Energy-Aware Data Compression for Multi-Level Cell (MLC) Flash Memory *

Yongsoo Joo, Youngjin Cho, Donghwa Shin and Naehyuck Chang †
Seoul National University
naehyuck@snu.ac.kr

ABSTRACT

We discover significant value-dependent programming energy variations in multi-level cell (MLC) flash memories, and introduce an energy-aware data compression method that minimizes the flash programming energy rather than the size of the compressed data. We express energy-aware data compression as an entropy coding with unequal bit-pattern costs. Deploying a probabilistic approach, we derive the energy-optimal bit-pattern probabilities and the expected values of the bit-pattern costs for the large amounts of compressed data which are typical in multimedia applications. Then we develop an energy-optimal prefix coding that uses integer linear programming, and construct a prefix code table. From a consideration of Pareto-optimal energy consumption, we make tradeoffs between data size and programming energy, such as a 35% energy saving for a 50% area overhead.

Categories and Subject Descriptors: C.4 [**Performance of systems**]: Modeling techniques; C.4 [**Performance of systems**]: Performance attributes
General Terms: Design, Measurement, Performance
Keywords: MLC, Compression, Flash memory

1. INTRODUCTION

High-capacity flash memory devices provide lightweight, reliable and high-performance non-volatile storage. Flash technology has already achieved significant commercial success, and generates more profit for the semiconductor vendors than SDRAMs. Thanks to evolutionary progress in capacity, flash memories continue to reach new application areas, especially those which involve multimedia data. Nevertheless, capacity is still one of the most important concerns for customers. That is why multi-level cell (MLC) technology has been widely applied to both NOR and NAND flash devices. Four-level MLC flash memories store two bits in a memory cell by using the threshold voltage (V_T) difference between the erased state and the programmed state of the single-level cell (SLC) flash to represent intermediate states.

*The ICT at Seoul National University provides research facilities for this study, and this work is supported by Samsung Electronics.

†Corresponding author.

DAC 2007, June 4–8, 2007, San Diego, California, USA.

Table 1: Programming energy variation of MLC flash memories.

Memory type	Operation	Time (μs)	Energy (μJ)
Intel MLC NOR 28F256L18	Program 00	110.00	2.37
	Program 01	644.23	14.77
	Program 10	684.57	15.60
	Program 11	24.93	0.38

Note: Program 00 (01, 10, 11) denotes a 32-word buffered programming with a repeated 00 (01, 10, 11) bit-pattern.

Consequently, it offers double the capacity for the same underlying semiconductor technology.

We made cycle-accurate energy measurements on an Intel 28F256L18 four-level MLC flash memory and confirmed that significantly more programming time and energy are required for particular bit-patterns than for others. As shown in Table 1, the time and energy required to program 01 and 10 are at least six times more than the requirements for programming 00 and 11.

Most existing data compression schemes focus on reducing the size of the compressed data. This approach achieves an energy-optimal data compression only if the programming energy is proportional to the compressed data size. This is true for conventional SDRAMs, which consume energy equally for all bit-patterns [9]. However, if each pattern consumes a different amount of energy, as shown in Table 1, it might be that programming four bits of data, '0101' would consume more energy than the eight bits '11110000'. In this case, conventional data compression techniques will be far from optimal in terms of energy consumption.

Previous authors [3] used energy consumption as the cost function instead of data size. But that work was primarily concerned with the read energy required for repeated accesses to code programmed in NOR flash, whereas we focus on the energy used in programming operations, which is orders of magnitude larger than that required for read operations. Recently, Intel have introduced an energy-efficient way of programming the MLC flash as though it were an SLC flash memory [1]. This involves programming a selected region of the MLC flash using only two out of the four possible states, so as simply to avoid the high-cost bit-patterns. However, this programming mode is not backed by any systematic attempt at optimization. Intel suggest that this technique would be suitable for storing data which is small and frequently updated, such as the index to a file system.

