This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JETCAS.2017.2777784, IEEE Journal on Emerging and Selected Topics in Circuits and Systems

SoC FPAA Hardware Implementation of a VMM+WTA Embedded Learning Classifier

Sahil Shah and Jennifer Hasler, *Senior Member, IEEE*

***Abstract*—This paper focuses on the circuit aspects required for an on-chip, on-line SoC large-scale Field Programmable Analog Array (FPAA) learning for Vector-Matrix Multiplier (VMM) + Winner-Take-All (WTA) classifier structure. We start by describing the VMM+WTA classifier structure, and then show techniques required to handle device mismatch. The approach is initially explained using a VMM+WTA as a two-input XOR classifier structure. The approach requires considering the entire mixed-mode system, including the analog classifier data path, control circuitry for weight updates, and digital algorithm for computing digital weight updates and resulting FG programming during the algorithm.**

I. FPAA enabled Embedded, On-Chip Learning

This paper focuses on the SoC large-scale Field Programmable Analog Array (FPAA) hardware implementation of a Vector-Matrix Multiplier + Winner-Take-All (WTA) [1] Embedded Learning Classifier. The SoC FPAA IC [2] was not designed or optimized for these classification, learning, or training tasks. The objective is to show the details of this novel learning algorithm as well as classifier implementation specifics. Unlike many machine learning applications, the SoC FPAA approach enables sensor (e.g. microphone), through analog preprocessing (e.g. frequency decomposition), and through the entire classifier and learning structure.

The on-chip embedded machine learning algorithm (Fig. 1) uses analog circuits for the classifier data path, analog infrastructure for sensing computed values into the microprocessor (μP), and μP computation for identifying learning updates as well as Floating-Gate (FG) node updates. A VMM+WTA learning algorithm connected to the FPAA hardware [3] can be trained one time or many times in the same IC infrastructure. The SoC FPAA IC was not designed or optimized for this learning algorithm (or most algorithms), but the SoC FPAA IC could be configured for these operations.

A VMM+WTA classifier, like at least a two-layer Neural Network (NN) classifier, is universal approximator. The VMM+WTA only requires a single layer [4]. The SoC FPAA has demonstrated hand-tuned VMM+WTA classifiers [1] for simple command word recognition [2], speech detection [5], and biometric classification [6], [7]. The classification requires less than 23μW of power, more than a factor of 1000$\times$ less custom digital solutions (vs analog computation) [8].

SoC FPAA devices enables an increase of 1000$\times$ in computational energy, and 100$\times$ in area efficiency to comparable digital computation, in a way that frees application engineers from custom IC design, similar to FPGAs for digital applications.

The authors are with the School of Electrical and Computer Engineering (ECE), Georgia Institute of Technology, Atlanta, GA 30332-250 USA (e-mail:jennifer.hasler@ece.gatech.edu).

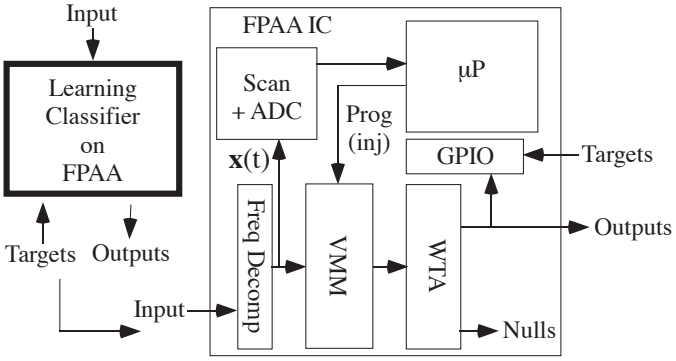


Fig. 1. This paper focuses on the circuit and related implementation aspects required for an on-chip, on-line SoC large-scale Field Programmable Analog Array (FPAA) learning algorithm utilizing a Vector-Matrix Multiplier + Winner-Take-All (WTA) classifier structure. The approach considers the entire mixed-mode system from analog input to analog output, including the analog classifier data path, control circuitry for weight updates, and digital algorithm for computing digital weight updates and resulting FG programming during the algorithm.

Implementation of custom ICs, particularly analog system ICs, takes years of development, requiring a large investment in time and highly specialized (and therefore expensive) people, that easily can miss a potential commercial or research target window opportunity. The heavy use of FPGAs, GPUs, and processors in digital processing directly comes from this reality for digital systems. FPAAs tend to be competitive in energy, area, frequency response [9] to custom devices, and the improvements from FPAAs to custom analog for a wide range of applications is less than the improvements from FPGAs to custom digital. One expects a significant demand in embedded machine learning systems, with all of the interest in learning networks [10], [11] and wearable devices. These opportunities will grow as FPAAs, and likely a family of FPAAs (e.g. [2], [12], [13]), become available.

II. VMM+WTA Circuit Classifier Structure

This section gives an overview of the fundamental operation of the VMM+WTA classifier structure and its SoC FPAA implementation. Figure 2 shows the measured operation for the WTA circuit embedded in a VMM + WTA learning classifier structure. The weight matrix (12x8) is programmed to an identity matrix illustrating the operation of each WTA input / output stage. This identity matrix is programmed (5nA) on top of a 10nA baseline current. This measurement uses on-chip DACs to enable each input (2.4V to 2.5V), in turn, to enable a single current for each WTA input. The VMM is implemented in routing fabric as mentioned elsewhere (e.g. [2]); further implementation details will be discussed in the following sections. Figure 2 shows the winners (and non-winners) controlled

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JETCAS.2017.2777784, IEEE Journal on Emerging and Selected Topics in Circuits and Systems

