LOW POWER MULTI-MODULE, MULTI-PORT MEMORY DESIGN FOR EMBEDDED SYSTEMS

Wen-Tsong Shiue
Electrical Engr. Dept.
Arizona State University
Tempe, AZ

Shashikiran Tadas
Electrical Engr. Dept.
Arizona State University
Tempe, AZ

Chaitali Chakrabarti
Electrical Engr. Dept.
Arizona State University
Tempe, AZ

Abstract – In this paper we describe a multi-module, multi-port memory design procedure that satisfies area and/or energy constraints. Our procedure consists of use of ILP models and heuristic-based algorithms to determine (a) the memory configuration with minimum area, given the energy bound, (b) the memory configuration with minimum energy, given the area bound, (c) array allocation such that the energy consumption is minimum for a given memory configuration (number of modules, size and number of ports per module). The results obtained by the heuristics match very well with those obtained by the ILP methods.

INTRODUCTION

Minimization of energy is one of the most important performance metrics in the design of portable systems. Traditionally, the main focus has been to reduce the energy consumption in the datapath components. However, in systems that involve multidimensional streams of signals such as images or video sequences, the majority of the area and energy cost is not due to the data-path or controllers but due to the global communication and memory interactions [1]. This is due to the fact that energy consumed in memory transfers is significantly larger than that due to data-path operations. This implies that with proper design, reduction in the memory related energy budget can far exceed the reduction due to voltage scaling and other energy saving transformations.

In some applications, assuming that the on-chip memory is a single module, single port memory is restrictive. In fact, some applications demand data parallel access capabilities which are necessary to complete the computations within a cycle budget. Such capabilities can only be supported by multi-port, multi-module memories.

In this paper we determine the memory configuration and the allocation of arrays to the different memory modules such that the energy and/or area constraints are satisfied. Given high-level code, area and energy models, and the system constraints of area, time, and energy, our aim is to find the best memory configuration (and array assignment). We develop ILP-based models and heuristics to address the following three problems:

- Determine the memory configuration (and array assignment) with minimum area, given the energy bound.
- Determine the memory configuration (and array assignment) with minimum energy, given the area bound.

0-7803-6488-0/00/$10.00 © 2000 IEEE

- Determine the allocation of arrays onto a fixed memory configuration (number of modules, size and number of ports per module is given) such that the energy consumption is minimum.

The heuristic-based algorithms are very efficient since they look at a small subset of configurations. The results obtained by our heuristics match very well with those obtained by the ILP models.

Several techniques to minimize the number of memory modules and determine which variables/arrays get grouped together exist in [2][3]. Methods to group scalar variables are either based on greedy heuristics [3] or formulated using the 0-1 integer linear programming [3]. Procedures for grouping arrays and assigning them to different memory modules has been presented in [4][5]. The technique presented in [4] is comprehensive. It creates an extended conflict graph (ECG) whose nodes represent arrays and whose edges represent compatible groups of arrays (those that can be mapped to the same physical memory). A search procedure based on area and power costs is used to determine the 'best' memory configuration. The method in [5] finds the best configuration by varying the number of modules and studying the tradeoffs between area and delay for different array assignments in each configuration.

The rest of the paper is organized as follows. Section 2 describes the basic terms, the area, energy models and the area-energy tradeoffs. Section 3 describes (i) how to find the best memory configuration and array assignment given area/energy bound and (ii) energy-efficient array assignment given the memory configuration. Section 4 concludes the paper.

BACKGROUND

Extended Conflict Graph (ECG) and Chromatic Number

Array	A	B	C	D	E	F	G
Size	100x8	200x32	100x16	100x16	200x8	300x32	100x16
Reads	100	300	300	100	0	0	100
Writes	100	200	0	0	200	300	100

(a)

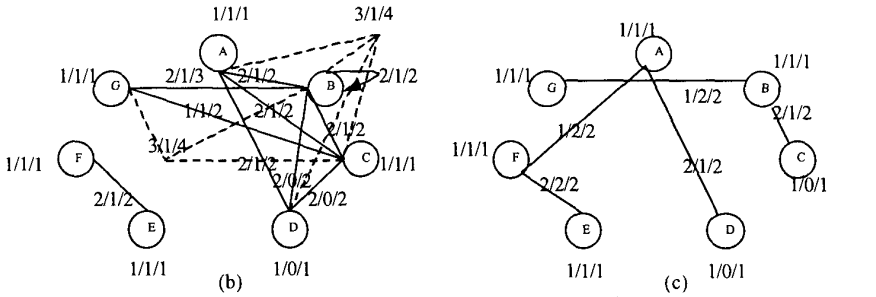


Figure 1. Example illustrating (a) size and number of accesses of each array, (b) initial ECG, and (c) final ECG after incorporating SBO technique.

