NeuADC: Neural Network-Inspired RRAM-Based Synthesizable Analog-to-Digital Conversion with Reconfigurable Quantization Support

Weidong Cao, Xin He, Ayan Chakrabarti and Xuan Zhang
Washington University in St.Louis

***Abstract*— Traditional analog-to-digital converters (ADCs) employ dedicated analog and mixed-signal (AMS) circuits and require time-consuming manual design process. They also exhibit limited reconfigurability and are unable to support diverse quantization schemes using the same circuitry. In this paper, we propose NeuADC — an automated design approach to synthesizing an analog-to-digital (A/D) interface that can approximate the desired quantization function using a neural network (NN) with a single hidden layer. Our design leverages the mixed-signal resistive random-access memory (RRAM) crossbar architecture in a novel dual-path configuration to realize basic NN operations at the circuit level and exploits smooth bit-encoding scheme to improve the training accuracy. Results obtained from SPICE simulations based on 130nm technology suggest that not only can NeuADC deliver promising performance compared to the state-of-art ADC designs across comprehensive design metrics, but also it can intrinsically support multiple reconfigurable quantization schemes using the same hardware substrate, paving the ways for future adaptable application-driven signal conversion. The robustness of NeuADC's quantization quality under moderate RRAM resistance precision is also evaluated using SPICE simulations.**

I. Introduction

Advanced technology scaling with lower supply voltage results in reduced intrinsic device gain, decreased signal swing, and aggravated device mismatch, making it more difficult to design scalable analog and mixed-signal (AMS) circuits [1], [2], [14]. As the quintessential example of an AMS circuit, the omnipresent analog-to-digital converter (ADC) faces the same design challenges when migrating to smaller highly-scaled technology nodes [2]. Traditional ADC circuits often require significant manual design iterations and re-spins to meet the desired performance specifications for a new process. Previous research has explored synthesizable and scalable ADC topologies to automate this expensive and time-consuming design process [2], [3]. One example is stochastic flash ADCs that make use of the intrinsic input offsets of minimum-sized digital comparators [2]. However, stochastic ADCs require a large number of hardware resources ($\sim$3840 comparators) and work only at relatively modest sampling rate ($\sim 8MS/s$) and resolution ($\sim$5.3-bit). Another example is synthesis-friendly time-domain delta-sigma ADCs [3], but they still require manual modifications of a standard cell and designer knowledge for floor-planning.

In addition to the design automation challenge, ADCs also face new demands from many emerging applications [4]–[6]. For example, in-memory computation in the analog domain using non-volatile memory (NVM) arrays has been proposed to accelerate neural network (NN) inference and training for deep learning applications [5], [6], where ADCs play a critical role at the A/D interface, yet little work specifically addresses NVM-compatible scalable ADC designs. The ability to support a flexible quantization scheme is another desirable ADC property that can benefit a variety of sensor frontend interfaces [4]. For instance, image sensors require logarithmic or square-root quantizations to realize a uniform distribution of exposure [10], instead of the uniform linear quantization implemented in standard ADCs. A reconfigurable ADC that can support different quantization schemes obviates the need to perform Gamma correction later in the digital domain, saving power and improving the energy efficiency of the backend image signal processor (ISP).

In this paper, we propose NeuADC — a novel design approach for synthesizable ADCs that addresses the aforementioned imminent challenges facing the traditional ADC design paradigm. Inspired by NN, NeuADC is founded on a deep learning framework and implemented using mixed-signal RRAM crossbar architecture. We consider RRAM, a promising NVM technology [5], [6], [8], [13], a perfect testbed to demonstrate the scalable and portable features of our method. We reformulate the ADC design as a NN learning problem, where the learning objective is to approximate multiple desirable quantization functions for A/D conversion. This approach allows us to take advantage of many training techniques developed for deep learning and seamlessly incorporate them into ADC design automation.

Key innovations and contributions of this paper are:

- NeuADC transforms traditional ADC design into a learning problem and enables the development of an automated design flow to synthesize ADC circuits. Our novel design approach opens new opportunities to employ learning techniques in AMS design automation.
- We propose a new dual-path RRAM crossbar architecture to facilitate the mixed-signal vector matrix multiplication (VMM) required by NeuADC in a scaling-compatible manner, along with an inverter voltage transfer characteristics (VTC) based activation function to implement the nonlinear activation function (NAF).
- We explore several hardware-oriented training techniques to account for device and circuit level characteristics and non-idealities in NeuADC design.
- We present SPICE simulation results that validate the competitive performance of our proposed NeuADC and its ability to support multiple reconfigurable quantization schemes. Robustness against RRAM resistance variation is also evaluated.

