memory as the NVM for executing BNNs. Specifically two *bit error tolerance (BET)* problems are studied in this paper:

- *BET Training Problem*: Given the FeFET temperature bit error model described in Section 2 and a set of labelled input data, the objective is to train a BNN for high accuracy. *The inference of the derived BNN does not have to be executed with any bit error countermeasures at run time.*
- *BET during Inference Problem*: Given the FeFET temperature bit error model described in Section 2 and a BNN, the objective is to execute the given BNN with bit error countermeasures to reduce the accuracy degradation of the BNN during runtime. *The given BNN does not have to be trained with bit flip injection.*

TABLE 1
The Regular BNN Execution With Many Buffer Writes to Memory and the Less-Buffer-Writes (LBW) Execution

Regular execution	In → C → buffer writes → buffer reads → MP → buffer writes → buffer reads → BN → buffer writes
LBW execution	In → 4C → MP → BN → buffer writes

Another layer configuration that we use is C → BN, in which case the thresholding of the BN is applied directly to the C result.

We note that solutions to the above two problems can be combined to yield better bit error tolerance.

4 BNN Execution With Less Buffer Writes

Before we address the two problems in Section 3.3, we first present the less-buffer-writes (LBW) BNN execution, in which the BNNs are executed in a way such that less layers are prone to bit errors compared to regular BNNs.

The FeFET bit errors can only have an effect on BNN accuracy when values are read from FeFET memory. For this reason, the buffering of data (which implies writes followed by *reads*) in FeFET memory should be avoided whenever possible. In the regular BNN execution, however, values are buffered multiple times, as shown in Table 1, which leads to many (avoidable) reads from FeFET memory. The values that are buffered to memory during execution are the intermediate results, i.e., the outputs of the convolution (C), maxpool (MP), and batch norm (BN) layer.

In order to minimize the buffering to FeFET memory, we use a less-buffer-writes (LBW) BNN execution. The LBW execution aims to reduce the reads from FeFET memory so that less values are affected by bit errors. When using the LBW execution, only the BNN parameters, inputs, and outputs of the BN layer (activations) suffer from bit errors.

In the LBW execution, we compute the C and MP in such a way that they are applied without buffer writes to memory. The LBW execution begins with the C layer. In the first iteration of the the LBW execution, the C layer computes the convolution results of the first four values $(0,0), (0,1), (1,0), (1,1)$ in the input. By MP, the maximum of the four values is then computed and thresholded with the BN layer to produce a binary output. After the threshold has been applied, the result has to be buffered in memory, so that the next C layer of the BNN can compute with it. In the next iteration of the LBW execution, the next four values in line $((0,2), (0,3), (1,2), (1,3))$ are processed the same way as in the first iteration. These iterations are repeated until the input to the C layer is fully processed. Due to the processing sequence of the LBW execution, only a buffer of two values is needed between the C → MP → BN computation. One value is needed for holding the current maximum and the second for the current MP result. For the case of C → BN layer compositions, the output of the convolutions do not need to be buffered because the thresholding can directly be applied to the output of the convolution.

If the regular execution is used instead of LBW, one would have to consider bit flips in the results of the C and MP as well, which also are signed integer values. For example, bit errors with any of the FeFET bit error rates in the outputs of the C layers (16 bit unsigned integers) before a

MP layer cannot be tolerated. The predictions of the BNNs become useless in this case. Bit flips in signed integers can naturally change the value in a much larger extent compared to binary values.

The number of executed operations in the LBW execution is equal to the number of executed operations in the regular way of execution. The operations of the LBW are simply executed in a different sequence and thus data from the memory is retrieved in a different sequence as well. The implications of the LBW execution on inference efficiency have to be investigated case-by-case for different inference systems during system design.

In this work, we do not assume any specific inference system. FeFET could e.g., be used as on-chip memory for BNN data while processing elements in an acceleratorr execute the BNN operations. In these cases, the operations of LBW BNNs are executed in ALUs and the values for accumulation and maxpool operations are stored in registers. We assume that once the values are in these components, they are not affected by bit errors anymore.

Since the correctness of the BNN outputs and the batch normalization thresholds (see [34]) are indispensable for BNN accuracy, we assume that this small fraction of values is protected by software or hardware measures such as Error-correcting code (ECC) or correcting implausible values with memory controller support (see [33]).

5 Methods for Achieving Bit Error Tolerance Against FeFET Bit Errors

In this section we present two different methods against the FeFET bit errors. We first describe how we take the asymmetry into account in bit flip training in Section 5.1, targeting the BET training problem posed in Section 3.3. Then, in Section 5.2, we present the novel bit error rate assignment algorithm (BERA) which exploits the asymmetry of the FeFET bit error model in a layer-wise manner with the goal of minimizing accuracy drop. This targets the BET during inference problem.

5.1 Bit Flip Injection During Training

In order to achieve high accuracy, we train the BNNs by minimizing the cross entropy loss, as described in Section 3. To also train BNNs for bit error tolerance against the FeFET bit errors, we use bit flip injection during training, as proposed in [4].

However, simply training for bit error tolerance without taking the asymmetry into account can lead to unacceptable accuracy drop of up to 10 percent, even when the well-known bit flip training method is used. Additional steps to the existing methods need to be taken. Specifically, the asymmetry needs to be taken into account evaluating all bit error rate settings $(p_{01}, p_{10}) \in \{(2.198, 1.090), (1.090, 2.198), (2.098, 0.190), (0.190, 2.098)\}$ by injecting bit flips with (p_{01}, p_{10}) into all the BNN data that is prone to bit errors. By doing this we can find out which setting leads to the least accuracy drop under temperature dependent bit errors.

5.2 Bit Error Rate Assignment Algorithm (BERA)

Instead of only evaluating the configurations in which all layers of the BNN are configured with the same bit error rates, we also aim for a more fine-grained method. In this work, we present the novel bit error rate assignment algorithm (BERA), which exploits the asymmetry of the FeFET temperature bit error model in a *layer-wise* manner. The goal of BERA is to reduce the effect of the bit errors by finding the layer-wise bit error rate configurations which maximize accuracy, without bit flip training.

