ICCAD 2002, San Jose

Energy Efficient Address Assignment Through Minimized Memory Row Switching

Sambuddhi Hettiaratchi, Peter Y.K. Cheung, Thomas J.W. Clarke
Department of Electrical and Electronic Engineering
Imperial College of Science, Technology and Medicine, London
Exhibition Road, London SW7 2BT, United Kingdom
s.hetti@ic.ac.uk, p.cheung@ic.ac.uk, t.clarke@ic.ac.uk

ABSTRACT

Data transfer intensive applications consume a significant amount of energy in memory access. The selection of a memory location from a memory array involves driving row and column select lines. A signal transition on a row select line often consumes significantly more energy than a transition on a column select line. In order to exploit this difference in energy consumption of row and column select lines, we propose a novel address assignment methodology that aims to minimize high energy row transitions by assigning spatially and temporally local data items to the same row. The problem of energy efficient address assignment has been formulated as a multi-way graph partitioning problem and solved with a heuristic. Our experiments demonstrate that our methodology achieves row transition counts very close to the optimum and that the methodology can, for some examples, reduce row transition count by 40-70% over row major mapping. Moreover, we also demonstrate that our methodology is capable of handling access sequences with over 15 million accesses in moderate time.

Categories and Subject Descriptors

B.5.1 [**Register-Transfer-Level Implementation**]: Design—*Memory design*; B.5.2 [**Register-Transfer-Level Implementation**]: Design Aids—*Automatic synthesis; Optimization*

General Terms

Algorithms, Design, Experimentation, Performance

Keywords

address assignment, data layout, memory synthesis

1. INTRODUCTION

In data transfer intensive applications, such as video and image processing, a significant fraction of the total energy consumption of the system is due to memory access [16]. Dynamic power dissipation is significant in CMOS circuits, and therefore, behavioural level energy minimization efforts often attempt to minimize signal transition counts, particularly on high capacitance nodes [13].

In order to minimize switching activity caused by memory access, it is necessary to have some knowledge about the access sequences. For many application-specific integrated circuits (ASICs), the access sequences are usually known *a priori*. ASICs may also contain data dependent access sequences. However, because the application is known, statistical information can be collected about the data dependent sequences. Information about the access sequences enable the application of energy optimizations to ASICs which would not be applicable to general purpose systems.

CMOS memory cell arrays are usually organized into rectangular blocks of memory cells. The selection of a memory location involves driving row, column and in some cases block select signals. Signal transitions on high capacitance signals such as the row and block select lines consume more energy compared to those on column select lines [3].

In this paper, we present a methodology for minimizing the energy consumption of memory access through address assignment that minimizes row switching. The novel contributions of this paper are: (1) formulation of the energy efficient address assignment problem as a multi-way graph partitioning problem; (2) application of an existing graph partitioning heuristic to solve the problem; (3) evaluation of the solution in terms of quality of the solution and run time.

This paper is organized as follows. Section 2 presents some of the previous work in the area of memory access energy minimization. Section 3 formulates the energy efficient address assignment problem as a graph partitioning problem. Section 4 contains a list of assumptions we have made in this work. Section 5 describes our address assignment methodology. Section 6 reports experimental results and Section 7 contains conclusions and indicates some possible future work.

2. PREVIOUS WORK

The relevant previous work of interest address the problem of reducing memory access energy in ASICs at the behavioural level.

In-place mapping attempts to reduce required memory size by sharing physical memory locations amongst signals whose lifetimes do not overlap [15]. Smaller memories consume less energy per access compared to larger memories [2]. Therefore, memory size minimization techniques such as in-place mapping and loop transformations [16] usually reduce energy consumption. Minimizing the number of memory accesses through loop transformations and packing several data items into a single memory word [14] also result in lower energy consumption.