In this paper, we introduce an energy-aware data compression technique for MLC NOR flash that is aimed at reducing the programming energy rather than the data size. If valid energy costs can be assigned to each bit-pattern, an energy-optimal data compression scheme can be achieved by adopting an optimal entropy coding for unequal alphabet cost. Many previous authors have addressed this problem, using approaches such as integer linear programming (ILP) [6], dynamic programming [5], or polynomial-time approximation schemes (PTAS) [4]. However, we cannot

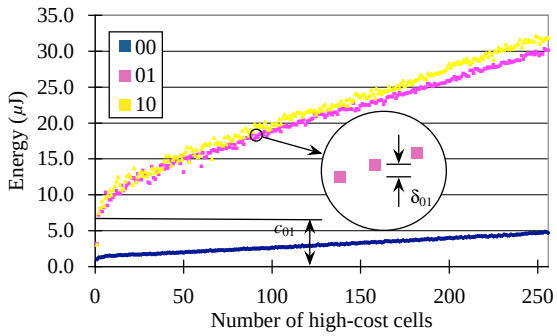

Figure 1: Programming energy variation versus the number of high-cost cells.

assign predefined energy costs to individual bit-patterns because the NOR flash memory programs many bit-patterns at the same time. The costs of bit-patterns become inter-related when those patterns are located in the same data unit, such as a word or a 32-word block in the case of the Intel NOR MLC flash. We therefore propose a new entropy coding that determines the expected energy cost of each bit-pattern by probabilistic estimation of energy and size requirements.

The contributions of this paper are as follows. First, we have performed cycle-accurate programming energy measurement of an MLC NOR flash memory, constructed its analytical energy model, and verified its accuracy to within 1% of average absolute error. Second, we have formulated a new model of entropy coding with unequal bit-pattern costs, i.e., programming energy costs. To use this model, we deploy a new probabilistic approach to energy-optimal entropy encoding. We start by introducing probabilistic energy and size estimation using, respectively, a joint probability mass function and results from information theory. Then we perform a heuristic search and obtain energy-optimal bit-pattern probabilities and the expected energy cost of each pattern. Finally, we deploy integer linear programming (ILP) to achieve an energy-optimal prefix coding for JPEG images. Third, we go on to show that our coding can be further extended to an energy tradeoff in programming MLC NOR flash. We demonstrate a 35% energy saving with a 50% size overhead.

2. MLC NOR FLASH ENERGY CHARACTERIZATION

We measured the programming energy of the Intel 28F256L18 four-level MLC flash memory using an automated cycle-accurate energy measurement tool [9]. We focus on 32-word buffered programming from among the available modes, because it is the most efficient programming method for large amounts of data.

A symbol 11 corresponds to the erased state (like 1 in SLC flash) and programming 11 requires the least amount of energy. If a data unit consists entirely of 11, MLC flash requires the least possible amount of energy to program the data, and therefore we use this as the reference case. First, we measured the change in energy consumption caused by varying the number of 00. We gradually increased the number of 00, filling the remainder with 11. We then repeated this experiment for 01 and 10. We also confirmed that the position of the high-cost 00, 01 and 10, does not change the energy consumption.

As shown in Figure 1, the energy required to program a data unit is approximately proportional to the number of high-cost symbols, and each high-cost symbol has a different energy cost. Amongst the high-cost symbols, 00 requires less energy than 01 or 10. The largest amount of energy is required to program 10, but it is not significantly different from the requirement of 01.

Table 2: Coefficients of Equation (1) for the Intel 28F256L18 MLC NOR flash.

Coefficients	Energy (nJ)	Coefficients	Energy (nJ)
δ_{00}	13.37	c_{00}	1299.60
δ_{01}	72.64	c_{01}	11494.00
δ_{10}	79.01	c_{10}	11850.20
δ_{11}	0.00	c_{11}	752.02

Table 3: Measured and estimated energy for the eight benchmarking sets. (energy unit: μJ)

Type	cpp	jpg	doc	zip	htm	wav	mp3	bin
Measured	3.72	3.59	2.45	3.59	3.51	3.19	3.39	3.20
Estimated	3.68	3.59	2.52	3.57	3.48	3.19	3.34	3.18
Error (%)	1.0	0.0	3.2	0.6	0.8	0.2	1.4	0.5

Figure 1 enables us to define the *incremental cost* of a symbol, δ_i, and also the *common-mode cost* of a symbol, c_i where i is one of the symbols 00, 01, 10 or 11. This incremental cost is the energy overhead determined by the number of high-cost symbols, while the common-mode cost is the energy overhead caused by presence of a high-cost symbol.