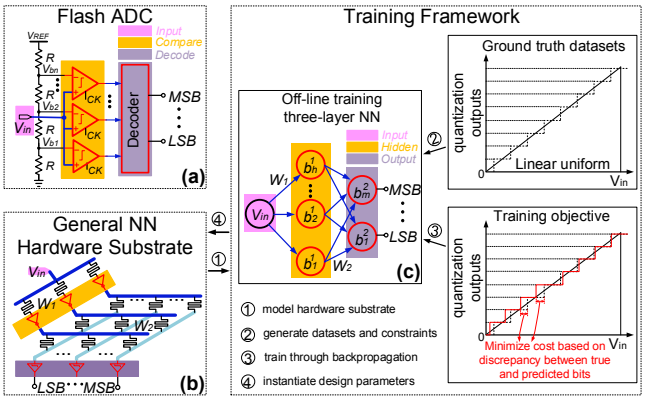

Fig. 1: Conceptual illustration of the proposed NeuADC design methodology. (a) Conventional flash ADC architecture. (b) Proposed NeuADC hardware substrate. (c) Proposed training framework that takes ideal quantization datasets as inputs during off-line training to find the optimal set of weights and derive the RRAM resistances in order to minimize the cost function and best approximate the desirable quantization function.

II. Design Methodology Overview

The cornerstone of our proposed design methodology is the universal approximation theorem [7] that proves a feed-forward NN with a single hidden layer, also known as a multi-layer perceptron (MLP), can approximate arbitrary complex functions. Coincidentally, an ideal ADC performs the mathematical transformation that quantizes the input analog value to its output bit representation. Therefore, if a NN can be trained to approximate the same, though highly nonlinear, input/output mapping of an ADC, then it is possible to construct a synthesizable ADC by implementing the trained NN in mixed-signal hardware.

Fig. 1 illustrates this basic concept and contrasts NeuADC with the conventional design method. The architecture of a conventional flash ADC is shown in Fig. 1(a). It relies on ideal comparators to obtain the desirable uniform staircase quantization function. Unfortunately, real circuits do not behave as ideal comparators, and their designs require expert knowledge, preventing automated synthesis. Our proposed NeuADC overcomes these difficulties with two integrated elements in its design methodology — a general NN hardware substrate and a hardware-oriented training framework, as illustrated in Fig. 1(b) and (c). The general NN hardware substrate performs NN operations, such as VMM and NAF, in the AMS domain. It is implemented using a dual-path RRAM crossbar architecture, whereas the off-line training framework learns the appropriate design parameters for the NN hardware substrate to approximate the desirable quantization behavior of an ADC. The overall design process can be summarized in four steps: ① the behavior of the hardware substrate is modeled as a MLP; ② the training datasets based on the desirable quantization function are fed to the optimization algorithm, along with customized objective functions and constraints to accurately reflect the hardware characteristics of the underlying circuits; ③ NN weights are iteratively trained through backpropagating the output errors; ④ the off-line-trained weights are used to instantiate the corresponding design parameters in the hardware substrate. Details of the hardware substrate design and the training framework development are presented in Sec. III and Sec. IV.

III. Hardware Substrate

Fig. 2(a) presents the overall architecture of NeuADC's general NN hardware substrate to realize a three-layer MLP. We use RRAM crossbar arrays and static CMOS inverters at each layer to perform VMM and NAF operations. The input analog signal represents the single "place holder" neuron in MLP's input layer, therefore the weight matrix dimensions are $1 \times H$ between the input and the hidden layer and $H \times M$ between the hidden and the output layer, assuming there are H and M neurons in the hidden and output layers.

A. RRAM Crossbar Array

Fig. 2(b) zooms on a RRAM crossbar array of positive path. The $1 \times H$ complementary input vector represented by input voltages $V_{i,1}$ to $V_{i,H}$ are fed to each row on the word lines (WL), and each element in the weight matrix is stored as the conductance of the RRAM device in each weight cell. The weight cell consists of one transistor and one RRAM device (1T1R) and can operate in both compute mode and program mode. In compute mode, the transistor is turned on and the RRAM crossbar performs the analog VMM computation by summing the currents on the shared bit line (BL). In program mode [15], the conductance of the RRAM is set to the desirable weight by the programming circuits (PC) and the address decoders (AD-DEC). The proposed RRAM crossbar differs from designs in previous work [5], [6] in eliminating operational amplifiers and analog inverters; thus it is more scalable and synthesis-friendly.

