Optimal Topology Exploration for Application-Specific 3D Architectures *

Ozcan Ozturk, Feng Wang, Mahmut Kandemir, and Yuan Xie
Computer Science and Engineering Department
Pennsylvania State University
e-mail: {ozturk,fenwang,kandemir,yuanxie}@cse.psu.edu

Abstract— As technology scales, increasing interconnect costs make it necessary to consider alternate ways of building integrated circuits. One promising option along this direction is 3D architectures where a stack of multiple device layers, with direct vertical tunneling through them, are put together on the same chip. In this paper, we explore how processor cores and storage blocks can be placed in a 3D architecture to minimize data access costs under temperature constraints. This process is referred to as the topology exploration. Using integer linear programming, we compare the best 2D placement with the best 3D placement, and show through experiments with both single-core and multi-core systems that the 3D placement generates much better results (in terms of data access costs) under the same temperature bounds. We also discuss the tradeoffs between temperature constraint and data access costs.

I. Introduction

As technology scales, the International Technology Roadmap for Semiconductors projects that on-chip communications will require new design approaches to achieve system level performance targets [15]. Three-dimensional integrated circuits (3D ICs) [6, 10, 11, 12] are attractive options for overcoming the barriers in interconnect scaling, offering an opportunity to continue the CMOS performance trend. In a three-dimensional (3D) chip, multiple device layers are stacked together with direct vertical interconnects tunneling through them. Consequently, one of the most important benefits of a 3D chip over a traditional two-dimensional (2D) design is the reduction on global interconnect. Other benefits of 3D ICs include: (i) higher packing density and smaller footprint due to the addition of a third dimension to the conventional two-dimensional layout; (ii) higher performance due to reduced average interconnect length; (iii) lower interconnect power consumption due to the reduction in total wiring length; and (iv) support for realization of mixed-technology chips.

As heat is generated by the power dissipated on the chip, the on-chip junction temperatures also increase, resulting in higher cooling/packaging costs, acceleration of failure mechanisms, and degradation of performance. For 3D ICs, the thermal issues are even more pronounced than the 2D designs due to higher packing density, especially for the inner layer of the die. It is often considered as a major hindrance for 3D integration [6]. For example, temperature increases force the design to slow down to cool the chip, such that the actual performance benefits from reducing global interconnect could be offset. Therefore, thermal-aware design is very critical to extract the maximum benefits from the 3D integration.

As technology moves towards 3D designs, one of the challenging problems in the context of multi-core systems is the placement of processor cores and storage blocks across the multiple layers available. This is a critical problem as both power and performance behavior of a design are significantly influenced by the data communication (data access) distances between the processor cores and storage blocks. In particular, if a computation block (e.g., a processor core) frequently accesses data from certain storage blocks, these storage blocks should be placed into positions close (in vertical or horizontal sense) to that processor core. Similarly, in a multi-core design, if two cores frequently share certain data residing in a given storage block, that storage block should be put close to both these cores to minimize the data communication distances, thereby, improving potentially both performance and power consumption.

The important point to note, however, is that each embedded application can require a different placement of processor cores and storage blocks for achieving the minimum data communication distances. Therefore, in this paper, our focus is on *application-specific placement* of processor cores and storage blocks in a 3D design space. For this purpose, we propose an integer linear programming (ILP) based processor core/storage block placement for single-core and multi-core embedded designs. This placement problem is also referred to as topology exploration. While ILP based solutions are known to take large solution times, this issue does not seem to be very pressing in our case because (i) we design a customized placement for a given embedded application, and thus, can afford large solution times as design quality is of utmost importance, and (ii) since a given design typically has only a small number of processor cores and storage blocks, the solution times can be kept under control.

We implemented our ILP based solution within a commercial solver [23], and performed experiments with different types of applications. The first group of applications are a set of six sequential Spec2000 codes and used for single-core designs in this work. The second group of applications, on the other hand, are for multi-core designs and include four array-based codes parallelized through an optimizing compiler. Our experiments that consider a set of cache lines as a storage block reveal two important results: (i) the best 3D designs significantly outperform the best 2D designs (also obtained using ILP) for a given application under the same temperature constraint for both single-core and multi-core applications, and

*This work is supported in part by NSF Career Award #0093082 and by a grant from GSRC.

