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.2017.2771392, IEEE Journal on Emerging and Selected Topics in Circuits and Systems

VMM + WTA Embedded Classifiers Learning Algorithm implementable on SoC FPAA devices

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

***Abstract*—This paper presents a learning algorithm for a VMM + WTA classifier one layer architecture on a Large-Scale Field Programmable Analog Array (FPAA). The technique enables opportunities for embedded, ultra-low power machine learning, techniques typically considered for large servers. To develop this training algorithm, the paper starts by understanding fundamental equivalent transformations for VMM +WTA classifier networks. A VMM + WTA structure can exactly compute a Self-Organizing Map (SOM) or Vector Quantization (VQ) operation, in addition to other transformations. SOM, VQ, and Gaussian Mixture Models (GMM) learning concepts are utilized for the training algorithm of this single one-layer network. An on-chip clustering step determines the initial weight set for ideal target and background values. Null symbols are important for the algorithm and are set from midpoints of the target values. The results are shown both as numerical simulation of the VMM+WTA learning network, illustrating some numerical ODE simulation limitations for this problem, as well as experimental measurements implemented on an SoC FPAA device.**

This paper focuses on training of classifiers for a single layer of a Vector-Matrix Multiplier (VMM) and a single layer of a k-Winner-Take-All (WTA) [1], built on our original foundational work on VMM + WTA classifiers being demonstrated experimentally as a universal approximator [2], and recent work demonstrating a single engineer-tuned example of wordspotting classification in the SoC large-scale Field Programmable Analog Array (FPAA) [3]. The experimental demonstration of the universal approximation concept [2], based on Maass' theoretically description a decade earlier [4], encourages the use of one-layer (or multiple layer) networks for embedded low-power classification. A universal approximator is a classifier that can represent any static function (with infinite resources) in a single layer of processing.

An automatic technique of learning the VMM classifier weights based on representative data sets is far preferred to hand tuning a particular classifier for a given application. This work develops a VMM+WTA training algorithm capable of the universal approximator functionality, and demonstrating the learning both in numerical simulation as well as experimentally in an SoC FPAA. The algorithm was developed through understanding the connections of VMM+WTA classifier to Self Organizing Maps (SOM) [5], [6], [7], Vector Quantization (VQ) and Learning VQ [8], [9], [10], [11], [12], [13], Gaussian Mixture Models (GMM) [14], and Support Vector Machine (SVM) capabilities [15]. Training multilayer networks, typically required for universal approximation, often has training issues due to error estimations in all but the last classificatier layer; avoiding this issue is central is most neural network (NN) [16], machine learning [17], and deep learning theory [18].

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).

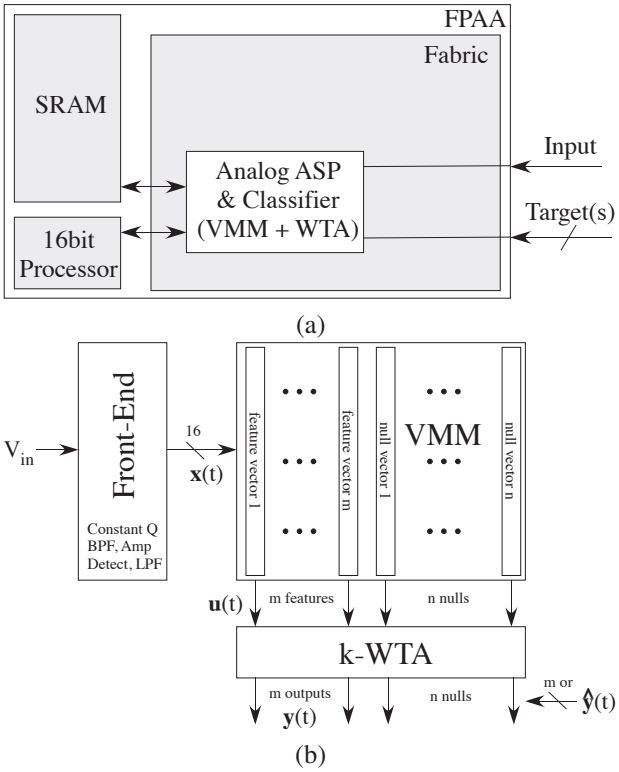


Fig. 1. On-chip classifier training algorithm on a Large-Scale Field Programmable Analog Array (FPAA). (a) Block Diagram for the on-chip, SoC FPAA based learning algorithm for the Vector-Matrix Multiplier (VMM) with dynamic k-Winner-Take-All (WTA) Embedded Classifier utilizing the on-chip 16-bit microprocessor (μP) and SRAM. (b) Block diagram for the Vector-Matrix Multiplier (VMM) with dynamic k-Winner-Take-All (WTA) Embedded Classifier [1], including the front-end circuitry, VMM (from $\mathbf{x}(t) \rightarrow \mathbf{u(t)}$, k-WTA (from $\mathbf{u}(t) \rightarrow \mathbf{y(t)}$) blocks, and training signals on desired outputs ($\hat{y}(t)$). We choose an acoustic demonstration problem utilizing front-end circuitry of constant Q filters, amplitude detection, and post filtering setting up the inputs classification.

This paper focuses on the algorithm implementation using the SoC FPAA IC [3], and classification will focus on audio classification tasks as a representative, although not exhaustive sample; we easily see these approaches extendable to sensor, vision, and communication tasks. FPAA devices enable the factor of 1000$\times$ improvement in energy efficiency over custom digital approaches [19], as well as improvements in area efficiency. Improvements over commercial FPGAs and related hardware is even more significant because of memory issues. The companion paper explaining particular hardware details is presented as part of this issue [20], and summarized in Section II. Recent FPAA devices utilize on-chip μP processors, even with programming in operation mode, one can utilize these

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.2017.2771392, IEEE Journal on Emerging and Selected Topics in Circuits and Systems