B. Dual-path Configuration

In order to accommodate negative weights, we propose a new dual-path configuration such that each NN layer consists of a positive path and a negative path. Each path uses a pair of complementary voltages with opposite polarity for signal representation and two RRAM crossbar sub-arrays to perform VMM, which we call the upper and the lower sub-array. We use the positive path crossbar shown in Fig. 2(b) as an example to explain the dual-path operation. Assume there are H pairs of complementary inputs:

$$V_{i,k}^{P} = V_{in,k}, \quad V_{i,k}^{N} = V_{DD} - V_{in,k} \tag{1}$$

where V_{DD} is the supply voltage, and $k = 1, 2, ..., H$. We represent the output voltages at the crossbar BL on the positive (negative) path as $V_{o,j}^{P}$ ($V_{o,j}^{N}$), and $j = 1, 2, ..., M$. Applying the Thevenin Theorem, the contribution of each pair of inputs is superimposed to obtain the output BL voltages:

$$V_{o,j}^{P} = \sum_{k=1}^{H} (W_{kj}^{PP} \cdot V_{i,k}^{P} + W_{kj}^{PN} \cdot V_{i,k}^{N}) \tag{2}$$

where $W_{kj}^{PP} = g_{kj}^{U} / \sum$, $W_{kj}^{PN} = g_{kj}^{L} / \sum$, and $\sum = \sum_{l=1}^{H} (g_{lj}^{U} + g_{lj}^{L})$. The first superscript of W_{kj}^{PP} denotes which path the weight belongs to, and the second superscript denotes which complementary input the weight acts upon. The superscript of g_{kj}^{U} denotes which sub-array the conductance belongs to. By replacing $V_{i,k}^{P}$ and $V_{i,k}^{N}$ with Eq. (1), the output

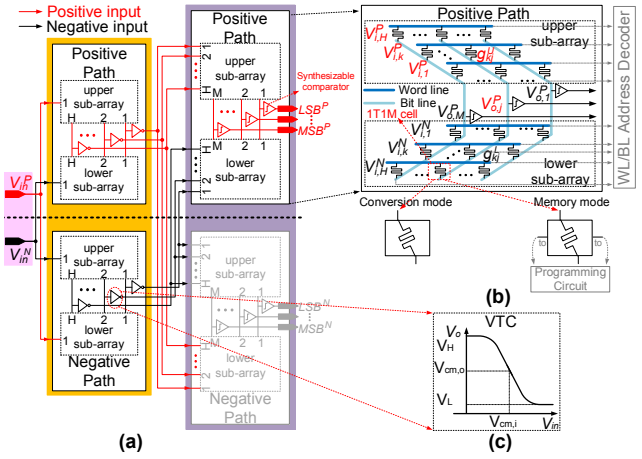

Fig. 2: NeuADC hardware implementation. (a) NeuADC Dual-path architecture; (b) Zoomed on RRAM $H \times M$ crossbar array; (c) Inverter VTC.

voltage $V^P_{o,j}$ of the positive path in Eq. (2) can be derived as $V^P_{o,j} = \sum_{k=1}^{H} W^P_{kj} \cdot V_{in,k} + V^P_{off,j}$. Here,

$$W^P_{kj} = W^{PP}_{kj} - W^{PN}_{kj} = \frac{g^U_{kj} - g^L_{kj}}{\sum} \tag{3}$$

and $V^P_{off,j} = \sum_{k=1}^{H}(W^{PN}_{kj} \cdot V_{DD})/\sum$. Thanks to the complementary voltage inputs, the effective weight W^P_{kj} is a subtraction of two conductances and thus can be negative. For the proposed conductance subtraction scheme to work, we need to generate $V^N_{o,j}$, the complementary version of $V^P_{o,j}$, by flipping the input voltage pair in the negative path.

C. VTC-based Activation Function

Modern deep learning methods have shown that many different forms of NAF can result in successfully-trained NN [11]. To maximally simplify the circuit implementation of our proposed NN hardware substrate, we leverage the intrinsic voltage transfer characteristics (VTC) curve of native CMOS logic circuits to perform NAF in the NN computation. As depicted in Fig. 2(c), VTC exhibits saturation at both ends of the input range, and can be considered as a flipped-and-shifted version of a general S-shaped curve, similar to the commonly-used sigmoid function. To provide flexibility to the training process, current-starved inverters are used as the NAF function in the hidden layer, as it allows the VTC curve to float in a range defined by V_H and V_L. The synthesizable comparators implemented with a 3-input NAND gate [2], [14] are used in the output layer to perform digitalization. Both the inverter and the 3-input NAND comparator are scalable and synthesizable, and thus they significantly reduce the design complexity and facilitate design automation.

IV. Training Framework

A. Learning Objective

The input-output relationship of the NeuADC circuit is modeled as a MLP with a single hidden layer:

$$\tilde{h} = L_1(V_{in}; \theta_1), h = \sigma_{\text{VTC}}(\tilde{h}); \tilde{b} = L_2(h; \theta_2), b = T(\tilde{b}) \tag{4}$$

Here, V_{in} is the scalar input signal and b is the output bit-vector. $\tilde{h}$ denotes voltages at the output of the crossbar array before the hidden neuron, and is modeled as a linear function L_1 of V_{in} with learnable parameters θ_1, which corresponds to the crossbar conductances. $\tilde{h}$ voltages are passed through the inverters acting as the NAF, whose input-output relationship is modeled by VTC curve $\sigma_{\text{VTC}}(\cdot)$, to yield the vector h. The linear function L_2 models the second layer of the crossbar to produce another vector $\tilde{b}$. The output bit-vector b is obtained by zero-thresholding $\tilde{b}$.