Most modern memory architectures are based on memory hierarchy [2]. Memory hierarchies consist of several layers of memories. The lower layers consume less energy per access than higher layers. Although, extra transfers are introduced to copy data from higher layers to lower layers, if there is sufficient temporal locality, energy is saved due to the decrease in the number of accesses to higher layers [17].

For a given number of memory accesses energy can be further minimized by address assignment that attempts to minimize signal transition counts, particularly high energy transitions on off-chip address busses [13]. Address and data bus encoding methods can also be used to minimize switching activity on off-chip busses [11].

None of the above mentioned work exploit the fact that different select lines in the memory cell array consume different amounts of energy. Our work reported in this paper exploits this property of the cell arrays to add a complimentary energy minimization approach to the existing architectural level energy minimization techniques.

3. PROBLEM DEFINITION

The problem of minimizing the activity on the row select lines can be thought of as a problem of clustering the accessed data items such that the number of transitions between the clusters is minimized. The cluster size is no greater than the number of memory columns and the number of clusters is equal to the number of memory rows. We now define, mathematically, the minimum row switching address assignment problem as a graph partitioning problem.

Given an undirected edge-weighted graph $G(V,E)$, where vertex set $V = \{v_i | i = 0,1,\ldots,n-1\}$ is the set of vertices, E is the set of weighted edges, and positive integers p and q where $p \times q \geq |V|$, find p subsets $V_0, V_1, \ldots, V_{p-1}$ of V such that:

1. $\cup_{i=0}^{p-1} V_i = V$ and $V_i \cap V_j = \emptyset$ for $i \neq j$
2. $|V_i| \leq q$ for $i = 0,1,\ldots,p-1$
3. the *cut size*, i.e., the sum of weights of edges crossing between subsets[1], is minimized.

4. ASSUMPTIONS

The following assumptions are made in this paper:

1. A signal transition on a row select line consumes more energy than a signal transition on a column select line.
2. Reducing switching activity on high capacitance signals such as row select lines reduces energy consumption.
3. Every individual data item has been mapped to a location in a physical memory but is not bound to a physical memory address.
4. There is no preference when switching from one row to another, i.e., for example switching from row 1 to row 3 is the same as switching from 1 to 2.
5. Address generators have not yet been synthesized.
6. Memory cell arrays are rectangular. p and q represent the number of rows and the number of columns respectively. w represents the width of a memory word. p and q are known and $p \times q$ gives the total number of locations of that cell array. $p \times q \times w$ gives the total capacity of the cell array in bits.
7. Memory arrays have one read/write port.
8. The data access sequences are known.
9. Memory cells are accessed by activating row and column select lines. We ignore block select lines.
10. Each data array is assigned to a separate memory cell array. And the memory cell array is just large enough to contain the assigned data array.

[1]The sum of weights of edges crossing between subsets is equal to the number of row transitions.

Assumptions 1- 6 are necessary for the problem formulation and its solution. Assumptions 7- 10 are simplifying assumptions. These assumptions hold through out this paper unless stated otherwise.

5. ENERGY EFFICIENT ADDRESS ASSIGNMENT METHODOLOGY

5.1 Input

The input sequences contain symbolic addresses. A symbolic address identifies a unique location within the memory cell array but is not bound to any physical address. A sequence of these symbolic addresses specify the access sequence into a single port of a memory cell array. The access sequences can be extracted from an executable specification or from a control data flow graph (CDFG) of the system.

For illustration purposes let us suppose that we have the two dimensional array $A[y][x]$ whose symbolic addresses are defined as $S_i = 2y + x$ where $y = 0,1$ and $x = 0,1$. Let us also suppose that this array produces the following symbolic address sequence: $S = (0,1,2,3,1,2,0,3,1,2,0,3)$.

5.2 Transition Graph

In Section 3 we defined the problem of finding an address assignment that minimizes the activity on the row select lines as a multi-way graph partitioning problem. Therefore, the input symbolic address sequences are converted to a *transition* graph for each memory cell array. The transition graph contains information regarding the unique memory locations accessed and the number of transitions between each pair of unique memory locations.

