Mixed-Signal Neural Network Branch Prediction

Owen Kirby Shahriar Mirabbasi Tor M. Aamodt
Dept. of Electrical and Computer Engineering,
University of British Columbia

8 June 2007*

Abstract

Accurate branch prediction is essential for modern microprocessors in order to maintain high instruction throughput. Neural networks have shown great promise in branch prediction. However, digital implementations require complex circuitry leading to low cycle times, which have precluded the use of these predictors in actual microprocessor designs.

This paper evaluates the potential for leveraging analog circuitry to efficiently implement neural network branch predictors. In particular, we propose to implement a perceptron branch predictor using simple analog blocks including a resistor network for the synaptic multiply/add function and a comparator for the non-linear output function. We have simulated this circuit using a C model embedded into SimpleScalar that calculates circuit voltages and currents. We have found that the prediction accuracy of the analog branch predictor is highly tolerant to resistor mismatch and noise. In the presence of realistic resistor matching and noise conditions, it achieves virtually the same accuracy as a digital implementation. The simplicity of the analog perception should enable it to operate at several GHz with lower power consumption than a digital implementation.

1 Introduction

Modern microprocessors depend on speculative execution to hide control flow dependencies and their associated latencies. High performance and high-frequency designs call for deeper pipelines,

*Manuscript last modified June 8, 2007; Posted online August 29, 2008.

which lead to costly misprediction penalties. Fast and accurate branch predictors are essential to avoiding these expensive penalties.

Some of the most promising work in branch prediction use artificial neural networks to predict branch outcomes. The use of these predictors has the potential for substantial performance gains over traditional counter- and history-based designs [11, 9, 10].

Unfortunately, digital logic is poorly suited to emulate neural networks, which require large synaptic multiply/add functions that are too complex to implement in a single cycle. Pipelining this addition is not an acceptable option either, because a delay in the branch prediction will lead to performance degradation.

The neurons and synapses, which make up our brains are not built of digital logic gates, but instead use a process of chemical transmission. The input signals to a neuron are weighted by the synapse's ability to conduct neurotransmitters. This behavior of biological neural networks is very similar to charge transfer through resistors. As branch predictors only provide microarchitecture *hints*, which are by their very nature subject to error, the use of analog circuitry, which is relatively sensitive to noise, is an interesting possibility for achieving high clock frequency at the possible expense of lost *instruction per cycle* performance.

Analog artificial neural networks have been studied and implemented before [19], including some very similar to what we propose here [5], but have not been considered for use in a high-performance branch predictor. In this paper we assess the feasibility of an analog circuit to perform the synaptic multiply/add function in the context of branch prediction. We propose a design and simulate it using a C model embedded into the SimpleScalar toolset. Because this circuit mainly consists of a resistor network and a comparator, this predictor should be capable of operating substantially faster than a purely digital implementation.

In Section 2, we discuss related work on neural network branch prediction and artificial neural networks. In Section 3, we describe the design of an analog perceptron. In Section 4, we explain our simulation methodology. We present our results in Section 5, then conclude in Section 6 with final thoughts and topics for further research.

2 Related Work

In this section we will discuss other work related to artificial neural networks and neural network branch prediction.

2.1 Perceptron Branch Predictor

The earliest proposal for using neural networks to predict branches we are aware of was by Vintan [20]. However, the prediction mechanism Vintan proposed, which was based upon an artifical neural network algrithm known as *Learning Vector Quantization* [13] achieved poorer accuracy than state-of-the-art branch predictor designs. Subsequently, Jimnénez and Lin proposed a neural network branch predictor based upon a single-layer *Perceptron* [17], which they demonstrated could achieve better prediction accuracy than prior branch prediction mechanisms [11]. The perceptron uses a neural network to compute the dot product between inputs and a trained weight vector, the result of which is passed through a nonlinear step function. The perceptron branch predictor is widely regarded as one of the most accurate branch predictors [12, 2, 7], but the summation in the dot product results in a long critical path, substantially lowering clock frequency making a straightforward *digital* implementation impractical.

2.2 Alternate Neural Network Predictors

In the search for a practical alternative to the perceptron branch predictor, Jimnénez proposed a neural path-based branch predictor which pipelines the perceptron in advance over multiple cycles to generate predictions in a single cycle [9]. The neural path-based predictor suffers decreased prediction accuracy relative to a single cycle version (it uses different information to make predictions) Further extending the concept of neural network branch prediction, Jimnénez combined the neural path-based predictor and the perceptron into a generalized *piecewise linear branch predictor* [10]. The piecewise linear predictor offers superior performance when large resource budgets are available, but still requires complex logic that may result in long critial paths and lower clock frequencies. We believe the technique we propose in this paper could be used to improve the circuit timing of piecewise linear branch prediction however an exploration of this possibility is beyond the scope of this work.