The learning objective is to find optimal θ_1, θ_2 such that for all values of V_{in} in the input range, NeuADC yields the corresponding bit-vectors b equal or close to the desired "ground-truth" vectors b_{GT}. Hence, a cost function is defined to measure the discrepancy between the predicted b and true b_{GT}. Note that b in Eq. (4) is non-differentiable, which prevents propagating gradients to θ_1, θ_2. Therefore, we define the differentiable cost $C(\tilde{b}, b_{\text{GT}})$ in terms of the un-thresholded bit-vector $\tilde{b}$. Now, given a set $\{V^t_{in}, b^t_{\text{GT}}\}_t$ of pairs of signal and bit-vector values, the training can be formally expressed as solving the following optimization problem:

$$[\theta_{1,2}] = \arg\min \sum_t C(L_2(\sigma_{\text{VTC}}(L_1(V^t_{in}, \theta_1)), \theta_2), b^t_{\text{GT}}) \tag{5}$$

B. Circuit Model for Training

Eq. (4) and (5) provide a sketch of how the training is formulated. To better reflect the hardware substrate behavior, we employ a more detailed circuit model in the training process. The first layer in our crossbar model has dual paths, each with H outputs. We denote these outputs as vectors $\tilde{p}$ and $\tilde{n}$, with $\tilde{h} = [\tilde{p}^T, \tilde{n}^T]^T$ being a $2H$ dimensional vector. Then we define the linear relationship L_1 between these outputs as

$$\tilde{p} = W_1 V_{in} + V_1, \quad \tilde{n} = V_{DD} - \tilde{p} \tag{6}$$

which is equivalent to

$$\begin{aligned} \tilde{p} &= \max(0, W_1) V_{in} + \max(0, -W_1)(V_{DD} \\ &\quad - V_{in}) + (V_1 - \max(0, -W_1) V_{DD}) \\ \tilde{n} &= \max(0, -W_1) V_{in} + \max(0, W_1)(V_{DD} \\ &\quad - V_{in}) + (V_{DD} - V_1 - \max(0, W_1) V_{DD}) \end{aligned} \tag{7}$$

Here, the learnable parameters are $\theta_1 = \{W_1, V_1\}$, where W_1 and V_1 are both H dimensional vectors. Additionally, since parameters in these models have real physical meanings as voltages and conductances, they have to abide by the following feasibility constraints on W_1 and V_1:

$$\begin{aligned} 0 &\le V_1 - \max(0, -W_1) \times V_{DD} \\ &\le V_{DD} \times (1. - \text{abs}(W_1)) \\ 0 &\le V_{DD} - V_1 - \max(0, -W_1) \times V_{DD} \\ &\le V_{DD} \times (1. - \text{abs}(W_1)) \end{aligned} \tag{8}$$

σ_{VTC} is linearly interpolated between the sampled points of the VTC using SPICE simulation to ensure both the value and the gradient exist for any input to σ_{VTC}. The L_2 function maps the inverter outputs $p = \sigma_{\text{VTC}}(\tilde{p})$ and $n = \sigma_{\text{VTC}}(\tilde{n})$ to the un-thresholded bit vector $\tilde{b}$ as $\tilde{b} = \max(0, W_2)(p - V_{cm,o}) + \max(0, -W_2)(n - V_{cm,o}) + V_2$. Here, the learnable parameters are $\theta_2 = \{W_2, V_2\}$, where W_2 is an $M \times H$ matrix and V_2 is an M-dimensional vector, with M the number of the output bits. Note that the hidden activations p and n are defined after subtracting the mid-point voltage

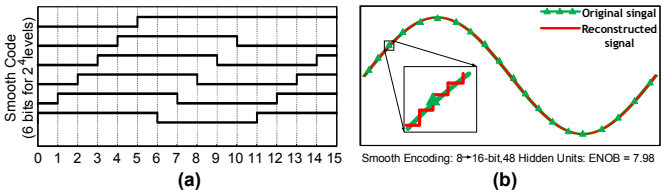

Fig. 3: Proposed smooth codes illustration. (a) Transition of different bits in our proposed smooth codes; (b) Example of reconstructed waveform from NeuADC outputs trained with smooth codes.

$V_{cm,o}$ of the inverter output range, because it leads to more stable training.

C. Bit Encoding and Decoding Scheme

Standard binary encoding is a straightforward way to define the "ground-truth" vectors b_{GT}:

$$\sum_{i=1}^{M} 2^{i-1} \cdot b_{\text{GT}i} = \text{round}\left(\frac{V_{in} - V_{min}}{V_{max} - V_{min}} \times (2^M - 1)\right) \quad (9)$$

where V_{min} and V_{max} are the minimum and maximum values of the scalar input signal. However, we find that a good NN approximation for the standard binary-encoded ADC is difficult to obtain with limited hidden neurons due to the high-frequency target function for the LSB (least significant bit), which must change signs 2^M times in the input range. Moreover, errors in the significant bits can cause large deviations in the reconstructed signal.

