Power-optimal Encoding for DRAM Address Bus

Wei-Chung Cheng and Massoud Pedram
University of Southern California
Los Angeles, CA 90089
{wccheng, massoud}@zugros.usc.edu

ABSTRACT

This paper presents Pyramid code, an optimal code for transmitting sequential addresses over a DRAM bus. Constructed by finding an Eulerian cycle on a complete graph, this code is optimal for conventional DRAM in the sense that it minimizes the switching activity on the time-multiplexed address bus from CPU to DRAM. Experimental results on a large number of testbenches with different characteristics (i.e. sequential vs. random memory access behaviors) are reported and demonstrate a reduction of bus activity by as much as 50%.

1. INTRODUCTION

Modern electronic systems have a dichotomy of simultaneously needing to be low power and high performance. This arises largely from their use in battery-operated portable (wearable) platforms. Even in fixed, power-rich platforms, the packaging and reliability costs associated with very high power and high performance systems are forcing designers to look at ways to reduce power consumption. Power-efficient design requires reducing power dissipation in all parts of the design and during all stages of the design process subject to constraints on the system performance and quality of service (QoS). Sophisticated power-aware high-level language compilers, dynamic power management policies, memory management and bus encoding techniques, and hardware design tools are required to satisfy these often conflicting design requirements [1] [2]. This paper focuses on the low power bus encoding problem.

In [3], Su et al proposed the use of *Gray code* to implement the program counter of a microprocessor to minimize the switching activity of sequential memory accesses. The authors showed that the Gray code is asymptotically optimal among all irredundant codes. *Bus-Invert code* [4] toggles the polarity of the signals according to the Hamming distance between two consecutive data values by using an additional bit line on the bus. Similarly, the *T0 code* [5] uses a redundant signal to indicate that the bus is in normal mode or in increasing address mode. In the latter case, it needs to change only one signal. Benini et al proposed an adaptive encoding scheme that switches between the T0 and Bus-invert codes depending on whether sequential or random data streams are encountered [8]. The *Working Zone code* [6] addresses the problem that the address bus does not behave completely sequentially because the accesses to different zones are usually interleaved. The *Beach code* [7] exploits the temporal correlations to further reduce the switching activity on the bus. All these techniques deal with simple address formats where the bus is not time-multiplexed. The key idea behind all of these techniques is to reduce the Hamming distance between consecutive addresses for a sequential memory access pattern, e.g., instruction fetching for large array access. However, these schemes cannot be applied to DRAM address bus encoding because of the time-multiplexed addressing scheme used therein.

Dynamic RAM (DRAM) is often layed out in a two-dimensional array. To identify a memory cell in the array, two addresses are needed: *row* and *column*. To reduce the address pin count and for legacy reasons, the row and column addresses are usually multiplexed. The row address is sent over the bus first and is latched in the DRAM decoder. Subsequently, the column address is sent to complete the address. Because the switching activity on a DRAM bus is totally different from that of a non-multiplexed bus, we need another Gray-code-like encoding scheme to minimize the switching activity for sequential memory access on a DRAM bus. In this paper, we present the problem formulation for the switching activity minimization, the Eulerian cycle isomorphism, and an optimal solution which we call the Pyramid code.

2. PROBLEM FORMULATION

Consider an address space of size 2^{2N}, which is represented by an ordered set $S_{2^{2n}}$={0, 1, … , $2^{2N}-1$}. We will use $S_{2^{2N}}$ to represent both the range and the increasing sequence of the 2^{2N} integer numbers. Each address $b \in S_{2^{2N}}$ has $2N$ bits: $< b_{2N-1}, b_{2N-2}, ..., b_N, b_{N-1}, ..., b_1, b_0 >$. Consider executing a sequence of RISC-style instructions with instruction addresses *S*={ $b^0, b^1, b^2, ...$ }. These addresses tend to be sequential unless branch instructions are present. Let S denote a *sequential* access pattern of arbitrary size. In the target system, if the code and data address busses are separated, then we can calculate the bus switching activity from *S* only. The switching activity between two addresses *x* and *y* is defined by the *Hamming* distance $H(x,y) = \sum_i x_i \otimes y_i$, where '$\otimes$' denotes the Boolean exclusive-OR operation. For a non-multiplexed bus (such as SRAM), the switching activity is $SA_{NOMUX}(S) = \sum_i H(b^i, b^{i+1})$.