2.3 Artificial Neural Networks

Neural network based branch predictors are relatively new to high-performance computing, but artificial neural networks have been implemented and studied for years. Examples of artificial neural networks can be found from as far back as the 1950's [15] when the simple single-layer network perceptron model was first proposed [17, 18, 1]. Artificial neural networks using analog

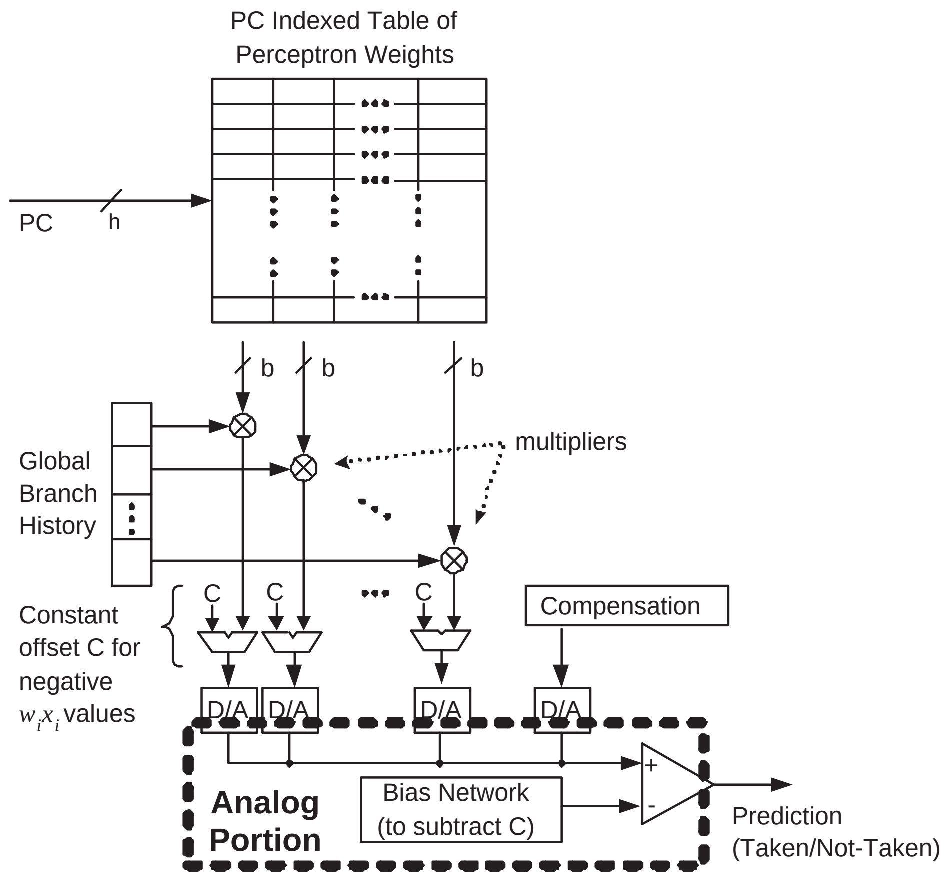
PC Indexed Table of Perceptron Weights
PC
h
b
b
b
Global Branch History
multipliers
Constant offset C for negative $w_i x_i$ values
C
C
C
Compensation
D/A
D/A
D/A
D/A
Analog Portion
Bias Network (to subtract C)
+
-
Prediction (Taken/Not-Taken)

Figure 1: The Mixed-Signal Perceptron Branch Predictor

electronics were a topic of a significant amount of research in the 1980's and early 1990's [19], but the pace of research in this field has slowed in recent years. In 2001, Camboni and Valle proposed a VLSI analog perceptron cell using current multipliers [5]. Their test circuit was implemented in 0.6 μm CMOS and was able to reach 125MHz with a small number of inputs. An analog neural network implementation in VLSI was demonstrated by Tarassenko [19]. Haikonen describes an analog artificial neural networks implementing a weighted sum function [8].

2.4 FPGA Perceptrons

Recently, Cadenas et al. implemented a perceptron branch predictor on a *field programmable gate array* (FPGA) [4, 3]. They suggest an interesting approach to reducing the critical path length of the overall prediction logic by grouping weights into blocks resulting in a cycle time improvement by a factor of 1.7 on an FPGA implementation [3].