A transition graph is created from a symbolic address sequence as follows [2]. Every unique symbolic address is mapped to a vertex in the transition graph. A transition between a pair of symbolic addresses is indicated with an undirected edge between the corresponding vertices. The edges are weighted, and the edge weight is equal to the number of transitions.

Figure 1(a) shows the transition graph for access sequence S. The numbers inside the vertices indicate the symbolic addresses and the numbers on the edges indicate the weights of the edges.

5.3 Multi-Way Graph Partitioning Heuristic

Once the transition graph for a memory cell array has been created, we require a mapping of each vertex to one of p rows such that no row contains more than q vertices.

For two-dimensional data arrays there are two simple address assignment methods: row major and column major. Figure 2 illustrates these two mapping schemes when array $A[y][x]$ is mapped to a memory with $p = 2$ and $q = 2$. Figure 1(b) and Figure 1(c) show the graph partitioning corresponding to row major and column major mappings respectively. The cut size for both row and column

[2]For an n-port memory cell array, it is necessary to consider n symbolic address sequences. However, since $n = 1$ from Assumption 7, only a single sequence is considered in this paper.

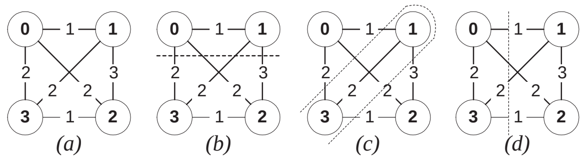

Figure 1: (a) Transition graph for the memory access sequence *S*, graph partitionings corresponding to (b) row major, (c) column major, and (d) minimal row switching assignments.

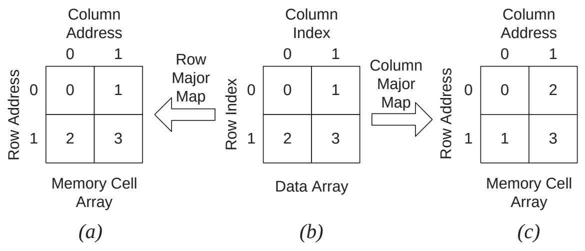

Figure 2: (a) Row major mapping of $A[y][x]$ to memory cell array (p=2 and q=2). (b) Data array $A[y][x]$ (c) Column major mapping of $A[y][x]$ to memory cell array (p=2 and q=2).

major mappings is 9. The third possible graph partitioning shown in Figure 1(d) results in a cut size of only 6.

In general, the problem can be viewed as a multi-way graph partitioning problem. Since the graph partitioning problem is NP-complete [5], even for the simplest case of graph bisection with unweighted edges and vertices, a heuristic approach is used to solve this problem.

There are many graph partitioning heuristics and software tools available [4]. Chaco graph partitioning tool developed by Hendrickson and Leland [7] was used for our application. Chaco provides several partitioning algorithms. We chose the multilevel-KL algorithm [8] as this offers low cut size for large problems in moderate time [7]. As data transfer intensive systems access large memories the ability to quickly partition large graphs into a large number of sets is very important. Chaco is controlled by a host of variables. One of the most important for our application is the KL_IMBALANCE variable. When KL_IMBALANCE is set to 0, set sizes for graphs whose vertices are not weighted vary by at most one [7]. With this guarantee we can prove that multilevel-KL algorithm meets the partitioning constraint $|V_i| \leq q$ defined in Section 3:

Proposition: If the largest and smallest set sizes, in a partitioned graph, differ by one and the size of the memory cell array that the vertices of the original graph are mapped to, is given by $p \times q$ where p is the number of rows and q is the number of columns, then for all positive integer values the largest set size will always be smaller than or equal to q.