Many repeated energy measurements contributed to the values of δ_i and c_i in Table 2. It turns out that the contribution of each incremental cost is independent, and thus the incremental costs can be summed to determine the unit programming energy. It also happens that the common-mode cost is determined by the largest common-mode cost from among the high-cost symbols, when a data unit contains more than one high-cost symbol. For example, if a data unit contains both 00 and 01, the common-mode energy is determined by 01, i.e. by c_{01}, because $c_{01} > c_{00}$.

We can now characterize the energy required to program one data unit in a four-level MLC NOR flash as $f_\varepsilon(n_{00}, n_{01}, n_{10}, n_{11})$, such that

$$f_\varepsilon(n_{00}, n_{01}, n_{10}, n_{11}) = \sum_{i=00}^{11} n_i \delta_i + \max\left(o_{00}c_{00}, \cdots, o_{11}c_{11}\right), \quad (1)$$

where n_i is the number of symbol i's in data unit, and o_i is defined by

$$o_i = \begin{cases} 0 & \text{if } n_i = 0 \\ 1 & \text{if } n_i > 0. \end{cases}$$

We have verified the energy model of Equation (1) by measuring the NOR flash programming energy using data from various real applications. Table 3 shows the comparison between the measured and derived energy that is expressed by Equation (1). We programmed 5,120 words (10 KB) of data and only had a 1.0% average absolute error.

3. ENERGY-AWARE ENTROPY CODING

3.1 Problem statement

Typical data compression schemes use the size of the compressed data as the cost function, since most semiconductor memories have the same cost of bit-patterns if their lengths are the same. In this paper, we consider the MLC flash programming energy rather than the size of the compressed data, because the costs of the bit-patterns are unequal, even if the resulting bit-patterns have the same length, as shown in the previous section. To produce a data compression that minimizes the MLC flash programming energy, we need to obtain an optimal entropy coding for unequal alphabet cost. To formulate an entropy coding with unequal symbol costs, the cost of each symbol (bit-pattern) in the alphabet (a set of symbols) should be given as a constant. For four-level MLC flash

memories, the symbols are defined by the bit-patterns 00, 01, 10, and 11, which are values that can be stored in one MLC flash cell. Thus, if we can measure the energy required to program these four symbols, we can formulate an energy-aware entropy coding for the MLC flash.

However, a typical NOR flash memory programs multiple bit-patterns at the same time, and there exist common-mode costs in the programming energy model of the NOR flash, as shown in Table 2. It means that the bit-pattern costs are correlated to each other, and thus we cannot assign predefined energy costs to the bit-patterns. These difficulties motivate a probabilistic approach to dealing with the unequal costs of MLC flash programming. Instead of trying to assign fixed costs to each symbol, we determine the expected cost of each symbol, which enable us to achieve an energy-optimal entropy coding.

3.2 Probabilistic energy estimation

We estimate the expected energy required to program a data unit when the probabilities of each symbol occurring are given. Let S be the number of symbols. A sequence of symbols to be programmed can be modeled as a sequence of independent throws of a dice with S faces, where the probability of the occurrence of i-th face is p_i.

Suppose we have a random vector variable $\mathbf{X} = (X_1, \cdots, X_S)$ with the outcome $\xi \in (x_1, \cdots, x_S)$, where $x_i \geq 0$ and $\sum_{i=1}^{S} x_i = L$, and L is the number of symbols in a data unit. $\mathbf{X}$ denotes the number of occurrences of the symbols in a data unit. The joint probability mass function (PMF) of $\mathbf{X}$ that specifies the probabilities of a product-form events such as $\{X_1 = x_1\} \cap \cdots \cap \{X_S = x_S\}$ is given by

$$p_{X_1,\cdots,X_S}(x_1, \cdots, x_S) = P[X_1 = x_1, \cdots, X_S = x_S]. \tag{2}$$

A family of conditional PMFs can be obtained from the joint PMF by conditioning on different subsets of the random variables as follows [7]:

$$\begin{aligned} p_{X_1,\cdots,X_S}(x_1, \cdots, x_S) &= p_{X_S}(x_S|x_1, \cdots, x_{S-1}) \\ &\quad \times p_{X_{S-1}}(x_{S-1}|x_1, \cdots, x_{S-2}) \\ &\quad \times \cdots \times p_{X_2}(x_2|x_1) \times p_{X_1}(x_1). \end{aligned} \tag{3}$$