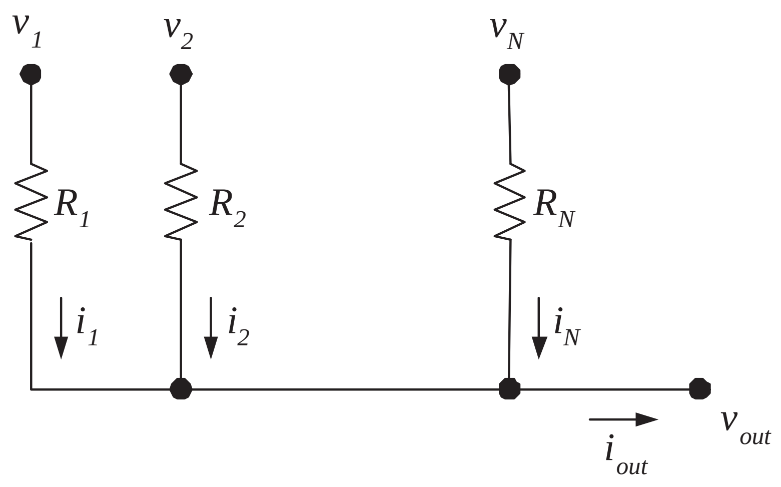

Figure 2: Weighted Average Resistor Network

3 Mixed-Signal Perceptron Design

Figure 1 shows a high-level view of the mixed-signal perceptron predictor. The details are described in the remainder of this section along with comments on the many design issues we considered.

Artificial neural networks typically implement a "synaptic" multipy/add function to weight and sum the input signals. This operation can be thought of as a vector dot-product operation between a vector of input signals and a vector of weights. The forumla for this multiply/add function is:

$$y = \sum w_i x_i \tag{1}$$

In the context of the perceptron branch predictor [11, 12], the w_i are weights trained based upon the past history of branch outcomes, and the x_i are the recent history of branch outcomes encoded as -1 for not-taken branches, and 1 for taken branches. The perceptron branch predictor algorithm predicts taken when y is greater than zero, and not-taken otherwise.

Each pair of terms can be multiplied together in parallel, performing the summation requires a large number of dependent additions which increase with the number of terms. While a tree structured network would have a critical path that grows as the logarithm of the number of terms, it is important to recognize that one of the strengths of the perceptron predictor is its ability to make effective use of long global branch histories (implying many terms). However, if we shift focus from the digital domain to the analog domain, we can effectively implement the addition with a resistor network assuming signals are voltage mode. If signals were current mode, then addition is accomplished by simply joining signal branches together as per *Kirchhoff's Current Law* (KCL). In this paper we focus on the prior approach.

Since the x_i take on values of {-1,1}, the multiplication in Equation (1), simply requires inverting the value of w_i when x_i is -1. The circuit in Figure 1 implements this by forming the 1's complement of the weight vectors using XOR gates. To implement the summation the circuit in Figure 1 computes the average of $w_i x_i$ in the analog domain. Note that since the average is over a fixed number of terms there is no information gained by multiplying the average by the (constant) number of terms to recover the actual sum.

3.1 Analog Summation

To understand how the analog circuit in Figure 1 implements the summation, consider the resistor network shown in Figure 2. Note that this circuit is a bit more general than we require as it computes a *weighted* average of the voltages v_i. The output voltage, v_{out}, is the average of the input voltages weighted by the conductance of their resistors ($1/R_n = G_n$).

$$i_{out} = \sum i_n = \sum \frac{(v_n - v_{out})}{R_n} = 0 \tag{2}$$

$$\therefore v_{out} = \frac{\sum G_n v_n}{\sum G_n} \tag{3}$$

If we take the resistor values in this diagram to correspond to the fixed output impediance of the Digital to Analog (D/A) converters in Figure 1 that in our design do *not* contain output buffers, then we can use this circuit to compute the required summation. Assuming all D/A converters in Figure 1 have the same output impedance, R, then setting $G_n = 1/R_n = 1/R$ in Equation (3), yields:

$$v_{out} = \frac{\sum(\frac{1}{R}v_n)}{\sum(\frac{1}{R})} = \frac{\frac{1}{R}\sum v_n}{\frac{N}{R}} = \frac{\sum v_n}{N} \tag{4}$$

Where N is the number of inputs. A prediction is then made by comparing v_{out} to a reference voltage. Normally, a perceptron performs decisions by checking the sign of the final sum. However, a typical D/A converter cannot produce a negative voltage. To effect negative values we simply add a reference value (labeled `C` in Figure 1) to the signed value to create an unsigned quantity and input it to the D/A. The sign of the sum can then be checked by comparing the output voltage to a reference voltage corresponding to our original offset.