BERA operates in two steps. In the first step (Algorithm 1), the accuracy drop of the entire NN is estimated by measuring the accuracy after injecting bit errors into one layer of the network (with the training set). Because this operation is non-deterministic we repeat it a couple of times and take the average in the end. In the second step, the setting with the lowest accuracy drop is chosen and assigned to the layer.

In Algorithm 1, we first set $bers$, the bit error rates, in Line 1. We then initialize an array for all layers that suffer from bit errors in Line 2, with a subarray for every bit error rate configuration, since we estimate the accuracy drop of every configuration. The value $reps$ is the number of repetitions through the entire training data set for each bit error rate setting. With this parameter, the precision of the accuracy drop estimation can be tuned. We measure the accuracy without errors on the training set in Line 4. The accuracy drop is estimated for every bit error rate setting in every layer individually in the loop. Finally, the results of the estimation are stored in an array called $adpl$ which is normalized with the number of repetitions after Line 12. Then, from the $adpl$ array, the setting with the lowest accuracy drop per layer is chosen and assigned to the layer at hand. We call this assignment the Greedy-A assignment.

Algorithm 1. Accuracy Drop Per Layer Estimation

```
    Input: $model, (X_{train}, y_{train})$
    Output: adpl
    //Initialize bit error rates
 1: $bers = \{c_1, \ldots, c_S\}$
    // Accuracy drop per layer (adpl)
 2: $adpl = \{\{ad_{1,1}, \ldots, ad_{1,S}\}, \ldots, \{ad_{L,1}, \ldots, ad_{L,S}\}\}$
    // Number of repetitions for a bit error rate
       setting and layer
 3: $reps = R$
    // Measure accuracy of model
 4: $acc_v = accuracy(model(X_{train}), y_{train})$
 5: for each layer $l \in \{0, \ldots, L\}$ do
 6:   for each $c_i$ in $bers$ do
 7:     set bit error rate tuple $c_i$ only for layer $l$
 8:     for each $r$ in $r \in \{0, \ldots, reps\}$ do
 9:       $acc_{l,c_i,r} = accuracy(model(X_{train}), y_{train})$
10:       $adpl[l][c_i] = adpl[l][c_i] + acc_{l,c_i,r}$
11:     reset bit error rates in $l$
12: for each layer $l \in \{0, \ldots, L\}$ do
13:   for each $c_i$ in $bers$ do
14:     $adpl[l][c_i] = acc_v - \frac{adpl[l][c_i]}{R}$
```

6 BNN Evaluation Setup

In this section, we first present the framework, the BNN architectures, and the datasets for evaluating the BNNs' tolerance against FeFET bit errors. Then, we discuss the

TABLE 2
Datasets Used for Experiments

Name	# Train	# Test	# Dim	# classes
FashionMNIST	60,000	10,000	(1,28,28)	10
CIFAR10	50,000	10,000	(3,32,32)	10
SVHN	73,257	26,032	(3,32,32)	10

methods used for bit flip injection. Afterwards, we present the datasets and the models in the evaluations of this work. At the end of this section, we discuss the bit error tolerant BNN execution used in the evaluations.

6.1 Bit Error Tolerance Evaluation Setup

In order to train the BNNs to tolerate the FeFET temperature errors, we use a BNN training environment that is set up with PyTorch , which provides efficient tensor operations for NN model optimization on GPUs. PyTorch also allows Cpp-access to the tensors with custom CUDA kernels, which we developed and use for bit flip injection into the various data types that BNNs use.

For the evaluation of accuracy over bit error rate, we inject bit flips with the bit error rates presented in Section 2, with the combinations $(p_{01}, p_{10}) \in \{(2.198, 1.090), (1.090, 2.198), (2.098, 0.190), (0.190, 2.098)\}$. With the temperature steps $T_{step} \in \{0, 1, \ldots, 16\}$, we evaluate the full bit error range. This corresponds to the entire range of operating temperature considered in our FeFET analysis. We incorporate these temperature steps in the evaluation by defining $T^*_{step} = \frac{1}{16} T_{step}$ in $BER(T^*_{step}) = T^*_{step} \cdot (p_{01}, p_{10})$. In the accuracy over bit error rates plots of following sections, we annotate the x-axis with T^*_{step}.

6.2 Bit Flip Injection

For bit flip injection into the BNN data, we call CUDA kernels to operate efficiently on the tensors of BNN weights, intermediate results, and inputs during each forward pass. To inject bit flips into the binarized values and intermediate results (in $\{-1, +1\}$), the CUDA thread changes the sign of a single entry if a randomly sampled floating point value v between 0 and 1 (using the NVIDIA cuRAND library) is below the bit error rate p in decimal. We inject bit flips anew in every forward pass for the BNN parameters and intermediate results. To inject bit flips into the 8 bit input values, we use the BFITT tool [35]. The functions in BFITT access the bit level representation and inject bit flips with CUDA kernels that execute the same comparison as above but iterate through the entire length of the bit representation. We inject bit flips into the entire input data batch every time a new batch is sampled.

6.3 Datasets, BNN Models, and LBW Execution

Datasets. In the evaluations, we use three standard object recognition, which are detailed in Table 2. FashionMNIST contains grayscale images of clothes in 10 classes sold on Zalando. CIFAR10 includes coloured images of 10 common objects and animals. SVHN contains images of house numbers in 10 classes from Google Street View.

TABLE 3
BNN Architectures and Bit Error Rates and Used in This Work

Parameter	Range
Fashion CNN	In → C64 → MP2 → BN → C64 → MP2 → BN → FC2048 → BN → FC10
CIFAR10 CNN	In → C128 → BN → C128 → MP2 → BN → C256 → BN → C256 → MP2 → BN → C512 → BN → C512 → MP2 → BN → FC1024 → BN → FC10
SVHN CNN	In → C128 → BN → C128 → MP2 → BN → C256 → BN → C256 → MP2 → BN → C512 → MP2 → BN → C512 → MP2 → BN → FC1024 → BN → FC10

BNN Architectures. For every dataset, we use a different BNN architecture. The description of the BNN architectures used in this work are collected in Table 3. To achieve reasonable accuracy, we use larger models for CIFAR10 and SVHN. For Fashion, a smaller BNN architecture is sufficient. The BNN architectures have similar basic building blocks (described in Section 3.1)) as commonly used NN models, such as VGG-16 and MobileNet, but without special layers such as foe example 1 × 1 convolutions.