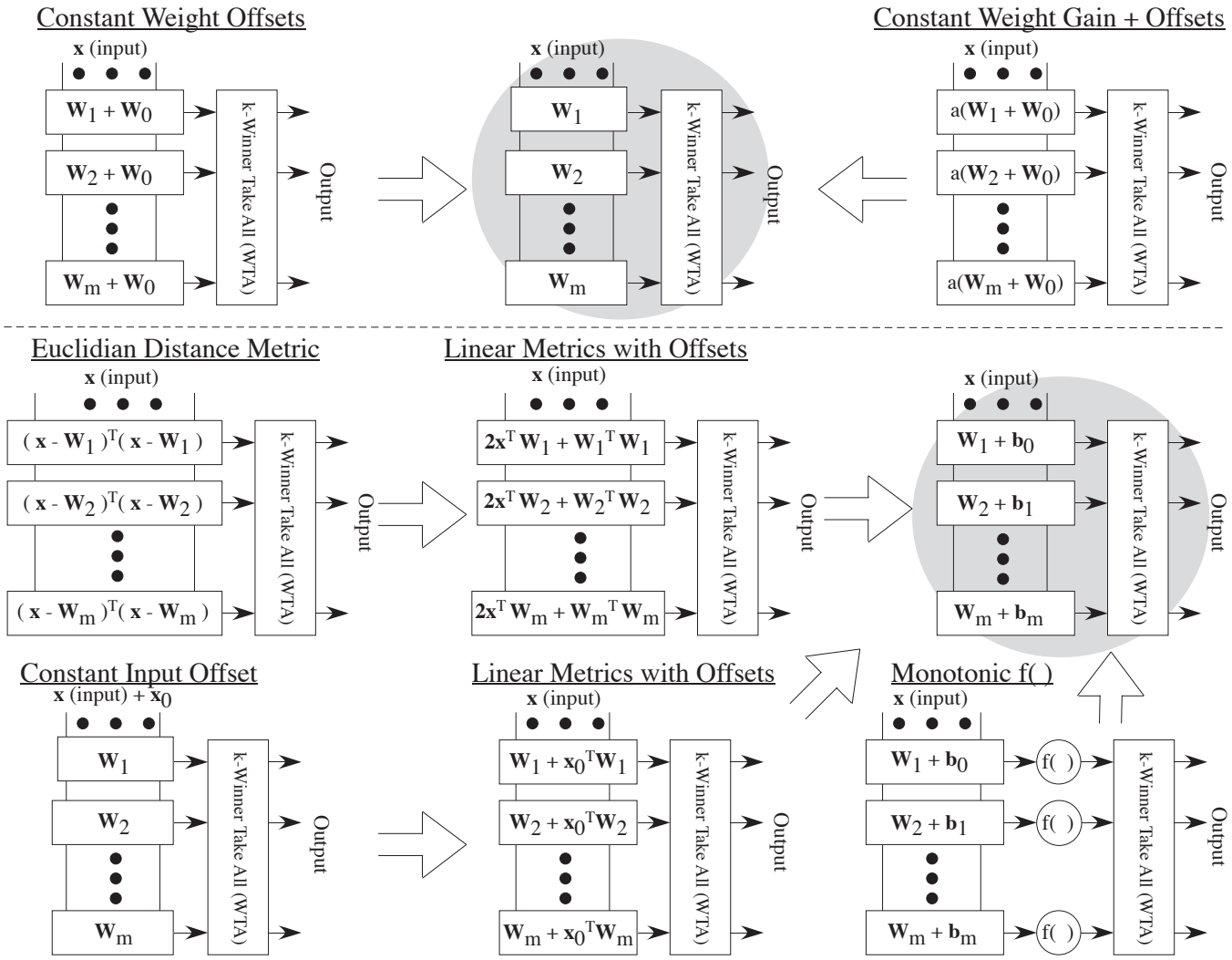


Fig. 2. Illustrating mathematical reductions possible for WTA classifiers. We assume the WTA classifier has sufficient precision for its comparisons. Constant Weight Offsets: Transforms to just VMM weights. Constant Weight Gain + Offsets: Transforms to just VMM weights. Euclidian Distance Metric: Transforms to linear VMM metric with particular offsets. Constant input offset: Transforms to linear VMM with weights with offsets. Monotonic f( ): Using a monotonic function (e.g. $e^{(\cdot)}$) after the linear VMM computation with offsets results in a reduction to a transformation of a linear VMM with weights and offsets.

features for on-line learning [3]. Given this framework, we will focus our on-chip learning for this system to involve training the classifier over a single statistical presentation of the data, or a single epoch, and then adapt the weights after each presentation (where the chip computes the error metrics, etc.). The input comes from a parallel (12 device) bandpass filter bank, resulting amplitude detectors, and filtering, enabling using a range of recorded and realistic auditory datasets (Section III).

This paper demonstrates the first on-chip learning algorithm for an on-chip FPAA classifier. The focus of this paper is on how to develop this on-chip learning algorithm, leaving the discussions about the particular application capability to their own focused discussion. Section I will describe the mathematical framework for the VMM + WTA classifier. Section II will summarize the SoC FPAA device aspects that are important for this implementation. Section III will describe the application and dataset used to demonstrate training this VMM +WTA classifier. Section IV will describe our VMM + WTA classifier, the VMM+WTA learning approach pointed towards an FPAA device. Section V summarizes our discussion.

I. Properties of the VMM +WTA Classifier

The WTA circuit [1] eliminates the multi-input common-mode signal, resulting in similarities between multiple classifier layers, similarities that are encapsulated in the VMM+WTA classifier. Building a VMM+WTA classifier mathematical framework requires understanding mathematical reductions of classifier layer(s) to this VMM+WTA block, enabling development components of a VMM+WTA learning algorithm.

Figure 2 shows graphical illustrations of these mathematical principles, particularly the mathematical simplifications from more complex to the basic VMM + WTA classifier. A mathematical framework to utilize to allow for transformations and simplifications that will reduce system complexity where not decreasing functionality as well as give a framework to understand potential learning algorithms for this system.

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.2017.2771392, IEEE Journal on Emerging and Selected Topics in Circuits and Systems

A VMM is the product of an input vector, $\mathbf{x}(t)$, and stored weight matrix, $\mathbf{W}$, giving the output vector $\mathbf{u}(t)$,

$$\mathbf{u}(t) = \mathbf{W}\mathbf{x}(t). \tag{1}$$

FPAA routing fabric performs energy-efficient (and extensively modeled) VMM computation [39].

Typically the difference of the inputs (typically to the largest input) into the k-WTA will undergo a high-gain transformation. The k-WTA allows for at most k winners. The particular output transistor current provides a non-zero confidence for each of the outputs, resulting from the *significant* input metrics. The digital outputs are achieved by scaling the output thresholds of the many-input comparison stage. The WTA computation compares between all of the input signals; one can find local WTA type operations (e.g. local spreading of inhibitory signals), but won't be the case treated in this discussion.

Limitations of the WTA circuit might have some practical limitations on the resulting classifier, and its analytic modeling. The WTA circuit input (to output) current gain is finite, limiting some approximations when input metrics are sufficiently close. We often need higher gain for the typical situation when all inputs have a common baseline (simplifying the *common mode* rejection) because we need to find smaller differences between inputs. The noise of the input signals as well as noise generated from the WTA circuit is also important. The following subsections go multiple key simplification properties.

A. Weight Offset effect on VMM+WTA Classification

For VMM weights transformed with the resulting output as

$$\mathbf{W} \rightarrow \mathbf{W} + \mathbf{W}_0, \mathbf{u} = \mathbf{W}\mathbf{x} + \mathbf{W}_0\mathbf{x}, \tag{2}$$

respectively, where $\mathbf{W}_0$ is a matrix of positive components that has identical valued rows (e.g. all values of $\mathbf{W}_0$ are equal), and the second term ($\mathbf{W}_0\mathbf{x}$) would be identical for all inputs. Because the WTA differentially compares the resulting inputs, the same value applied to all inputs, even a time varying value, has no effect on the resulting classification. Similarly one can prove that if we scale the weights all by a constant value.

This property allows entirely positive computation weights without any theoretical loss of behavior, requiring half of the VMM FG devices. This property eliminates the need for single-ended to differential-ended conversions. This property enables learning algorithms to only require positive weight changes (e.g. only hot-electron injection [23]) assuming that the weights in the same columns (for a same input) all increase by the same amount.

B. Input Offsets Effects for VMM +WTA classifier

For inputs transformed by a shift in the input vector, $\mathbf{x}_o$,

$$\mathbf{x} \rightarrow x\mathbf{x} + \mathbf{x}_o, \mathbf{u} = \mathbf{W}\mathbf{x} + \mathbf{W}\mathbf{x}_0, \tag{3}$$

resulting in a constant term for the k-th input into the WTA as ${\mathbf{w_k}}^T\mathbf{x}_0$. Either one can compensate by a separate constant per row to equalize the differences in ${\mathbf{w_k}}^T\mathbf{x}_0$ (e.g. adding current / element), or constrain this metric in weights to be constant. This result enables unsigned-only inputs (non-zero dc bias point), while varying around a constant unsigned level, for both computation and training. The unsigned (i.e. positive) input signals for a spectrum classifier structure reduces the complexity of the VMM infrastructure.

C. Equivalence of Euclidian Distance and VMM Metric for VMM+WTA classifier

Starting with a typical Euclidian distance metric

$$u_k = -\sum_l \left(x_l - w_{l,k}\right)^2 \tag{4}$$

as the input into our WTA block, where x_k is the k-th value in the vector $\mathbf{x}$, and u_l is the l-th value in the vector $\mathbf{u}$. We use a minus sign to convert between finding a winning metric versus finding a minimum value typical of VQ [8] and SOM [5] networks. Expanding out the square terms gives

$$u_k = -\mathbf{x}^T\mathbf{x} + 2{\mathbf{w_k}}^T\mathbf{x} - {\mathbf{w_k}}^T\mathbf{w_k} \tag{5}$$

where we notice that for all values of k, $\mathbf{x}^T\mathbf{x}$ is the same value added to each metric, and therefore can be eliminated. Therefore, the WTA compares with the metric

$$u_k = {\mathbf{w_k}}^T\mathbf{x} - \left({\mathbf{w_k}}^T\mathbf{w_k}\right)/2 \tag{6}$$

Because the offset is critical for equivalent operation with VQ / SOM networks, one typically adds additional currents to equalize the differences in ${\mathbf{w_k}}^T\mathbf{w_k}$. This solution brings the theory used for VQ, as well as associated theory for SOM, into the operation of this classifier's learning / training.