3.2 Offset Compensation

No analog circuit can ever be fabricated perfectly, factors such as resistor tolerances, temperature dependence, and leakage can alter the reference reference voltage. Because the reference voltage is used to make predictions, errors in the reference voltage can bias the predictions, and in extreme cases will render the perceptron useless.

To compensate for errors in the bias network, we add a compensator circuit, as shown in Figure 1. The compensator circuit consists of an additional D/A converter and a compensator register. The compensator register continually adjusts itself to try and correct for any errors in the reference voltage.

The compensator circuit checks the reference voltage by inputting a test pattern into the perceptron and generating a prediction. If the prediction is incorrect the compensation register is adjusted accordingly. Because the compensator circuit is essentially a form of feedback control, the compensator circuit should be also able to compensate for time-varying changes in the resistor networks (e.g., temperature dependency of resistors).

3.3 Analog Designs Issues

Unlike digital circuitry, issues such as resistor matching, temperature dependence, noise, leakage current, and circuit dynamics can distort or corrupt an analog signal. We will attempt to quantify the effects of these issues on the accuracy of the mixed-signal perceptron branch predictor.

Not all resistors can be manufactured with perfect precision. Natural variations in the silicon and lithography process, among other factors, can create variations in expected resistor values. VLSI resistors are typically manufactured using a thin film of doped polysilicon. Using this technology, absolute resistances typically have tolerances of 20%, or worse, but relative tolerances can be 2% or better [14, 16, 6]. Fortunately, the accuracy of D/A converters depends on the relative matching of its resistors and not the absolute accuracy.

The Temperature Coefficient of Resistance (TCR) for doped silicon is between -200 ppm/C and -700 ppm/C. Assuming a maximum T of 40C, this corresponds to at most a 2% change in resistor matching [14, 6]. We will not simulate temperature depenancies in this paper, but their effect should be similar to resistor mismatching.

There are two soures of noise in analog electronics: thermal noise, and interference noise. Thermal, or inherent, noise is a result of thermal agitation of charge carriers in the resistor. The magnitude of thermal noise depends on the capacitance present in this circuit, and is independant of it's resistance. Equation 5 relates the variance and magnitude of thermal noise to the circuit capacitance. Inherent noise is unwanted signals coupled into the circuit from nearby noise sources.

$$\sigma^2 = V_{rms}^2 = \frac{kT}{C} \tag{5}$$

We found that the capacitance of our circuit is dominated by the input capacitance of the comparator, and we estimate that a high-speed comparator would have an input capacitance of approximately 200 fF, adding interference noise, the overall amount of nose in the circuit is given by Equation 6, where $V_{interference}$ is the RMS value of the interference noise.

$$\sigma^2 = \sum V_{rms}^2 = \frac{kT}{C} + V_{interference}^2 \tag{6}$$

3.4 Perceptron Speed

Because the analog perceptron is merely a large resistor network, the speed of operation is based off of the RC delay. Based on 0.18 μm technology, we found that the parasitic capacitances are negligable compared to the input capacitance of the comparator.

We assume an operating frequecy of 4GHz is possible and size our resistors for five τ settling time of 250 ps. Based upon our noise measurements, we expect this will be sufficient settling time to provide accurate branch predictor operation.

4 Simulation

In this section we describe the methodology used to simulate the mixded-signal perceptron branch predictor.

The analog circuitry is simulated using a C model embedded int the SimpleScalar toolset, and computes the voltages at every node on a cycle-by cycle basis. Noise is generated using a normal distribution based off of expected circuit parameters, and injected into the mixed-signal perceptron. Using this model, we are able to determine prediction accuracy, power consumption, and noise levels which are highly dependant on the inputs to the circuit, and could not be simulated with a traditional circuit simulator.

2Bc-gshare	6kB gshare predictor table. 8kB 2-bit counter table. 8kB 2-bit meta predictor.
Digital Perceptron	1024 table entires. 24-bits global history. 8-bits local history. 8-bit weights.
Static Predictor	Always Taken.
Analog Perceptron	1024 table entries. 24-bits global history. 8-bits local history. 8-bit weights. 8-bit compensation register.

Table 1: Simulated Branch Predictors

Using this C model to simulate the mixed-signal perceptron, we tested the circuit using 10 of the SPECint2000 bechmarks compiled for the PISA instruction set and run on the SimpleScalar branch predictor simulator, sim-bpred. Table 1 describes the branch predictors simulated.

The impact of resistor matching was tested by running the simulator across a wide range of resistor matching values and recording the prediction accuracy. Every test point was simulated 25 times to generate an average result and calculate yeild estimates.