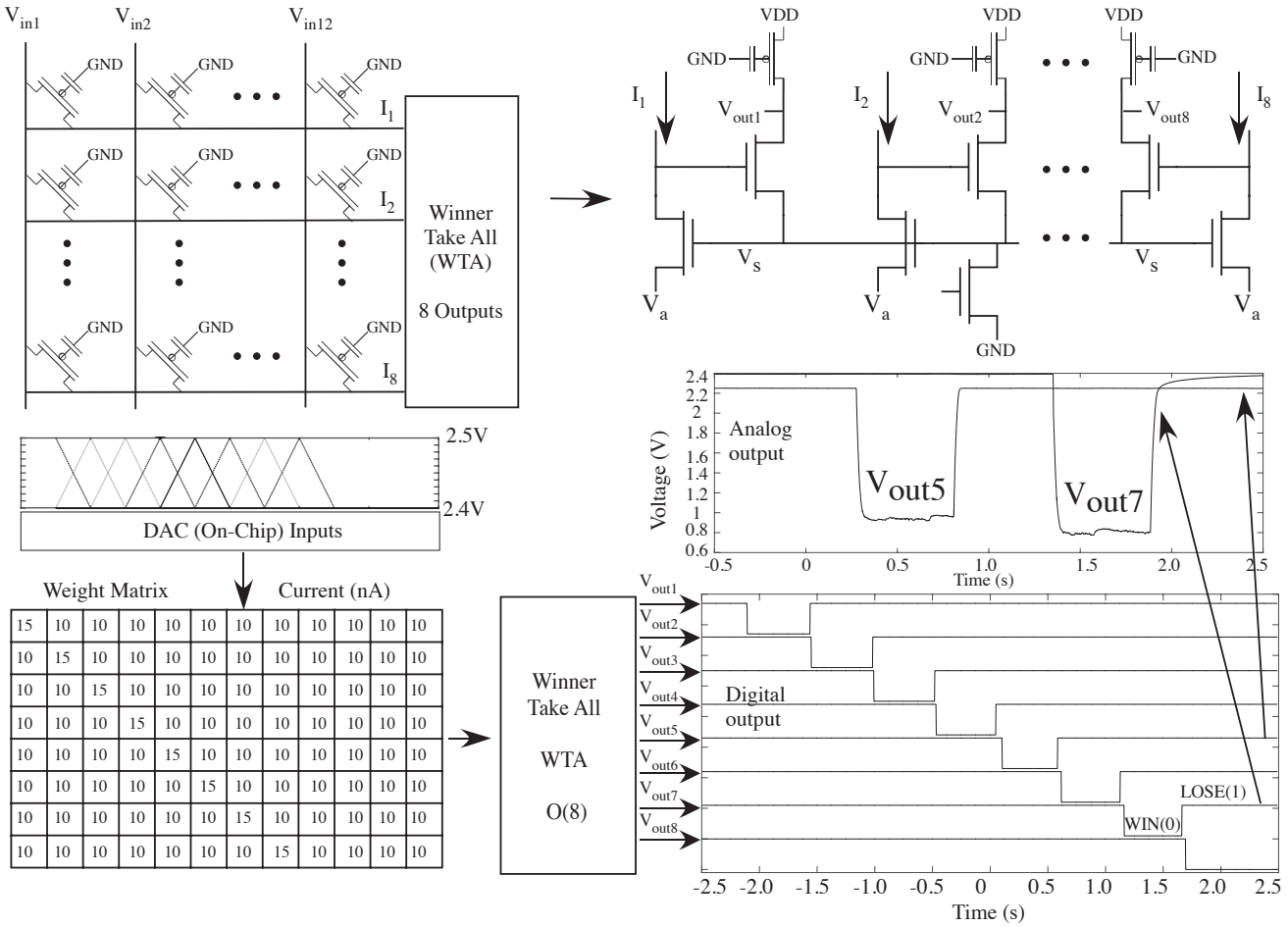


Fig. 2. Illustration of the WTA functionality. The VMM is programmed to an identity matrix (programmed to 10nA) for the entire function. The DAC inputs are ramped between 2.4V and 2.5V, each in sequence; the DACs come from explicit 7-bit signal DACs in the FPAA infrastructure. The eight WTA outputs all each win in sequence. The particular measured output waveform moves between a losing signal (between 2.2 and 2.5V) and a winning signal (below 1.2V). The winning signal is limited by the voltage of the common bias (V_s) on the WTA line. V_a was held at GND for this experiment.

by the simple classifier structure. Given the input pattern, we expect the outputs to win, in sequence, from the first output through the eighth output, corresponding to the experimental measurements. The core circuit derives from Lazzaro's WTA circuit using FG pFET devices to enable programmable load devices [14]. The FG pFETs are programmed independently, setting up threshold levels for each k-WTA stage. The outputs can *win* based on their relative computed metrics.

Figure 3 shows the particular VMM+WTA implementation for moderate size weight matrices in multiple SoC FPAA Computational Analog Blocks (CAB). Each compiled WTA stage, one per CAB, has one weight vector of the VMM operation as well as the resulting offset value. The resulting architecture just requires connecting a series of CABs together. The FG values, including the routing fabric weights, are programmed through a known infrastructure on the SoC FPAA IC [15]. FG programming is shown to be better than 0.80% for target currents between 5nA to 10μA [15].

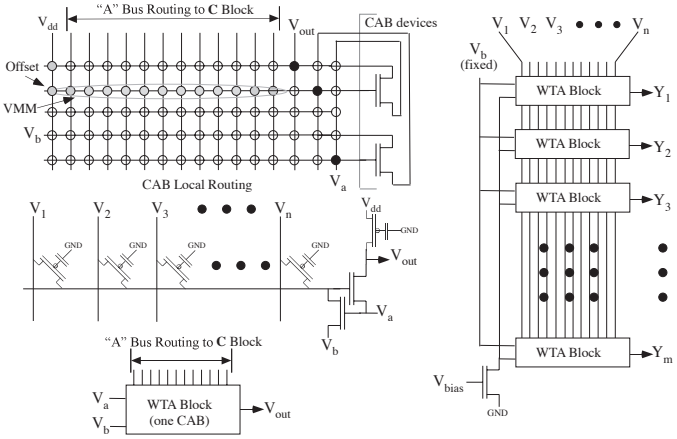


Fig. 3. Physical FPAA implementation of the VMM + WTA module in the FPAA. The VMM and the offset implementation are implemented as a row of FG switches connected to the input of the two nFET transistor (current conveyer) configuration. The reduced routing, circuit, and block representation are all shown. This block, implemented in a single CAB (with its two nFET transistors), is replicated in multiple CABs, one CAB per each output. Future implementations might consider fully integrated WTA stages in the CABs.

III. Classifier Mismatch: The Role and Removal of Mismatch for On-Chip Learning

Device mismatch impact physical classifiers. Not everything can be trained in a learning system; some absolute references are almost always required. Mismatch will occur between transistors of the compiled WTA circuit, of the ADC element, and resulting infrastructure elements. These approaches require

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JETCAS.2017.2777784, IEEE Journal on Emerging and Selected Topics in Circuits and Systems

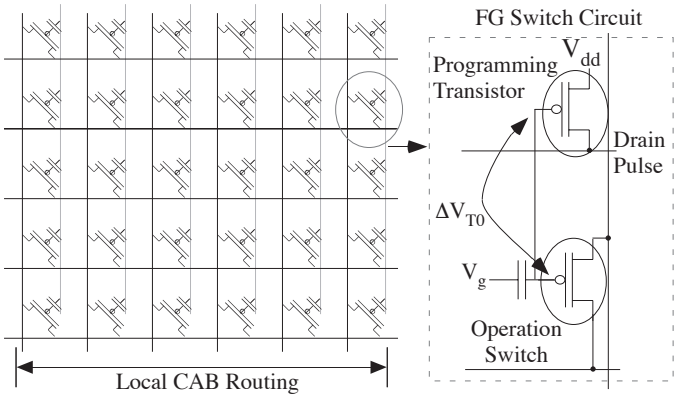


Fig. 4. Threshold voltage mismatch between pFET transistors for the indirect FG switch element is one of the sources of error. This error can be calibrated and incorporated in the programming infrastructure. The change in V_{T0}, measured directly through V_{out} in the programming infrastructure, remains roughly constant with the life of the chip. This is the primary point of error for the weights of VMM, stored as a charge on the floating node, after the first iteration of data through the array, or if the learning computation is performed off chip and downloaded to the device.

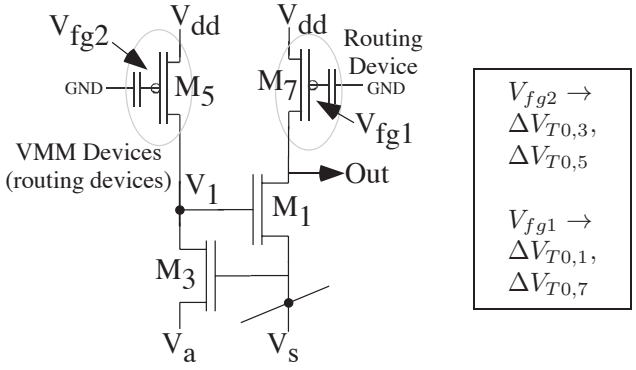


Fig. 5. WTA (+VMM) circuit diagram for addressing mismatch in the classifier structure. The Floating-Gate (FG) voltages can directly account for transistor mismatch for both high-gain sections.