0-7803-9451-8/06/$20.00

(ii) optimized placement of blocks is very important in the 3D domain as the difference between the optimized and random core/storage block placements (in 3D) is very significant in all the cases tested.

The rest of this paper is organized as follows. The next section discusses the related work on 3D and customized memory design. Section III explains the thermal model used in this work. Section IV presents the details of our ILP formulation, and Section V shows the working of our approach through an example. Section VI gives an experimental evaluation of our approach. Finally, Section VII presents our concluding remarks.

II. Related Work

We discuss the related efforts in two categories: 3D design and customized memory design for embedded systems. Design techniques and methodologies for 3D architectures have been investigated to efficiently exploit the benefits of 3D technologies. Recent efforts have focused on developing tools for supporting custom 3D layouts and placement tools [11]. In Deng et al [12], the technology and testing issues are surveyed and a 3D integration framework is presented. However, the investigation of the benefits of 3D design at the architectural level is still in its infancy. Das et al [10] study the energy and thermal behavior of 3D designs under a supplied time constraint, and their tool is based on a standard-cell circuit layout. A recent paper provides an overview of the potential benefits of designing an Itanium processor in the 3D technology [6]. However, it does not provide details of the design of the individual components. Several recent efforts also study employment of 3D designs in reconfigurable fabrics [1, 3]. Our work is different from all these prior efforts as we are interested in application-specific placement of processor cores and storage blocks in 3D.

Memory system design and optimization has been a popular area of research. The prior work mostly focus on single processor based systems [5, 8, 9, 18, 19, 20]. An embedded system consisting of a VLIW processor, instruction cache, data cache, and second-level unified cache has been investigated by Abraham and Mahlke [2]. A hierarchical approach has been used to partition the system into components. Meftali et al [16] attack the memory space allocation (partitioning) problem. In their work, an integer linear programming model has been applied to obtain an optimal distributed shared memory architecture. The objective is to minimize the global cost to access the shared data in the application and the memory cost. A packet routing switch example has been used to test the effectiveness of the proposed approach. Embedded memory design for application-specific multi-core system-on-chips has been investigated by Gharsalli et al [13]. In this methodology, they try to integrate the standard memory components. Our work is different from these prior efforts discussed above since we focus on a 3D architecture and study optimal block placement under temperature constraints.

III. 3D Thermal Model

In order to facilitate the thermal-aware processor core/storage block placement process, a compact thermal model is needed to provide the temperature profile. Numerical computing methods (such as finite difference method (FDM) [22]) are very accurate but computationally intensive, while the simplified close-form formula [14] is very fast but inaccurate. Skadron et al proposed a thermal model called Hotspot [21], which is based on lumped thermal resistances and thermal capacitances. It is more efficient than the prior low-level approaches since the variances at temperature are tracked at a granularity of functional block level. In our research, we use a simplified analytical model called the 3D resistor mesh (shown in Figure 1), which is similar to the approach taken by Hotspot, to facilitate the thermal analysis. The model employs the principle of thermal-electrical duality to enable efficient computation of the thermal effects at the block level. The transfer thermal resistance $R_{i,j}$ of block i with respect to block j can be defined as the temperature rise at the block i due to one unit of power dissipated at block j:

$$R_{i,j} = \frac{\Delta T_{i,j}}{\Delta P_{i,j}}.$$

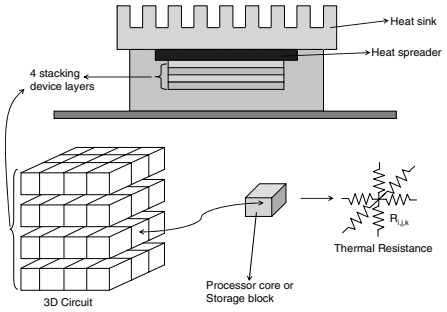

Fig. 1. 3D resistor mesh model.

In this simplified model, the device layers are stacked together with the heat sink as the bottom layer, and each device layer consists of the blocks, which are the storage (cache memory) blocks or the processor core(s). These blocks define the thermal nodes of the thermal resistor mesh. (For more details, the reader is referred to [21].) A 3D resistor mesh consists of vertical and lateral thermal resistors, which model the heat flow from wafer to wafer, wafer to heatsink, and heat transferring between the blocks. These vertical and lateral resistances are calculated based on Hotspot [21].