Noise is generated from two sources: thermal, or inherent, noise is a product of the resistors and other elements which make up the circuit, while interference noise is coupled into the circuit from surrounding noise sources. The amount of thermal noise in an R-C circuit is a product of the capacitance, and is independent of the resistance. Equation 5 relates the variance of the thermal noise to the temperature and capacitance of the circuit. We found that the capacitance of the comparator dominates the circuit parameters, and we estimate that a high-speed comparator would have an input capacitance of approximately 200 fF.

Noise is simulated by injecting gaussian white noise into the comparator. The amplitude of this noise model is derived from Equation 5, and includes both thermal noise and interference noise coupled in from other circuits. Random resistors are generated using a normal distribution with 1-σ tolerances [1].

[1]Typically stated tolerances are 3-σ values, meaning the tolerance value is equal to 3 standard deviations. We choose 1-σ values as a worst-case parameter.

5 Results

In this section we describe the performance of the analog perceptron branch predictor when faced with a variety of noise and tolerance parameters.

The digital perceptron and static predictor are included for comparison as best and worse-case performances of the analog perceptron and the 2Bc-gshare predictor is included as a 'competing design.'

5.1 Resistor Tolerance

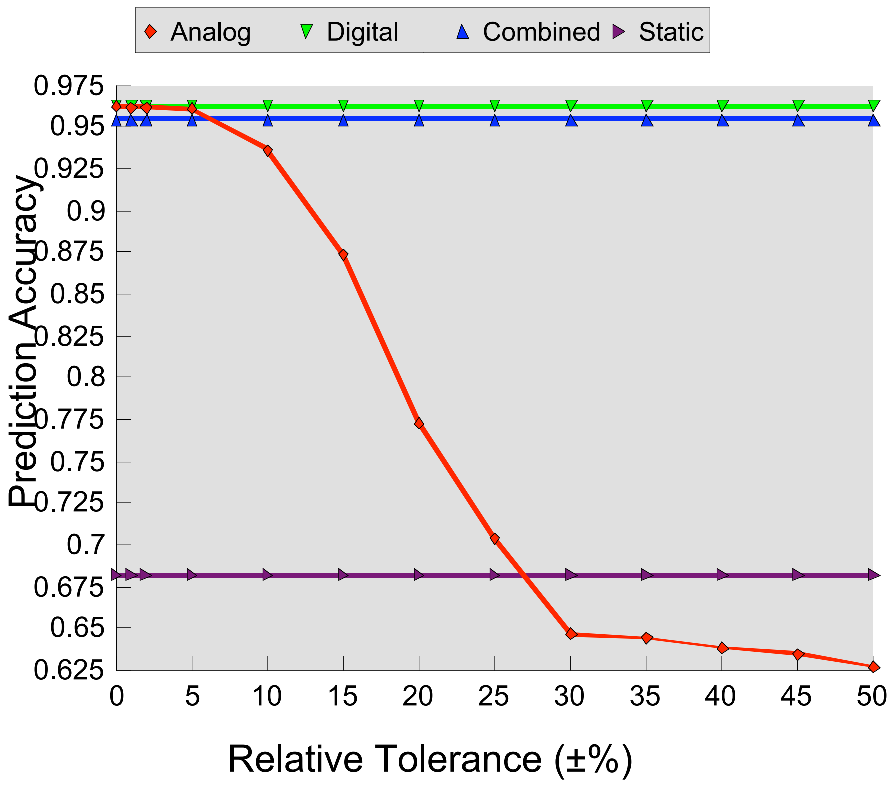

Figure 3: Robustness of Prediction Accuracy to Resistor Matching (Average)

Figure 3 plots the arithmetic mean prediction accuracy against resistor matching over all the simulated benchmarks. The horizontal lines plot the baseline predictors. With resistor matching as bad as 5% there is no noticeable difference between the analog and digital perceptrons. At 10% matching accuracy drops below the 2Bc-gshare predictor, and performance becomes unacceptable at 15%.