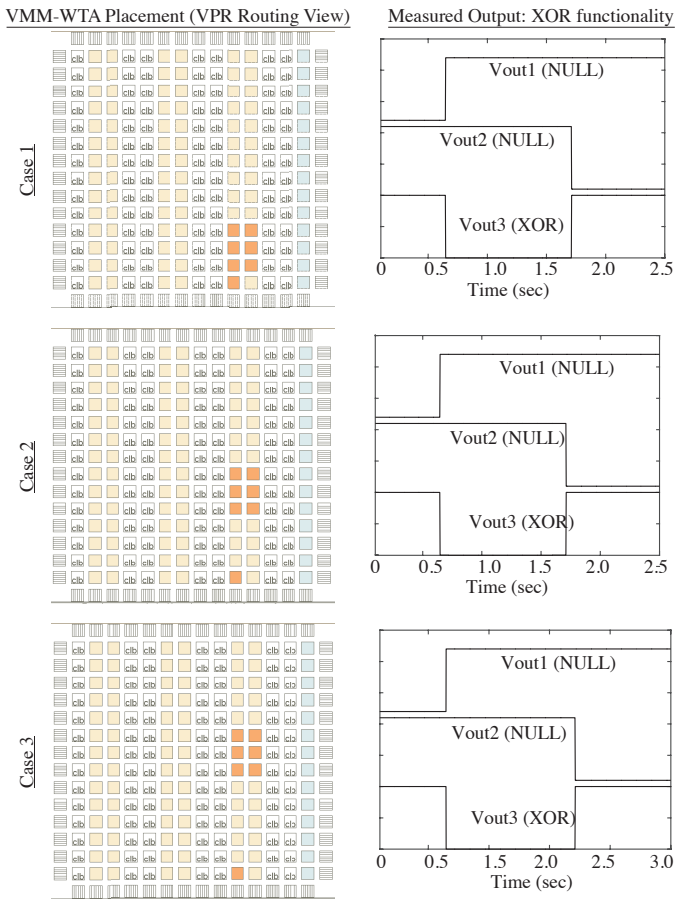


Fig. 6. The XOR classifier data is repeated for the VMM at three different locations, as seen by the three VPR routing views, and similar results. Multiple locations show the calibration eliminates effects of ΔV_{T0} due to indirect programming. The location of the VMM weight matrix has little effect on the resulting computation due to initial measurements that calibrate the V_{T0} mismatch from indirect programming.

other FG devices to remotely compensate for these effects. The primary mismatch issue, typical of most current ICs, is V_{T0} mismatch. The front-end circuitry is typically programmed and tuned separately [16]. Fortunately, within the SoC FPAA, one has roughly half a million analog FG parameters to account for these issues, parameters that often directly correct for threshold voltage (V_{T0}) mismatch. Some existing techniques are already possible, including system calibration with some mismatch map modeling [17], as well as initial built-in self testing [16].

FPAA Mismatch occurs because of indirect FG programming. The crossbar array of FG elements have two transistors per FG node (Fig. 4). The transistor to measure current in programming is different than the transistor used in the array. Two identically drawn devices have a threshold voltage (ΔV_{T0}) difference. This mismatch only needs to be characterized once for critical devices, such as VMM FG routing devices; these values might be useful even during learning operations.

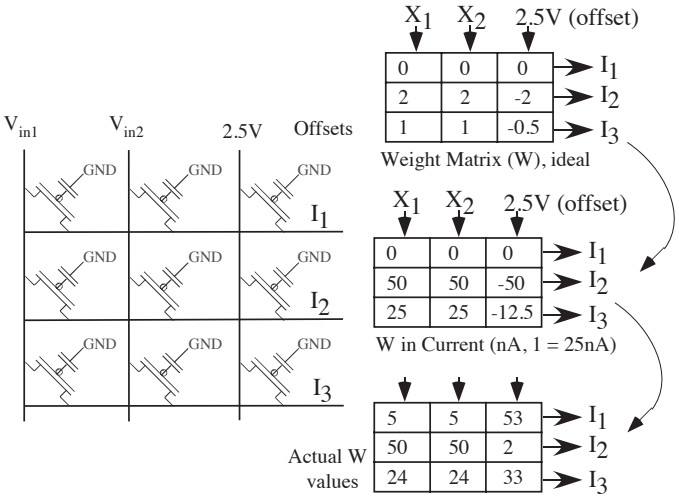
Figure 5 shows the WTA section, including the VMM, to use FG devices to compensate for mismatch. This compensation, by performing simple measurement of the switches used as VMM [17], enables results seen in Fig. 6, where one gets identical responses for three circuits compiled in three different locations (XOR classification application). The FG voltages address V_{T0} mismatch (Fig. 5) as $V_{fg2} \rightarrow \Delta V_{T0,3}, \Delta V_{T0,5}, V_{fg1} \rightarrow \Delta V_{T0,1}, \Delta V_{T0,7}$. V_{T0} mismatch from the gate term could be handled by the FG VMM pFET devices or FG pFET load transistor, both typically routing elements. The resulting circuit gain between V_{fg1} to V_1 is $\frac{\kappa_5}{\sigma_3}$. The resulting gain between V_1 to the Out node is $\frac{\kappa_1}{\sigma_7}$, where $\sigma_7 = \kappa_7(C_{ov}/C_T)$ because of the FG capacitive network.

The V_{T0} mismatch is the dominant mismatch in a transistor. Typically, mismatch in W and L tends to be 0.5% or less, and capacitor mismatch tends to be below 1% range. Mismatch in capacitances might have a small effect on the FG node, but in those cases, one programs the FG charge, accounting for these differences. When operating transistors with sub threshold bias currents, the percentage current change ($I_{mismatch}$ / I_{bias}) due to threshold voltage mismatch, ΔV_{T0}, is described for small to moderate mismatch ($\Delta V_{T0} < U_T$) as

$$\frac{I_{mismatch}}{I_{bias}} = e^{\kappa \Delta V_{T0}/U_T} \approx 1 + \frac{\kappa}{U_T}\Delta V_{T0} \qquad (1)$$

To have mismatch at 1%, it would require ΔV_{T0} <0.3mV, levels that are 1-2 orders of magnitude from realistic devices, particularly to scaled down devices. Most practical analog

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JETCAS.2017.2777784, IEEE Journal on Emerging and Selected Topics in Circuits and Systems



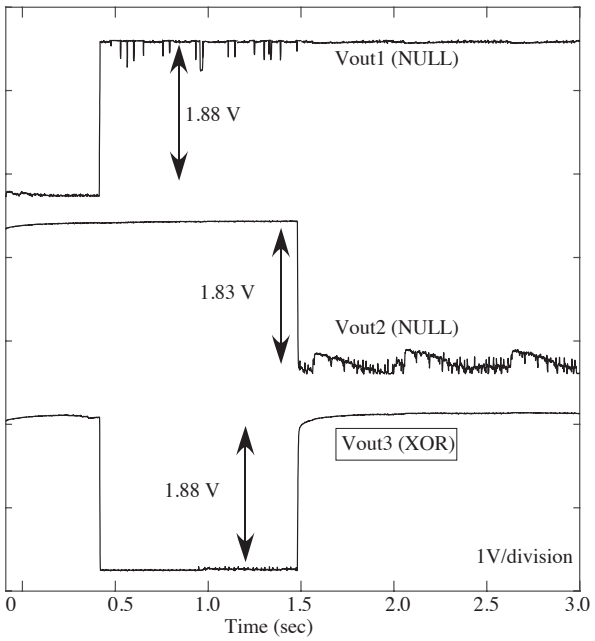


Fig. 7. VMM+WTA classifier illustrating the XOR functionality from a 2-input, 1 constant and 3 output classifier. The weights are transformed into programmed currents by assuming a weight of 1 normalizes to 25nA for this problem; the normalization is optimally chosen for the required frequency response for the VMM, although a much higher value is used for this illustration. The actual programmed currents are also shown, including offset currents required for all positive values. The third output is programmed to the XOR output; the first two outputs are nulls in the overall classification space.