After the 3D thermal resistance network has been determined, the temperature rise at each block can be estimated by solving the following equation:

$$T = R \times P,$$

where T is the temperature of each block, R is a matrix of thermal resistances, and P is the power consumption of each block. One of the nice properties of this high-level abstraction of temperature behavior is that it can easily be embedded within an ILP-based opimization framework (since it is linear), as will be discussed in the next section.

IV. ILP Formulation

Our goal in this section is to present an ILP formulation of the problem of minimizing data communication cost of a given

TABLE I
THE CONSTANT TERMS USED IN OUR ILP FORMULATION. THESE ARE EITHER ARCHITECTURE SPECIFIC OR PROGRAM SPECIFIC. NOTE THAT C_Z CAPTURES THE NUMBER OF LAYERS IN THE 3D DESIGN. BY SETTING C_Z TO 1, WE CAN MODEL A CONVENTIONAL 2D (SINGLE LAYER) DESIGN AS WELL. THE VALUES OF $FREQ_{p,m}$ ARE OBTAINED BY COLLECTING STATISTICS THROUGH SIMULATING THE CODE AND CAPTURING ACCESSES TO EACH STORAGE BLOCK.

Constant	Definition
P	Number of processor cores
M	Number of storage blocks
C_X, C_Y, C_Z	Dimensions of the chip
P_X, P_Y	Dimensions of a processor core
$SIZE_m$	Size of a storage block m
$FREQ_{p,m}$	Number of accesses to storage block m by processor p
$R_{l,v}$	Thermal resistance network
T_B	Temperature bound

application by determining the optimal placement of storage blocks and processor cores under a given temperature bound. A storage block in this paper corresponds to a set of consecutive cache lines. The data cache is assumed to be divided into *storage blocks* of equal size. In this paper, our focus is on the data cache only; however, the proposed approach can be applied to the instruction cache as well.

ILP provides a set of techniques that solve those optimization problems in which both the objective function and constraints are linear functions and the solution variables are restricted to be integers. The 0-1 ILP is an ILP problem in which each (solution) variable is restricted to be either 0 or 1 [17]. Table I gives the constant terms used in our ILP formulation. We used *Xpress-MP* [23], a commercial tool, to formulate and solve our ILP problem, though its choice is orthogonal to the focus of this paper. In our ILP formulation, we view the chip area as a 3D grid, and assign storage blocks and cores into this grid.

Assuming that P denotes the total number of processor cores, M the total number of storage blocks, (C_X, C_Y, C_Z) the dimensions of the chip, (P_X, P_Y) the dimensions of the processor core, our approach uses 0-1 variables to specify the coordinates of each storage block and processor core.

We use PC to identify the coordinates of a processor core. More specifically,

• $PC_{p,x,y,z}$: indicates whether processor core p is in (x, y, z).

Similarly, MC is used in our formulation to identify the coordinates of a storage block.

• $MC_{m,x,y,z}$: indicates whether storage block m is in (x, y, z).

Although the size of a storage block is given, its dimensions may vary. We use MD to capture the dimensions of a storage block. Specifically, we have:

• $MD_{m,x,y}$: indicates whether storage block m has dimensions of (x, y).

The mapping between the coordinates and the blocks is ensured by variable $MMap$ for the storage blocks, and variable $PMap$ for the processor cores. That is,

• $MMap_{x,y,z,m}$: indicates whether coordinate (x,y,z) is assigned to storage block m.

• $PMap_{x,y,z,p}$: indicates whether coordinate (x,y,z) is assigned to processor core p.

The distances between a processor core and a storage block on each axis (x,y and z) are captured by $Xdist_{p,m,x}$, $Ydist_{p,m,y}$, and $Zdist_{p,m,z}$. Specifically, we have:

• $Xdist_{p,m,x}$: indicates whether the distance between processor core p and storage block m is equal to x on the x-axis.

• $Ydist_{p,m,y}$: indicates whether the distance between processor core p and storage block m is equal to y on the y-axis.

• $Zdist_{p,m,z}$: indicates whether the distance between processor core p and storage block m is equal to z on the z-axis.