LBW Execution of BNNs. In the experiments we use LBW BNNs, which we proposed in Section 4. Here, we conduct evaluations for the sake of demonstrating that our assumption of using the LBW (Less Buffer Writes) execution of BNNs is reasonable and, additionally, to quantify the impact of such an assumption on system performance. The LBW execution of BNNs buffers less intermediate results to the memory during execution than the usual way of execution. Our goal is to demonstrate that the LBW execution of BNNs does not lead to significantly higher execution times compared to the regular way of execution. We use this assumption (LBW execution of BNNs) in the error tolerance evaluations, to justify bit flip injection in the weights, input images, and activations (instead of injecting bit flips in all intermediate results, which are the outputs of every layer). Here, we evaluate the execution time of LBW BNNs and compare it to regular ones using commonly available CPUs for the BNN operations. As experiment platforms for the execution time measurements of machine learning models we use the same setup as the work in [36]. Furthermore, we generate C++ code from PyTorch models with the same framework as the study in [37]. For Intel we used a Intel i7-8550U CPU with 1.80 GHz and 16 GB RAM. For ARM we used an ARM Cortex-A53 with 1.4 GHz and 1 GB RAM (RaspberryPi 3B+). For PPC we used a QorIQ T4240 PowerPC CPU with 1.67 GHz and 6 GB RAM. To summarize the results in Table 4, the execution times for FASHION and CIFAR only differ by a large factor for PPC, and for the other settings the execution times do not differ by a large margin.

7 Experiments for FeFET Temperature Bit Error Tolerance

In Section 7.1 we first assess the impact of FeFET bit errors on BNN accuracy without any countermeasures. In the next step, to protect the BNNs, we apply the well-known bit flip

TABLE 4
Average Execution Times Evaluation for the Regular and LBW BNNs on Different Platforms and Datasets

Platform	BNN-type	FASHION (ms/el.)	CIFAR10 (ms/el.)
Intel	Regular	1.09	26.47
	LBW	1.05	32.15
ARM	Regular	12.28	305.09
	LBW	11.03	312.79
PPC	Regular	4.55	210.03
	LBW	12.45	306.30

The values are in ms per one BNN evaluation. Each BNN was evaluated 10^4 *times as compiled C++ code.*

training while taking the asymmetry of the FeFET bit error model into account (addressing the BET Training Problem). In Section 5.2 we evaluate the novel bit error rate assignment algorithm that exploits the asymmetry by operating in a layer-wise manner to minimize the impact of the FeFET bit errors (addressing the BET during inference problem).

7.1 The Impact of FeFET Bit Errors on BNN Accuracy

We use the configurations shown in Table 3 for the three datasets FASHION, CIFAR, and SVHN. We run the Adam optimizer for 50 epochs for FASHION and SVHN and for 200 epochs for CIFAR. We use a batch size of 256 and an initial learning rate of 10^{-3} for all cases. To stabilize training we decrease the learning rate every epoch for FASHION, after every 50 epoch for CIFAR, and after every 25th epoch for SVHN, by 50 percent in all cases. All training and testing experiments are repeated 10 times. We inject bit flips with the bit error rates configuration $(p_{01}, p_{10}) \in \{(2.198, 1.090), (1.090, 2.198), (2.098, 0.190), (0.190, 2.098)\}$ in the forward pass. We plot the accuracy over bit error rate for all BNNs we tested in Fig. 7.

Impact on BNNs With no Countermeasures. For BNNs trained without errors (left column) the impact of the temperature bit errors can be substantial if no bit error training is used and when no attention is paid to the asymmetry of the bit error rates. We find accuracy degradation of over 25 percent for FASHION, over 30 percent for CIFAR, and over 7 percent for SVHN at the highest operating temperature $T^*_{step} = 1$.

Bit Flip Injection During Training. When training with bit flip injection (right column of Fig. 7), we achieve bit error tolerance for the entire range of operating temperature. The asymmetry of the bit error model, however, plays a key role in these experiments. The differences among the highest and the lowest FASHION curves is below 0.2 percent for $T^*_{step} = 1$. For $T^*_{step} = 1$ this difference is 0.85 percent. For CIFAR, these gap are larger. First, for $T^*_{step} = 0$, the difference between the plot with the highest and the lowest accuracy is 1.75 percent. For $T^*_{step} = 1$, this difference is 9.6 percent. Furthermore, when comparing the CIFAR plots without bit flip training (left column) with the bit flip trained plots at $T^*_{step} = 0$, the highest plots differ by 1.1 percent. In this case, the bit flip training drops the accuracy by more than one percent. For the other datasets, this accuracy trade-off amounts to merely around 0.2 percent. For SVHN, the difference among the plots at $T^*_{step} = 0$ is below 0.2 percent and for $T^*_{step} = 1$ it is 1.4 percent when comparing the highest accuracy plot with the lowest. In all three datasets, the setting $(2.098, 0.190)$ dominates among the four bit error rate settings.

In summary, our experiments show that bit flip training is necessary for avoiding accuracy drop to unacceptable levels. The asymmetry of the bit error model also cannot be ignored. The accuracy drop at the highest operating temperature can still amount to around 10 percent in some cases.

7.2 Bit Error Rate Assignment Algorithm (BERA)

We present the output of Algorithm 1 with $reps = 10$ in Fig. 8 and plot the accuracy drop for the layers affected by bit errors. In general, we observe that bit errors in the first few layers have the largest impact on BNN accuracy. In the input image and the parameters of the first layer, the bit errors have the most significant accuracy drop. To give an intuition for the impact of the bit errors, consider that the probability for no errors in one value is $(1 - \frac{p}{100})^8$, where p is the bit error rate in percentage points and the exponent is 8 because the input values have 8 bits. In the symmetric case $p = 2\%$, for one or more bit flips in one value the probability is approximately 15 percent. One bit flip in an unsigned 8 bit value can have a large impact (depending on the position of the flip) and the flip can have a large impact on the accuracy. On the other hand, the weights of the first layer also have a large accuracy drop since the number of weights in this layer is the smallest. In the first convolutional layer, our NNs that use coloured images have $3 \times 3 \times 3$ parameters. In the second convolutional layer, the number of parameters is $3 \times 3 \times 128$, which may be more robust.