D. Abstracting away Identical nonlinear WTA input functions to WTA

Having a monotonic function, $f(y_k)$, for all k inputs into the WTA, does not change the classification function. Assuming high enough gain, the resulting monotonic function does not change the relative size of the incoming signals; even though transistor gain typically is sufficient, in practice the gain required should be taken into account. The practical implication of this result is looking at metrics, say as those used for *bump* regions

$$u_k = \exp\left(\sum_l \sigma_l \left(x_l - w_{l,k}\right)^2\right) \tag{7}$$

often resulting in the computation of a product of input signals from the guassian functions. As a result of all elements having the same $\exp()$ nonlinear function, we can reduce the resulting input to the WTA block as

$$u_k = \sum_l \sigma_l \left(x_l - w_{l,k}\right)^2 \tag{8}$$

This case returns the VMM block mentioned before. For arbitrary distributions, we will need additional terms for the resulting input power variances; an alternate way of handling

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.2017.2771392, IEEE Journal on Emerging and Selected Topics in Circuits and Systems

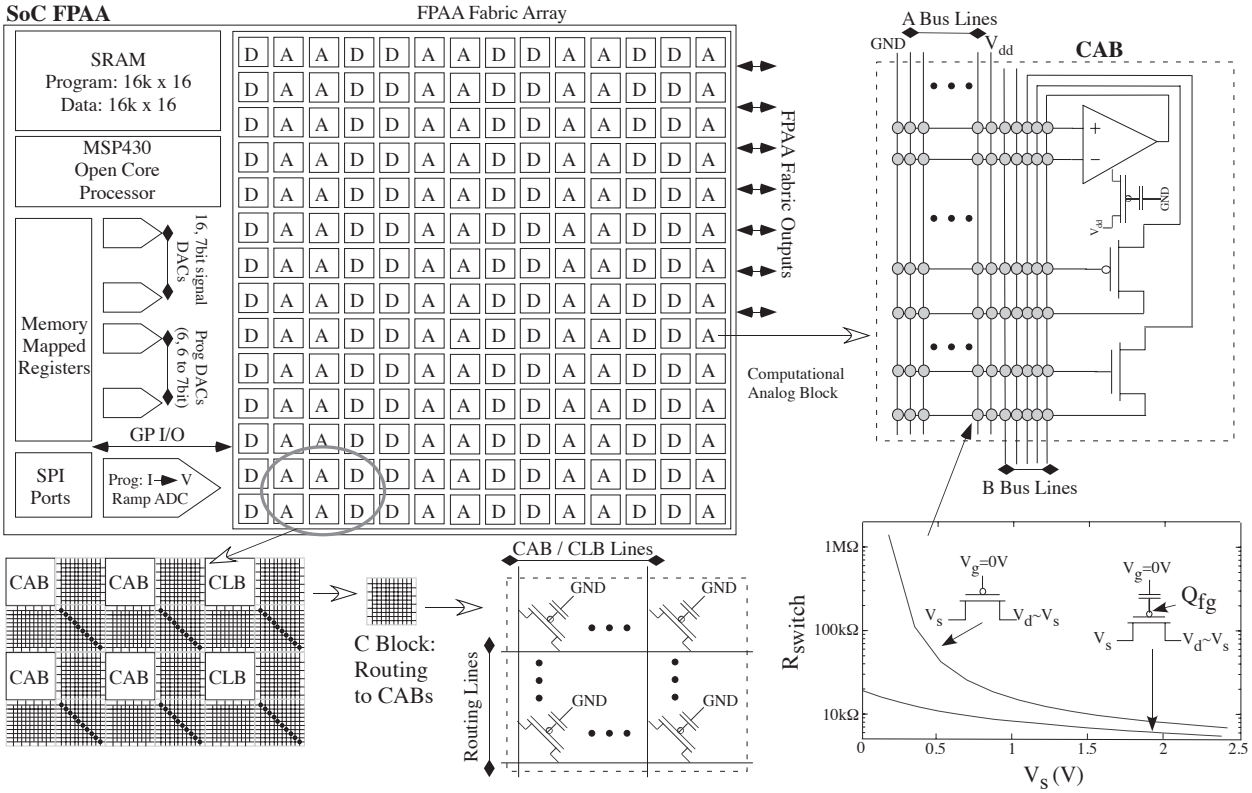


Fig. 3. Top Level View of the SoC FPAA IC, focusing on components used for the VMM+WTA learning application. The SoC FPAA has 98 Configurable Analog Blocks (CAB) intermixed with 98 Configurable Logic Blocks (CLB) in a Manhattan-style routing architecture. The manhattan routing utilizes connection (C) block routing and switch block routing to reach the CAB / CLB local routing (crossbar) fabrics. The routing elements are Floating-Gate (FG) devices, capable of storing voltages outside of the power supply rail, providing a single-transistor rail-to-rail switch.

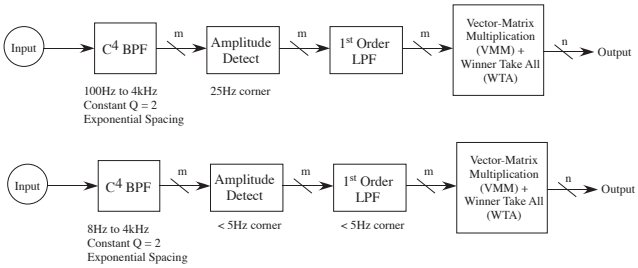


Fig. 4. Acoustic low-power classification architecture compiled on our SoC FPAA device. *Upper plot*: Block diagram for analog auditory word classification, in a similar representation used for our tool framework, using a BPF with center frequencies that are scaled evenly on a log-frequency scale between 100Hz and 4kHz with a nearly constant Q (Q≈2). *Lower plot*: Block diagram for classifying acoustic sensor signals for field tested devices being turned on and off. This system has a similar architecture, but with significantly lower corner frequencies (from 8Hz to 5KHz) to detect the majority of signal energy of these mechanical structures.

these issues is utilizing nulls to sharpen the resulting distributions as required for different σ values. When we generalize values for $\sigma_{l,k}$, typical for a GMM metric as

$$u_k = \exp\left(\sum_l \sigma_{l,k}\left(x_l - w_{l,k}\right)^2\right) \tag{9}$$

which then reduces to comparing metrics

$$u_k = \sum_l \sigma_{l,k}\left(x_l - w_{l,k}\right)^2, \tag{10}$$

that requires handling nonlinearities (quadratic terms), or additional metrics (nulls) to deform the decision boundary. This property allows the VMM+WTA classifier to perform SOM [5], [6], [7], GMM [14], and SVM capabilities [15].

II. SoC FPAA Hardware Overview

Figure 3 shows the top-level overview of the SoC FPAA IC. We summarize the important operational aspects for the learning application; more detailed information can be found elsewhere (e.g. [3]). The SoC FPAA IC utilizes a Manhattan FPAA architecture, including the array of computation blocks and routing, composed of Connection (*C*) and Switch (*S*) blocks. This configurable fabric effectively integrates analog (A) and digital (D) components in a hardware platform easily mapped towards compiler tools. The switchable analog and digital devices are a combination of the components in the Computational Analog Blocks (CAB), in the Computational Logic Blocks (CLB), and in the devices in the routing architectures that are programmed to non-binary levels. The architecture is based on Floating-Gate (FG) device, circuit, and system techniques. VMM computation in routing fabric is enabled by this unique routing device. Integrating these capabilities with an on-chip microprocessor (μP) component (open source MSP 430) and 16 on-chip 7-bit signal DACs, completes the picture that this FPAA is a SoC computing device, not just an analog signal-conditioning device. SoC FPAA has dynamically modifiable routing fabric using shift register control signals, directed by locally routed signals in the fabric, to rapidly between programmed aspects . The approach

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.2017.2771392, IEEE Journal on Emerging and Selected Topics in Circuits and Systems

enables a configurable scan chain throughout the SoC FPAA fabric.

SoC FPAA devices enable the factors of 1000× computational energy, and 100× area, efficiency to comparable digital computation, in a way that frees application engineers from custom IC design, similar to FPGAs for digital applications. 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 (scaling paper) 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. FPAA also enables the dream of analog circuits, not to mention analog signal processing, implemented as reusable IP blocks, enabling a designer's circuit or system creation to be used by engineers. The capabilities of design tools [42] as well as initial analog computing development [43], are essential parts of these emerging capabilities. One expects learning classifiers, with all of the interest in learning networks [17], [18], would have high demand. These opportunities will grow as FPAAs, and likely a family of FPAAs, become available.

III. Example Dataset for Classification Learning

This section discusses the dataset and experiment used for demonstrating the learning process for this classification structure. Figure 4 shows a block diagram of application dataset that we will use to develop our training algorithm. The dataset was obtained by Lincon Labs to measure classification for the *Nzero* DARPA program. The low-power systems were required to classify acoustic (and other) signals after two minutes of data was presented. This problem has continued to be a challenging problem, with other groups just barely classifying the most elementary situations after years of effort [40]. This dataset is modified to have unto twenty, 1s sound sources, in a two minute acoustic window. The new classification problem is significantly harder than the original problem, and is learned and classified in low-power hardware.