We derive a generalized conditional PMF for MLC NOR flash memories using Equation (3).

$$\begin{aligned} p_{X_1,\cdots,X_S}(x_1, \cdots, x_S) = &\binom{L - \sum_{i=1}^{S-1} x_i}{x_S} {p_S}^{x_S} (1-p_S)^{L-\sum_{i=1}^{S} x_i} \\ &\times \binom{L - \sum_{i=1}^{S-2} x_i}{x_{S-1}} {p_{S-1}}^{x_{S-1}} (1-p_{S-1})^{L-\sum_{i=1}^{S-1} x_i} \\ &\vdots \\ &\times \binom{L-x_1}{x_2} {p_2}^{x_2} (1-p_2)^{L-x_1-x_2} \\ &\times \binom{L}{x_1} {p_1}^{x_1} (1-p_1)^{L-x_1}. \end{aligned} \tag{4}$$

We define the programming energy cost of a data unit as $f_\varepsilon(x_1, \cdots, x_S)$. The expected value of the programming energy of a data unit can then be expressed as:

$$\begin{aligned} E[f_\varepsilon(X_1, \cdots, X_S)] = \sum_{x_1=0}^{L} \cdots \sum_{x_{S-1}=0}^{L-\sum_{i=1}^{S-2} x_i} \Bigg(& f_\varepsilon(x_1, \cdots, x_{S-1}, L - \sum_{i=1}^{S-1} x_i) \\ & \times p_{X_1,\cdots,X_S}(x_1, \cdots, x_{S-1}, L - \sum_{i=1}^{S-1} x_i) \Bigg). \end{aligned} \tag{5}$$

3.3 Size estimation using information theory

With a given set of symbol costs, $(c_1, \cdots, c_S)$, the entropy coding results in a probability for each symbol p_i, such that

$$p_i = Z^{-c_i}, \tag{6}$$

where Z is the largest real solution of the following equation [8]:

$$\sum_{i=1}^{S} Z^{-c_i} = 1. \tag{7}$$

In general, the following condition holds for semiconductor memories:

$$c_i = c_j, \quad \text{for all } i, j \text{ where } 1 \leq i, j \leq S. \tag{8}$$

Therefore, the size-optimal entropy coding is achieved when $p_i = S^{-1}, 1 \leq i \leq S$.

Let us define the *information quantity* as the amount of information that can be programmed in a data unit I, and this can be expressed in the following terms:

$$I = L \cdot H_S(p_1, \cdots, p_S) = -L \sum_{i=1}^{S} p_i \log_2 p_i, \tag{9}$$

where $H_S(p_1, \cdots, p_S)$ is the entropy of the symbol whose probabilities of occurrence are $p_1, \cdots, p_S$ [8]. Suppose we perform an entropy coding with particular symbol costs, i.e., energy costs $(\hat{c}_1, \cdots, \hat{c}_S)$, and consequently generate the symbol probabilities, $(\hat{p}_1, \cdots, \hat{p}_S)$. The information quantity of this particular entropy coding, $\hat{I}$, is always less than or equal to I, which is the size-optimal entropy coding such that $p_i = S^{-1}$. The normalized information quantity, $\tilde{I}$, is defined by

$$\tilde{I}(\hat{p}_1, \cdots, \hat{p}_S) = \frac{H_S(\hat{p}_1, \cdots, \hat{p}_S)}{H_S(S^{-1}, \cdots, S^{-1})}. \tag{10}$$

The inverse of the normalized information quantity $1/\tilde{I}(\hat{p}_1, \cdots, \hat{p}_S)$ specifies the normalized size of that particular entropy coding with $(\hat{p}_1, \cdots, \hat{p}_S)$.