We present the result for BERA in the accuracy over bit error rate plots in Fig. 7 as Greedy-A (orange plots). In the left column, we show the plots for only using BERA to protect the BNNs and without bit flips during training. The Greedy-A plot is able to protect the BNN from bit errors to a similar extent as using bit flip training. The difference at $T^*_{step} = 1$ is 0.66 percent when compared to BNNs retrained with Greedy-A. For the other two datasets, the Greedy-A assignment can only reach the same accuracy as $(2.098, 0.190)$, since that setting dominates for every layer.

We also evaluate the combination of BERA and bit flip training. In the right column we show the plots (in orange) for retraining with the Greedy-A setting after training without bit flips. We retrained FASHION BNNs for 10 epochs, CIFAR for 200, and SVHN for 50. For FASHION, the Greedy-A assignment improves accuracy by 0.3 percent compared to the other settings. This eliminates the trade-off in accuracy when injecting bit flips during training.

In conclusion, for asymmetric bit error rate models, in some cases BERA can be used without bit flip training for protecting BNNs from accuracy drop. It however can only exploit the asymmetry if one bit error rate configuration does not dominate in all layers.

7.3 Discussion of Experiment Results

Our methods in general are fully applicable to other datasets like CIFAR 100 and ImageNet. However, examining such more complex datasets inevitably necessitates employing extremely powerful servers during training and long

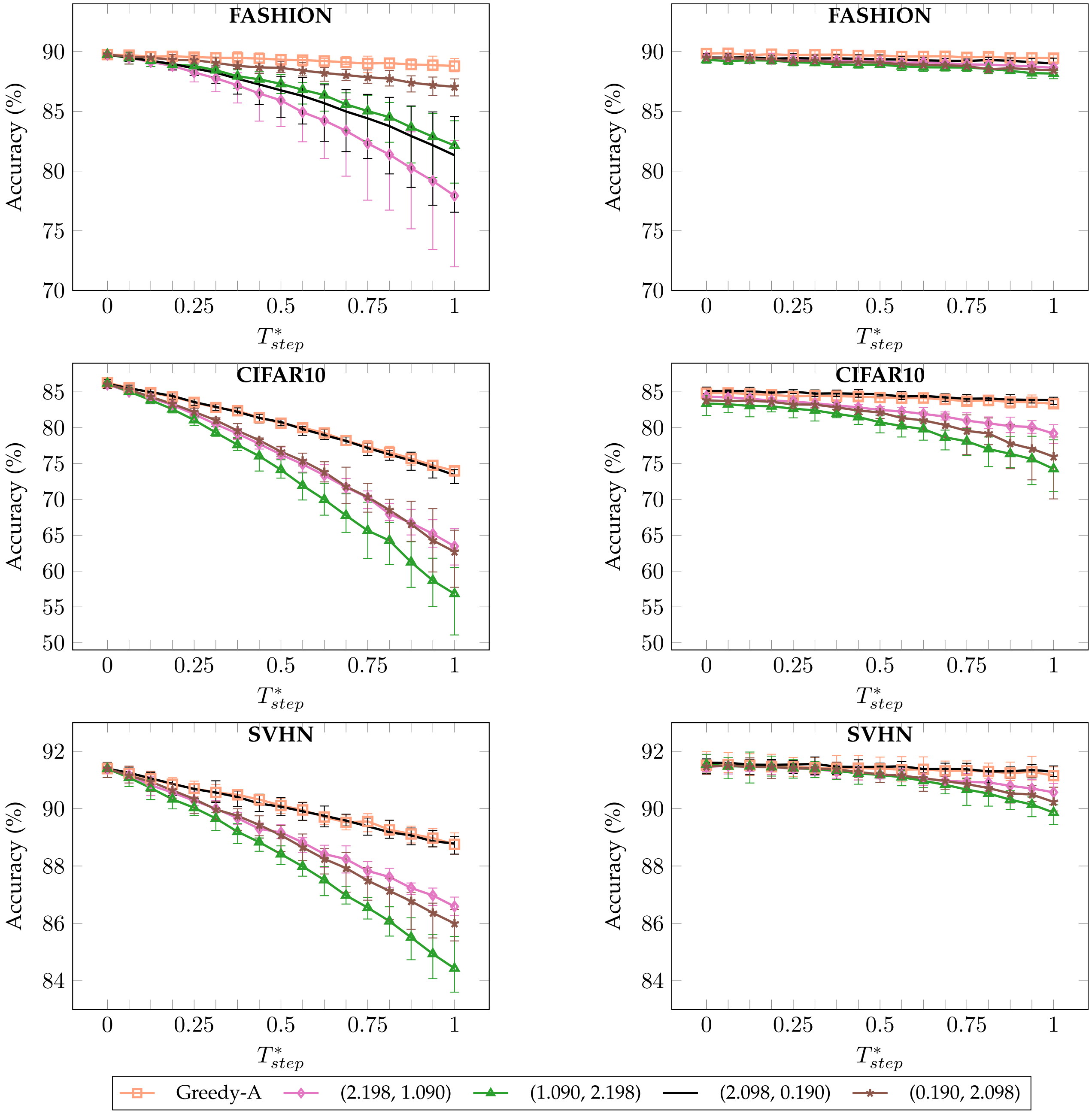


Fig. 7. Accuracy over temperature-dependent error rates. Left column: No bit flips during training, and in the orange plot (Greedy-A) the result for applying BERA after training. Right column: Bit flip training with all bit error rate configurations and in orange (Greedy-A) the retraining with BERA. $T^*_{step} = \frac{1}{16}T_{step}$ and $T_{step} \in \{0, 1, \ldots, 16\}$. $BER = T^*_{step} \cdot (p_{01}, p_{10})$ yields the temperature dependent bit error rate setting. Greedy-A is the accuracy optimal assignment acquired after executing BERA. In the other four settings, every layer of the BNN is configured with the same bit error rate. E.g., when we write $(2.198, 1.090)$, then every layer of the BNN is configured with these bit error rates.

experiment runtimes. In the following, we discuss the operations that lead to a high resource demand: (1) Training BNNs needs more memory than training traditional floating-point NNs. This is because during BNN training, we need, on the one hand, to store the floating point weights (because they are used for gradient updates in the backward pass) and, on the other hand, to store the binarized weights in inference for the forward pass. Hence, training BNNs comes with a larger memory requirement than training floating-point NNs. (2) For training BNNs, we need to execute a larger number of operations. Before a convolution is executed, the weights needs to be binarized, which adds additional overhead every time the weights are accessed in the forward pass. (3) For achieving bit-error tolerance, we apply the bit error model during training. This causes additional overheads, because every time we access the weights, we need to inject the bit flips, which is performed by random number samplings, comparisons, and flipping bits.