In order to facilitate the thermal-aware core/storage block placement, power and temperature values need to be calculated. Temperature of each block (T_i) is obtained using the resistance vector (R) and the corresponding power consumption values (obtained through a Wattch-based simulation [7]).

• $Power_m$: is the power consumption for block m.

• $Temp_m$: is the temperature of block m. A processor core needs to be assigned to a single coordinate:

$$\sum_{i=0}^{C_X-1}\sum_{j=0}^{C_Y-1}\sum_{k=0}^{C_Z-1} PC_{p,i,j,k} = 1, \quad \forall p. \tag{1}$$

In the above equation, i, j and k correspond to the x, y and z coordinates, respectively. A storage block also needs to be assigned to a unique coordinate:

$$\sum_{i=0}^{C_X-1}\sum_{j=0}^{C_Y-1}\sum_{k=0}^{C_Z-1} MC_{m,i,j,k} = 1, \quad \forall m. \tag{2}$$

A storage block needs to have unique dimensions:

$$\sum_{i=1}^{C_X}\sum_{j=1}^{C_Y} MD_{m,i,j} = 1, \quad \forall m. \tag{3}$$

Each storage block should have dimensions in such a way that its size, $SIZE_m$ (given as input), will fit into the allocated space. That is, $SIZE_m = width \times height$:

$$SIZE_m = \sum_{i=1}^{C_X}\sum_{j=1}^{C_Y} MD_{m,i,j} \times i \times j, \quad \forall m. \tag{4}$$

Storage blocks should be mapped to the chip based on the coordinate and dimensions of the corresponding storage block. This requirement can be captured as follows:

$$MMap_{x,y,z,m} \geq MC_{m,x_1,y_1,z-1} + MD_{m,dx,dy} - 1,$$
$$\forall m, x, x_1, dx, y, y_1, dy, z, z_1, dz \text{ such that}$$
$$x_1 + dx \geq x > x_1, \text{ and } y_1 + dy \geq y > y_1. \tag{5}$$

In this expression, x_1, y_1 and z_1 denotes the x, y, and z coordinates of a storage block. Similarly, dx and dy denote the dimensions of the storage block. Based on these values, $MMap$ assigns the corresponding coordinates to the storage block m. Similarly, processor cores should be mapped to the chip, which can be expressed as follows:

$$PMap_{x,y,z,p} \geq PC_{p,x_1,y_1,z-1}, \forall p, x, x_1, y, y_1, z, z_1$$
$$\text{such that } x_1 + P_X \geq x > x_1, \text{ and } y_1 + P_Y \geq y > y_1. \tag{6}$$

In order to prevent multiple mappings of a coordinate in our grid, we force a coordinate to belong a single processor core or

TABLE II
THE ACCESS PERCENTAGE OF EACH BLOCK BY DIFFERENT PROCESSORS.

Processor	B_1	B_2	B_3	B_4	B_5	B_6	B_7	B_8	B_9	B_{10}
1	0.20%	0.18%	61.76%	0.19%	0.17%	0.20%	3.48%	2.86%	0.20%	0.20%
2	0.20%	0.19%	61.86%	0.19%	0.22%	4.07%	2.30%	0.18%	0.19%	0.18%
3	0.18%	0.18%	61.83%	0.18%	0.18%	4.05%	2.34%	0.19%	0.21%	0.19%
4	0.18%	0.22%	61.76%	0.19%	0.18%	4.05%	2.32%	0.18%	0.20%	0.18%
Processor	B_{11}	B_{12}	B_{13}	B_{14}	B_{15}	B_{16}	B_{17}	B_{18}	B_{19}	B_{20}
1	22.83%	0.22%	0.20%	0.18%	0.20%	0.19%	0.18%	0.19%	1.83%	4.55%
2	22.64%	0.20%	0.18%	0.18%	0.17%	0.19%	0.18%	1.91%	4.49%	0.28%
3	22.65%	0.20%	0.20%	0.22%	0.19%	0.20%	0.18%	1.89%	4.47%	0.29%
4	22.59%	0.17%	0.18%	0.18%	0.19%	0.21%	0.18%	1.89%	4.49%	0.31%

a single storage block:

$$\sum_{i=1}^{M} MMap_{i,x,y,z} + \sum_{i=1}^{P} PMap_{i,x,y,z} = 1, \forall x, y, z. \quad (7)$$