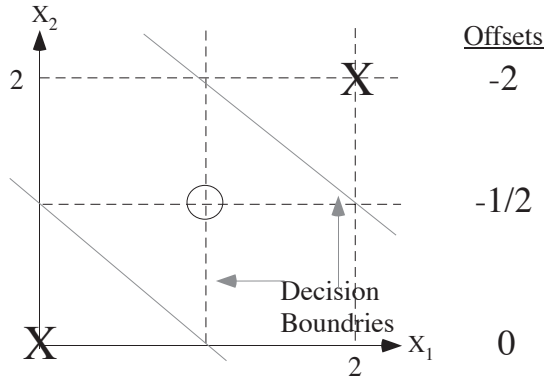


Fig. 8. Illustration of the input space for the XOR classification including the one desired (o) metric and the two null (X) metrics. This structure determines the decision boundaries for the XOR classification. The offsets are also included for all three computed metrics. The lower figure shows the transformation of the weight matrix and offset values from ideal matrix values to programmable current (positive) values. Weights are normalized to a value of 25nA (W=1 $\rightarrow$ 25nA). Then offsets need to be positive, so we need to add a constant offset to these offsets.

design tends to be sub threshold, near sub threshold, or within a gate voltage overdrive of 200-300mV. In all of these cases, ΔV_{T0} dominates the resulting device mismatch. Fortunately in these cases, FG capabilities can directly program out these errors, and correcting these errors also reduces temperature sensitivity due to device mismatch. These approaches also eliminates the need for any specialized layout techniques to create the necessary matching beyond usual techniques (transistors in the same orientation, same size devices, etc.).

IV. Classification and Learning Example: Two-Input XOR Classifier

This section looks at the learning structure for a simple classifier problem to illustrate the key concepts for circuit operation. Figure 7 shows the two-input XOR classifier measured output, repeatable (measured) for multiple locations on the SoC FPAA (Fig. 6). The WTA is programmed to have only a single winner. Weight values between 0 and 2 are scaled between bias currents of 0nA and 50nA (weight = 1 $\rightarrow$ 25nA). Inputs between 0 and 2 are scaled between 2.4V (0) and 2.5 (2). For this implementation, the inputs are applied externally (Analog Discovery USB Device). The values in Fig. 8 were obtained through learning the pattern from a labeled data set [3]. One can train on the weights off-line and download where useful for the application; the resulting adaptation could improve the results (after cluster step) as desired.

Figure 8 shows the two-dimensional classification space (X_1,X_2), including the two boundary lines between the three regions required for the XOR problem, as well as the resulting XOR metric output and two null metric outputs. The XOR computation has a single output, which makes for a conceptually clear example. A single winning output is also the exception for specifying the number and location of nulls in the classification space. Often a single output requires a noise level null as well as another null, typically with a starting position above the found null, adapting to the desired system solution. The input signals are randomly chosen values from a uniform distribution between 0 and 2. The initial solution for the input clustering, (X_1, X_2) = (1,1), equivalent to taking the moment inside the decision boundary region (solved easily by symmetry). The initial solution for the noise null would be to take the minimum actual measured values with the system noise applied; one would expect a value near (X_1,X_2) = (0,0).

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JETCAS.2017.2777784, IEEE Journal on Emerging and Selected Topics in Circuits and Systems

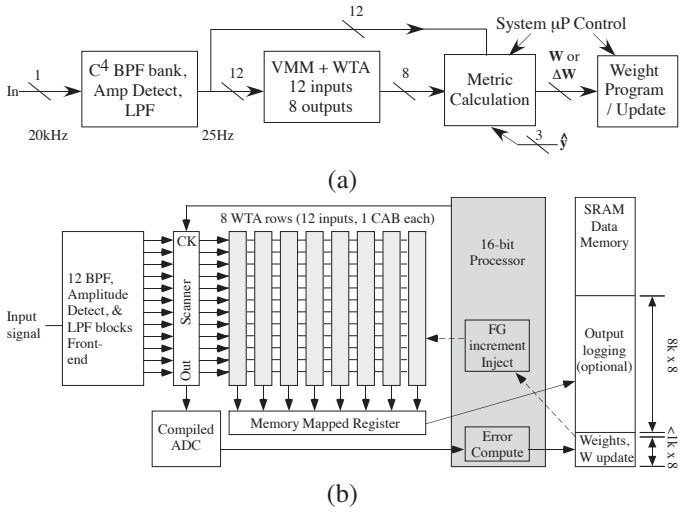


Fig. 9. VMM+WTA classifier Soc FPAA learning algorithm and implementation. (a) Block diagram similar to the tool level description. (b) Structured block diagram illustrating the various required SoC FPAA computations; the target signals and inputs from external and synced together.

The remaining null value would be selected at a higher point, likely at the upper right corner, (X_1,X_2) = (2,2). In such a fortunate case, one has arrived at the ideal solution and no further adaptation is required.

V. Hardware Specifics for On-Chip Learning Algorithm

This section will describe in detail the training algorithm for a 12-input VMM and corresponding WTA FPAA learning classifier, including measured data for this system. The learning algorithm has two steps. First, a clustering stage, using the first epoch of data, sets the initial weight and offset values. The starting weight values correlate to the resulting clustered positions. Second, a weight adaptation stage, sets the network goes through a modified LMS stage, where the errors in the training algorithm create shifts in the weights corresponding to positions in the classification space.

The input is initially processed through bandpass filter acoustic front-end processing, so the VMM input signals come from the peak detector / LPF output (Fig. 1). The learning and classification structure demonstration used a dataset obtained by Lincoln laboratory to perform classification for the *Nzero* DARPA program. The datasets were processed through a constant Q filter bank (from 1.6Hz to 5KHz), amplitude detection, and LPF (5Hz) structure, similar to Fig. 1. A FGOTA based LPF level shifts the signal between 2.4 and 2.5V.

The Soc FPAA implementation (Fig. 9a) includes the feedforward computation, spectral decomposition and classification, as well as the basic training approach. The digital processor computes the weight values after the first epoch and after the subsequent weight updates (Fig. 9a). The flow graph diagram is similar to the graphical *code* used to implement this function [2]. The hardware level implementation floor plans (Fig. 9b) the compilation of the FPAA components (e.g. VMM) in the routing fabric, as well as digital memory for the weight update computation. The input signals come from multiplexed compiled ADC and the target signals (digital) come directly into the processor. The weight updates originate from an 8-bit signal ADC, accumulated based on target signals for training stored in memory, and transformed into the resulting 14-bit target current (and therefore weight) value.