In summary, we have demonstrated our methods for relatively small datasets (e.g., CIFAR 10 and FASHION). The analysis of such relatively-small datasets is, for instance, around 3 weeks (500 hours on a machine with Intel Core i7-8700 K 3.70 Ghz, 32 GB RAM, 2x GeForce GTX 1080 8 GB). This demonstrates the indispensable need to have more powerful server machines when larger datesets (e.g., ImageNet and CIFAR 100) are targeted.

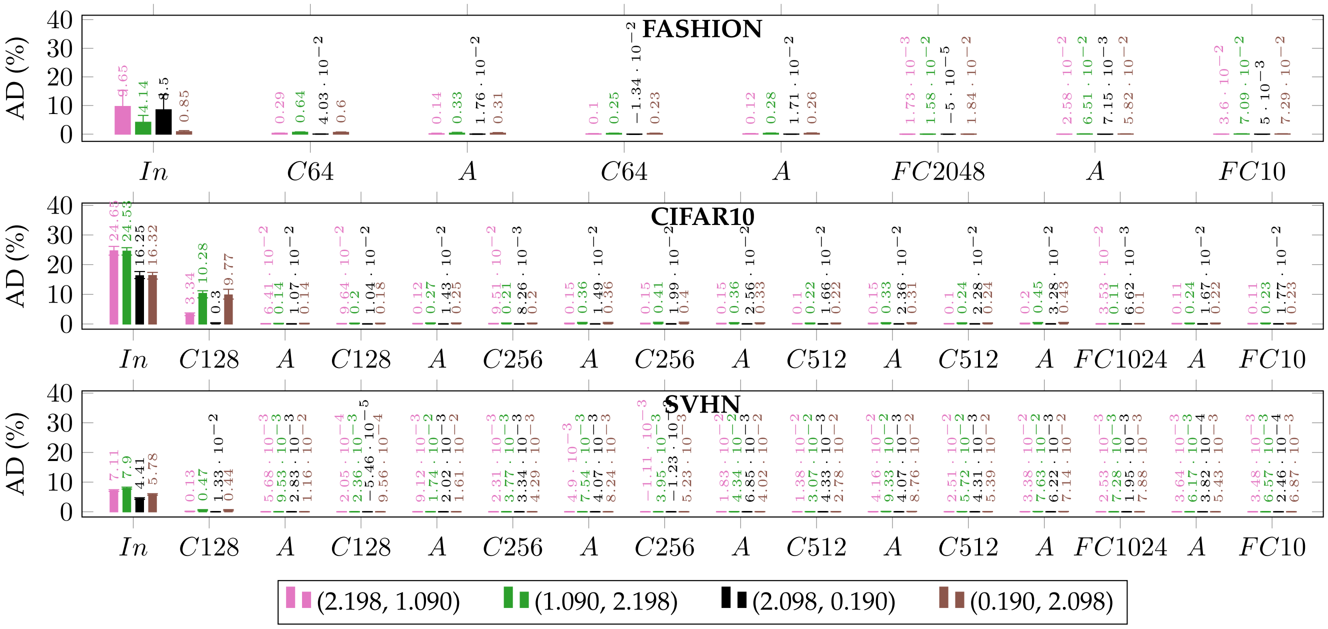


Fig. 8. Accuracy drop (AD) per layer. All BNN layers that suffer from bit errors are shown, i.e., In: Input, C: Convolution, A: Activation, FC: Fully Connected. To measure AD per layer, bit flips are injected only into the layer under analysis and then the accuracy is evaluated. The bar plots show the impact of bit flips on the layers individually.

For evaluating the overhead of binarization, bit flip injection, and the bit error rate assignment algorithm (BERA), we measured our experiment run times on the machine we used for this work (see above). The results are presented in Table 5. We observe that the binarization-aware training takes significantly more time than training floating point NNs. Injecting bit flips into the BNNs also increases the run times, but not as much as binarization. Bit error tolerance evaluation takes only a few minutes. The run time of BERA can take a few hours, depending on the size of the BNNs.

8 Related Work

In this section we review studies that mainly exploit the reduced requirements of NNs on (A) non-volatile memory technologies, i.e., RRAM, STT-RAM or MRAM, FeFET and CCT, and (B) the volatile memory technologies SRAM and DRAM.

In our work, we focus on the modeling of temperature effects in FeFET memory and evaluating the impact of the errors on BNN accuracy. We are the first to evaluate impact of temperature in FeFET memory. The related studies do not deal with temperature issues, but with errors (in general) stemming from other effects, such as voltage scaling or tuning of timing parameters. We present these related studies in the following.

TABLE 5
Experiment Run Times in Our Framework

Measure	FASHION	CIFAR10	SVHN
Train one epoch (floating-point NN, without BFI)	3.20 s	47.09 s	65.34 s
Train one epoch without BFI (BNN)	3.84 s	51.26 s	70.63 s
Train one epoch with BFI (BNN)	4.28 s	53.15 s	72.95 s
Bit error tolerance evaluation (16 error rates, BNN)	6.31 s	92.06 s	186.52 s
BERA (Bit error rate assignment algorithm, BNN)	9.59 min	269.18 min	306.92 min

BFI: Bit flip injection. In all entries are for BNNs except the ones specified with "floating point NN". Epochs were runs 50 times then averaged, bit error tolerance evaluations were run ten times and BERA once. Experiments were run on a machine with Intel Core i7-8700K 3.70 Ghz, 32 GB RAM, 2x GeForce GTX 1080 8 GB.