Figure 4 shows the same same data as Figure 5, but for individual benchmarks. The same

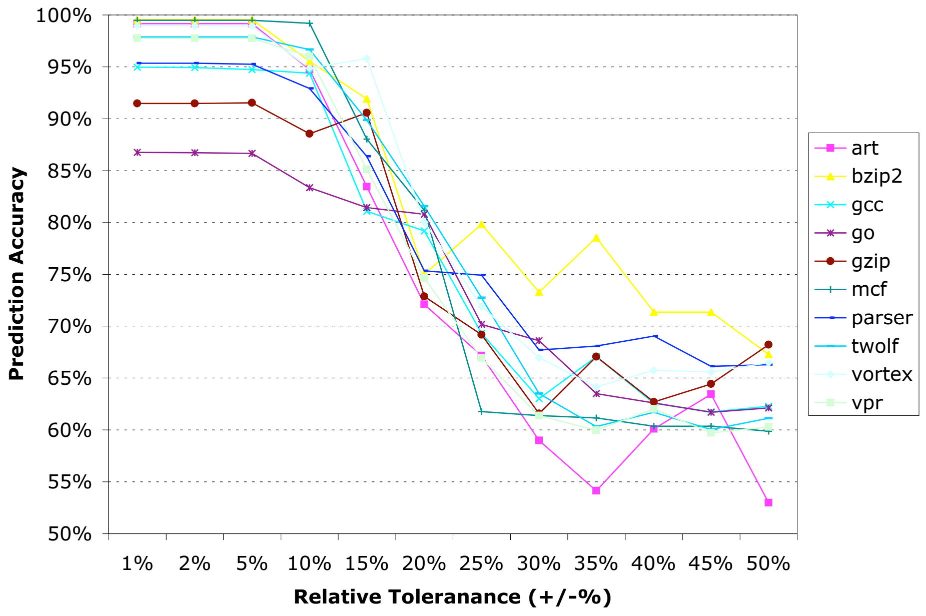

Figure 4: Robustness of Prediction Accuracy to Resistor Matching (per Benchmark)

trend can be seen in each of the benchmarks: there is no noticeable performance loss for resistor matching of up to 5%. Performance falls off sharply at 10-15%, and quickly becomes unacceptable.

With mismatched resistors it is possible that some chips may be produced with errors too great for the branch predictor to operate correctly. To quantify this effect we define the yield as the number of simulation runs which performed better than the baseline 2Bc-gshare predictor. In most benchmarks this means only those chips performing extremely close to the ideal perceptron will pass.

Figure 5 plots the yield vs. resistor matching. For expected matching (2-5%), the yield is over 98%. Even at 15% matching where the average performance was very poor, the yield still is over 50%. Despite terrible average accuracy, the majority of chips still perform nearly identical to the ideal perceptron.

5.2 Noise Sensitivity

Figure 6 shows the arithmetic mean prediction accuracy vs. injected noise power measured in dBm for individual benchmarks. Prediction accuracies fall as noise power is increased. For even most intolerant benchmarks, noise levels less than 15dBm have no noticeable effect on prediction accuracy.