For the learning investigations, as well as data analysis, the datasets were processed through a constant Q filterbank (from 1.6Hz to 5KHz), amplitude detection, and LPF (5Hz) structure, similar to Fig. 4. This technique provides a good representation of acoustic signals, similar to human cochlea systems, and retaining desired acoustic time-spectral information. This circuit approach has been efficiently implemented on the SoC FPAA device previously (e.g. [3]) and is utilized again for this discussion. High-level MATLAB / Scilab level modeling, based on experimental measurements, closely agrees with these measurements, and can be nearly interchangeably used for these approaches. These spectrally separated outputs were recorded to be used for theoretical and experimental development. The input acoustic signal, going through a typical input ac coupling, was converted from pressure as 40mV / Pascal (from the original measurements), similar to the front-end sensors in the testbed measurement (which include high-pass corner frequencies around 20Hz).

Figure 5 shows the 16 output signals from the bandpass filter block for generator datasets in both urban (concrete) and rural (dirt) conditions, showing the strong narrow response just under 100Hz. Rural datasets have higher attenuation than the corresponding urban datasets, probably because sound travels better through concrete than dirt, often resulting in narrower spectrums for classification. Car datasets have a wider band of higher frequency signals, differing its results from the truck and generator signals. In these figures, the straight-line elements are the extracted background noise level (with occasional discussions by those setting up the measurements) obtained from the quiet urban and rural datasets. Signal below the noise could still be extracted using techniques other than simple amplitude metrics.

The task will learn the correct classification of one of these single datasets of steady-state acoustic signals. The dataset uses multiple signals and resulting background that is far more challenging than originally proposed for training and classification. The test input is composed of signals from urban environment for 3 objects and background "quiet" sound setting a typical background noise constructing a realistic labeled data set. The resulting blocks of datasets give us the opportunity to generate a wide range of labeled datasets. These nulls in the space highly reduce the number of false alarm cases. The classifier files are available on our GT website[1]. [2]

IV. VMM+WTA Learning and Computation

This section develops a batch on-chip learning algorithm based on utilizing the connection between a VMM+WTA classifier and VQ, SOM, GMM, and SVN classifiers (Section I), and experimentally implemented on the SoC FPAA IC (Section II) as well as in numerical simulation. The application dataset was described in Section III. The VMM + WTA gives a structure quite similar to VQ / GMM although with lower complexity, because the input metrics only require a VMM operation implemented in the routing fabric of the SoC FPAA.

This work demonstrates the first embedded classification algorithm for this ultra-low energy VMM+WTA classification. The input is processed through a 12 BPFs log-spaced from 1.6Hz to 5kHz range. The hardware demonstrates the entire end-to-end system operating with less than 25μW (measured) average power. These acoustic classifier measurements are consistent with other acoustic classifier measurements, such as command-word classification [3], or knee-wrap classification [44]. We calibrate out some mismatch from the front-end BPF filter and amplitude detector structures (as in [3]), precisely programming the bias currents, as well as account for capacitor mismatch.

This rich structure of the VMM +WTA classifier points towards some potential training / learning algorithms. We develop a supervised approach for VMM + WTA training,

[1]Files can be found on the website: hasler.ece.gatech.edu

[2]We will make available our files for this classifier after acceptance and publication of this paper on our GT website.

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.2017.2771392, IEEE Journal on Emerging and Selected Topics in Circuits and Systems

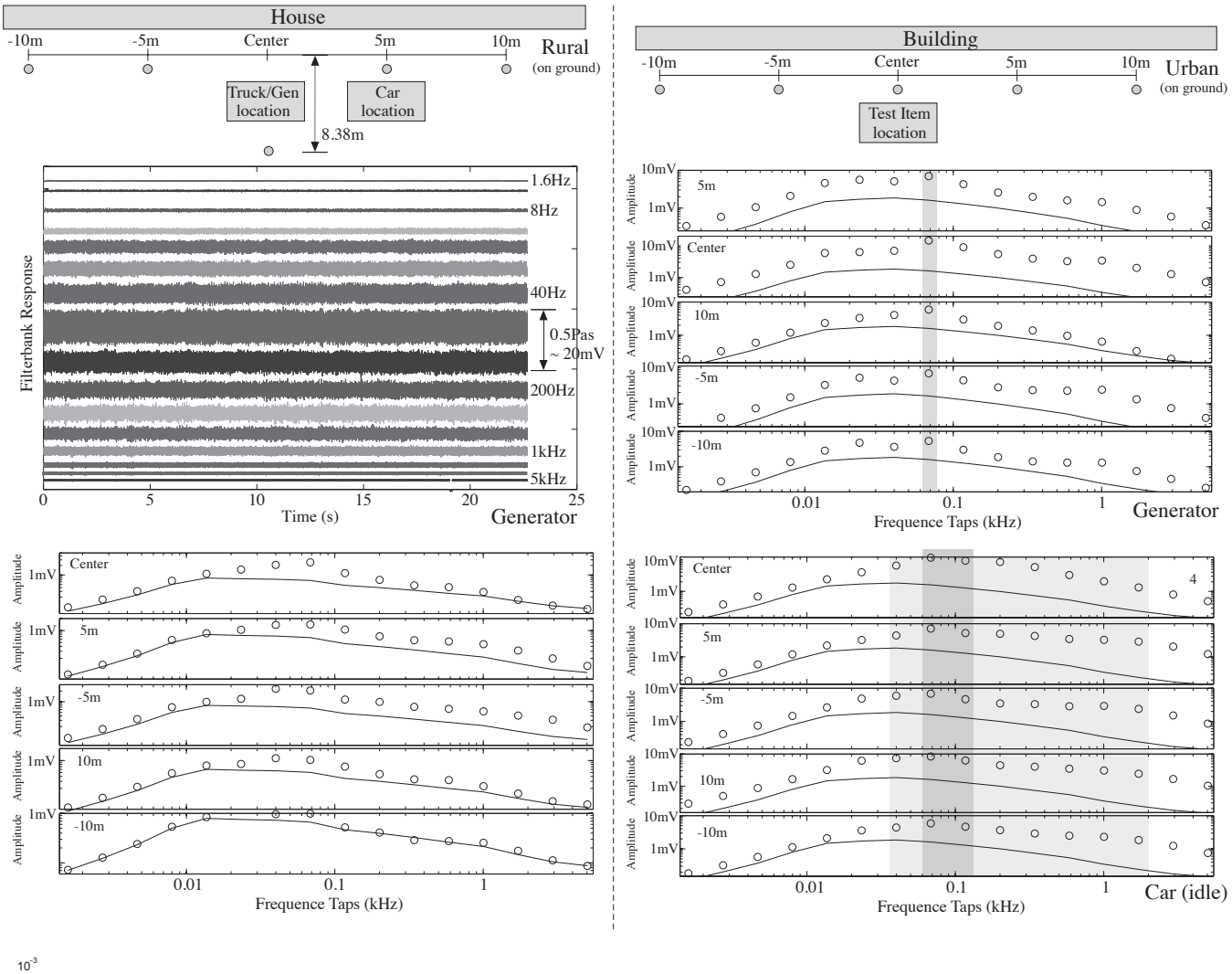


Fig. 5. Analysis for an acoustic field test data developed by DARPA for acoustic classification in both for rural and urban condition datasets. We summarize the experimental conditions for the acoustic measurements for both the rural (dirt) as well as urban (concrete) situations. We are showing sample curves from the datasets we analyzed using circuit modeled MATLAB data set (similar to what is used in our Scilab design tools), including filterbank output from the rural generator, as well as input classifier results from a rural truck data set, an urban generator set, and urban idle car dataset. Solid lines represent our extracted background levels obtained from quiet datasets provided, showing significant signals for classification as well as a sense of how to pattern signals for training and recognition. The generator seems to have very strong signals just below 100Hz with significant signals around it. For the idle car data, we see far more higher frequency signals to distinguish its behavior. For rural signals, we see a stronger single peak frequency (or small group of frequencies) due to the higher attenuation through a dirt (rather than concrete) environment.