Figure 10 illustrates the detailed infrastructure used for the on-chip classification and learning. The weight values scale between 10nA (0) and 40nA (1), and the inputs are applied to the source voltage between 2.4V (0) and 2.5V (1). Only positive inputs and positive weights are requires for this VMM+WTA classification structure. Source voltage of 2.5V gives a current value near the programmed device level. Adding a constant to the same inputs of each weight vector results in a common-mode term to each WTA [3]; common term to all WTA inputs is effectively eliminated from the computation. The designer selects the particular current level and source input voltage levels based on the system constraints (e.g. frequency response, energy). Initially the weights are programmed below 10nA. This programming step accuracy is not significant (e.g. 50% accuracy) as long as it is below 10nA. The first iteration performed after initial programming will cluster the weights around the inputs.

A. First Iteration: Clustering Step

The first iteration learning step requires clustering each input when that vector in the training sequence is selected (Fig. 10). Digitally, this just requires adding input signals throughout the entire epoch as in (2). The inputs ($0 \rightarrow 1, 2.4V \rightarrow 2.5V$), measured through a ramp ADC, give $\approx$ 6bit accuracy for each summation (value between 0 and 1). The incoming data rate into the processor for acoustic signals (e.g. 10kHz) is 120KSPS. The input vector could be selected for the entire sequence; 10kSPS for 110s ($\approx 2^{20}$ samples) requires 26 bits to avoid worst case overflow.

The resulting target vector weight is the clustered value,

$$\mathbf{W_1} = \frac{\mathbf{1}}{\mathbf{samples}} \sum_{\mathbf{k}} \mathbf{x}\hat{\mathbf{y}}^{\mathbf{T}} \tag{2}$$

divided by the total number of times the clustered value appears. One must count the number of times each input is in the particular input class, a number between 0 and 2^{20}. After summation, this value is converted to the programmed weight value. The weight value corresponds to current between 10nA and 40nA, corresponding to measured V_{out} of the programming infrastructure is 1.3V to 1.4V, corresponding to 14-bit ADC code between 5936 and 6560. The span between the two numbers is 625 values, slightly more than 9bit representation. These 9-bit numbers of the summation are added to the lowest code (10nA, 1.3V, 5936). We just add the top 9-bits, 6-bits at the signal level and 3-bits after the decimal, scaled by a factor of 8 (giving an integer code), because a constant weight gain shift does not affect the resulting operation. The top 10 bits can be used with scaling. Computations for null starting points are kept within this same representation. Midpoints and noise floors for starting null positions are computed on the processor. The minimum of the unselected signals sets the noise level, a null is positioned at that location.

FG Programming for adaptation only requires increasing a current, an incremental hot-electron injection step (ms

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JETCAS.2017.2777784, IEEE Journal on Emerging and Selected Topics in Circuits and Systems

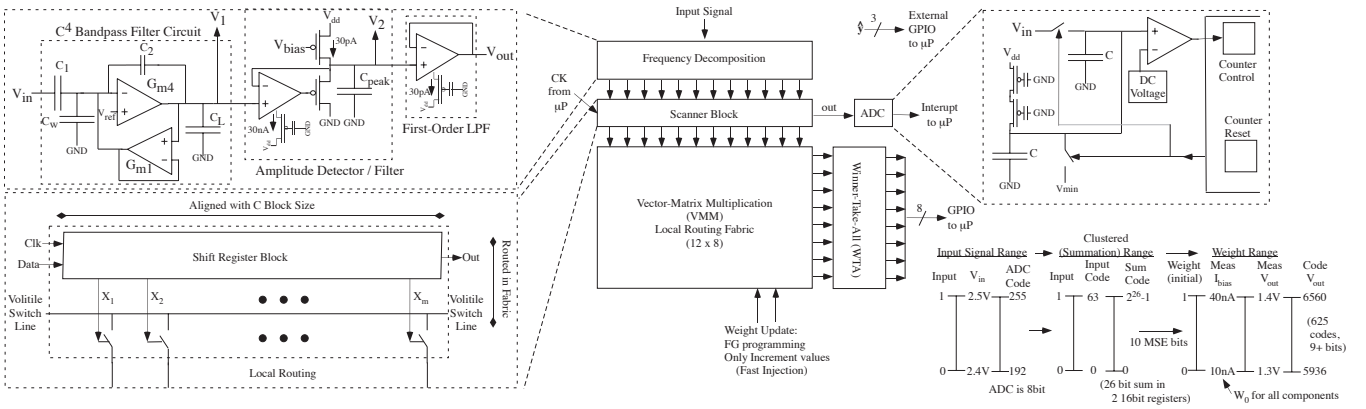


Fig. 10. Detailed block diagram for the acoustic learning problem. The feedforward flow moves through a frequency decomposition, through a scanner block, through a VMM computation in routing fabric and finally through the WTA elements before connecting to memory-mapped General Purpose I/O (GPIO). The frequency decomposition is a Bandpass filter, amplitude detect circuit, and first-order LPF, all compiled in a single CAB. The scanner block is built from volatile switch lines between the local and *C* block routing; the output goes through a compiled ramp ADC (8bit) that communicates through the processor through interrupt lines. The processor takes this data to compute weight updates, requiring shifts between various representations; the representation transformation for the first iteration (clustering step) is explicitly shown; further iterations scale from these representations.

timescale), Decreasing a current, requiring erasing an entire block (or the entire IC), and reprogramming the IC, including the desired value, by hot-electron injection (minutes). The programmed currents for targets are within a factor of 4 (10nA to 40nA) of the lowest target current. Most signals should be less than 40mV change in FG voltage on any adaptation step. Programming controls the injection process through a sequence of measurements and pulses of fixed time, to hit the desired target in as few pulses as possible without overshooting the target. The pulses are modeled to for finding a drain pulse that would satisfy the solution of the resulting FG voltage [15]. Each pulse will approach, but underestimate, the target.

The fixed-point processor based computation finds the next significant error, then finds the resulting drain DAC code to minimize the error. Drain voltage results in an exponential factor for the V_{fg} change per iteration, enabling the system to improve on MSB as well as LSB through a compressed, linear drain voltage. The measuring ADC (14-bit) is the component that requires accuracy to program the FG to a precise value. The theoretical limitation in accuracy comes from using a 14-bit ADC over the (roughly) 2V output voltage range, resulting from 1V shift in FG voltage for the measured device. The LSB for the 14-bit ADC results in 61μV in FG voltage accuracy, resulting in 0.166% error for subthreshold currents.

B. Later Iterations:Weight Adaptation Error Steps

Error metrics are computed in the processor as the data arrives. The computation of the weight updates (Fig. 11) start from target and output signals through the 8-bit signal ADC, accumulated as training selects, placed in memory, and , transformed at the epoch end into the LSB changes for the weight update (14-bit). Particular error metrics are signed quantities; the signed errors typically cancel some samples. The total number of potential samples normalizes averages the error metric. The weight updates are again programmed into VMM FG devices, in batch, after the computation of the epoch update metrics. This adaptation could be used after any initial programmed initial condition of the weights. One might adapt to slight difference from physical implementation to

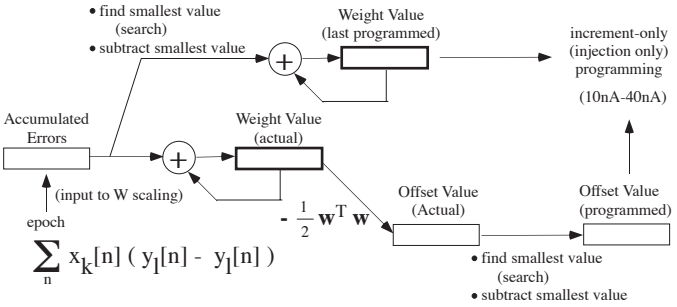


Fig. 11. Processor Update Algorithm Description. Bold boxes are continuously kept registers throughout each epoch iteration, and regular boxes are generated registers by each epoch iteration. The weight updates are computed from inputs, outputs, and target outputs throughout the epoch, the other operations occur at the end of an epoch. Computing the new weights is the first step in computing the new offsets. Updates are programmed as increasing values, requiring finding the smallest (typically largest negative number) value and subtracting it from all other values.

physical implementation if an off-line solution was developed. The clustering step finds a good initial condition when needed.