3.4 Energy-optimal symbol probabilities

Based on the energy and size estimation from the occurrence probabilities of the symbols in the alphabet, we explore the Pareto-optimal points through heuristic search. We can derive two functions of the symbol occurrence probabilities from the results in Sections 3.2 and 3.3. One is the energy consumption per data unit, $E[f_\varepsilon(X_1, \cdots, X_S)]$, and the other is the normalized size, $1/\tilde{I}(\hat{p}_1, \cdots, \hat{p}_S)$. The cost function defined by $\dfrac{E[f_\varepsilon(X_1, \cdots, X_S)]}{\tilde{I}(\hat{p}_1, \cdots, \hat{p}_S)}$ is the programming energy consumption with particular symbol occurrence probabilities. The problem statement for the energy-optimal symbol occurrence probabilities can then be described as follows:

$$\text{Find } (\hat{p}_1, \cdots, \hat{p}_S) \text{ that minimizes } \frac{E[f_\varepsilon(X_1, \cdots, X_S)]}{\tilde{I}(\hat{p}_1, \cdots, \hat{p}_S)}. \tag{11}$$

Additionally, we can make a problem statement for the energy-optimal symbol occurrence probabilities with a normalized size constraint s_c, by adding the inequality:

$$\frac{1}{\hat{I}(\hat{p}_1, \cdots, \hat{p}_S)} < s_c. \tag{12}$$

We need to employ a heuristic approach to solve this problem, because this is a non-convex optimization with a non-linear constraint. However, by exploring the solution space we have found that this problem has few local optima, allowing us to use direct search to obtain a solution.

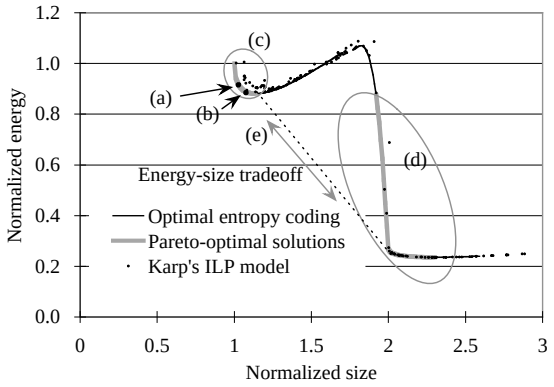

Figure 2: Pareto-optimal prefix coding solutions for JPEG images that minimize the Intel MLC flash programming energy.

3.5 Expected values of symbol costs

Now we find the expected energy costs of each symbol from the optimal symbol probabilities found in Section 3.4 using Equations (6) and (7). Let the optimal symbol probabilities be $(\hat{p}_1, \cdots, \hat{p}_S)$, which is a result from Section 3.4. We can now obtain the optimal costs, $(\hat{c}_1, \cdots, \hat{c}_S)$, such that

$$E\left[f_\varepsilon(\hat{X}_1, \cdots, \hat{X}_S)\right] = L\sum_{i=1}^{S} \hat{p}_i \hat{c}_i. \tag{13}$$

From Equation (6), the expected energy value, $\hat{c}_i$, is given by

$$\hat{c}_i = -\frac{\log_2 \hat{p}_i}{\log_2 Z}. \tag{14}$$

And from Equations (9), (13) and (14), we derive

$$\log_2 Z = \frac{-\sum_{i=1}^{S} \hat{p}_i \log_2 \hat{p}_i}{E\left[f_\varepsilon(\hat{X}_1, \cdots, \hat{X}_S)\right]} = \frac{L \cdot H_S(\hat{p}_1, \cdots, \hat{p}_S)}{E\left[f_\varepsilon(\hat{X}_1, \cdots, \hat{X}_S)\right]}. \tag{15}$$

Therefore,

$$\hat{c}_i = -\frac{E\left[f_\varepsilon(\hat{X}_1, \cdots, \hat{X}_S)\right]}{L \cdot H_S(\hat{p}_1, \cdots, \hat{p}_S)} \cdot \log_2 \hat{p}_i. \tag{16}$$

4. ENERGY-OPTIMAL PREFIX CODING

4.1 Pareto-optimal prefix coding solutions

We will now explore the theoretical bound on the energy gain that can be achieved by the result in Section 3.4. Figure 2 shows the Pareto-optimal points of the Intel 28F256L18 four-level MLC flash. With a symbol probability set of $(p_{00}, \cdots, p_{11}) = (0.302, 0.185, 0.176, 0.337)$, we achieved a 9% programming energy reduction with a 3% size overhead (Point (a)), and a 12% energy reduction with a 10% size overhead (Point (b)) with a symbol probability set of $(p_{00}, \cdots, p_{11}) = (0.333, 0.135, 0.123, 0.408)$.