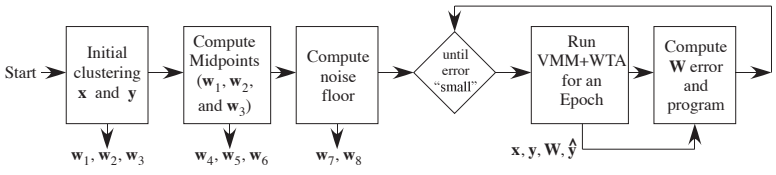
where the resulting output classifier has multiple outputs that are uninteresting for further stages, which we will identify as null outputs. Figure 7 presents the expected centers, nulls, and decision boundaries for this problem in two dimensions. These outputs, even for labeled training sets, would not have any identification of these resulting output, and we would want to fill the resulting space. Figure 6 gives the block diagram for the weight update algorithm, including the on-chip metric calculation and weight update per epoch. This discussion assumes a labeled dataset for the desired features (only) indicating the duration that they are correct.

The following subsections will demonstrate the resulting VMM+WTA learning classifier. The test input will be 110s of data from the dataset mentioned in Section III. The first subsection walks through the VMM+WTA learning algorithm to be implemented on the SoC FPAA IC. The second subsection illustrates many of the classifier behaviors through a careful MATLAB level simulation learning the referenced data set. The third subsection shows experimental data taken from the SoC FPAA IC showing on-chip preprocessing, classification, and learning using this data set.

A. VMM+WTA Learning Algorithm

The learning algorithm takes inspiration from a VQ supervised algorithm [8], [9] when the desired outputs occur, and from an unsupervised clustering algorithm [5], [6] when we have no desired output. For our training sequence, we will repeat the resulting data set, typical of training sequences; each sequence would be defined as an epoch of data; similar approach could be taken for data that is varying (and labeled) assuming the statistics are similar between each epoch. The network trains in batch mode taking data from each epoch and using it to improve the resulting weights for the classifier; this approach allows for occasional reprogramming (or in this

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.2017.2771392, IEEE Journal on Emerging and Selected Topics in Circuits and Systems



(a)

Dataset creation ("Quiet" and three urban data)
Subband Computation for VMM input
Initialize the Weights by clustering, midpoints
a: Compute VMM + WTA classifier on dataset
Compute Error Metrics
if error is not small enough, jump to **a**

(b)

Fig. 6. Details of the FPAA learning algorithm for VMM+WTA classifier. (a) Operation of the μP for the VMM+WTA classifier learning algorithm, including the initial weight clustering as well as error-driven weight improvements. (b) Pseudocode description of the VMM+WTA learning algorithm.

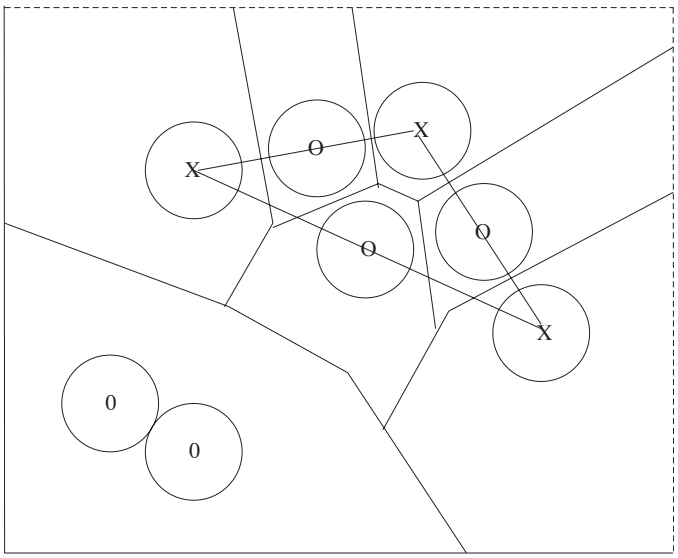


Fig. 7. Illustration in two-dimensions, of the decision boundary created by the desired targets and null signals, in this two-dimensional projected space. We illustrate the 3 classes, 3 midpoints for the 3 classes, and the background noise level nulls with an 8-output VMM+WTA classifier.

case incrementing or decrementing) the FG weight values. Figure 6a shows a block diagram of computations for the weight update procedure, both for the first data epoch, and for additional iterations until convergence is reached. Figure 6b gives a pseudo-code description for the learning algorithm.

The first epoch is handled as a special case to develop good initial conditions for the target and null weights. The target weights train on the input data stream based on the training signals, clustering each target weight based on the center position of the resulting input (within an additive constant) for the starting weight values for the desired output signals. We assume we have *m* target weight values which are set by the *m* training signals. Often GMMs start by clustering their initial data for their initial conditions for learning.

We compute as many null vectors as possible after computing the starting target weight values using the starting position for the null vectors as the midpoint between all target weights. The total number of null vectors is (m/2) (m-1). For problems where we have no signal and just baseline background *noise*, we want to have at least one null at these levels.

In cases with more midpoints then possible nulls, one computes the distance between centers and uses the centers farthest away from each other. Sometimes centers are close to each other; in a case with four centers in a square in 2 dimensional space, one has two nearly identical / redundant midpoints in the center. After computing these weights, one runs the input signal into the resulting classifier; often, the weight initialization puts weights near the desired target therefore we adapt from a good initial condition.

For remaining epochs until we see negligible change in the network error, which can be generalized to a wide set of measures used in network training, we modify the resulting weights based on the resulting errors in the solution. The algorithm for target values will follows a typical VQ learning with weight decay terms following circuit dynamics for finding centers (e.g. [13], [22]). We want a cluster of nulls to fill the space given the total number of nulls available; null outputs are unsupervised adaptation. Nulls should not go to an unbounded W value (which would swamp out anything else); need similar strength of target and null blocks. The nulls result in different σ for each weight vector in a typical GMM type calculation.

The measured errors result in a modification of the clustering algorithm in directions to correct the resulting errors; we iterate our batch learning for the next step in $\mathbf{W}$ per epoch resulting in changes from the original clustering that hopefully gives a good initial starting condition for the classifier. The amount of change in the weights should be proportional to the percentage of errors in the presented data epoch. If 2% of a data epoch have training errors, the weight correction should be proportional to this 2% correction. We expect to see errors in the following cases:

- Target a should have been Target b. Add that signal as a clustering term into Target b weights, and subtract that signal as a clustering term into Target a weights.
- Null c should have been Target b. Add that signal as a clustering term into Target b weights, and subtract that signal as a clustering term into Null c weights. We assume that the two coordinates are close in multidimensional space compared to other coordinates.
- Target b should not have been activated (false positive). This case requires negatively clustering the term into Target b weights.

All of these terms are summed up, proportional to the length of the input sequence, and then modify the resulting VMM weight. Only the changes are needed to be computed.

B. VMM+WTA Simulation Results

We start by describing the resulting algorithm and show results from a simulation model for a realistic situation. Then we continue by describing the resulting FPAA (RASP 3.0) on-chip implementation of the learning network for a 12input,

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.2017.2771392, IEEE Journal on Emerging and Selected Topics in Circuits and Systems

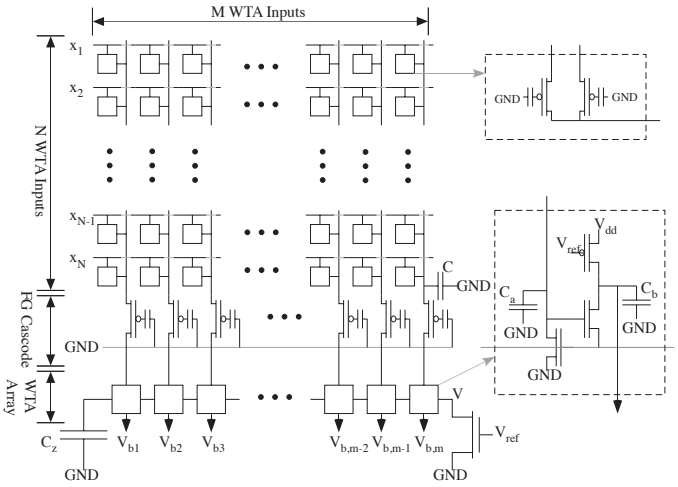


Fig. 8. Block diagram and resulting circuits for the VMM + WTA combined block, typical of what we would compile for an FPAA device. Some of the circuits could be modified (like the WTA components) depending on the particular availability of components.

8output VMM+WTA block trained on learning to identify the three different mechanical systems being activated along with 5 nulls symbols (3 midpoints and 2 background noise level) in the resulting space. We first process the bandpass filter, amplitude detection, LPF, and then subsample the resulting signals to a 500Hz sample rate from the original 44.1kHz sample rate, reducing the amount of computation going forward.