To address this problem, we propose training with more redundant "smooth" $A \rightarrow B$ codes that replace an A-bit binary encoding with $B > A$ bits, where each of the 2^A levels is represented with B-bit unique codewords with two properties: 1) only one bit changes between subsequent levels (similar to Gray code); and 2) bit flips are minimized. This leads to a smoother bit-encoding with fewer transitions as shown in Fig. 3(a). Fig. 3(b) illustrates the high fidelity of signal reconstruction achieved by NeuADC circuits trained with the proposed smooth encoding. Given a target bit-vector b_{GT} for each V_{in} defined by the smooth encoding, our cost function for training can be set to the cross-entropy loss commonly used for classification:

$$C(\tilde{b}, b_{\text{GT}}) = [\sum_{i=1}^{M} b_{\text{GT}i} \log(1 + e^{-\tilde{b}_i}) + (1 - b_{\text{GT}i}) \log(1 + e^{\tilde{b}_i})]^2 \quad (10)$$

We use the square of the sum of the cross-entropy losses for individual bits to engineer a higher penalty for multiple bit errors in the same bit-vector. We apply a decoding scheme to calibrate deviations in the learned mapping after training, where the learned parameters are used to compute the set $\{V_{in}^t, b^t\}$ for a finely sampled set of values V_{in}. We construct a lookup table that maps an encoded b to an estimated V_b' based on $V_b' = \text{Mean}\{V_{in}^t : \forall b^t = b\}$.

D. Hardware-oriented Training

We are able to train the parameters using stochastic gradient descent [9] following the learning objective and the differentiable cost $C(\tilde{b}, b_{\text{GT}})$ defined earlier. The parameters θ_1, θ_2 are initialized randomly and updated iteratively based on the gradients computed by the mini-batches of

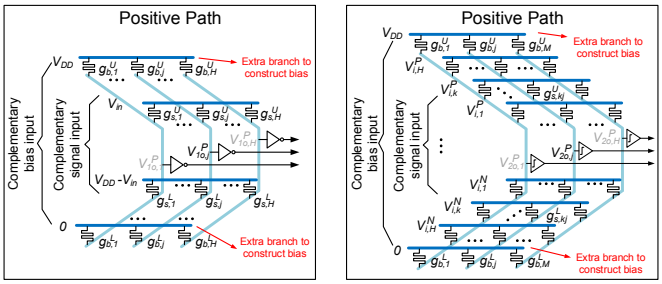

Fig. 4: Hardware model to map with training model. (a) Crossbar array for the first layer; (b) Crossbar array for the second layer.

$\{(V_{in}, b_{\text{GT}})\}$ pairs sampled from the input range. To enforce the feasibility constraints on θ_1 specified in Eq. (8), we clip the positive and negative path biases (the third term in Eq. (7)) to match Eq. (8), so that the final layer does not learn to depend on an infeasible combination of inputs in each iteration of training. Moreover, we periodically clip values of W_1 to the feasible set between [-0.5,0.5] and scale V_1 accordingly every 256 iterations.

This hardware-oriented training procedure allows us to confine the trained weight and bias parameters within the feasible range for circuit instantiation. To realize the precise trained bias, we add an extra row (word line) in the RRAM crossbar in both the hidden layer and the output layer as illustrated in Fig. 4. We instantiate the biases in Eq. (7) by providing the supply voltage V_{DD} as an input, in addition to the signal inputs V_{in} and $V_{DD} - V_{in}$. For instance, for each output of the positive path, i.e., $\tilde{p}_i,\ i \in \{1, \dots H\}$, we denote $g_{s,i}^U, g_{s,i}^L, g_{b,i}^U$, and $g_{b,i}^L$ as the conductances connecting the output to V_{in}, $V_{DD} - V_{in}$, the supply V_{DD} and GND as illustrated in Fig. 4(a). The conductance values can be calculated from the learned weights W_1 and V_1 as

$$\begin{aligned} g_{s,j}^U &= C_1 \times \max(0, W_{1,j}) \\ g_{s,j}^L &= C_1 \times \max(0, -W_{1,j}) \\ g_{b,j}^U &= C_1 \times (V_1 - \max(0, -W_1) \times V_{DD})/V_{DD} \\ g_{b,j}^L &= C_1 - g_{s,j}^U - g_{s,j}^L - g_{b,j}^U \end{aligned} \quad (11)$$

where $C_1 = g_{s,j}^U + g_{s,j}^L + g_{b,j}^U + g_{b,j}^L$ is a scaling factor to bring the RRAM conductances within a reasonable range. The process is repeated to instantiate the conductances for the negative path in the first layer. For the second layer, we adopt a similar strategy by first normalizing both W_2 and V_2 proportionally so that the sum of the positive and negative values across all columns is less than 1.