8.1 NN Inference Systems With NVM Technologies

RRAM. The closest works are about error tolerant BNNs that operate with reduced requirements on the memory in order to benefit in terms of power, performance, area, lifetime, etc. Hirtzlin *et al.* [4] propose to compute BNN operations with RRAM that features in-memory processing capabilities. They set the write energy of RRAM low and show that BNNs can tolerate the resulting errors by error tolerance training. This low energy setting also increases the RRAM cell lifetime since the low energy writes stress the cells less. The work by Yu *et al.* [15] also uses RRAM to implement on-chip BNNs. They show that under limited bit yield, BNNs can still operate with satisfying accuracy.

MRAM or STT-RAM. Another branch in the literature is about NNs on STT-RAM or MRAM. Hirtzlin *et al.* [8] propose deploying BNNs on MRAM with a low energy programming setting that causes relatively low error rates, no significant accuracy drop, but decreases write energy by a factor of two. Tzoufras *et al.* [9] also propose operating BNNs on MRAM with reduced voltage with similar results. They test a wide range of error rates and discuss the implications of BNN bit error tolerance on lifetime, performance, and density of MRAM.

FeFET. Work on FeFET by Chen *et al.* [11], Long *et al.* [12] and Zhang *et al.* [13] explore the in-memory processing capabilities of FeFET and compare it to other CMOS-based circuits. Yoon *et al.* [14] investigate the effect of FeFET device limitations on NN accuracy. However, these works do not investigate the temperature effects of the designs and do not exploit the error tolerance of NNs.

8.2 NN Inference Systems With Volatile Memories

SRAM. For NN inference systems using on-chip SRAM, the works in the literature mainly employ scaling of various device parameters. To reduce energy consumption, the SRAM voltage is scaled in [38], [39]. Yang *et al.* [40] seperately tune weight and activation values of BNNs to achieve fine-grained control over energy comsumption.

DRAM. For DRAM, the study by Koppula *et al.* [33] provide an overview over studies related to NNs that use different DRAM technologies and proposes a framework to evaluate NN accuracy for using approximate DRAM in various different settings and inference systems. Specifically, they show that DRAM parameters can be tuned such that energy and performance are optimized to achieve significant improvements, whereas the NN accuracy drop stays negligible due to the NNs' adaptations in retraining.

9 Conclusion

In this work, we first analyzed the effects of variable temperature on FeFET memory and proposed an asymmetric bit error model that exhibits the relation between temperature and bit error rates - high temperature leads to high bit error rates. We then evaluated the impact of FeFET asymmetric temperature bit errors on BNN accuracy if no countermeasures are applied and showed that the accuracy can drop to unacceptable levels. To deploy BNNs with high accuracy using FeFET memory despite the temperature effects, we proposed two countermeasures to the bit errors: (1) Bit flip training while the asymmetry into account and (2) a bit error rate assignment algorithm (BERA) which estimates accuracy drops per layer and assigns layer-wise the bit error rate configuration with the lowest accuracy drop. With these methods, the BNNs achieve bit error tolerance for the entire range of operating temperature. These results indicate that FeFET memory can be used on the low-power edge for BNNs despite the temperature-dependent bit errors.

The bit error tolerance methods proposed in our study can also be applied to other types of DNNs, such as quantized or floating point DNNs. However, we did not run bit error tolerance analyses on these DNNs yet. These extensions are not trivial, since dedicated tools need to be developed, while parameter tuning and model training is necessary, which takes considerable amounts of additional time. We leave these extensions as future work.

Another interesting subject is the study of the effects that error tolerance training has in the BNNs. For example, in [41], the effects of bit error tolerance training on the BNNs on the neuron and inter-neuron level is explored. In that study, the goal is to explain the achieved bit error tolerance with metrics, to gain understanding of the changes that bit error tolerance training causes in NNs. Advancements in this area, incorporating the findings of this work, are left as future work as well.

Acknowledgments

This work was supported in part by Deutsche Forschungsgemeinschaft (DFG) Project OneMemory 405422836, in part by the Collaborative Research Center SFB 876 "Providing Information by Resource-Constrained Analysis" under Project 124020371, subproject A1 (http://sfb876.tu-dortmund.de), and in part by the German Competence Center for Machine Learning Rhine Ruhr (ML2R, https://ml2r.de), through the German Federal Ministry for Education and Research. The author would like to thank Thomas Kaempfe from Fraunhofer IPMS for the provided device measurements, as well as Kai Ni from the Rochester Institute of Technology for the valuable discussions on ferroelectric devices.

References

[1] S. Han *et al.*, "EIE: Efficient inference engine on compressed deep neural network," *SIGARCH Comput. Archit. News*, vol. 44, no. 3, pp. 243–254, 2016.

[2] S. Kim, P. Howe, T. Moreau, A. Alaghi, L. Ceze, and V. S. Sathe, "Energy-efficient neural network acceleration in the presence of bit-level memory errors," *IEEE Trans. Circuits Syst. I: Regular Papers*, vol. 65, pp. 4285–4298, Dec. 2018.

[3] P. Chi *et al.*, "PRIME: A novel processing-in-memory architecture for neural network computation in ReRAM-based main memory," *ACM SIGARCH Comput. Archit. News*, vol. 44, no. 3, pp. 27–39, 2016.

[4] T. Hirtzlin *et al.*, "Outstanding bit error tolerance of resistive RAM-based binarized neural networks," in *Proc. Int. Conf. Artif. Intell. Circuits Syst.*, pp. 288–292, 2019.

[5] A. F. Vincent *et al.*, "Spin-transfer torque magnetic memory as a stochastic memristive synapse for neuromorphic systems," *Trans. Biomed. Circuits Syst.*, vol. 9, pp. 166–174, 2015.

[6] Y. Pan *et al.*, "A multilevel cell STT-MRAM-based computing in-memory accelerator for binary convolutional neural network," *IEEE Trans. Magn.*, vol. 54, pp. 1–5, Nov. 2018.

[7] X. Sun, X. Peng, P. Chen, R. Liu, J. Seo, and S. Yu, "Binary neural network with 16 MB RRAM macro chip for classification and online training," in *Proc. Asia South Pacific Des. Automat. Conf.*, 2018, pp. 574–579.

[8] T. Hirtzlin *et al.*, "Implementing binarized neural networks with magnetoresistive RAM without error correction," 2019, *arXiv:1908.04085*.

[9] M. Tzoufras, M. Gajek, and A. Walker, "Magnetoresistive RAM for error resilient XNOR-Nets," 2019, *arXiv:1905.10927*.