Initial investigations on the VMM+WTA learning classifier started with computer modeled simulations, which is a typical approach for early adopters to these techniques. Figure 8 show the circuit block for the one-layer VMM + WTA classifier block for N inputs and M outputs. The VMM+WTA modeling is built as one equation for the VMM modeling, and three vectorized equations for the WTA modeling.

The VMM voltage output to current modeling follows the original framework shown for the VMMs of adaptive filters elsewhere [26]. A VMM is differential input signal and single-ended output current, with each output current term allowing for four-quadrant multiplication due to the output bias current (which is dependent on the programmed the weights). The initial question is what is the resulting input range and voltage input representation

$$V_{in}^{+} = V_{bias} + xV_a,\ V_{in}^{-} = V_{bias} - xV_a \tag{11}$$

where the input, x, is represented between 1 and -1. The resulting output current for a single node would be

$$I_{out} = I_{bias}\left(W^{+}e^{xV_a/U_T} + W^{-}e^{-xV_a/U_T}\right) \tag{12}$$

exponential is U_T because we are biasing the source voltage; we define $a = V_a/U_T$, where typically, $a < 1$, setting the input range of the input signals. Weight per stage is defined

$$W^{+} = 1 + (w/2), \text{and } W^{-} = 1 - (w/2); \tag{13}$$

where having baseline of 1 is always possible by defining I_{bias} for a particular circuit. The output current gets summed with other elements to get the resulting case for row k as

$$I_k = I_{bias}\sum_{l}^{N} 2\cosh(ax_l) + w_{k,l}\sinh(ax_l),$$

$$\mathbf{I} \approx I_{bias}\left(2N(1 + a^2\mathbf{x}^T\mathbf{x}) + \mathbf{W}\sinh(a\mathbf{x})\right), \tag{14}$$

where $\mathbf{W}$ is a weight matrix, and $\mathbf{I}$, and $\mathbf{x}$ are column vectors. Analyzing around the cascade voltage sets the first-state variable, by define $\tau_1 = CU_T/(NI_{bias})$, and normalize the voltage level by U_T to give

$$\tau_1\frac{d\mathbf{u}}{dt} + \mathbf{y} = 2(1 + a^2\mathbf{x}^T\mathbf{x}) + \mathbf{W}\sinh(a\mathbf{x})/N.$$

The VMM computation can be modeled through the nonlinear, vector multiplication operation followed by a first-order LPF.

The differential equations for the WTA current conveyer block finishes the classifier model. Setting κ all equal to make a more tractable form and simplifies the numerics. The first set of ODEs for the WTA system state variables for position k ($V_{a,k}$) as well as the equation of the middle node, V_z, as

$$\frac{C_a}{I_{s01}}\frac{dV_{a,k}}{dt} = u_k - e^{\Delta V_z/U_T}\left(1 - re^{-\Delta V_{a,k}/U_T}\right)$$

$$\frac{C_z}{I_{so3}}\frac{dV_z}{dt} = e^{-\Delta V/U_T}\sum_{l=1}^{M} e^{\Delta V_{a,l}/U_T} - 1 \tag{15}$$

where I_{s01} is equal to N I_{bias}, V_{max} is the highest voltage for V_b values after the DC shift, I_{so3} is the bias current for the constraining current source, and $r = e^{-V_{a0,k}/U_T}$ is a constant relating a coefficient related to biasing values around their steady-state point. To build a polynomial model for computation, as well as analysis, we define

$$A_k = e^{\Delta V_{a,k}/U_T}, Z = e^{\Delta V_z/U_T}, \tau_A = \frac{C_aU_T}{I_{s01}}, \tau_Z = \frac{C_zU_T}{I_{s03}}$$

resulting in the equation set

$$\tau_A\frac{dA_k}{dt} = y_kA_k - Z(A_k - r), \tau_Z\frac{dZ}{dt} = \sum_{l=1}^{N} A_k - Z, \tag{16}$$

We will assume the output termination has negligible dynamics, where we can compare the resulting output current (value for A_k) against a threshold ($\text{sign}\left(A_k/(\sum_{l=1}^{N} A_k) - A_{th}\right)$) , where the threshold sets the number of k maximum winners for this k-WTA circuit (a *1* is for a winning node, and a *0* is for non-winning nodes).

The VMM+WTA classifier is trained on the 110s acoustic dataset where a sound source of $approx$ 1s would turn on a particular time. The training should learn a classifier to correctly classify these sounds. Figure 9 shows the results from these closely modeled physical IC system. The network measurements are shown after the training converged in five iterations. The resulting simulation showed all classes were correctly classified. The numerical simulations with the 500Hz sampling rate was just at the edge of numerical stability, due to some internal high magnitude derivates, illustrating the trade-off comparison between the digital and analog computation of these structures.

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.2017.2771392, IEEE Journal on Emerging and Selected Topics in Circuits and Systems

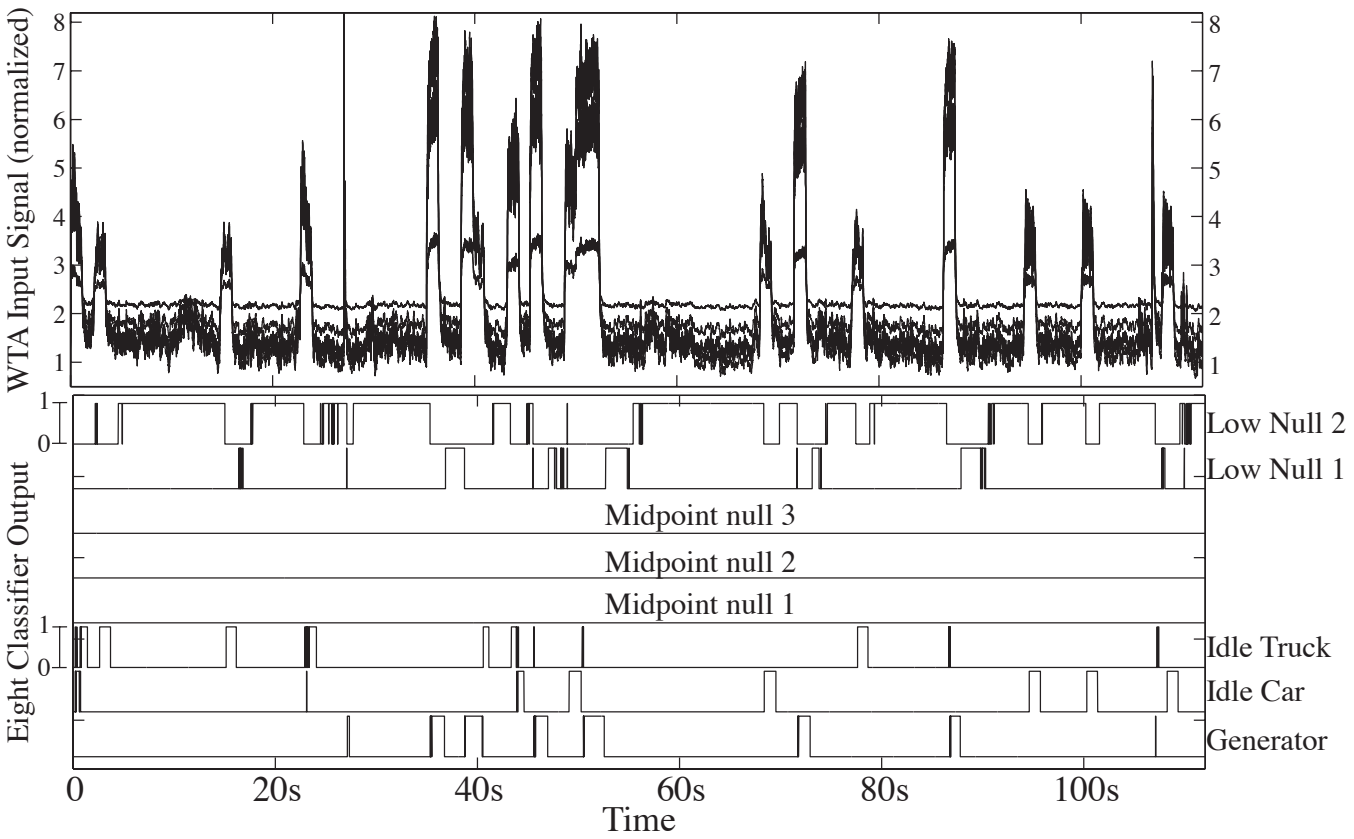


Fig. 9. Realistic simulation of the auditory classification pathway with 12 filter bank inputs and amplitude detectors, and a 12 x 8 VMM + WTA classifier block for measured test range data. The output measurement shows the output of the VMM, the computed metrics that are input into the WTA, and the raw output of the WTA block. We built a random dataset by having one of a generator, idle car, or idle truck sound turn on for a short interval and classify the resulting signal. Our classification accuracy was greater than 95%, where all of the errors are in predictive timing and issues in the ODE software modeling. The accuracy assuming a minimum of 30ms timing, as dictated by the front-end structure (simple post processing) is 100%.