Programming a positive value into the FG array avoids the need to erase the resulting array. FG Programming requires only hot-electron injection pulses, even when negative values are used. Training the weight values so every increment is a positive step is essential to optimize programming times. If the same offset value is added to all of the weights for the VMM + WTA classifier, the classification remains unchanged. For a given set of weight updates, the smallest (likely negative) update would be subtracted from each weight update, including those that are 0. Adding a constant to all the weights requires taking the most negative components of every weight change for all weight vectors and use it as the baseline value (0), and all of the rest of the values are positive. Therefore, every weight value would either increase or remain the same, enabling only a small hot-electron injection weight update. Adding this constant has no effect on the required offset computation; the offsets are created from the actual weight values without constants applied. Weight changes often require offset changes. One must store (digital) the actual weight value, and programmed value. The FG voltage update should

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JETCAS.2017.2777784, IEEE Journal on Emerging and Selected Topics in Circuits and Systems

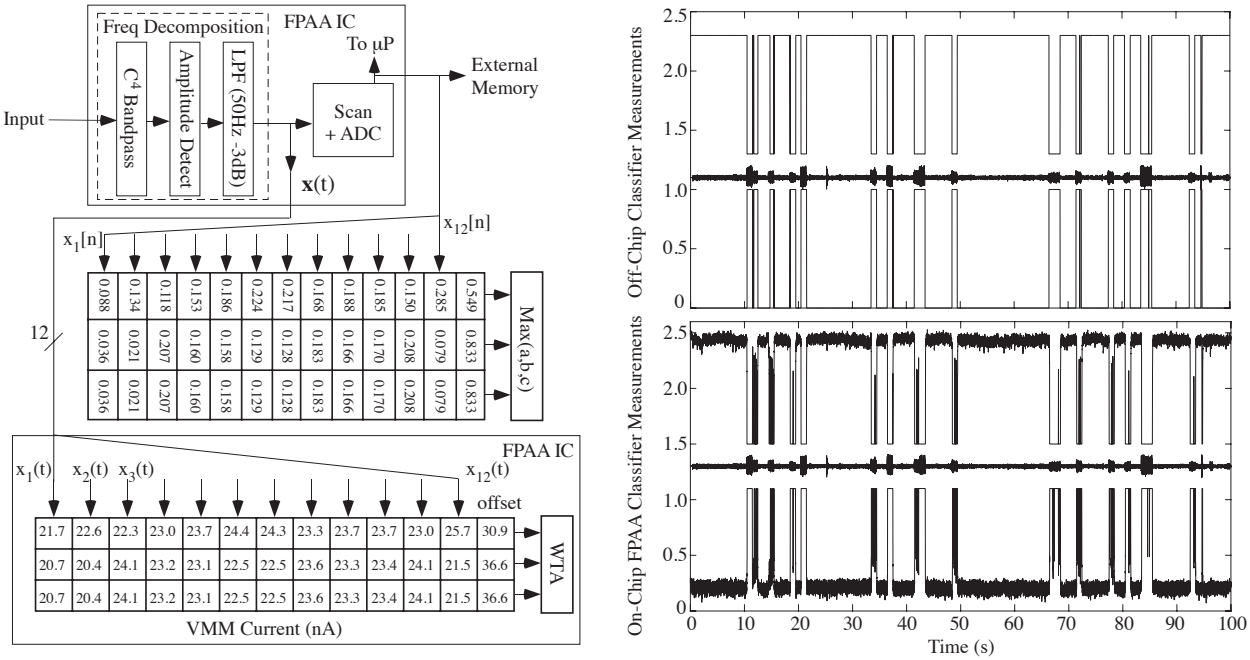


Fig. 12. VMM+WTA classification of an acoustic dataset created using a series of 1s data inputs, identifying the presence of a sound source, whether it be a generator, truck, or car. The classifier used a 12x3 VMM classifier followed by a 3 input, 3 output WTA. Both on-chip training (training and classification on the SoC FPAA) and off-chip training (training and classification numerically simulating SoC ODEs) using circuit models are shown for comparison. Both cases used the same on-chip frequency decomposition of 12 overlapping bandpass (C^4) filters, 12 amplitude detectors, and 12 low-pass filters (LPF, 50Hz corner for spectrum representation). The data measurements were offset to show the input signal, the WTA output (top vector), and one WTA null (third vector) on the same plot. These two approaches yield similar results, and assuming a minimum time for any symbol of 40ms, the classifier correctly recognized the results every time.

	Classify Energy(J)	# of Bands	IC Process	Metric Value	ADC ?	On-IC Learn
This Work	23μW	12	350nm	**1nJ**	N/A	**Yes**
[39]	16-20μJ	39	130nm	1.2μJ	no	no
[40]	124μJ	8	130nm	41.7μJ	no	no
[41]	$\approx$ 11μJ	15	40nm	6.42μJ	no	no

Fig. 13. Comparison of time-dependent signal detection and classification. Every system is solving a similar problem, requiring a number frequency decomposition bands. Acoustic classification $\approx$ 1k classifications per second, the closest number for continuous-time values, and the value used to interpolate where needed. The computation efficiency is consistent with general digital neural classification engines [42] . For a digital implementation the ADC is an essential part of the computation, although not included with our examples. The analog approach does not require this additional step. Our IC includes on-chip learning, a feature not discussed in other implementations.

be significantly less than 10mV increase on any adaptation step when required at all. Few pulses per element are required per iteration, resulting in fast injection programming times. When the first iteration reaches a reasonable starting solution, the number of errors is a small fraction of the measurements.

C. Twelve-input Classifier Learning Experimental Measurements

Figure 12 shows a comparative experimental measurement for the learning and training of these networks, one completely on-chip, and one where learning and classification is performed off the FPAA for comparison. The input dataset utilized a larger dataset composed of measured background acoustic sounds and additional measurements of generators, idle cars, and trucks in this environment. The input dataset was then composed of multiple 1s bursts of a generator, idle car, or truck on a 110s background; constructing the dataset in this way produces a labeled dataset. All learning and classification occurred on this dataset passing through the same frequency decomposition stage: 12 overlapping bandpass (C^4) filters, 12 amplitude detectors, and 12 low-pass filters (LPF, 50Hz corner for spectrum representation). The LPF function block also provides a DC shift for the VMM+WTA blocks. The approach shows a sensor-to-classified signal processing chain, unlike most classification algorithms, including hardware based classification and training algorithms.

Figure 12 shows the measured results of a single epoch after training converged. The network was trained to identify the presence of a sound source, whether it be a generator, truck, or car. One might use this representation to do further classification, similar to identification of speech over noise. The weights in Fig. 12 were the trained weights, one case for computer based simulation of this computation, and one case for on-chip learning of this computation. Two nulls (3-input and 3-three output single WTA device) starting near the classification the noise level after the first epoch.