[10] M. Donato, B. Reagen, L. Pentecost, U. Gupta, D. Brooks, and G.-Y. Wei, "On-chip deep neural network storage with multi-level ENVM," in *Proc. Des. Automat. Conf.*, 2018, pp. 1–6.

[11] X. Chen, X. Yin, M. Niemier, and X. S. Hu, "Design and optimization of FeFET-based crossbars for binary convolution neural networks," in *Proc. Des., Automat. Test Eur. Conf. Exhib.*, 2018, pp. 1205–1210.

[12] Y. Long *et al.*, "A ferroelectric FET based power-efficient architecture for data-intensive computing," in *Proc. Int. Conf. Comput.-Aided Des.*, 2018, pp. 1–8.

[13] X. Zhang, X. Chen, and Y. Han, "Femat: Exploring in-memory processing in multifunctional FeFET-based memory array," in *Proc. Int. Conf. Comput. Des.*, 2019, pp. 541–549.

[14] I. Yoon, M. Jerry, S. Datta, and A. Raychowdhury, "Design space exploration of ferroelectric FET based processing-in-memory DNN accelerator," 2019, *arXiv:1908.07942*.

[15] S. Yu *et al.*, "Binary neural network with 16 MB RRAM macro chip for classification and online training," in *Proc. IEEE Int. Electron Devices Meeting*, 2016, pp. 16.2.1–16.2.4.

[16] D. Reis *et al.*, "Design and analysis of an ultra-dense, low-leakage, and fast FeFET-based random access memory array," *IEEE J. Explor. Solid-State Comput. Devices Circuits*, vol. 5, no. 2, pp. 103–112, Dec. 2019.

[17] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv , and Y. Bengio, "Binarized neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2016, pp. 4107–4115.

[18] S. Dünkel *et al.*, "A FeFET based super-low-power ultra-fast embedded NVM technology for 22nm FDSOI and beyond," in *Proc. IEEE Int. Electron Devices Meeting*, 2017, pp. 19.7.1–19.7.4.

[19] M. Trentzsch *et al.*, "A 28nm HKMG super low power embedded NVM technology based on ferroelectric FETs," in *Proc. IEEE Int. Electron Devices Meeting*, 2016, pp. 11.5.1–11.5.4.

[20] A. Gupta, K. Ni, O. Prakash, X. S. Hu, and H. Amrouch, "Temperature dependence and temperature-aware sensing in ferroelectric FET," in *Proc. IEEE Int. Rel. Phys. Symp.*, 2020, pp. 1–5.

[21] Accessed: Aug. 1, 2020. [Online]. Available: https://www.synopsys.com/silicon/tcad.html

[22] S. Natarajan, *et al.*, "A 14nm logic technology featuring 2 nd-generation FinFET, air-gapped interconnects, self-aligned double patterning and a 0.0588 μm 2 SRAM cell size," in *Proc. Int. Electron Devices Meeting*, 2014, pp. 3–7.

[23] K. Ni *et al.*, "Critical role of interlayer in Hf 0.5 Zr 0.5 o 2 ferroelectric FET nonvolatile memory performance," *IEEE Trans. Electron Devices*, vol. 65, no. 6, pp. 2461–2469, Jun. 2018.

[24] A. Gupta, K. Ni, O. Prakash, X. S. Hu, and H. Amrouch, "Temperature dependence and temperature-aware sensing in ferroelectric FET," in *Proc. IEEE Int. Reliability Phys. Symp.*, 2020, pp. 1–5.

[25] K. Ni, A. Gupta, O. Prakash, S. Thomann, X. S. Hu, and H. Amrouch, "Impact of extrinsic variation sources on the device-to-device variation in ferroelectric FET," in *Proc. IEEE Int. Rel. Phys. Symp.*, 2020, pp. 1–5.

[26] N. Zagni, P. Pavan, and M. A. Alam, "A memory window expression to evaluate the endurance of ferroelectric FETs," *Appl. Phys. Lett.*, vol. 117, no. 15, 2020, Art. no. 152901.

[27] P. R. Genssler, V. Van Santen , J. Henkel, and H. Amrouch, "On the reliability of FeFET on-chip memory," *IEEE Trans. Comput.*, to be published, doi: 10.1109/TC.2021.3066899.

[28] Y. Higashi *et al.*, "Impact of charge trapping on imprint and its recovery in HFO 2 based FeFET," in *Proc. IEEE Int. Electron Devices Meeting*, 2019, pp. 15–6, IEEE.

[29] V. Sessi *et al.*, "A silicon nanowire ferroelectric field-effect transistor," *Adv. Electron Materials*, vol. 6, no. 4, 2020, Art. no. 1901244.

[30] K. Ni *et al.*, "Critical role of interlayer in Hf 0.5 Zr 0.5 o 2 ferroelectric FET nonvolatile memory performance," *IEEE Trans. Electron Devices*, vol. 65, no. 6, pp. 2461–2469, Jun. 2018.

[31] T. Ali, *et al.*, "A study on the temperature-dependent operation of fluorite-structure-based ferroelectric HfO 2 memory FEFET: Pyroelectricity and reliability," *IEEE Trans. Electron Devices*, vol. 67, no. 7, pp. 2981–2987, Jul. 2020.

[32] M. Courbariaux and Y. Bengio, "Binarynet: Training deep neural networks with weights and activations constrained to +1 or -1," 2016, *arXiv:1602.02830*.

[33] S. Koppula *et al.*, "EDEN: Enabling energy-efficient, high-performance deep neural network inference using approximate dram," in *Proc. Int. Symp. Microarchit.*, 2019, pp. 166–181.

[34] E. Sari, M. Belbahri, and V. P. Nia, "How does batch normalization help binary training?," 2019, *arXiv:1909.09139*.

[35] BFITT. Accessed: Aug. 1, 2020. [Online]. Available: https://github.com/myay/bfitt

[36] S. Buschjager, K. Chen, J. Chen, and K. Morik, "Realization of random forest for real-time evaluation through tree framing," in *Proc. IEEE Int. Conf. Data Mining*, 2018, pp. 19–28.

[37] S. Buschjäger, K. M. Jens Bu ß, and L. Pfahler, "On-site gamma-hadron separation with deep learning on FPGAs," in *Proc. Joint Eur. Conf. Mach. Learn. Knowl. Discov. Databases*, 2021, pp. 478–493.