C. VMM+WTA Experimental FPAA Measurements

Figure 10a shows the high-level hardware level FPAA implementation. The input data waveform was input using an analog discovery (14bit, upto 100MSPS) [46], processed through the initial subtending classifier, and then input into the VMM+WTA classifier. Hardware demonstration illustrates the close agreement between theory, matlab simulation. Figure 10a shows the computation for the weight values (first iteration) and the weight updates are computed through the digital processor; the input signals come from multiplexed compiled ADC and the target signals (digital) come directly into the processor. We expect more optimal future architectures, although this architecture is sufficient to demonstrate the hardware capability, as well as the mixed-signal FPAA signal processing. The computation of the weight updates for 8-bit signal ADC that are accumulated, based on target signals, as required for training, in multiple memory registers, transformed at the end of the epoch into the resulting LSB changes for the 14bit target value for the current, and therefore weight, measurement. Figure 10c shows the VMM circuit topology used for this classifier structure.

Figure 11b shows on-chip FPAA experimental measurement for the learning and training of these networks. The input dataset was to classified the combination of generators, idle cars, and trucks in this environment. All learning and classification occurred on this dataset passing through the same frequency decomposition stage as previously described. The approach shows a sensor-to-classified signal processing chain. Figure 11b shows the measured results of a single epoch after data set training converged. The weights in Fig. 11b were the trained weights for on-chip learning of this computation. Two nulls are used (3-input and 3-three output single WTA device), both starting around at the noise level of the classification after the first epoch. Assuming a minimum time for any symbol of 40ms, the classifier correctly recognized every input correctly with no errors.

We can add a constant to all the weights; therefore we add the most negative weight change for all 8 weight vectors to get a small, but positive increment. This approach works because the number of learning iterations are small. The network initialization starts with current levels small enough (e.g. 10nA), within a factor of 4-8 of the maximum value in current. All programming steps are a small increasing step to every possible weight, easily utilizing the on-chip, μP controlled programming infrastructure [23]. while still significant enough to allow incremental injection.

V. Summary and Discussion about the VMM+WTA Learning Classifier

We presented a learning algorithm for a VMM + WTA classifier one layer architecture, an architecture known to be a universal approximator. The approach starts by understanding

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.2017.2771392, IEEE Journal on Emerging and Selected Topics in Circuits and Systems

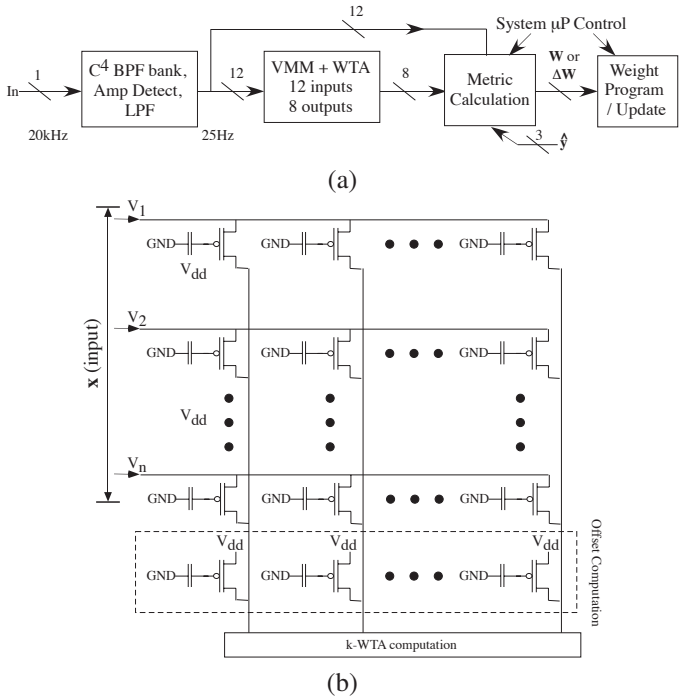


Fig. 10. Details of the FPAA learning algorithm for VMM+WTA classifier and its implementation on the SoC FPAA. (a) Block diagram similar to the tool level description of the SoC FPAA learning algorithm. (b) Circuits for the VMM circuit. Add current offset for the resulting offset; can have a constant, so we are only adding positive quantities.

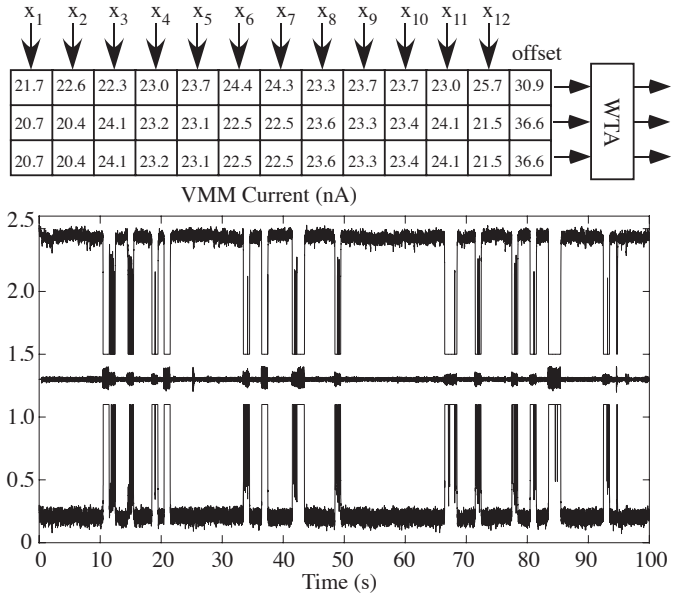


Fig. 11. VMM+WTA classification of the acoustic classification task of a series of 1s data inputs, identifying the presence of a sound source, whether it be a generator, truck, or car. The classifier used a 12x3 VMM classifier followed by a 3input, 3 output WTA. On-chip classification and training used the on-chip 12-band frequency decomposition. The experimental data measurements were offset to show the input signal, the WTA output (top vector), and one WTA null (third vector) on the same plot. Assuming a minimum time for any symbol of 40ms, the classifier correctly recognized the results every time.

fundamental equivalent transformations for VMM + WTA classifier networks; in particular, showing that a VMM + WTA structure can exactly compute a SOM or VQ operation, in addition to other transformations, enables utilizing SOM, VQ, and GMM learning approaches for this single one-layer network. The results are shown both as numerical simulation of the VMM + WTA learning network, illustrating some numerical ODE simulation limitations for this problem, as well as experimental measurements implemented on an SoC FPAA device [3]. This discussion focuses on classifiers with analog inputs looking for symbolic outputs; issues around spiking input networks requires different conceptual frameworks (for example, see [24]).

A. Placing VMM + WTA learning classifier with other Batch Learning Approaches

Developing the learning for the VMM + WTA is a combination of learning of adaptive filters and neural networks in custom hardware [25], [26], [41], as well as learning for SOM and VQ algorithms. A clustering step determines the initial weight set for ideal target and background values; the starting point for the remaining null symbols are set from midpoints of the target values. The learning approach follows one aspect of a traditional hardware learning seen for weight-perturbation type algorithms [30], [31], [32], [33], [34]. in running an epoch of the dataset after each change of weights. These approaches differ in using he 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 [35], although different training and network structure. Further, previous approaches mostly showed training for an XOR problem or a few phase shifted sine waves; our approach trained in a DARPA dataset modified to identify these acoustic symbols present for a short time ($\approx 1s$).

B. 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 MAC unit operating near the energy efficiency wall [19] 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 [28]. Implementation on an embedded processor, say that requires 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[29], 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 [3], [44], [45] as well as for this computation.

One might ask what the maximum problem size capable in 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

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.2017.2771392, IEEE Journal on Emerging and Selected Topics in Circuits and Systems

/ 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. Future SoC FPAA devices will likely enable an order of magnitude more classification with specialized WTA nodes directly compiled into the CABs. Further, future SoC FPAA devices built in advanced technology nodes [27] would expect a quadratic increase in capability (synapse, WTA outputs) with process node (roughly $10\times \rightarrow$ 130nm, and roughly $100\times \rightarrow$ 40nm). There would be significant differences (improvements) for data from scanned two-dimensional sensors (e.g. imagers) that are beyond the scope of this paper

C. Energy-Efficient One-Layer Classifier Discussions

The VMM + WTA classifier enables a one-layer training approach, not only computes an equivalent computation, enables an elegant and dense training approach, elegantly implemented in FG-based FPAA approaches. The computation and resulting learning removes the need for more complex GMM type hardware [13], [14], [36], including those cases built as part of machine learning approaches [36]. FPAA approach enables these learning algorithms to be widely available IP blocks that can be compiled on this and related family of FPAA devices.