Figure 12 shows measured results for a 12 input, 3 output VMM+WTA comparing the difference between emulating this structure on-chip, as well as implementing this classifier off-chip. The off-chip computation was done in MATLAB, experimentally modeling a subset of the analog functions; the WTA block was modeled as a $\max(\cdot)$ operation. Effectively, the results are similar, although the on-chip weights have additional offsets as expected from the training approach. These two approaches yield similar results. The on-chip learning

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/JETCAS.2017.2777784, IEEE Journal on Emerging and Selected Topics in Circuits and Systems

method required additional offsets to be applied as expected by the learning algorithm (no negative increments). Assuming a minimum time for any symbol of 40ms, the classifier correctly recognized every input correctly with no errors.

VI. VMM+WTA Hardware Discussion

A. Computation required for VMM + WTA learning classifier

The equivalent digital computation of this classifier, between the bandpass filter operation as well as the equivalent 12x8 VMM operating at a slow rate of 200SPS (for this problem) is roughly 4MMAC (/s). A Multiply ACcumulate (MAC) unit operating near the energy efficiency wall [18] will take roughly 1mW of power, consistent with the similar processing and energy requirements of digital hearing aid devices. The resulting memory access is likely a factor of 2 to 8 larger than this computation [19]. Implementation on an embedded processor, would require 250pJ/Op, typical of low power processors, would require roughly 4-8mW for these numeric computations. A typical ADC for this computation, such as ADI7457[20], would require 1mW at 3.3V supply to transform the resulting acoustic signal to the digital processor. The required classifier levels (23μW) are significantly less than the required, dedicated digital computation; these energy requirements have been consistent across multiple acoustic applications [2], [6], [7] as well as for this computation.

The computation and resulting learning removes the need for more complex GMM type hardware [21], [22], [23], including those cases built as part of machine learning approaches [23]. This classifier system, compiled on this FPAA, is consistent with the x1000 improvement factor in computation (measured in MAC operations), is similar to systems developed for VMMs (custom and compiled) [24], as well as other custom classifier networks [1], [22], [23].

Figure 13 shows the comparison of this classifier with digital classification equivalents for acoustic classification. The resulting numbers are consistent with the 1000$\times$ energy efficiency of analog computing versus digital approaches, including the original analog VMM computational efficiency [25] compared to the digital energy efficiency wall [18]. The digital ICs are all custom implementations, where the analog values come from a configurable SoC FPAA IC [2]. Classification energy is the energy required for a single classification step. The metric used is Classification Energy per frequency band normalized to the IC Process (350nm CMOS). Scaling for IC process is between linear to quadratic function; we utilize the conservative view of linear scaling improvement on energy efficiency for this table.

The power required during training for this implementation is higher due to using only the digital processor for digital updates. The feedforward classifier chain is below the typical 30μW of power required for classification. The digital computation requires 12 input samples every 100μs during training, as well as operations based on these values. Assuming roughly 10-one clock op per data sample, the computation is a 1MOp/s calculation, requiring roughly 2mW of power from the μP [15]. These computations are not typically high-power, but higher than the feedforward classifier chain. After learning convergance, the processor can sleep (clock set $\approx$0Hz).

B. Size of Classifier Implementations on SoC FPAA device

The section describes the maximum size of a neural network that can be compiled on a single SoC FPAA device. A network could be built as a single layer network, or as a combination of layers; each VMM+WTA classifier is a universal approximator for its input / output space. The maximum problem size depends on the number of WTA stages and then on the number of synapses and inputs. One can get between 1-2 WTA stages per CAB, with 98 CABs on the IC. The current implementation uses 16 inputs per CAB, although this number can be increased significantly by using the *C* block switches in addition to the local routing switches. Configurable fabric can allow for sparse patterns, which could potentially improve the computational complexity as in digital systems; in this case, we look at fully connected local arrays to provide one possible metric on this design. Conservatively (just 16 inputs per block), one could get roughly 200 WTA stages and 6000 synaptic (multiply + add) connections, operating from bandwidths less than 1Hz to greater than 5MHz on this IC. The VMM computation would be 30GMAC(/s) at 5MHz in this case, requiring 3-6mW of power. One can extend these approaches with multiple devices.

C. VMM + WTA learning classifier and other Classifiers

The learning algorithm running an epoch of the dataset after each change of weights. Earlier hardware algorithms have utilized this particular feature, including weight-perturbation type algorithms [26], [27], [28], [29], [30]. These approaches differ in using the signals to compute a weight update, typical of SOM and VQ type maps, to explicitly minimizing these errors for each step, resulting in typically few iterations. These concepts superficially relate to earlier learning SOM in hardware for 2 layer networks [31], although different training and network structure. These structures would differ from only programmed networks (e.g. [32], [33], [34]) or continuous online learning [35], [36], [37], [38].

VII. Summary on VMM+WTA Hardware Implementation

This paper focused on the circuit aspects required for an on-chip, on-line SoC FPAA learning for Vector-Matrix Multiplier + Winner-Take-All (WTA) classifier structure. The VMM+WTA classifier FPAA implementation, including techniques required to handle device mismatch, set the foundation for the learning efforts. Learning considerations started by considering VMM+WTA as a two-input XOR classifier structure. The approach requires considering the entire mixed-mode system, including the analog classifier data path, control circuitry for weight updates, and digital algorithm for computing digital weight updates and resulting FG programming during the algorithm. The approach was demonstrated on a larger (12-input, 3-output) VMM+WTA classifier structure. The SoC FPAA IC was not designed or optimized for these classification, learning, or training tasks.

Unlike many machine learning applications, the SoC FPAA approach enables going from sensor (e.g. microphone),

through the resulting analog preprocessing stages like frequency decomposition, as well as the entire classifier and learning structure. The on-chip embedded machine learning algorithm requires using analog circuits for the classifier data path, analog infrastructure for sensing computed values into the μP computation, and resulting μP computation for identifying learning updates as well as FG node updates.

This work, and resulting algorithmic modeling [3], are just the beginning of what is possible using these embedded on-chip, FPAA compilable algorithms. The on-chip classification and learning open opportunities for many areas in embedded computing, particularly sensor-input (e.g. acoustic, accelerometer, image, and RF). Additional hardware and algorithmic development enables wider use by applying these techniques towards multiple focused classification problems.

REFERENCES

[1] S. Ramakrishnan and J. Hasler, "Vector-Matrix Multiply and Winner-Take-All as an Analog Classifier," *IEEE TVLSI*, vol. 22, no. 2, 2014, pp. 353-361.

[2] S. George, S. Kim, S. Shah, J. Hasler, M. Collins, F. Adil, R. Wunderlich, S. Nease, and S. Ramakrishnan, "A Programmable and Configurable Mixed-Mode FPAA SoC," *IEEE VLSI*, June 2016.

[3] J. Hasler and S. Shah, "VMM + WTA Embedded Classifiers Learning Algorithm implementable on SoC FPAA devices," *JETCAS*, in press, 2017.

[4] W. Maass, "On the computational power of winner-take-all," *Neural Computation*, vol. 12, no. 11, pp. 25192535, 2000.

[5] S. Shah and J. Hasler, "Low Power Speech Detector On A FPAA," *IEEE ISCAS*, May 2017.

[6] S. Shah, H. Treyin, O. T. Inan, and J. Hasler, "Reconfigurable analog classifier for knee-joint rehabilitation," *IEEE EMBC*, August 2016.

[7] S. Shah, C. N. Teague, O. T. Inan, and J. Hasler, "A proof-of-concept classifier for acoustic signals from the knee joint on an FPAA," *IEEE Sensors*, October 2016.