A conflict graph consists of a set of nodes that correspond to the arrays and there is an edge between nodes whenever the corresponding arrays are in conflict. An extended conflict graph contains additional information regarding read, write and read/write ports, and also about self-conflict and hyper edges. If more than two arrays are accessed in the same cycle then the corresponding nodes are linked by a hyper edge. In Figure 1(b), there are hyper edges between nodes (A,B,C,D) and between (B,C,G). These edges are shown by dashed lines. If the same array is being accessed more than once in the same cycle, then it has a self-conflict. In Figure 1(b), there is a self-conflict edge on node B. If there are self-conflict and hyper edges in an ECG, then the conflicting arrays have to be assigned to either different memories or to a multiport memory. In such cases, if only single-port modules are available, then the number of modules is equal to the chromatic number of the extended conflict graph. If multi-port modules are available, then the number of ports should be larger than the chromatic number of the ECG. For instance, since the chromatic number of the ECG in Figure 1(b) is 4, we need 4 single-port memory modules. The number of self-conflict edges, hyper edges, and chromatic number can be reduced by applying the storage bandwidth optimization (SBO) technique of [4]. Figure 1(c) shows the ECG after the SBO technique has been applied to Figure 1(b).

Area and Energy Models

In order to compare the different memory configurations, it is necessary to develop good area and energy models. The area model that we developed is based on Mulder's model [6] and modified according to the data points in Wuytack et al's paper [4] for 1.2μm CMOS technology. Given the bit width, word length and R/W/RW port-information, we can calculate the area for a multi-port module as follows.

$A=TF*bits*(1+\alpha*p1)*sqrt(words)*PF*K;$ [unit:mm^2]
where $PF=[1+0.25*(p1+p2-2)]$; $p1=R+W$; $p2=RW$;

A	: Area.
TF	: technology scaling factor (min-geometry[μ]/2)^2.
PF	: empirical factor for the number of ports in SRAM.
words	: the number of words.
bits	: the width of a word in bits.
p1	: the total number of single-end ports.
p2	: the total number of ports.
α	: 0.1
K	: 3.9174e-2

We also developed an energy model that is a modified version of the model in [7] for multi-port on-chip memory architectures. Given the bit/word length of arrays, read/write accesses, the clock frequency, and the number of ports, we can calculate the energy consumption using this model.

E=(Eread+Ewrite)*ports*1e6; [unit: uJ]
where
Cread =(9707+108*words+1126*bits+6*words*bits)*1e-15;
Cwrite =(7994+117*words+759*bits+9*words*bits)*1e-15;
Eread =0.5*Vdd^2*Cread*Raccess; (Here Vdd=5V).

Ewrite =0.5*Vdd^2*Cwrite*Waccess;
E : energy.
Eread : energy due to read.
Ewrite : energy due to write.
Raccess : number of read accesses.
Waccess : number of write accesses.
words : the number of words.
bits : the width of a word in bits.
ports : the total number of ports.

Area-Energy Tradeoff

Since small memory modules consume less energy, assigning individual arrays to separate memory modules results in the smallest energy consumption. However this configuration results in a large area. Figure 2(a) describes this configuration for the example in Figure 1(c). Assigning arrays with the same bit width to the same module is very area-efficient as shown in Figure 2(d). The energy consumption of this configuration is however larger than Figure 2(a), as is expected. Configurations that illustrate the area-energy tradeoffs are shown in Figure 2(b) and 2(c).

Grouping all the arrays in to a single memory module results in both large energy consumption and area as shown in Figure 2(e). This is because the resulting module has two ports and grouping arrays with different bit widths results in loss of memory.

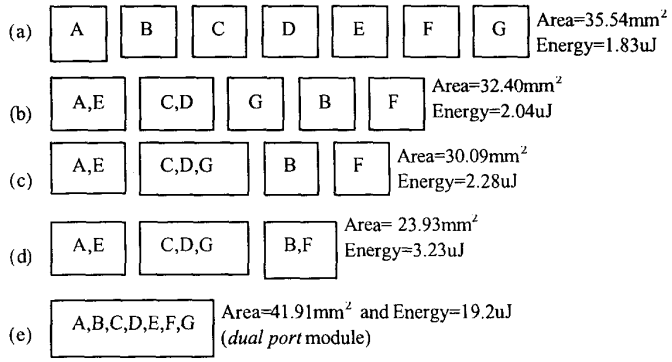


Figure 2. Area and energy values for different memory configurations for the example in Figure 1(c).