PROOF. Let a be the number of sets of size x and b be the number of sets of size $x-1$, then because the total number of vertices has to be less than or equal to the size of the cell array we have:

$$ax + b(x-1) \leq pq \qquad (1)$$

Since p is the total number of partitions, simplifying and substituting p for $a+b$ gives:

$$x - b/p \leq q \qquad (2)$$

Now $0 \leq b/p \leq 1$ and x and q are positive integers. Therefore, for $0 \leq b/p < 1$

$$x \leq q \qquad (3)$$

Note that if $b/p = 1$ then the size of the largest set is $x-1$. And we have

$$x - 1 \leq q \qquad (4)$$

□

5.4 Output

The output is a mapping of symbolic addresses to rows of the memory cell array. It should be noted that the exact row and column addresses are not fixed after graph partitioning. In other words, graph partitioning only performs a *partial* address assignment. The row and column addresses can be sequentially assigned. However, this sequential assignment may not be optimal for address generators. There is further optimization opportunity here for reducing energy consumption in the address generators.

6. EXPERIMENTAL RESULTS

We performed experiments on several memory access sequences to evaluate our energy efficient graph based address assignment method. The total number of row transitions caused by memory accesses was used as an energy consumption metric for each of the row major, column major and graph based address assignment schemes. A row transition in a larger memory may consume more energy than a row transition in a smaller memory. However, in our experiments row transition counts are not weighted to take account of the memory size.

Memory access sequences used for our experiments are obtained from the following examples: Gauss-Seidel formula (GSR) [12], Successive Over Relaxation (SOR) [12], Compress algorithm [12], separable Discrete Cosine Transform (DCT) [1], two dimensional Convolution (Conv) [1], Lowpass [12] and image flip algorithm [10]. Some of the examples contain several data arrays, exhibiting different access sequences. From such examples we have manually selected the data array with the most number of accesses for our experiments. For simplicity, two dimensional data arrays in our examples are $K \times K$ square arrays. We varied the data array dimension K from 10 to 1000 at intervals of 10 (16 to 1000 at intervals of 8 for DCT) and studied the effects on the row transition count (RTC) when the arrays are mapped with different mapping schemes to memory cell arrays with different numbers of columns.

Table 1 shows the average percentage reductions in row switching achieved through graph based mapping over row major mapping when the data arrays are mapped to a memory with 32 columns ($q = 32$). For some examples, average reductions of 40-70% are achievable over row-major mapping. Row major mapping was chosen as the benchmark, because for all our access sequences row major mapping produced lower RTCs than column major mapping when $q = 32$. Table 1 also shows absolute numbers for row transition count when $K = 256$.

The magnitude of reduction in the row transition count over row major mapping achievable through our method depends on the particular access sequence. For some access sequences row major mapping may be optimal and our method would not yield any improvement in row transition count. Where reductions in RTC are possible the amount of reduction additionally depends on the number of columns in the memory. Figure 3 shows row transition count for SOR against number of columns when $K = 256$. When the number of columns is one, all three mapping schemes produce the same RTC. And also if the number of columns is large enough to

Example	RTC ($K = 256$) row-major	RTC ($K = 256$) this work (graph based)	Avg. % Red.
Compress	133619	57075	53.1
Conv	277287	234033	14.5
DCT	8191	2047	73.8
GSR	197339	114304	40.5
Lowpass	268731	148428	43.6
SOR	98551	48463	47.7

Table 1: Average reductions in row transition count achieved by graph partitioning address assignment. q=32.

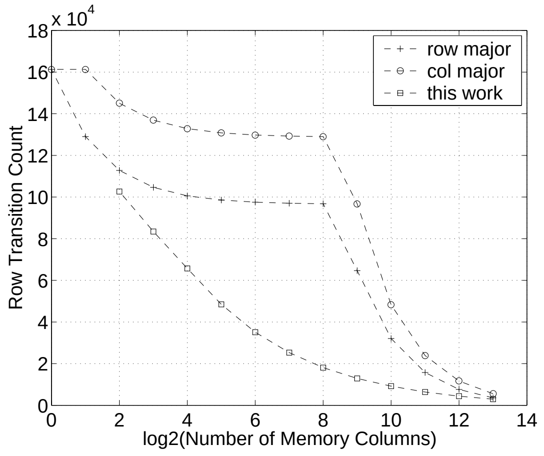

Figure 3: Row transition count against number of memory columns (q). K = 256.