E. Quantization Schemes

The same training framework can be extended beyond a normal linear uniform quantization scheme to learn parameters for other quantization schemes, to reflect the desired precision for specific applications. To accommodate alternative schemes, the only update needed is changing the definition of b_{GT} in Eq. (9) by using a function of V_{in} instead of V_{in} itself. We can use $V_{in,log} = c \cdot log_2(a \cdot V_{in} + b) + d$ instead of V_{in} for logarithmic quantization, and $V_{in,sq} = c\sqrt{a \cdot V_{in} + b} + d$ for square-root quantization. Here, a, b, c, and d are the desired application-specific quantization parameters.

V. Experimental Methodology

A. Experiment Configurations

Training setup: The NeuADC NN model is trained via related techniques in [16] and stochastic gradient descent with the Adam optimizer [9] using TensorFlow. The batch size is 4096, and the projection step is performed every 256 iterations. We train for a total of 5.12×10^4 iterations (except for certain smooth codes that converge much faster), varying the learning rate from 10^{-3} to 10^{-4} across the iterations.

Technology model: We use the HfOx-based RRAM device model [5], [6], [8] to simulate the crossbar array. We consider the finite conductance/resistance resolution of the RRAM weight cells, and present the evaluation results in Sec. VI. The transistor model is based on a standard 130nm CMOS technology. The inverters, output comparators, and transistor switches in the RRAM crossbars are simulated with the 130nm model using Cadence Spectre.

B. Synthesis Flow

Fig. 5 illustrates our automated synthesis flow for the NeuADC circuits which consists of three phases.

Characterization phase: First, characterization at the basic device and circuit level is performed by a SPICE simulator — Cadence Spectre in our case, to extract the CMOS inverter and 3-input NAND comparator VTC curve and the RRAM conductance distributions. The characterization data are then fed to the training framework.

Training phase: The NN model of the NeuADC circuits can be fully captured by a group of hyper-parameters (H, N_B, N_S). H denotes the number of hidden neurons. N_B, N_S denote the number of binary bits and the number of smooth bits. Given the desired ADC resolution, the ground truth datasets can be generated according to the desired ADC quantization and are used to train the MLP. The training iterations are monitored to ensure the convergence of the learning objective, and the reconstruction quality is verified at the end of each training. If the reconstructed signals match well with the labeled ground-truth signals, the trained model parameters (W_1, V_1, W_2, V_2) are saved for later verification using SPICE simulation. Otherwise, hyper-parameters are updated for retraining until satisfactory performance is met.

Verification phase: The SPICE netlist of NeuADC is automatically synthesized according to the trained model parameters (W_1, V_1, W_2, V_2). The synthesis script instantiates the device/circuit design parameters, such as RRAM conductance and inverter sizes, to perform circuit-level simulation for verification. A comprehensive sets of circuit analysis are performed to rigorously evaluate and verify the performance of the synthesized NeuADC circuits. In our experiment, typical ADC metrics, such as the effective number of bits (ENOB), differential non-linearity (DNL), integral non-linearity (INL), and different ADC figures-of-merit (FoM) are used for the evaluation and verification.

VI. Experimental Results

A. Signal Reconstruction

First, we illustrate the reconstruction ability of NeuADC under three different quantization schemes in Fig. 6(a). For each quantization scheme, we reconstruct the signal using the decoding scheme in Sec. IV-C. The reconstructed signals (labeled as log, square-root and linear) match well with the original signal, shown in yellow, under different schemes, demonstrating that NeuADC can perform high-fidelity signal reconstruction with multiple reconfigurable quantization support using exactly the same hardware substrate.

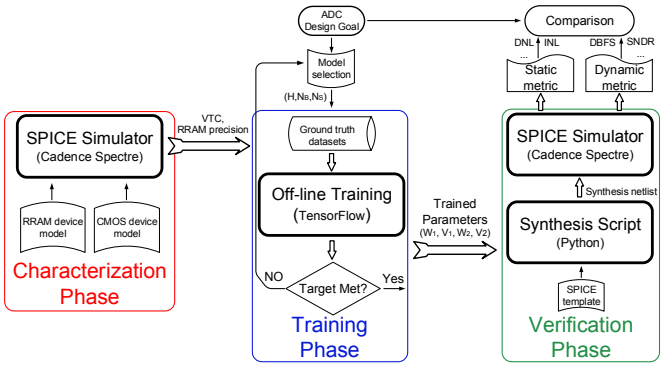

Fig. 5: The automated NeuADC design flow.