We performed optimal prefix coding of real JPEG images using the unequal alphabet programming energy costs of the Intel four-level MLC flash by the result in Section 3.5. We borrow the source symbols and their probabilities from the Huffman table for the luminance AC coefficients in the JPEG standard [2]. The expected values of the symbol costs are derived from Equation (16). We use Karp's ILP model [6] and the CPLEX ILP solver.

Figure 2 shows the Pareto-optimal solutions of the prefix code sets, which are very close to the theoretical Pareto-optimal points. The reasons that the actual energy-optimal JPEG encodings are not exactly identical to the theoretical optimal solutions, are mismatches between the alphabet energy costs and the probabilities of the source symbols, quantization error of the alphabet costs for Karp's ILP, and the limitation of ILP itself.

The Intel four-level MLC flash has quite high common-mode costs, allowing us only a small degree of freedom in choosing a Pareto-optimal solution. The group of solutions represented by Curve (c) in Figure 2 has an overhead of less than 10% and an energy saving of around 5% to 12%. On the other hand, the solution group of Curve (d) shows almost 80% energy saving, but the size has nearly doubled, reducing the storage efficiency to near that of an SLC flash.

4.2 Energy-size tradeoff

To achieve further energy reductions while making a reasonable compromise on data size, we propose the energy-size tradeoff represented by Line (e) in Figure 2, which is a coarse-grain mixture of the Pareto-optimal solutions of Curves (c) and (d) (i.e. some of the data is encoded using Curve (c), and the rest using Curve (d)). Adjusting the proportion of the two encoding groups achieves different energy-size tradeoffs. For instance, we can save more than 35% programming energy if we accept a 50% size overhead.

Suppose we have a digital camera. The maximum number of pictures that can be taken is determined either by out of battery or out of memory, whatever comes first. By tracking the remaining battery budget and flash memory capacity, an on-line tradeoff between energy and data volumes can maximize the number of pictures that can be taken with the remaining resources.

5. CONCLUSION

MLC flash memories are rapidly expanding their market share due to their strong cost-effectiveness. This paper reports a pioneering energy-aware data compression for MLC flash memories that minimizes the programming energy rather than the size of the compressed data. This is specifically applicable to new paradigm low-power memory systems, and achieves significant energy saving with no additional hardware cost. We provide a generalized problem formulation and a solution method that reflect practical operating conditions such as the simultaneous programming of multiple cells. Our solution is applicable to any multi-level cell memory devices that have unequal read or write (programming) energy costs, and achieves the optimal programming energy consumption in the case of NOR flash. We have further extended the applicability of the proposed method by quantifying the relationship between programming energy and compressed data size. The resulting tradeoffs, such as a 35% programming energy saving for a 50% size overhead, can help to maximize the operational lifetime of mobile devices that use MLC flash memories.

6. REFERENCES

[1] *Intel StrataFlash Cellular Memory (M18) Datasheet*. Intel.

[2] *Information Technology - Digital Compression and Coding of Continuous-tone Still Images - Requirements and Guidelines*. ISO/IEC 10918-1, ITU T.81, September 1992.

[3] L. Benini, A. Macii, and E. Macii. Exact and heuristic algorithms for low-energy code compression in performance and memory constrained embedded systems. In *Proceedings of the 44th IEEE 2001 Midwest Symposium on Circuits and Systems*, volume 2, pages 552–555, August 2001.

[4] M. J. Golin, C. Kenyon, and N. E. Young. Huffman coding with unequal letter costs. In *Proceedings of the ACM Symposium on Theory of Computing*, pages 785–791, May 2002.

[5] M. J. Golin and G. Rote. A dynamic programming algorithm for constructing optimal prefix-free codes with unequal letter costs. *IEEE Transactions on Information Theory*, 44(5):1770–1781, September 1998.

[6] R. M. Karp. Minimum-redundancy coding for the discrete noiseless channel. *IEEE Transactions on Information Theory*, 7(1):27–38, January 1961.

[7] A. Leon-Garcia. *Probability and Random Processes for Electrical Engineering*. Addison-Wesley, Reading, MA, 1994.

[8] C. E. Shannon. A mathematical theory of communication. *The Bell System Technical Journal*, 27:379–423, 623–656, 1948.

[9] H. Shim, Y. Joo, Y. Choi, H. G. Lee, and N. Chang. Low-energy off-chip SDRAM memory systems for embedded applications. *ACM Transactions on Embedded Computing Systems*, 2(1):98–130, 2003.