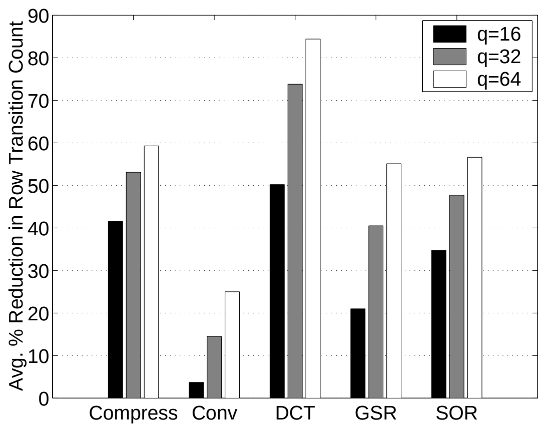

Figure 4: Average reductions in row transition count for different numbers of memory columns.

Example	sequence length (millions)	No. edges (millions)	Run Times graph part (s)	Run Times s2g (s)
Compress	5.0	3.0	105.7	39.7
Conv	15.9	3.0	123.2	82.7
DCT	8.0	1.1	155.1	57.2
GSR	5.0	5.0	138.0	55.0
Lowpass	10.0	4.0	110.1	62.0
SOR	3.5	2.5	109.3	25.3

Table 2: Run times when K = 1000 and q = 32. Number of vertices is about 1 million for all examples.

contain all the data items then RTC would be one. However, the three mapping schemes take different paths between these two convergent points as q is varied. Figure 4 shows the average percentage reduction in RTC achieved for several of our examples when q is 16,32 and 64.

Graph partitioning address assignment aims to achieve a near optimal row transition count. In order to evaluate our graph based mapping with respect to the optimal mapping we have carried out the following experiment. *source*$[i][j]$ array in the image flip algorithm shown below has a row major access sequence. Therefore, row major mapping is optimal for this access sequence.

```
$for(i = 0;\ j < K;\ i++)$
    $for(j = 0;\ j < K;\ j++)$
        $dest[i][K-1-j] = source[i][j];$
```

When *source*$[i][j]$ is row major mapped to a memory cell with $p = q = K$, this results in an RTC of p. When *source*$[i][j]$ is column major mapped, that results in an RTC of p^2, which is the maximum possible RTC for this access sequence. Our experiments have shown that for image flip example, graph based mapping on average only performs 0.08% worse than the optimum.

Table 2 shows the run times for graph partitioning and access sequence to transition graph conversion (s2g). The run times reported are for the case where $K = 1000$ and $q = 32$. For this magnitude of data array size s2g handles input access sequences of up to 16 million memory accesses. And Chaco operates on graphs containing about a million vertices and up to about five million edges. The partitioning and conversion tools were executed on a 1GHz AMD Athlon processor with 776Mbytes of memory running Linux.

7. CONCLUSION AND FUTURE WORK

In this paper we have presented a methodology for energy efficient address assignment through minimization of memory row switching. Row transition counts for many commonly found access sequences in multimedia applications can be reduced by 40%-70% over row major mapping with our methodology. We have also demonstrated that our methodology can achieve row transition counts very close to the optimum.

The methodology is directly applicable to address generator synthesis methods which require an expanded address sequence, such as counter based methods [6], shift register based methods [9] and finite state machine based methods [9].

Energy consumption in the address generators was not considered. It should be noted that the methodology presented performs a grouping of data array variables which should be assigned to the same memory row. It does not fix row and column addresses. We could simply sequentially assign row and column addresses to mapped data variables. However, this sequential assignment could be further optimized for a given address generator architecture.