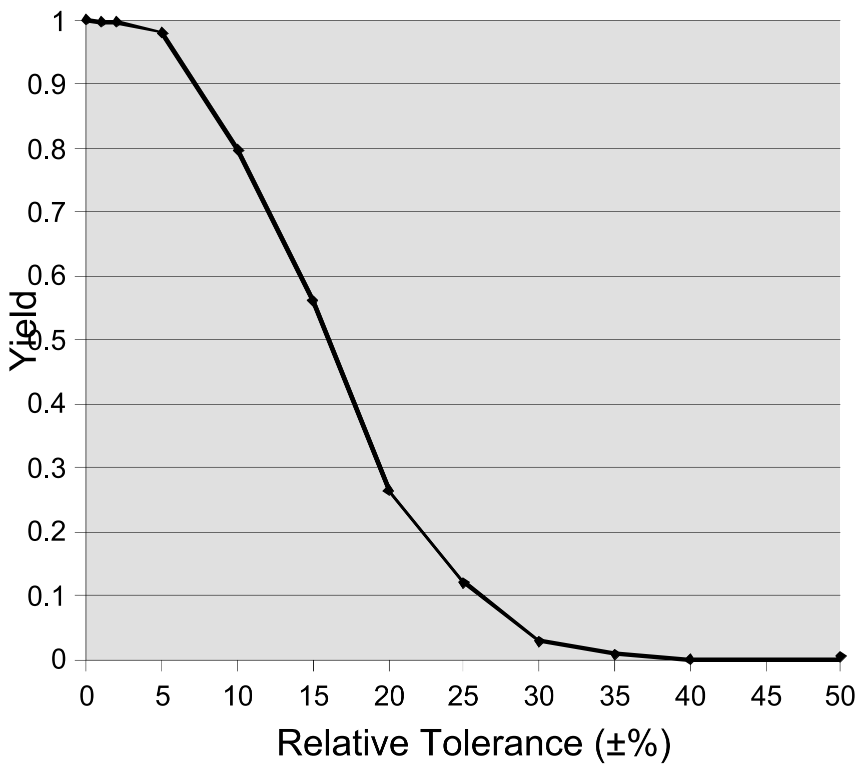

Figure 5: Yield versus Resistor Matching Tolerance

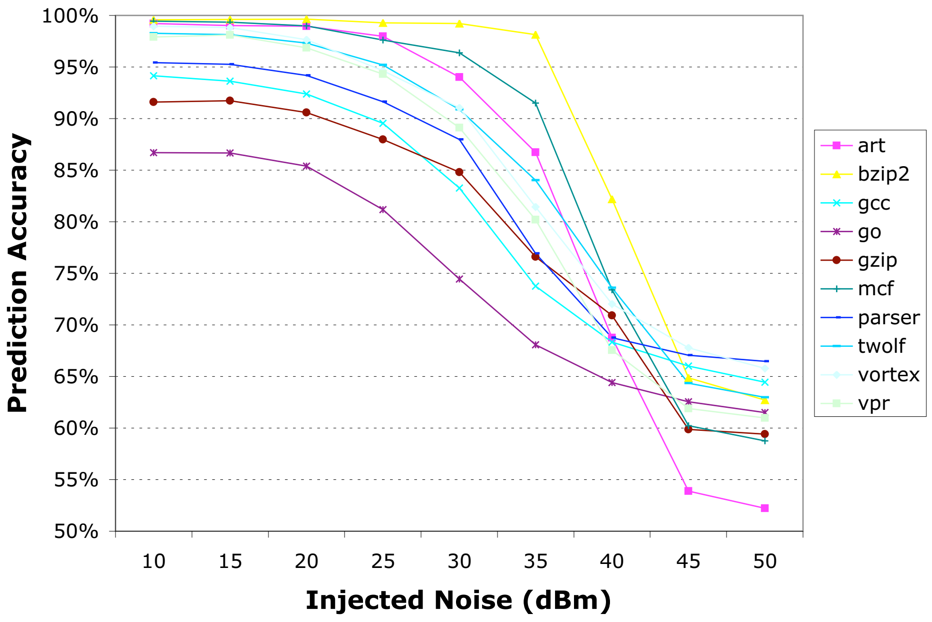

Figure 6: Prediction Accuracy vs. Signal-to-Noise Ratio.

5.3 Power

Figure shows the static power consumed by the mixed-signal perceptron. Although power consumption varies per benchmark, the power consumption scales linearly resistance. We find that for all resistance values, the total static power is reasonable (less than 1 Watt).

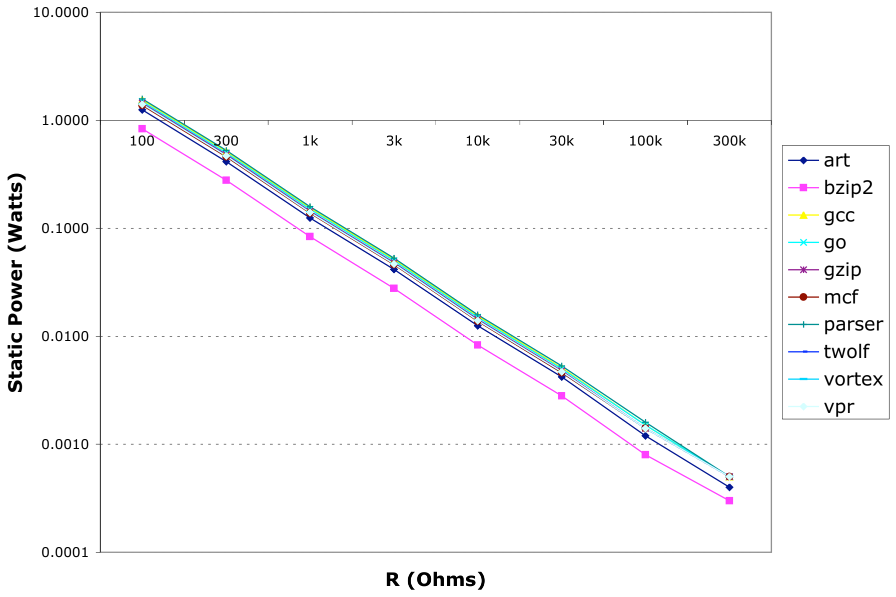

Figure 7: Static Power Consumption vs. D/A Impedence.