[38] X. Sun *et al.*, "Low-VDD operation of SRAM synaptic array for implementing ternary neural network," *IEEE Trans. Very Large Scale Integr. Syst.*, vol. 25, no. 10, pp. 2962–2965, Oct. 2017.

[39] S. Henwood, F. Leduc-Primeau , and Y. Savaria, "Layerwise noise maximisation to train low-energy deep neural networks," 2019, *arXiv:1912.10764*.

[40] L. Yang, D. Bankman, B. Moons, M. Verhelst, and B. Murmann, "Bit error tolerance of a CIFAR-10 binarized convolutional neural network processor," in *Proc. Int. Symp. Circuits Syst.*, 2018, pp. 1–5.

[41] S. Buschj äger *et al.*, "Bit error tolerance metrics for binarized neural networks," 2021, *arXiv:2102.01344*.

Mikail Yayla is currently working toward the PhD degree with the Design Automation for Embedded Systems Group, TU Dortmund University, Germany. His current research interests include reliable machine learning on unreliable emerging devices. He was the recipient of one Best Paper Nomination at DATE'21

Sebastian Buschjäger (Student Member, IEEE) is currently working toward the PhD degree with the Artificial Intelligence Group, TU Dortmund University, Germany. His research interests include resource efficient machine learning algorithms and specialized hardware for machine learning. He focuses on ensemble methods and randomized algorithms combined with specialized hardware such as FPGAs.

Aniket Gupta received the BTech degree from the Department of Electronics Engineering, NIT, Uttarakhand, India, in 2020. From June 2019 to December 2019, he was a research intern and chair of embedded systems with the Department of Computer Science, Karlsruhe Institute of Technology, Germany. He was the recipient of Deutscher Akademischer Austauschdienst (DAAD: German Academic Exchange Service) scholarship in 2019, the Mitacs Globalink scholarship in 2019, and the Best Startup Paper Award at VDAT international conference.

Jian-Jia Chen (Senior Member, IEEE) received the BS degree from the Department of Chemistry, National Taiwan University, in 2001, and the PhD degree from the Department of Computer Science and Information Engineering, National Taiwan University, Taiwan, in 2006. He is currently a professor with the Department of Informatics, TU Dortmund University, Germany. From May 2010 to March 2014, he was a junior professor with the Department of Informatics, Karlsruhe Institute of Technology, Germany. Between January 2008 and April 2010, he was a postdoc researcher with ETH Zurich, Switzerland. His research interests include real-time systems, embedded systems, energy-efficient scheduling, power-aware designs, temperature-aware scheduling, and distributed computing. He was the recipient of European Research Council (ERC) Consolidator Award in 2019 and more than ten best paper awards and outstanding paper awards and has involved in Technical Committees in many international conferences.

Jörg Henkel (Fellow, IEEE) received the diploma and PhD (summa cum laude) degrees in electrical engineering from the Technical University of Braunschweig, Braunschweig, Germany. He is currently the chair professor of embedded systems with Karlsruhe Institute of Technology. His research interests include co-design for embedded hardware or software systems with respect to power, thermal, and reliability aspects. He was the editor-in-chief of *ACM Transactions on Embedded Computing Systems* and is currently the editor-in-chief of *IEEE Design and Test Magazine*. He has led several conferences as a general chair, including ICCAD, ESWeek and is currently a Steering Committee chair or member for leading conferences and journals for embedded and cyber-physical systems. He coordinates the DFG program SPP 1500 Dependable Embedded Systems and is a site coordinator of the DFG TR89 collaborative research center on Invasive Computing.

Katharina Morik (Member, IEEE) received the habilitation degree with the TU Berlin University in 1988. She established the chair of artificial intelligence with the TU Dortmund University in 1991. In 2011, she acquired the Collaborative Research Center SFB 876 providing information by resource-constrained data analysis, of which she is the spokesperson. It consists of 12 projects and a graduate school of more than 50 PhD students. She is currently a spokesperson of the Competence Center for Machine Learning Rhein Ruhr and coordinator of the six German AI competence centers. She is an editor of the international journal the *Data Mining and Knowledge Discovery* and was a founding member, the program chair, several times the vice chair of the IEEE ICDM, and the program chair of the European Conference on Machine Learning . Since 2015, she has been a member of the Academy of Technical Sciences and of the North Rhine-Westphalian Academy of Sciences and Arts since 2016. She was the recipient of the Fellow of German Society of Computer Science GI e.V. in 2019.

Kuan-Hsun Chen (Member, IEEE) received the PhD degree from the Department of Informatics, TU Dortmund University, Germany, in 2019. Since August 2019, he has been a postdoc researcher with the Department of Informatics, TU Dortmund University, Germany. His research interests include dependable computing, cyber-physical systems, resource-constrained data analysis, and real-time operating systems. He was the recipient of the Best Student Paper Award at RTCSA 2018 and a Dissertation Award at TU Dortmund University in 2019.

Hussam Amrouch (Member, IEEE) received the PhD degree with distinction (Summa cum laude) from KIT in 2015. He is currently a junior professor for the Semiconductor Test and Reliability (STAR) chair at the computer science, electrical engineering faculty at the University of Stuttgart as well as a research group leader at the Karlsruhe Institute of Technology (KIT), Germany. He has authored or coauthored more than 135 papers, including 50 in journals in multidisciplinary research areas across the entire computing stack, starting from semiconductor physics to circuit design all the way up to computer-aided design and computer architecture. His research interests mainly include design for reliability and testing from device physics to systems, machine learning, security, approximate computing, and emerging technologies with a special focus on ferroelectric devices. He is currently an associate editor for the *Integration* and *Very Large Scale Integration* Journals. He was with the Technical Program Committees of many major EDA conferences such as DAC, ASP-DAC, and ICCAD and a reviewer in many top journals like the *IEEE Transactions on Electron Devices*, *IEEE Transactions on Circuits and Systems*, *IEEE Transactions on Very Large Scale Integration Systems*, *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, and *IEEE Transactions on Computers*. He was the recipient of seven HiPEAC paper awards and three best paper nominations at top EDA conferences: DAC'16, DAC'17, and DATE'17 for his work on reliability.