Manhattan Distance is assumed to be the cost of the *data communication* between a storage block and a processor core. This is also referred to as the *data access cost* in this paper, and is the metric whose value we want to minimize. Note that in our architecture processor cores communicate by sharing data in storage blocks. To capture the Manhattan Distance, we use two variables, namely, $Xdist_{p,m,x}$ and $Ydist_{p,m,y}$, and employ the following constraints:

$$Xdist_{p,m,x} \geq PC_{p,x_1,y_1,z_1} + MC_{m,x_2,y_2,z_2} - 1,$$
$$\forall p, m, x, x_1, x_2, y_1, y_2, z_1, z_2 \text{ such that } x = |x_1 - x_2|. \quad (8)$$

$$Ydist_{p,m,y} \geq PC_{p,x_1,y_1,z_1} + MC_{m,x_2,y_2,z_2} - 1,$$
$$\forall p, m, x_1, x_2, y, y_1, y_2, z_1, z_2 \text{ such that } y = |y_1 - y_2|. \quad (9)$$

In addition to the x and y dimensions, there is a communication cost due to z dimension, which captures the vertical dimension. Data communication across the layers is not the same as the communication within a given layer, because it only depends on the wafer thickness, which does not change with the block size, so it needs to be captured separately as follows:

$$Zdist_{p,m,z} \geq PC_{p,x_1,y_1,z_1} + MC_{m,x_2,y_2,z_2} - 1,$$
$$\forall p, m, x_1, x_2, y_1, y_2, z, z_1, z_2 \text{ such that } z = |z_1 - z_2|. \quad (10)$$

In wafer-bonding 3D technology, the dimensions of the vertical through-wafer interconnect are not expected to scale at the same rate as feature size, because wafer-to-wafer alignment tolerances during bonding pose limitations on the scaling of the through-wafer interconect. Current dimensions of through-wafer via sizes vary from 1μm-by-1μm to 10μm-by-10μm [6, 11, 12]. The relatively large size of via makes the interconnect delay going through wafer to be relatively much smaller.

We next calculate the temperature of each block using the $Temp = R \times Power$ equation. More specifically,

$$Temp_m = \sum_{j=1}^{P+M+1} R_{m,j} \times Power_j, \quad \forall m. \quad (11)$$

Finally, the temperature constraint is enforced using the following expression:

$$Temp_m \leq T_B, \quad \forall m. \quad (12)$$

TABLE III
SINGLE-CORE BENCHMARK CODES USED IN THIS STUDY.

Benchmark Name	Source	Description	Number of Data Accesses
ammp	Spec	Computational Chemistry	86967895
equake	Spec	Seismic Wave Propagation Simulation	83758249
mcf	Spec	Combinatorial Optimization	114662229
mesa	Spec	3-D Graphics Library	134791940
vortex	Spec	Object-oriented Database	163495955
vpr	Spec	FPGA Circuit Placement and Routing	117239027

TABLE IV
MULTI-CORE BENCHMARK CODES USED IN THIS STUDY.

Benchmark Name	Source	Description	Number of Data Accesses
3step-log	DSPstone	Motion Estimation	90646252
adi	Livermore	Alternate Direction Integration	71021085
btrix	Spec	Block Tridiagonal Matrix Solution	50055611
tsf	Perfect Club	Nearest Neighbor Computation	54917732

Having specified the necessary constraints in our ILP formulation, we next give our objective function. We define our cost function as the sum of the data communication distances in all 3 dimensions. X_{Cost}, Y_{Cost}, and Z_{Cost} denote the total data communication distances traversed along dimensions x,y, and z, respectively. The communication cost due to dimension x is:

$$X_{Cost} = \sum_{i=1}^{P} \sum_{j=1}^{M} \sum_{k=1}^{C_X - 1} FREQ_{i,j} \times Xdist_{i,j,k} \times k. \quad (13)$$

Similarly, the cost due to dimension y can be expressed as:

$$Y_{Cost} = \sum_{i=1}^{P} \sum_{j=1}^{M} \sum_{k=1}^{C_Y - 1} FREQ_{i,j} \times Ydist_{i,j,k} \times k. \quad (14)$$