Dynamic power consumption, which is caused by capacitive loading, is a product of the capacitance, operating frequency and voltage swings. Equation 7 is used to calculate the dynamic power consumption.

$$P_{dynamic} = \frac{Cf(\Delta V)^2_{rms}}{2} \tag{7}$$

We measured the worst-case $(\Delta V)_{rms}$ to be 63mV. By using our estimated capacitance, 200 fF, and a typical high-frequency, we can use Equation 7 to calculate our dynamic power consumption:

$$P_{dynamic} = \frac{Cf(\Delta V)^2_{rms}}{2} = (200fF)(4GHz)(63mV)^2 = 72\mu V \tag{8}$$

6 Summary

We have simulated a perceptron implemented using analog circuits in a mixed-signal IC and subjected our model to a wide range of noise and resistor accuracies. With expected resistor matching (2-5%) an analog perceptron performs nearly identical to a digital perceptron. Interference noise of less than 15 dBm is required for correct operation.

Based on 0.18 μ m technology, we expect that our mixed-signal perceptron will be capable of 4 GHz operation, while consuming less than 100 mW of power.

A topic for future research is investigating the relative merits of a perceptron predictor based upon current mode D/A converters which may provide better noise tolerance than the voltage mode converters we explored in this paper.

References

[1] H. D. Block. The perceptron: A model for brain functioning. *Reviews of Modern Physics*, 34:123–135, 1962.

[2] E. Brekelbaum, J. Rupley, C. Wilkerson, and B. Black. Hierarchical scheduling windows. In *Proc. Int'l Symp. on Microarchitecture*, pages 27–36, 2002.

[3] O. Cadenas, G. Megson, and D. Jones. Implementation of a block based neural branch predictor. In *Proc. Euromicro Conf. on Digital Systems Design (DSD)*, pages 235–238, August 2005.

[4] O. Cadenas, G. Megson, and D. Jones. A new organization for a perceptron-based branch predictor and its fpga implementation. In *Proc. IEEE Symp. on VLSI New Frontiers in VLSI Design*, pages 305–306, May 2005.

[5] F. Camboni and M. Valle. A Mixed Mode Perceptron Cell for VLSI Neural Networks. In *Int'l Conf. on Electronics, Circuits and Systems*, volume 1, pages 377–380, September 2001.

[6] H.-M. Chuang, K.-B. Thei, S.-F. Tsai, C.-T. Lu, X.-D. Liao, K.-M. Lee, H.-R. Chen, and W.-C. Liu. A comprehensive study of polysilicon resistors for CMOS ULSI applications. *Superlattices and Microstructures*, 33:193–208, July 2003.

[7] A. Falcon, J. Stark, A. Ramirez, K. Lai, and M. Valero. Prophet/critic hybrid branch prediction. In *Proc. Int'l Symp. on Computer architecture*, pages 250–263, 2004.

[8] P. Haikonen. Associative neuron in an artificial neural network. United States Patent Number 6,625,588, September 2003.

[9] D. A. Jimnézez. Fast path-based neural branch prediction. In *Proc. Int'l Symp. on Microarchitecture (MICRO)*, pages 243–252, December 2003.

[10] D. A. Jimnézez. Piecewise Linear Branch Prediction. In *Proc. Int'l Symp. on Computer Architecture*, pages 382–393, June 2005.

[11] D. A. Jimnézez and C. Lin. Dynamic Branch Prediction with Perceptrons. In *Proc. Int'l Symp. on High-Performance Computer Architecture (HPCA)*, pages 197–206, January 2001.

[12] D. A. Jimnézez and C. Lin. Neural methods for dynamic branch prediction. *ACM Trans. on Computer Systems*, 20(4), 2002.

[13] T. Kohonen. Improved versions of learning vector quantization. In *IJCNN International Joint Conference on Neural Networks*, volume 1, pages 545–550, June 1990.

[14] W. A. Lane and G. T. Wrixon. The Design of Thin-Film Polysilicon Resistors for Analog IC Applications. *IEEE Transactions on Electronic Devices*, 36(4):738–744, 1989.

[15] M. Minsky. *Neural nets and the brain-model problem.* PhD thesis, Princeton University, 1954.

[16] A. Murray. Analog CMOS Fundamentals. http://www.see.ed.ac.uk/˜afm/teaching/notes_gif/vlsi4/ANALOG1/index1.htm, April 2007.

[17] F. Rosenblatt. The Perceptron: A Perceiving and Recognizing Automaton, Project PARA. Technical Report 85-460-1, Cornell Aeronautical Laboratory, 1957.

[18] F. Rosenblatt. *Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms.* Spartan Books, 1962.

[19] L. Tarassenko and A. F. Murray. VLSI Implementation of Neural Networks. In *IEE Colloquium on Current Issues in Neural Network Research*, pages 5/1–5/6, May 1989.

[20] L. Vintan. Towards a high performance neural branch predictor. In *Int'l Joint Conference on Neural Networks*, pages 868–873, July 1999.