CHOOSING BEST MEMORY CONFIGURATION AND ARRAY ASSIGNMENT

Problem 1: Find the memory configuration with minimum area, given the energy bound.

ILP Model

The ILP model is used to find the number of modules and the size of each module such that the area is minimum and the energy constraint is satisfied. This translates

to finding which arrays are housed in each module. We refer to the set of arrays that are assigned in each module as an array grouping. The first step in the procedure is to identify the permissible array groupings. The arrays in an array grouping are assigned to the same module. For instance, for the example in Figure 3, the permissible array groupings are {A}, {B}, {C}, {D}, {AC}, and {AD}.

The next step is to derive the ILP model. Let x_i be a {0,1} integer which assumes a value 1 if array grouping i is chosen. Let S_i be the area cost of array grouping i. Then the objective is to minimize $\Sigma_i S_i x_i$ (where i=1,N), given the energy bound, where N is the number of permissible groupings. The constraint equations consist of (i) array constraint that ensures that an array can be assigned to only one of the modules, (ii) 0-1 constraint that ensures that x_i is either 0 or 1, and (iii) energy constraint that ensures that the energy of the assignment satisfies the energy bound.

<ILP Model>

S: cost of area, T: cost of energy, Y_{bound}= energy bound.
N: number of possibilities, N_a: number of arrays.

(i) Objective function:

$$\min : \sum_{i=1}^{N} S_i x_i;$$

(iii) 0-1 constraint:

$$\forall 1 \le i \le N, x_i \le 1;$$

(v) Integer constraint:

$$\forall 1 \le i \le N, \text{int } x_i;$$

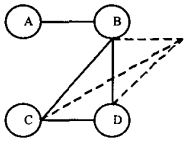
(ii) Array constraint:

$$\forall 1 \le j \le N_a, \sum_{i=1}^{N} x_i = 1;$$

(iv) Energy constraint:

$$\sum_{i=1}^{N} T_i x_i \le Y_{bound};$$

Figure 3 traces the steps of the ILP based procedure on an example. For Y_{bound} =0.45uJ, the ILP solver finds three array groupings, x2, x4 and x5 to be equal to 1. Thus the final assignment has three memory modules with {B} in one module, {D} in one module and {A} and {C} in another module. The area of this assignment is 10.24mm^2 and the energy is 0.41uJ while is less than the energy bound.



A=100 x 8 (100 reads, 0 writes)
B=200 x16 (0 reads, 200 writes)
C=50 x 8 (50 reads, 50 writes)
D=100 x 16 (150 reads, 50 writes)

(a)

Variables	Allowed groupings	Area (mm^2)	Energy (uJ)
X1	A	1.69	0.04
X2	B	4.79	0.18
X3	C	1.2	0.03
X4	AC	2.07	0.11
X5	D	3.38	0.12
X6	AD	4.79	0.26

(b)

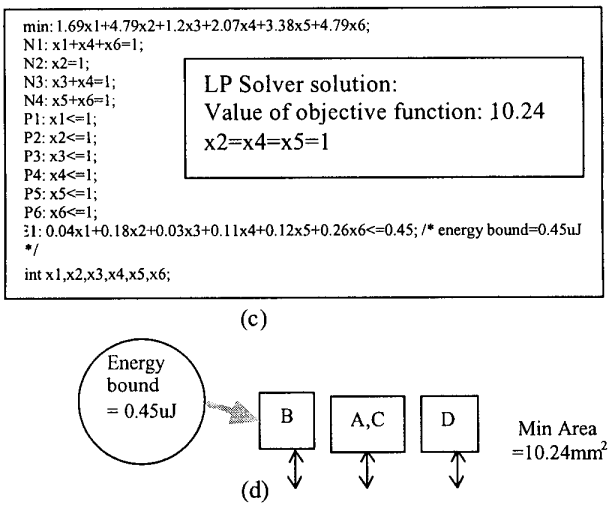


Figure 3. Example illustrating the ILP model. (a) ECG, (b) Area and energy costs of the allowed groupings, (c) ILP formulation, and (d) Chosen configuration.

