# A Machine-learning Classifier Implemented in a Standard 6T SRAM Array

Jintao Zhang, Zhuo Wang, and Naveen Verma
*Princeton University, Princeton, NJ, USA*

## Abstract

This paper presents a machine-learning classifier where the computation is performed within a standard 6T SRAM array. This eliminates explicit memory operations, which otherwise pose energy/performance bottlenecks, especially for emerging algorithms (e.g., from machine learning) that result in high ratio of memory accesses. We present an algorithm and prototype IC (in 130nm CMOS), where a 128×128 SRAM array performs storage of classifier models and complete classifier computations. We demonstrate a real application, namely digit recognition from MNIST-database images. The accuracy is equal to a conventional (ideal) digital/SRAM system, yet with 113× lower energy. The approach achieves accuracy >95% with a full feature set (i.e., 28×28=784 image pixels), and 90% when reduced to 82 features (as demonstrated on the IC due to area limitations). The energy per 10-way digit classification is 633pJ at a speed of 50MHz.

## System Overview

Recent trends towards in-/near-memory computing are motivated by limitations posed by memory operations [1]. For example, with 16b words and 32kB SRAMs, SRAM-access vs. multiplication consumes 17pJ vs. 1pJ in 45nm. In the demonstrated system, computation is performed in-place by the bit cells of an SRAM, avoiding energy-intensive accesses. This faces two key challenges: (1) the constrained structure of standard 6T arrays limits the computations possible; and (2) circuit non-idealities, especially high variability in bit cells, degrades the quality of outputs. To overcome (1), we exploit an idea from machine learning called boosting. In boosting, outputs from multiple *weak classifiers* are combined (e.g., via weighted voting) to form a *strong classifier* (weak/strong classifier implies inability/ability to fit arbitrary training data). We focus on the algorithm Adaptive Boosting (AdaBoost), where weak classifiers are iteratively trained to correct the fitting errors of previous iterations. Theory shows that very weak classifiers (marginally better than 50/50 guessing) can be used [2], enabling use of the limited structures possible via SRAM bit cells. To overcome (2), we employ an algorithmic extension to AdaBoost we previously developed called Error-Adaptive Classifier Boosting (EACB) [3]. EACB exploits the actual weak-classifier implementations within the iterative training to adaptively correct not only fitting errors but also errors due to any static (possibly random) non-idealities in the implementations; analysis and experimental validation show that substantial non-idealities can be overcome [3].

Fig. 1 shows the block diagram. The system operates in two modes: SRAM Mode and Classify Mode. In SRAM Mode, the system behaves as a standard SRAM, enabling read/write of the classifier model (derived from training) into bit cells. For classification, algorithms typically utilize a comparison metric between input data and the classifier model. A common metric is the inner product. A binary linear classifier is simply based on taking the inner product between an input feature vector $\vec{x}$ and a weight vector $\vec{w}$, and then performing sign thresholding: $sign(\vec{x}\cdot\vec{w})$. In Classify Mode, the system implements a very weak form of linear classifier in every column, where the elements of $\vec{w}$ are restricted to be +/-1, according to data stored in the bit cells. First BL/BLB's are precharged. Then all the WL's are driven at once to analog voltages corresponding to the elements of $\vec{x}$ (i.e., the features); this is done via the peripheral WLDACs. Thus, each bit cell pulls a current modulated by its WL voltage from either BL or BLB, depending on its stored state. This approximates multiplication by +/-1. The bit-cell currents $I_{BC}$ from the column then add together, as in an inner product, discharging BL or BLB. Finally, a comparator provides sign thresholding of the differential signal. Below we consider circuit non-idealities, and how these are overcome either by the architecture itself or via the training algorithm.

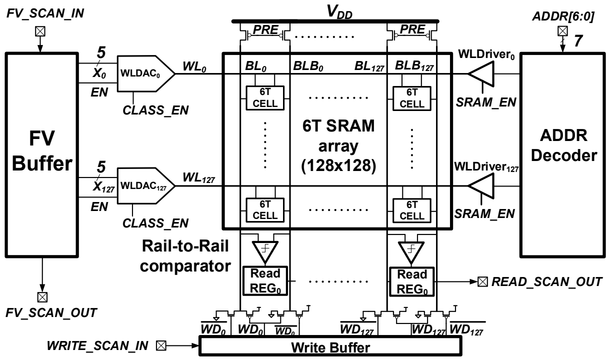


Fig. 1: System Block Diagram.