B. ADC Metric Evaluation

Second, we evaluate the performance of our proposed NeuADC across a range of ADC metrics and against both state-of-the-art and alternative synthesizable ADC designs. Table. I summarizes the ENOBs that can be obtained with different NeuADC designs. We report ENOB based on its standard definition ENOB=(SNDR-1.76)/6.02, where the signal to noise and distortion ratio (SNDR) is measured from NeuADC's output spectrum. Note that NeuADC is able to achieve close to ideal ENOB with a modest number of hidden and output neurons at 6, 7, and 8-bit. We then choose a specific NeuADC model (8→16 bits, 48 hidden units) as an example to report other representative metrics based on SPICE simulation. As shown in Fig. 6(b), the worst DNL and INL are -0.42LSB and -0.81LSB, well within the range of conventional linearity requirements. Fig. 6(c) exhibits -0.5 decibels relative to full scale (dBFS) of a 76.33MHz input with an SNDR of 49.68dB. Fig. 6(d) shows the linear trend of SNDR with changing input amplitude. Fig. 6(e) plots the SNDR degradation with signal frequency, verifying the bandwidth of the 8-bit NeuADC is well above 150MHz. All these metrics suggest good static and dynamic behaviors of our NeuADC circuits based on SPICE simulations.

We evaluate another 6-bit NeuADC model (6→8 bits, 12 hidden units) following similar methods, and the metric evaluation results are summarized in Table. II, together with comparative data for previously-published ADC designs. ISSCC14 represents the state-of-the-art manually-designed ADC [12], and Sto1 [2], Sto2 [14] are two representative synthesizable stochastic flash ADCs. NeuADC not only delivers superior performance at much higher sampling frequencies than the alternative synthesizable ADCs, but it can also achieve competitive FoMs comparable to those of the state-of-the-art ADC with an automated design flow.

C. Robustness Against Finite RRAM Precision

Previous work has shown that RRAM can be programmed to different resistance precisions, ranging from 6-bit to 12-

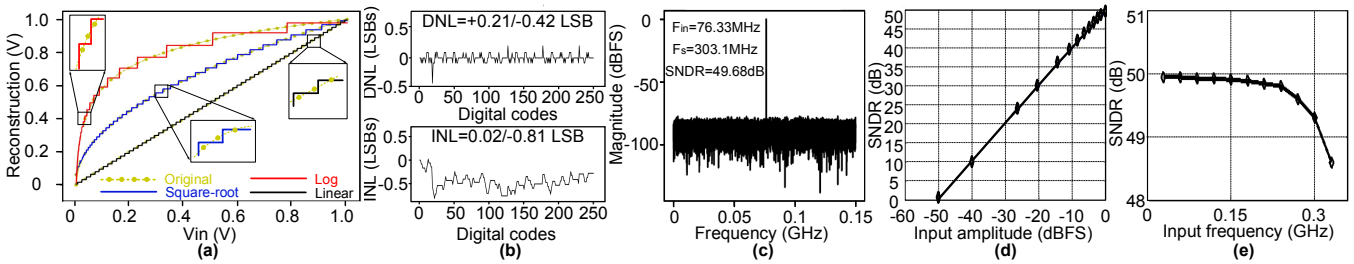

Fig. 6: (a) 6→8 *bits*, 12 *hidden units* NeuADC multi-quantization example. (b)-(e) are the simulated metrics of 8→16 *bits*, 48 *hidden units* NeuADC. (b) DNL and INL; (c) Output spectrum with a -0.5$dBFS$, 76.33MHz input, 303.1MHz sampling rate; (d) Linear SNDR trend with increasing input amplitudes; (e) SNDR trend with increasing input frequency.

TABLE I: Learned performance of NeuADC with Smooth Encodings measured by ENOB.

# Bits	# Hidden	ENOB
6→ 8	12	5.95
6→ 8	16	5.98
7→ 15	12	6.91

# Bits	# Hidden	ENOB
7→ 15	16	7.00
8→ 16	48	7.98
8→ 16	64	7.99

TABLE II: Performance Comparison.

	ISSCC14' [12]	Sto1 [2]	Sto2 [14]	**This work**
Supply Voltage(V)	1.1	1.3	1.2	**1.2**
Technology(nm)	40	180	90	**130**
Area(mm^2)	0.052	0.43	0.18	**0.005/0.04**
Power(mW)	27.4	0.631	34.8	**18/30**
$f_s(S/s)$	2.2G	8M	0.21G	**1G/300M**
$BW(Hz)$	1.1G	4M	105M	**0.5G/150M**
Resolution(bits)	7	6	6	**6/8**
ENOB(bits)	5.94	5.28	5.08	**5.95/7.96**
$FOM_W(fJ/c)$[a]	205.7	1970	3255	**291/401**
$FOM_S(dB)$[b]	143.4	132	121	**141.8/149**
Auto. Synthesis?	No	Yes	Yes	**Yes**
Reconfigurable?	No	No	No	**Yes**

[a] $FOM_W = P/(2^{ENOB} \cdot f_s)$. Walden FoM. Smaller is better. P is power.
[b] $FOM_S = SNDR + 10 \cdot log(BW/P))$. Schreier FoM. Larger is better.

bit when employed in RRAM-based NN accelerators [5], [6], [13]. Taking finite RRAM precision into consideration, we examine how the ENOBs of different NeuADC designs may vary with RRAM resistance precisions, as presented in Fig. 7. We observe that generally (B+1)-bit RRAM resistance precision is sufficient to achieve a target resolution of B-bit. Another finding is that NeuADC designs using more hidden or output neurons exhibit more robustness against the precision degeneration, suggesting a trade-off between robustness and resources/redundancy.