Heuristic Algorithm

The algorithm iteratively finds the memory configuration with minimum area given the energy bound by either grouping arrays or splitting already grouped arrays based on whether the energy bound is closer to the minimum area configuration or the minimum energy configuration. Let U correspond to the configuration with minimum area (obtained by assigning arrays with same bit width to the same memory module) and maximum energy, and L correspond to the configuration with minimum energy (obtained by assigning each array to a separate module) and maximum area. Let X be the desired configuration and E(X) the given energy bound.

The algorithm decides whether to group arrays or split already grouped arrays based on the value of E(X) with respect to E(L) and E(U). If E(X) is closer to E(L) (than E(U)), we group arrays. On the other hand, if E(X) is closer to E(U), we split already grouped arrays. To help in deciding which arrays should be grouped or split, the arrays in the candidate set R are prioritized. If E(X) is very close to E(L), then arrays with lower energy value (lower area value if their energy values are equal) have higher priority to be fused. If E(X) is close to E(L) but not very close, then arrays with higher energy value (lower area value if their energy values are equal) have higher priority to be fused. If E(X) is very close to E(U), then arrays with lower energy value (higher area value if their energy values are equal) have higher priority to be split. If E(X) is close to E(L) but not very close, then arrays with higher energy value (higher area value if their energy values are equal) have higher priority to be split.

After a successful grouping (energy < E(X)), candidate set R and configuration L are updated. Similarly after a successful split (energy > E(X)), set R and configuration U are updated. If the grouping is unsuccessful (energy > E(X)), we

evaluate the following possible configurations and pick the one which is the closest to the energy bound: (i) configuration A obtained by splitting the second most recently merged group in the current configuration, (ii) previous configuration B, and (iii) configuration C obtained by fusing arrays in B with the lowest priority in set R. Similarly, if the splitting is unsuccessful (energy is a lot less than E(X)), we evaluate the following configurations and pick the one which is the closest to the energy bound: (i) current configuration A, (ii) configuration B obtained by fusing the second most recently split group in A, and (iii) configuration C obtained by splitting arrays in previous configuration with the lowest priority in set R. The procedure is iterative and continues until the set R is empty.

We illustrate this with the example in Figure 4 for an energy bound, E(X) = 2.5uJ. U corresponds to the following groupings {(A,E),(C,D,G),(B,F)} and has an energy of E(U)=3.2275uJ. L corresponds to the groupings {A,B,C,D,E,F,G} and has an energy of E(L) =1.8289uJ. The values of d1 is E(U) – E(X)= 0.7275 and the value of d2 is E(X) – E(L) = 0.6711. Since d1 >= d2 and d2/(d1+d2) >= ¼, we first fuse arrays B, F (with high priority) in set R. The energy for this configuration {(BF),C,E,G,A,D} is 2.7739uJ, is larger than E(X). So we go back to previous solution and remove arrays B and F in set R. Now R={C,E,G,A,D}. We fuse arrays C and G since they are of the same bit width. The energy for the configuration {B,F,(CG),E,A,D} is 1.9634uJ, that is smaller than E(X). We update the value of E(L) to 1.9634uJ and remove the arrays C and G in set R. Now R={E,A,D}. The process is repeated. The final configuration is {B,F,(CGD),(EA)} with an energy of 2.2825uJ < 2.5uJ. Figure 4 describes the algorithm trace for this example.

We simulate the example in Figure 1(c) for different energy bounds. The results are given in Figure 5. Note that the results obtained by the heuristic match very well with those obtained by the ILP method.

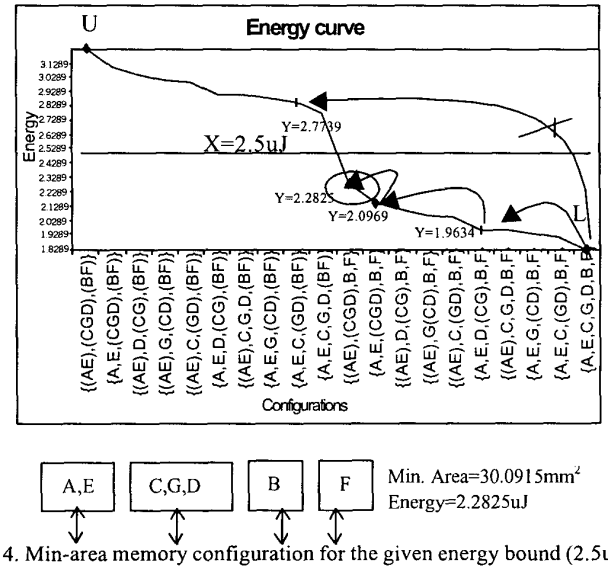


Figure 4. Min-area memory configuration for the given energy bound (2.5uJ).