## Circuit Design

For SRAM Mode, standard circuit structures are used (precharge devices, address decoders, WL drivers, write drivers). For Classify Mode, the key circuit blocks, described below, include: WLDACs, bit cells, and comparators.

Fig. 2 shows the WLDAC circuit and simulated output (WL) waveform (note, WL drivers are disabled in Classify Mode). Digital features are provided as 5 bits *X[4:0]*, to select binary-weighted PMOS current sources; an additional PMOS current source is included to enhance the linearity of the charge drawn by a bit cell ($Q_{BC} = \int I_{BC}dt$), as described below. The current from the PMOS sources is used to bias a replica of the bit cell, consisting of a $V_{DD}$-connected driver transistor ($MD_R$) and diode-connected access transistor ($MA_R$), which drives the WL. Thus, all bit cells in the row provide a corresponding current $I_{BC}$ (i.e., scaled by the replica sizing ratio). With the WLDAC output impedance ~$1/g_{m,MAR}$, up sizing the replica easily yields the desired charging times for the WL capacitance (~120fF). Note that with all WLs driven this way, a potentially large number of bit cells drive BL/BLB at the same time (128 in the prototype). To not saturate discharge of BL/BLB capacitances, $Q_{BC}$ of each is designed to be low <10fC, amounting to WL voltages <0.4V.

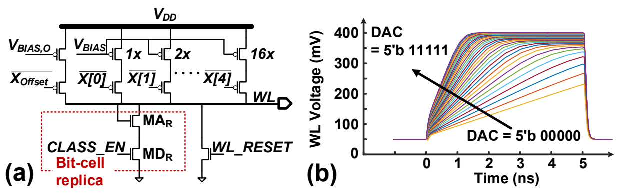


Fig. 2: (a) WLDAC circuit, and (b) WL transient voltage with different WLDAC input codes.

Bit-cell sizing is identical to standard 6T cell (for read/write margin). We note that in Classify Mode an upset condition is possible. Namely, we like to maximize the dynamic range of BL/BLB for inner-product computation. This implies that cells can be exposed to low BL/BLB voltages, as in a write condition. But, given the low WL voltages in Classify Mode, Monte Carlo simulations show that, with typical cell sizing, less stringent margin is needed than for standard read/write.

BL/BLB discharge also faces two sources of non-linearity. The first is the $Q_{BC}$ transfer function in Fig. 3a (shown with BL/BLB statically connected to $V_{DD}$). This is mitigated by (1) including a constant offset current source in the WLDAC, and (2) applying the inverse of the nominal transfer function to input features. The second is non-constant BL/BLB discharge, shown in Fig. 3b. This is due to both $I_{BC}$ reduction with reducing $V_{DS}$ across the access transistors of bit cells pulling down, and competing charging current from bit cells pulling up. However, such monotonic effects due to the discharge level do not impact the final BL/BLB comparison required for classification (we note that with finite comparator offset there will be some impact, but this can be addressed by EACB).

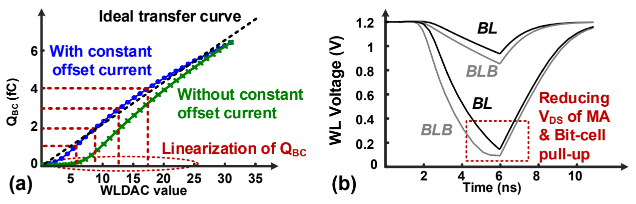


Fig. 3: (a) BL/BLB discharge ($Q_{BC}$) transfer function, and (b) non-constant bit-line discharge due to pull-down/-up current variation.

To support the large dynamic range on BL/BLB, the system must incorporate a comparator capable of rail-to-rail inputs. The circuit used is shown in Fig. 4a. A simple mechanism for offset compensation is described below to enhance the system.

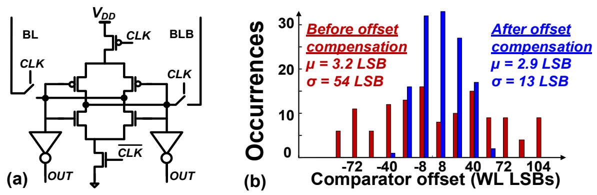


Fig. 4: (a) Rail-to-rail comparator circuit, and (b) measured offset histograms (128 comparators) before/after compensation.