Consider next a DRAM address bus with *N* signals. An address b is transmitted over this bus by sending the row address $Row(b) = < b_{2N-1}, b_{2N-2}, ..., b_N >$ followed by the column address $Col(b) = < b_{N-1}, ..., b_1, b_0 >$. The *external switching activity* of two consecutive addresses b^i and b^{i+1} is $SA_E(b^i, b^{i+1}) =$

$H(Row(b^i), Col(b^{i+1}))$. The *internal switching activity* of an address b^i is $SA_I(b^i) = H(Row(b^i), Col(b^i))$. The total switching activity of a multiplexed bus is

$$SA_{MUX}(S) = \sum_i SA_E(b^i, b^{i+1}) + \sum_i SA_I(b^i) \quad (1)$$

A *2N-digit fixed-length code* or simply *code* is a permutation of $S_{2^{2N}}$, e.g., a bijection from $S_{2^{2N}}$ to itself. The idea of irredundant memory bus encoding is to find a function *f* such that sequence *S* can be replaced with sequence *f*(*S*) and the resulting activity $SA(S)$ can be reduced to $SA(f(S))$. For example, to exploit the sequentiality of the instruction addresses, Gray code *g* was used in [3] to minimize the switching activity from $SA_{NOMUX}(S_{2^{2N}})$ to $SA_{NOMUX}(g(S_{2^{2N}}))$, which is the minimum possible. Similarly, our goal is to find a code *f* such that $SA_{MUX}(f(S_{2^{2N}}))$ is minimum.

3. OPTIMALITY AND EULERIAN CYCLE

From equation-(1), the tight lower bound on $SA_{MUX}(S_{2^{2N}})$ is $SA_I(S_{2^{2N}})$, which is achieved when $SA_E(S_{2^{2N}}) = 0$. This is because (i) $SA_I(S_{2^{2N}})$ is a fixed value since a different *f* function only affects $SA_E(S_{2^{2N}})$ (ii) the best lower bound on $SA_E(S_{2^{2N}})$ is 0, when $SA_E(b^i, b^{i+1}) = 0$, $0 \le i < 2^{2N}$, which in turn implies $Col(f(b^i)) = Row(f(b^{i+1}))$. We call any *f* that forces $SA_E(S_{2^{2N}}) = 0$ an *optimal multiplexed code*. We will find the class of optimal multiplexed codes by transforming the minimization problem into a graph problem.

Given a directed graph *G*=(*V*, *E*) where *V*={0, 1, ... , 2^N-1}, a *cycle c* of length *k* on *G* is a sequence of vertices [v_0, v_1, v_2, …, v_{k-1}] if $v_i v_{i+1} \in E$ for *i* = 0, 1, …, *k*-1 (modulo *k*). An *Eulerian cycle* is a cycle that traverses every edge in *E* exactly once. K_n is the *complete graph* (clique) of *n* vertices whereas $K_{m,n}$ denotes the *complete bipartite graph* on *m* and *n* vertices. The problem of finding an Eulerian cycle on K_n is denoted as ECP$_n$. For each edge $v_i v_{i+1} \in E$, we label it with the function $L: E \to S_{2^{2N}}$, $L(v_i v_{i+1}) = b$ such that $Row(b) = v_i$ and $Col(b) = v_{i+1}$. Intuitively, each edge represents an address in the memory space $S_{2^{2N}}$.

Consider an Eulerian cycle $c_{2^{2N}}$=[$v_0, v_1, v_2, \ldots, v_{2^{2N}-1}$] on a complete graph K_{2^N} with 2^{2N} directed edges. The length of *c* is 2^{2N}. It is easily proved that the ordered set $\Lambda(c_{2^{2N}}) = \{L(v_0v_1), L(v_1v_2)...\} = \{b^0, b^1, b^2, ..., b^{2^{2N}-1}\}$ (the labels of all the edges on an Eulerian cycle of K_{2^N}) is a permutation of $S_{2^{2N}}$. In other words, $\Lambda(c_{2^{2N}})$ is a code of $S_{2^{2N}}$. More importantly, $\Lambda(c_{2^{2N}})$ is a code without external switching activity, i.e., an optimal multiplexed code, because it is constructed from an Eulerian cycle.

Theorem 1. Any Eulerian cycle *c* on a complete graph K_{2^N} (*V*,*E*) with ordered vertex set *V*={0, 1, ... , 2^N-1} produces an optimal multiplexed code $\Lambda(c)$ of $S_{2^{2N}}$.

Proof. Because (i) $\Lambda(c)$ is a code of $S_{2^{2N}}$ (ii) $SA_E(S_{2^{2N}}) = 0$, $\Lambda(c)$ is an optimal multiplexed code.