Energy bound	Results of Heuristic algorithm (Approximate solutions)		Results of ILP (Optimal solutions)		% error
	Configurations	Area (mm^2)	Configurations	Area (mm^2)	
3.5	(AE),(CGD),(BF)	23.9301	(AE),(CGD),(BF)	23.9301	0
3.2	A,E,(CGD),(BF)	25.0845	A,E,(CGD),(BF)	25.0845	0
2.8	A,E,C,G,D,(BF)	29.3759	A,E,C,G,D,(BF)	29.3759	0
2.5	(AE),(CGD),B,F	30.0915	(AE),(CGD),B,F	30.0915	0
2.1	(AE),C,(GD),B,F	32.4003	(AE),C,(GD),B,F	32.4003	0
1.95	A,E,C,(GD),B,F	33.5547	A,E,G,(CD),B,F	33.5547	0
1.9	A,E,C,G,D,B,F	35.5373	A,E,C,G,D,B,F	35.5373	0

Figure 5. Minimum area memory configuration given energy bound for the example in Figure 1(c).

Problem 2: Find the memory configuration with minimum energy, given the area bound.

The formulation of this problem (both ILP and heuristic) is very similar to that of Problem 1 and we will not discuss it here. For the example in Figure 1(c), we quote the results obtained by the heuristic and the ILP formulations for different area bounds in Figure 6. A maximum error of 2.5% indicates that our result from heuristic algorithm is very close to that from ILP model.

Area bound	Results of Heuristic algorithm (Approximate solutions)		Results of ILP (Optimal solutions)		% error
	Configurations	Energy (uJ)	Configurations	Energy (uJ)	
34.5	(AE),C,G,D,B,F	1.9624	(DG),C,A,E,B,F	1.9124	2.5
32.5	(AE),C,(GD),B,F	2.0459	(AE),C,(GD),B,F	2.0459	0
31.5	A,E,(CGD),B,F	2.1490	A,E,(CGD),B,F	2.1490	0
30.5	(AE),(CGD),B,F	2.2825	(AE),(CGD),B,F	2.2825	0
29.0	(AE),C,G,D,(BF)	2.9074	A,E,C,(GD),(BF)	2.8574	1.7
27.5	A,E,C,(GD),(BF)	2.8574	A,E,C,(GD),(BF)	2.8574	0
25.5	A,E,(CGD),(BF)	3.0940	A,E,(CGD),(BF)	3.0940	0

Figure 6. Minimum energy memory configuration given area bound for the example in Figure 1(c).

Problem 3: Map the arrays to given single-/multi-port memory modules such that the energy consumption is minimum, given the memory configuration (number of modules, size and the number of ports per module).

Heuristic Algorithm

The proposed algorithm tries to assign frequently accessed arrays to modules that are small in size and/or have few ports. This is because the energy consumption is a function of the size of the memory, number of ports and the number of accesses. Specifically,

Energy = (Eread+Ewrite)*ports = (Cread*Raccess+Cwrite*Waccess)*ports
= ports*Cread*(Raccess+Ratio*Waccess) = Cr*y
where Cr= ports*Cread, y= Raccess+Ratio*Waccess, Ratio= Cwrite/Cread ≅ 1.2.

Thus the parameter Cr is a function of the size and the number of ports, and the parameter y is a function of the number of accesses. Note that while the ratio Cwrite/Cread does not vary with memory size, Cread is a function of the memory size (number of words and bit width).

The heuristic algorithm operates on R, a set of arrays that are prioritized based on the value of y, and S, a set of modules that are prioritized based on the value of Cr. The algorithm is greedy and tries to assign an array with a high value of y to a module with a low value of Cr. An assignment is valid only if (i) there is enough space in the module to house that array and (ii) the limitations imposed by the conflict graph matches the number of available ports. For instance, if array A has conflict with an array that has already been housed in the high priority module, M, and if M has a single port, then A can not be assigned to M. If, on the other hand, M has two ports, then A can be assigned to M. Finally, if the bitwidth of the array, ba, is larger than that of the module, bm, then multiple memory words (i.e. $\lceil (ba/bm) \rceil$ words) have to be dedicated for each array entry. If ba < bm, then the memory is under-utilized. A solution is not feasible if the number of arrays connected by the largest hyper edge is larger than the total number of ports.