Furthermore, the methodology presented can be extended to data dependent memory access sequences through the use of statistical methods for construction of the transition graph. Also our methodology can be easily extended to more complex memory organizations [3] that use block select lines in addition to row and column select lines by first assigning data variables to blocks and then to rows.

8. ACKNOWLEDGMENTS

This work was funded by LSI Logic Corporation, US and the Overseas Research Students Awards Scheme, UK. The authors would like to thank Thomas Niemann, Christos Bouganis and Andy Royal for their help with the implementation of the s2g programme and Nishanth Kulasekeram for his comments on the paper.

9. REFERENCES

[1] P. Baglietto, M. Maresca, M. Migliardi, and N.Zingirian. Image processing on high-performance RISC systems. *Proceedings of the IEEE*, 84:917–930, July 1996.

[2] L. Benini and G. de Micheli. System-level power optimization: Techniques and tools. *ACM Transations on Design Automation of Electronic Systems*, 5(2):115–192, Apr. 2000.

[3] R. J. Evans and P. D. Franzon. Energy consumption modeling and optimization for SRAM's. *IEEE Journal of Solid-State Circuits*, 30:571 – 579, May 1995.

[4] P. Fjallstrom. Algorithms for graph partitioning: A survey, 1998.

[5] M. Garey, D. Johnson, and L. Stockmeyer. Some simplified NP-complete graph problem. *Theoretical computer science*, 1:237–267, 1976.

[6] D. M. Grant and P. B. Denyer. Address generation for array access based on modulus m counters. In *Proceedings of the European Design Automation Conference*, pages 118 – 122, Feb. 1991.

[7] B. Hendrickson and R. Leland. The Chaco user's guide, version 2.0. Technical Report SAND95-2344, Sandia National Laboratories, July 1995. User Manual.

[8] B. Hendrickson and R. Leland. A multilevel algorithm for partitioning graphs. In *Proceedings of the conference on Supercomputing*, 1995.

[9] S. Hettiaratchi, P. Y. Cheung, and T. J. Clarke. Performance-area trade-off of address generators for address decoder-decoupled memory. In *Proceedings of the Design Automation and Test in Europe Conference*, pages 902–908, Mar. 2002.

[10] H. R. Myler and A. R. Weeks. *Computer Imaging Recipes in C*. Prentice-Hall, Englewood Cliffs, New Jersey, 1993.

[11] P. R. Panda, F. Catthoor, N. D. Dutt, K. Danckaert, E. Brockmeyer, C. Kulkarni, A. Vandercappelle, and P. G. Kjeldsberg. Data and memory optimization techniques for embedded systems. *ACM Transactions on Design Automation of Electronic Systems*, 6(2):149 – 206, Apr. 2001.

[12] P. R. Panda and N. Dutt. Low power memory mapping through reducing address bus activity. Technical Report 95-32, University of California, Irvine, Nov. 1995.

[13] P. R. Panda and N. D. Dutt. Low-power memory mapping through reducing address bus activity. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 7(3):309–319, Sept. 1999.

[14] H. Schmit and D. E. Thomas. Synthesis of application-specific memory designs. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 5(1):101–111, Mar. 1997.

[15] I. Verbauwhede, F. Catthoor, J. Vandewalle, and H. D. Man. Background memory management for the synthesis of algebraic algorithms on multi-processor DSP chips. In *Proceedings of the International Conference on VLSI*, pages 209–218, Aug. 1989.

[16] S. Wuytack, F. Catthoor, F. Franssen, , L. Nachtergaele, and H. D. Man. Global communication and memory optimizing transformations for low power systems. In *IEEE International Workshop on Low Power Design*, pages 203–208, Apr. 1994.

[17] S. Wuytack, Jean-Philippe, F. V. Catthoor, and H. J. D. Man. Formalized methodology for data reuse exploration for low-power hierarchical memory mappings. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 6(4):529–537, Dec. 1998.