VII. Conclusion

We present NeuADC — a novel automated design approach for synthesizable A/D conversion with reconfigurable quantization support using the same hardware substrate. Inspired by NN, NeuADC is built upon a general NN hardware substrate enabled by a novel dual-path mixed-signal RRAM crossbar architecture. The design parameters are "learned" through NN training. We exploit a new smooth encoding scheme to improve the training accuracy and develop hardware-oriented circuit models and constraint formulations in the training process. The entire synthesis process of NeuADC can be automated without human designers in the loop. Comprehensive ADC performance metrics are evaluated using circuit-level SPICE simulation. The results demonstrate that our automatically-synthesized NeuADC can indeed be reconfigured for different quantization schemes with high-fidelity reconstruction and achieve performance comparable to state-of-the-art ADCs despite limited RRAM resistance precision.

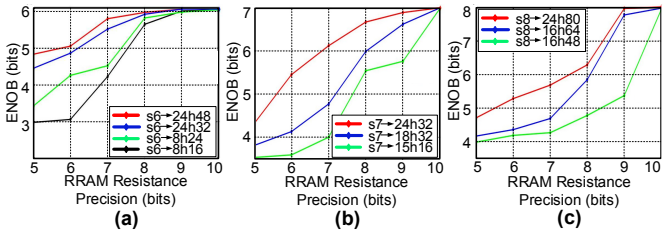

Fig. 7: Performance degradation for different NeuADC designs with decreasing RRAM resistance precision. (a) 6-bit models; (b) 7-bit models; (c) 8-bit models.

Acknowledgement

This work was partially supported by the National Science Foundation (CNS-1657562).

References

[1] AJ. Annema et al, "Analog Circuits in Ultra-Deep-Submicron CMOS," *IEEE JSSC*, vol. 40, no. 1, pp. 132-143, 2005.
[2] S. Weaver et al, "Stochastic Flash Analog-to-Digital Conversion," *IEEE TCAS-I*, vol. 57, no. 11, pp. 2825-2833, 2010.
[3] B. Xu et al, "A scaling compatible, synthesis friendly VCO-based delta-sigma ADC design and synthesis methodology," in *IEEE/ACM DAC*, 2017, pp. 1-6.
[4] L. Robert et al, "RedEye: Analog ConvNet Image Sensor Architecture for Continuous Mobile Vision," in *IEEE/ACM ISCA*, 2016, pp. 255-266.
[5] P. Chi et al, "PRIME: A Novel Processing-in-Memory Architecture for Neural Network Computation in ReRAM-Based Main Memory," in *IEEE/ACM ISCA*, 2016, pp. 27-39.
[6] B. Li et al, "RRAM-Based Analog Approximate Computing," *IEEE TCAD*, vol. 34, no. 12, pp. 1905-1917, 2015.
[7] Kurt Hornik, "Approximation capabilities of multilayer feedforward networks," *Neural Networks*, vol. 4, issue. 2, pp. 251-257, 1991.
[8] H. Li et al, "A SPICE Model of Resistive Random Access Memory for Large-Scale Memory Array Simulation," *IEEE EDL*, vol. 35, no. 2, pp. 211-213, 2014.
[9] Kingma et al, "Adam: A method for stochastic optimization," arXiv preprint arXiv:1412.6980, 2014.
[10] E. Reinhard et al, "Color Transfer Between Images," *IEEE Computer Graphics and Applications*, vol. 21, no. 5, pp. 34-41, 2001.
[11] B. Karlik et al, "Performance Analysis of Various Activation Functions in Generalized MLP Architectures of Neural Networks," *IJAE*, vol. 1, no. 4, pp. 111-122, 2011.
[12] M. Miyahara et al, "A 2.2GS/s 7b 27.4mW time-based folding-flash ADC with resistively averaged voltage-to-time amplifiers," *IEEE ISSCC*, 2014, pp. 388-389.
[13] M. Prezioso et al, "Training and operation of an integrated neuromorphic network based on metal-oxide memristors," *Nature*, vol. 521, no. 7550, pp. 61-64, 2015.
[14] S. Weaver et al, "Digitally synthesized stochastic flash adc using only standard digital cells," *IEEE TCAS-I*, vol. 61, no. 1, pp. 84-91, 2014.
[15] F. Bedeschi et al, "A bipolar-selected phase change memory featuring multi-level cell storage," *IEEE JSSC*, vol. 44, no. 1, pp. 217-227, 2009.
[16] X. He et al, "AxTrain: Hardware-Oriented Neural Network Training for Approximate Inference,"*ISPLED*, 2018.