4. PYRAMID CODE

The sufficient and necessary conditions for an Eulerian cycle to exist on a directed graph are (i) the graph is strongly connected (ii) for every vertex the in-degree is the same as the out-degree [9]. There exist a large number of solutions to ECP$_n$, each of them leading to an optimal multiplexed code. One can apply algorithms such as depth-first search or breadth-first search to easily obtain an arbitrary solution. However, the memory bus encoding and decoding functions will have to be realized in hardware. Simple, yet efficient, functions are necessary for practical implementation. These functions should not be too complex so as to offset the power savings from the reduction of switching activity.

Our approach for finding a suitable solution to ECP$_n$ is based on dynamic programming. Assume that W_i is the solution to ECP$_i$ on complete graph K_i, and that $W_{1\le j<i}$ has been obtained previously. If we consider K_i as being decomposed into two parts $K_{i-1} \cup \{i\}$, then the cut set is the edge set of the complete bipartite graph $K_{i-1,1}$. So, W_i can be decomposed into three parts W_{i-1}, $W_{i-1,1}$, and W_1 corresponding to solutions on K_{i-1}, $K_{i-1,1}$ and K_1, respectively. Note that $W_{i-1,1}$ is an Eulerian cycle on $K_{i,1}$ and can be constructed easily by traversing back and forth between W_{i-1} and W_1. The final solution is constructed by joining the three Eulerian cycles on K_{i-1}, $K_{i-1,1}$, and K_1. The formal generating formulas are:

$$W_{i=1} = [0]$$
$$W_{i>1} = W_{i-1} \,\&\, [0, \underbrace{\underbrace{i-1,1}_{1}, \underbrace{i-1,2}_{2}, ..., \underbrace{i-1, i-2}_{i-2}, \underbrace{i-1, i-1}_{i-1}}_{(i-1)\, pairs}]$$

where '&' denotes concatenation of two strings. For example:

W_2 = [0, 0, 1, 1] = [0] & [0, 1, 1]
W_3 = [0, 0, 1, 1, 0, 2, 1, 2, 2]
W_4 = [0, 0, 1, 1, 0, 2, 1, 2, 2, 0, 3, 1, 3, 2, 3, 3]

Imagine a tetrahedron (a 4-vertex, 3-dimensional pyramid) with six edges of the same non-zero length. Then W_4 is a way of traversing every edge in both directions exactly once.
Given W_{2j}, we obtain the *Pyramid Code* for $S_{2^{2j}}$, PC_{2j}, from $\Lambda(W_{2j})$ directly[1]. For example:

PC_2 ={0000, 0001, 0101, 0100}
PC_4 ={0000, 0001, 0101, 0100, 0010, 1001, 0110, 1010, 1000, 0011, 1101, 0111, 1110, 1011, 1111, 1100}

[1] For our DRAM model, the dimension must be an even number. Thus, only PC_{2j} are useful in our application.

Theorem 2. The Pyramid code is an optimal multiplexed code.

Proof. We only provide the proof outline due to space limitation. We have to show that the Pyramid code generates an Eulerian cycle on $G(V,E)$. This in turn is proven by showing that (i) the encoding is a one-to-one and onto mapping of the edge set E, and (ii) the consecutive edges are connected so that they form a cycle.

We believe that Pyramid code is not only an optimal multiplexed code, but also superior to other codes in this class because of the following two properties: (1) N-independence: notice that PC_j (W_j) is a prefix of PC_i (W_i) if $j<i$, which implies that the encoding/decoding functions are independent of N, e.g., the size of the address space (2) Rugularity: the rule of generating PC_i (W_i) is simple and regular, so the encoding/decoding functions can be implemented efficiently.

Unlike the Gray code, Pyramid code is only optimal in one direction. If the sequential access pattern is reversed, then the Pyramid code has to be modified accordingly to be optimal. The required modification is to switch the row and column addresses in the encoding and decoding functions.

5. EXPERIMENTAL RESULT

The purpose of our experiments is to quantitatively assess the performance of the Pyramid code compared to the Binary code. We do not compare to the Gray code because its performance is very similar to Binary code performance on time-multiplexed busses.

We assume that the total memory space is 64K byte (16-bit address). The address bus is 8-bit wide and row/column multiplexed. We also assume that the code address bus and data address bus are different, so the data addresses do not disturb the sequential access pattern of the code addresses. Each instruction is four-byte long. Because the address is increased by four each time, we have to make the addresses consecutive by right-rotating them two bits before the encoding. The rotation operation has low overhead and can be integrated into the encoder.