[8] J. Hasler, "Opportunities in Physical Computing driven by Analog Realization," *ICRC*, October 2016.

[9] J. Hasler, S. Kim, and F. Adil, "Scaling Floating-Gate Devices Predicting Behavior for Programmable and Configurable Circuits and Systems," *JLPEA*, vol. 6, no. 13, 2016, pp. 1-19.

[10] Nils J. Nilsson, *Introduction to Machine Learning*, 2005.

[11] A. Mohamed, G. E. Dahl, and G. Hinton, "Acoustic Modeling Using Deep Belief Networks," *IEEE transactions on Audio, Speech, and Language Processing*, Vol. 20, no. 1, 2012, pp. 14-22.

[12] Rumberg, B.; Graham, D.W. "Reconfiguration Costs in Analog Sensor Interfaces for Wireless Sensing Applications." IEEE MWSCAS, Columbus, OH, USA, 2013. pp. 321–324.

[13] N. Guo, Y. Huang, T. Mai, S. Patil, C. Cao, M. Seok, S. Sethumadhavan, and Y. Tsividis, "Energy-efficient hybrid analog/digital approximate computation in continuous time," IEEE JSSC, 2016, pp. 1–11.

[14] J. Lazzaro, S. Ryckebusch, M. A. Mahowald, and C. A. Mead, "Winner-take-all networks of O(N) complexity," in *Advances in Neural Information Processing Systems 1*, Morgan Kaufmann, 1989.

[15] S. Kim, J. Hasler, and S. George, "Integrated Floating-Gate Programming Environment for System-Level ICs," *Transactions on VLSI*, vol. 24, no. 6, 2016. pp. 2244-2252.

[16] S. Shah and J. Hasler, "Tuning of multiple parameters with a BIST system," *IEEE CAS I*, Vol. 64, No. 7, July 2017. pp. 1772-178

[17] S. Kim, S. Shah, and J. Hasler, "Calibration of Floating-Gate SoC FPAA System," *Transactions on VLSI*, September 2017.

[18] B. Marr, B. Degnan, P. Hasler, and D. Anderson, "Scaling energy per operation via an asynchronous pipeline," *IEEE TVLSI*, vol. 21, no. 1, pp. 147151, 2013.

[19] J. Hasler, "Energy Constraints for Building Large-Scale Neuromorphic Systems," *GOMAC*, March 2016.

[20] http://www.analog.com/media/en/technical-documentation/data-sheets/AD7457.pdf. Last visited August 31, 2017.

[21] P. Hasler, P. Smith, C. Duffy, C. Gordon, J. Dugger, and D. Anderson, "A floating-gate vector-quantizer," *MWCAS*, vol. 1, 2002, pp. 196199.

[22] S.-Y. Peng, P. Hasler, and D.V. Anderson, "An analog programmable multi-dimensional radial basis function based classifier," *IEEE CAS I*, Vol 54, No. 10, 2007. pp. 2148-2158.

[23] J. Lu, S. Young, I. Arel, and J. Holleman, "A 1 TOPS/W Analog Deep Machine-Learning Engine With Floating-Gate Storage in 0.13 μm CMOS," *IEEE Journal of Solid-State Circuits*,vol. 50, no. 1, 2015.

[24] C. Schlottmann, and P. Hasler, "A highly dense, low power, programmable analog vector-matrix multiplier: the FPAA implementation," *IEEE Journal of Emerging CAS*, vol. 1, 2012, pp. 403-411.

[25] R. Chawla, A. Bandyopadhyay, V. Srinivasan, and P. Hasler, "A 531 nW/MHz, 128 x 32 current-mode programmable analog vector-matrix multiplier with over two decades of linearity," *IEEE Custom Integrated Circuits Conference*, October 2004, pp. 651-654.

[26] M. Jabri and B. Flower, "Weight Perturbation: An Optimal Achitecture and learning Technology for Analog VLSI Feedforward and Recurrent Multilayer Networks" *IEEE Transactions on Neural Networks*, vol. 3, no. 1, 1992.

[27] Philip H.W. Leong, and M. A. Jabri, "A Low-power trainable analogue neural network classifier chip," *IEEE Custom Integrated Circuits Conference*, 1993, pp. 4.5.1 - 4.5.4.

[28] K. Hirotsu, and M.A. Brooke, "An Analog Neural Network Chip With Random Weight Change Learning Algorithm," *IJCNN*, vol. 3, Nagoya, 1993, pp. 3031-3034.

[29] G. Cauwenberghs, "Neuromorphic Learning VLSI Systems: A survey" *Neuromorphic systems engineering*, Springer, 1998.

[30] G. Cauwenberghs, "An analog VLSI recurrent neural network learning a continue-time trajectory," *IEEE Transactions on Neural Networks*, vol. 7, no. 2, 1996, pp. 346-361.

[31] B. Zhang, M. Fu, H. Yan, and M. A. Jabri, "Handwritten Digit Recognition by Adaptive-Subspace Self-Organizing Map," IEEE Transactions on Neural Networks, Vol. 10, No. 4, JULY 1999 pp. 939-945.

[32] J. C. Platt and T. P. Allen, "A neural network classifier of the I1000 OCR Chip," *NIPS*, 1996. pp. 938-944.

[33] J. Chen and T. Shibata, "A Neuron-MOS-Based VLSI Implementation of Pulse-Coupled Neural Networks for Image Feature Generation," *IEEE CAS I*, vol. 57, no. 6, 2010. pp. 1143-1153.

[34] B. Larras, C. Lahuec, F. Seguin, and M. Arzel, "Ultra-Low-Energy Mixed-Signal IC Implementing Encoded Neural Networks," *IEEE TCAS I*, vol. 63, no. 11, 2016. pp. 1974-1985.

[35] P. Hasler and J. Dugger, "An analog floating-gate node for supervised learning," *IEEE TCAS I*, vol. 52, no. 5, 2005. pp. 834845.

[36] J. H. Poikonen, M. Laiho, "A mixed-mode array computing architecture for online dictionary learning," *IEEE ISCAS*, 2017.

[37] M. A. Petrovici, et. al, "Pattern representation and recognition with accelerated analog neuromorphic systems," *IEEE ISCAS*, 2017.

[38] N. Qiao, H. Mostafa, F. Corradi, M. Osswald, F. Stefanini, D. Sumislawska, and G. Indiveri, "A reconfigurable on-line learning spiking neuromorphic processor comprising 256 neurons and 128K synapses," *Frontiers in Neuroscience*, April 29, 2015.

[39] J. Kwong, and A. P. Chandrakasan, "An Energy-Efficient Biomedical Signal Processing Platform," *IEEE JSSC*, vol. 46, no. 7, July 2011. pp. 1742-1753.

[40] K. H. Lee, and N. Verma, "A Low-Power Processor With Configurable Embedded Machine-Learning Accelerators for High-Order and Adaptive Analysis of Medical-Sensor Signals," *IEEE JSSC*, vol. 48, no. 7, July 2013. pp. 1625-1637.

[41] M. Shah, et. al, "A Fixed-Point Neural Network Architecture for Speech Applications on Resource Constrained Hardware," *Journal of Signal Processing Systems*. Nov. 25, 2016. pp. 1-15.

[42] J. K. Kim, P. Knag, T. Chen, and Z. Zhang, "A 6.67mW sparse coding ASIC enabling on-chip learning and inference,"' *IEEE VLSI Circuits*, 2014, pp. 1-2.