Algorithm: Array Assignment (given memory configuration)

(i) R = the set of arrays to be assigned. List the priority of arrays in set R based on the value of y. Higher value of y has higher priority, where y is related to the number of memory accesses.

(ii) S = the set of modules. List the priority of modules in set S based on the value of Cr. Smaller value of Cr has higher priority, where Cr is related to the size and the number of ports.

(iii) Repeat till set R is empty.
- Assign the array with higher priority in set R to a module with higher priority in set S only (a) there is enough space to house the corresponding and (b) it satisfies the extended conflict graph (ECG) constraints.
- Remove the visited array from set R.
- Update the size of the visited module.

ILP Model

We also develop a simple ILP model based on the ball-hole assignment problem. Our objective is to minimize the energy consumption which is a function of the number of memories, the number of ports, and the memory size. Our constraint equations consist of (i) array constraint that ensures an array (ball) can be assigned to only one of the modules (holes), (ii) edge constraint that ensures that the ECG edge constraints and the number of ports match, and (iii) size constraint that ensures that the sum of the sizes of the assigned arrays is less than or equal to the module size. The ILP model can be formulated as follows. Let x_{ij} be an {0,1} integer value, which assumes a value of 1 if array i is assigned to module j and is 0 otherwise and the number of ports of module j.

In Figure 7, we show the array assignment for different memory configurations for the example in Figure 1(c). For these examples, the results of the final assignment are the same as that obtained by the heuristics for algorithm of array

assignment. Note that the frequently access arrays (i.e. arrays with higher y) are always assigned to the module with smaller Cr (which is a function of size and number of ports).

Given modules	Size (mm^2)	Ports	Final assignment	Energy (uJ)
M1	22	2	{B,C,F,G}	4.04
M2	2	1	{A}	0.08
M3	5	1	{D,E}	0.26
M1	21	2	{B,E,F}	3.02
M2	5	2	{A,D}	.31
M3	5	1	{C,G}	.43
M1	29	3	{B,C,E,F,G}	5.06
M2	7	2	{A,D}	0.26

Figure 7. Results of array assignment for different memory configurations for the example in Figure 1(c).

CONCLUSION

In this paper, we have developed ILP and heuristic-based procedures to design multi-port, multi-module memory configurations that satisfy area and/or energy constraints. We have assumed that the entire memory is on chip and that two arrays do not share memory space even if they have non-overlapping life times. Currently, we are extending our method to handle non-overlapping arrays.

REFERENCES

[1] F. Catthoor, S. Wuytack, E De Greef, F. Balasa, L. Nachtergaele, and A. Vandecappelle, "Custom Memory Management Methodology – Exploration of Memory Organization for Embedded Multimedia System Design," Norwell, MA: Kluwer, 1998.

[2] M. Balakrishnan, A. Majmudar, D. Banerji, J. Linders, and J. Majithia, "Allocation of Multiport Memories in Data Path Synthesis," IEEE Transactions on CAD, vol. 7, no. 4, pp. 536-540, April 1988.

[3] P. Lippens, J. van Meerbergen, W. Verhaegh, and A. van der Werf, "Allocation of Multiport Memories for Hierarchical Data Streams," IEEE International Conference on Computer Aided Design, CA, Nov. 1993.

[4] S. Wuytack, F. Catthoor, G. de Jong, and Hugo De Man, "Minimizing the Required Memory Bandwidth in VLSI System Realizations," IEEE Transactions on VLSI Systems, Vol. 7, No. 4, Dec. 1999.

[5] P. R. Panda, "Memory Bank Customization and Assignment in Behavioral Synthesis," IEEE International Conference on Computer Aid Design, 1999.

[6] J. M. Mulder, N. T. Quach, and M. J. Flynn, "An Area Model for On-chip Memories and its application," IEEE Journal on Solid-State Circuits, Vol. 26. Pages 98-105, Feb. 1991.

[7] D. Lanneer, M. Cornero, G. Goosesens, H. De Man, Data Routing: A Paradigm for Efficient Data-path Synthesis and Code Generation," Proc. 7th ACM/IEEE International Symposium on High-Level Synthesis, pages 17-22, May 1994.