We assume that the total size of the code segment is 1024 bytes. To quantitatively evaluate the effectiveness of the different degrees of address sequentiality, we divide this code segment into blocks of 4, 8, ..., 1024 bytes. For example, if the block size is 8, it means that we have 128 blocks with random starting addresses and within each block we have 8 sequential addresses.

We apply statistical sampling techniques to report the results. More precisely, we define a unit of sampling to be the total number of bit switchings in a code segment of 1024 instructions. We then form a sample by taking the mean of the switching activity values for 30 randomly generated sampling units. We report the expected value. The total number of bit switchings per code segment of 1024 instructions by analyzing 3 sample results. In our experience, the sample size and number of samples is sufficient to provide high confidence (90% or higher) and low error (5% or lower) for the reported results.

The Pyramid code is worse than binary code only in the case when the block size is four (no sequential addressing whatsoever).

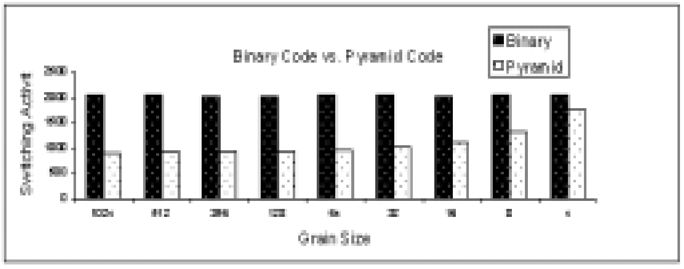

Figure 1. Total switching activity per code segment of 1024 bytes

Note that this is highly unlikely. The case of blocks of 8 or 16 bytes is more typical. Once the block size is larger than eight, the reduction of switching activity converges to near 50%, because Pyramid code virtually eliminates all external switching activity. We also notice that the binary code has similar internal and external switching activity.

Therefore, if the access pattern exhibits pure sequential pattern, the pyramid code will cut half of the switching activity by a factor of two by eliminating the external switching activity.

6. CONCLUSION

In the paper, we addressed the switching activity minimization problem on conventional DRAM busses, and formulated it as an Eulerian cycle problem on a complete graph. To make the implementation practical, we proposed an efficient traversal algorithm for this problem, and presented very simple encoding and decoding functions. Experimental results show that the Pyramid code can reduce the switching activity on the bus by as much as 50% if the access pattern exhibits sequential behavior. For more advanced DRAM technology such as EDO DRAM, the Pyramid code can be extended by modifying the formulation of ECP_n and applying a similar traversal algorithm.

7. REFERENCES

[1] E. Macii, M. Pedram and F. Somenzi, "High level power modeling, estimation and optimization", *IEEE Trans. on Computer Aided Design*, Vol. 17. No. 11, November 1998, pages 1061-1079.

[2] M. Pedram, "Power minimization in IC design: principles and applications," *ACM Trans. on Design Automation of Electronic Systems*, Vol. 1, No. 1 (1996), pages 3-56.

[3] C. L. Su, C. Y. Tsui, A. M. Despain, "Saving Power in the Control Path of Embedded Processors," *IEEE Design and Test of Computers*, Vol. 11, No. 4, pp. 24-30, 1994.

[4] M. R. Stan, W. P. Burleson, "Bus-Invert Coding for Low-Power I/O," *IEEE Transactions on VLSI Systems*, Vol. 3, No. 1, pp. 49-58, 1995.

[5] L. Benini, G. DeMicheli, E. Macii, D. Sciuto, C. Silvano, "Address Bus Encoding Techniques for System-Level Power Optimization", *DATE-98: IEEE Design Automation and Test in Europe*, pp. 861-866, Paris, France, Feburary 1998.

[6] E. Musoll, T. Lang, J. Cortadella, "Exploiting he Locality of Memory References to Reduce the Address Bus Energy," ISLPED-97: *ACM/IEEE International Symposium on Low Power Electronics and Design*, pp. 202-207, Monterey, CA, August 1997.

[7] L. Benini, G. DeMicheli, E. Macii, M. Poncino, S. Quer, "System-Level Power Optimization of Special Purpose Applications: The Beach Solution", *ISLPED-97: ACM/IEEE International Symposium on Low Power Electronics and Design*, pp. 24-29, Monterey, CA, August 1997.

[8] L. Benini, G. DeMicheli, E. Macii, D. Sciuto, and C. Silvano, "Address Bus Encoding Techniques for System-Level Power Optimization," *Design Automation and Test in Europe*, pp. 861-866, Feb. 1998.

[9] D. E. Knuth, *Fundamental Algorithms*, vol. 1 of "*The Art of Computer Programming*," Addison-Wesley, 1973.