This classifier system, compiled on this FPAA, is consistent with the improvement factor in computation (measured in equivalent multiplication and accumulate, or MAC, operations), is similar to systems developed for VMMs (custom and compiled) [37], as well as other custom classifier networks [2], [13], [14], [36]. The VMM+WTA classifier, being a universal approximator [2], can generate the same classification functions as other Radial Basis Function networks, SVN, or Guassian Mixture Model networks [13], [14], as well as related algorithms [36]. This case shows a full system for an embedded classifier structure, going from sensor input (audio) to classified sound, and further, is experimentally demonstrated in configurable analog hardware utilizing high-level design tools.

D. Remaining open VMM+WTA algorithm Questions / Opportunities

Multiple open questions still exist for the VMM + WTA algorithm, including questions of comparison of classification on a wider range of application problems, such as speech and vision, typical of classifier problems, and question of nulls in terms of this learning structure. But this approach provides further opportunities for learning systems, particularly on configurable platforms.

REFERENCES

[1] 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.

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

[3] 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 Transactions on VLSI*, vol. 24, no. 6, 2016. pp. 2253-2261.

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

[5] T. Kohonen, Teuvo (1982). "Self-Organized Formation of Topologically Correct Feature Maps," *Biological Cybernetics*, vol. 43, no.1, 1982, pp. 5969.

[6] T. Kohonen, "Self-Organization and Associative Memory", *Springer-Verlag*, 1989.

[7] P. Somervuo and T. Kohonen, *Self-Organizing Maps and Learning Vector Quantization for Feature Sequences*, Kluwer Academic Publishers, 2004, pp. 1-10.

[8] R. M. Gray, "Vector Quantization," *IEEE ASSP Magazine*, vol. 1, no. 2, pp. 429.

[9] T. Kohonen, "Learning vector quantization," *M.A. Arbib, editor, The Handbook of Brain Theory and Neural Networks, MIT Press*, 1995, pp. 537540.

[10] A. Sato and J. Tsukumo, "A Criterion for Training Reference Vectors and Improved Vector Quantization," *ICNN*, Vol. 1, 1994, pp.161-166.

[11] M. Biehl, A. Ghosh, B. Hammer, "Dynamics and Generalization Ability of LVQ Algorithms," *Journal of Machine Learning Research* vol. 8, 2007. pp. 323-360.

[12] A. Sato, and K Yamada, "Generalized Learning Vector Quantization," *NIPS 7*, pp. 423-429, 1995.

[13] P. Hasler, P. Smith, C. Duffy, C. Gordon, J. Dugger, and D. Anderson, "A floating-gate vector-quantizer," *Midwest Symposium on Circuits and Systems*, vol. 1, 2002, pp. 196199.

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

[15] A. Ben-Hur, D. Horn, H. T. Siegelmann, V. Vapnik, "Support Vector Clustering," *Journal of Machine Learning Research*, vol. 2, 2001. pp. 125-137.

[16] S. Haykin, *Neural networks - A comprehensive foundation*, 2nd ed, Prentice-Hall, 1999.

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

[18] 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.

[19] B. Marr, B. Degnan, P. Hasler, and D. Anderson, Scaling energy per operation via an asynchronous pipeline, IEEE Transactions on Very Large Scale Integration (VLSI) Systems, vol. 21, no. 1, pp. 147151, 2013.

[20] S. Shah and J. Hasler, "SoC FPAA Hardware Implementation of a VMM+WTA Embedded Learning Classifier," submitted to special issue for *JETCAS*, June 2017.

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

[22] P. Hasler, "Continuous-Time Feedback in Floating-Gate MOS Circuits," *IEEE Transactions on Circuits and Systems II*, vol. 48, no. 1, 2001.

[23] 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.

[24] J. Hasler and B. Marr, "Finding a roadmap to achieve large neuromorphic hardware systems," *Frontiers in Neuromorphic Engineering*, September 2013. pp. 1-29. doi:10.3389/fnins.2013.00118.

[25] B. Furman, J. White, and A. Abidi, "CMOS analog implementation of back propagation algorithm," 1988, pp 381.

[26] P. Hasler, and J. Dugger, "An analog floating-gate node for supervised learning," IEEE Transactions on Circuits and Systems I, vol. 52, no 5, pp. 2005. pp. 834-845.
[27] J. Hasler, S. Kim, and F. Adil, "Scaling Floating-Gate Devices Predicting Behavior for Programmable and Configurable Circuits and Systems," *Journal of Low Power Electronics Applications*, vol. 6, no. 13, 2016, pp. 1-19.
[28] J. Hasler, Energy Constraints for Building Large-Scale Neuromorphic Systems, *GOMAC*, March 2016.
[29] http://www.analog.com/media/en/technical-documentation/data-sheets/AD7457.pdf. Last visited August 31, 2017.
[30] 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.
[31] 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.
[32] 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.
[33] G. Cauwenberghs, "Neuromorphic Learning VLSI Systems: A survey" *Neuromorphic systems engineering*, Springer, 1998.
[34] 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
[35] 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.
[36] 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.
[37] 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.
[38] PSoC5 Data Sheet, Cyprus Semi, 2011.
[39] C. Schlottmann, S. Shapero, S. Nease, and P. Hasler, "A Digitally-Enhanced Reconfigurable Analog Platform for Low-Power Signal Processing," *IEEE Journal of Solid State Circuits*, September 2012, vol. 47, no. 10, pp. 2174-2184
[40] S. Tin, J. Kriz, B. Oliver1, T. Fabian1, A. Weidling1, G. Niemela2, J. Sangid2, A. Baruah2, K. Harris2, B. Blalock, J. Holleman, and V. Yantchev3, "N-Zero Integrated Analog Classifier (NINA)," *GOMAC*, 2017, pp. 35-38.
[41] A. Carusone and D.A. Johns, "Analogue adaptive filters: past and present," *IEE Proceedings Circuits Devices and Systems*, vol. 147, no. 1, 2001, pp. 82-90.
[42] M. Collins, J. Hasler, and S. George, "An Open-Source Toolset Enabling Analog–Digital–Software Codesign," invited paper *Journal of Low Power Electronics Applications*, Vol. 6, no. 1, February 2016, pp. 1-15.
[43] J. Hasler, "Opportunities in Physical Computing driven by Analog Realization," *ICRC*, October 2016.
[44] S. Shah, H. Treyin, O. T. Inan, and J. Hasler, "Reconfigurable analog classifier for knee-joint rehabilitation," *IEEE EMBC*, August 2016.
[45] 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.
[46] Analog Discovery Reference Manual, https://reference.digilentinc.com/reference/instrumentation/analog-discovery-2/reference-manual, last checked September 8, 2017.

PLACE PHOTO HERE

Jennifer Hasler is a Professor in the School of Electrical and Computer Engineering at Georgia Institute of Technology. Dr. Hasler received her M.S. and B.S.E. in Electrical Engineering from Arizona State University in 1991, and received her Ph.D. From California Institute of Technology in Computation and Neural Systems in 1997. Her current research interests include low power electronics, mixed-signal system ICs, floating-gate MOS transistors, adaptive information processing systems, "smart" interfaces for sensors, cooperative analog-digital signal processing, device physics related to submicron devices or floating-gate devices, and analog VLSI models of on-chip learning and sensory processing in neurobiology. Dr. Hasler received the NSF CAREER Award in 2001, and the ONR YIP award in 2002. Dr. Hasler received the Paul Raphorst Best Paper Award, IEEE Electron Devices Society, 1997, IEEE CICC best paper award, 2005, Best student paper award, IEEE Ultrasound Symposium, 2006, IEEE ISCAS Sensors best paper award, 2005, and best demonstration paper, ISCAS 2010.

PLACE PHOTO HERE

Sahil Shah received his B.S from Manipal University, India, and MS from Arizona State University, Tempe. He is currently pursuing a PhD in electrical engineering at Georgia Institute of Technology, Atlanta. His research interest involves design of analog and mixed signal circuits, design of integrated biosensors, and microfluidics.