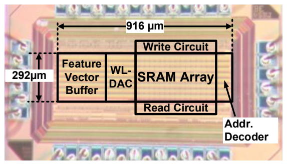
## Classifier Training

EACB (and AdaBoost) are meta-algorithms, which bias the way weak classifiers are trained at each iteration by increasing the emphasis on data instances that are incorrectly classified. However, an appropriate base training algorithm is needed for the weak classifiers themselves. For linear classifiers, L2 linear regression is often employed to train the weights $\vec{w}$, via the cost function: $\sum_i \| \vec{x_i} \cdot \vec{w} - y_i \|_2^2$ (where $x_i$ are the training feature vectors and $y_i$ are the corresponding training labels). This leads to quadratic convex optimization. However, in this system, weights can only be +/-1. Simple quantization after training would yield highly sub-optimal performance. Instead, we add an explicit optimization constraint, restricting the elements of $\vec{w}$ to be +/-1. This leads to mixed integer-programming optimization, for which fast methods exist, integrated into readily available solvers.

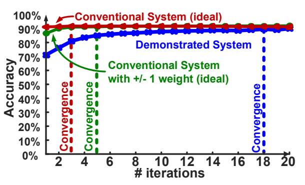
Though EACB has previously shown to overcome nonlinearities (as in the $Q_{BC}$ transfer function) and offsets (as in the comparators) [3], minimizing these can reduce the number of weak-classifier iterations needed. $Q_{BC}$ nonlinearity is compensated as described previously (Fig. 3a). Comparator offset is compensated by allocating some SRAM rows to an offset-compensation set. All rows in this set receive the same WLDAC code, resulting in the same nominal $Q_{BC}$. During offset compensation, other rows in the array are selected to discharge BL/BLB by a nominally equal amount. Preferably, various random selections of rows are employed and averaged, to mitigate bit-cell variability. Then, for all columns (done in parallel), data is written to bit cells in the offset-compensation set to equalize the probability of either comparator state (i.e., setting the trip point). This is done in a binary-search manner for rapid convergence. For the demonstration, 32 rows are designated to the offset-compensation set, with WLDAC code of 16. Fig. 4b shows the measured pre-/post-compensated offset over 128 comparators, in terms of the difference in WLDAC code for BL/BLB discharge at the trip point.

## Measurement Results and System Demonstration

The prototype is shown in Fig. 5, with a measurement summary. Though bit-cell device sizing in the 128×128 SRAM array is that conventionally used, the cell layout is larger to meet logic design rules. MNIST digit classification is demonstrated by employing 45 binary classifiers between all pairs of digits, and performing all-vs.-all (AVA) voting. Due to prototype column limitations, 128 classifiers are tested at a time for the boosting iterations required in EACB (ideally, enough SRAM columns would be implemented for all classifiers and iterations needed). Due to prototype row limitations, the 28×28 images are down-sampled to a set of 82 pixel features (this reduces classification accuracy from 95% to 90%). Using these features, the classification accuracy vs. boosting iterations is shown for: (1) an ideal conventional SRAM/digital-MAC system based on linear classifiers, requiring 10b weights (from simulations); (2) an ideal conventional SRAM/digital-adder system based on linear classifiers restricted as proposed to +/-1 weights; and (3) the demonstrated system. The energy of each system is shown for 10-way digit classification, taking into account the differences in iterations (assuming 5.3pJ for MAC, 0.42pJ for adder, and 0.12pJ per SRAM bit access, all measured or post-layout simulated). The proposed system employs 18 iterations of EACB, achieving 113× and 12× energy savings, respectively.



| CHIP SUMMARY | |
|---|---|
| Technology | 130 nm |
| SRAM Size | 128 × 128 bits |
| Bit cell Size | 1.26 μm × 3.44 μm |
| Energy / 10-way Class. | 633.4 pJ |
| Speed: | 50MHz |
| Accuracy | 90% (18 iterations) |
| Energy saving | 113 × / 12 × |
| Feature Resolution | 5b |



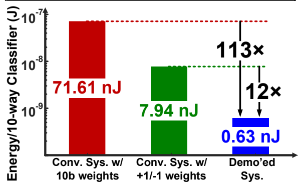


Fig. 5: Die photo and measurement summary.

[1] M. Kang, et al., *ISCAS*, pp. 2505-2508, May 2015.
[2] R. Schapire, et al., Boosting: Foundations and Algorithms, 2012.
[3] Z. Wang, et al., TCAS-I, vol.62, no.4, pp.1136-1145, April 2015.