Finally, the cost due to dimension z can be written as:

$$Z_{Cost} = \sum_{i=1}^{P} \sum_{j=1}^{M} \sum_{k=1}^{C_Z - 1} FREQ_{i,j} \times Zdist_{i,j,k} \times k. \quad (15)$$

Consequently, our objective function can be expressed as:

$$\min \quad (\alpha \times (X_{Cost} + Y_{Cost}) + \beta \times Z_{Cost}). \quad (16)$$

To summarize, our topology exploration problem can be formulated as "minimize $\alpha \times (X_{Cost} + Y_{Cost}) + \beta \times Z_{Cost}$ under constraints (1) through (15)." It is important to note that this ILP formulation is very flexible as it can accomodate different number of processor cores, storage blocks, and layers.

V. EXAMPLE

In this section, we give an example demonstrating the effectiveness of our approach. As our example, we use 3step-log,

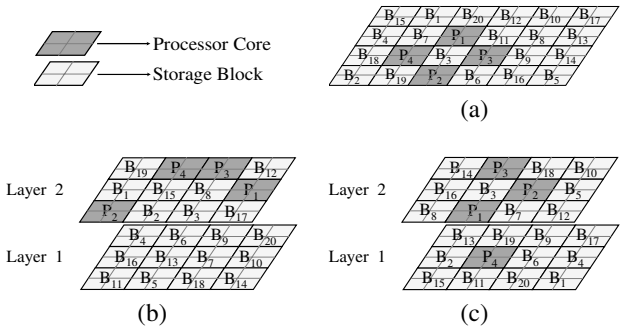

Fig. 2. The topologies generated with 4 processor cores (3step-log). (a) Optimal 2D. (b) Random 3D. (c) Optimal 3D. $P_1 \ldots P_4$ are processor cores and $B_1 \ldots B_{20}$ are storage blocks.

one of our benchmarks, with 4 processor cores and 20 storage blocks. In the 2D case, we assume the dimensions of the chip (C_X, C_Y and C_Z) as $12 \times 8 \times 1$, and in the 3D case, we assume the dimensions to be $8 \times 6 \times 2$. In both the cases, the total number of coordinates (total chip area) is equal to 96. Both the storage blocks and the processor cores are assumed to be 2×2, though they can be set to any value in our formulation. Table II shows the percentage of accesses to each block by different processor cores.

In Figure 2, the topologies generated by three different schemes are depicted. Figure 2(a) illustrates the best 2D placement (determined using ILP). On the other hand, a randomly-generated storage block/processor core placement is shown in Figure 2(b) for 3D. Finally, Figure 2(c) shows the best placement for 3D (determined using ILP). Note that, each processor core and each storage block is mapped to 4 coordinates since the size of a storage block/processor core is assumed to be 4. To explain these mappings, let us consider Figure 2(c). In this topology, B_{15} is mapped to coordinates (0,0,0), (0,1,0), (1,0,0), and (1,1,0). As it can be seen from Table II, data block B_3 is the most frequently accessed block by all the processor cores, and that is why it is put into a very close position to all 4 processor cores in optimal 2D and optimal 3D. We also see that, B_{11} is the next most frequently accessed block by all the processor cores in this example.

VI. Experimental Evaluation

A. Setup

We performed several experiments with two different set of benchmarks. The first group of applications are sequential Spec2000 codes and used for single-core designs in this work. This benchmark set consists of six applications randomly-selected from the Spec2000 benchmark suite. Table III lists these codes and their important characteristics. In collecting the statistics on accesses to storage blocks, for each benchmark, we fast-forwarded the first 2 billion instructions, and simulated the next 300 million instructions. The fourth column of Table III gives the number of data accesses for each application. The second group of applications, on the other hand, are for multi-core designs and include several array-based codes parallelized through an optimizing compiler built upon SUIF [4]. The exact mechanism used in parallelizing the code is orthogonal to the focus of this paper. What we mean by parallelization in this context is distributing loop iterations across the processors; each processor is typically assigned a subset of loop iterations. Table IV lists the important characteristics of this second group of benchmarks. As before, the third column gives the description of the benchmarks, and the last column gives the number of data accesses. The ILP solution times on an Intel Pentium III Xeon Processor of 549MHz with 1GB of RAM varied between 144 seconds and 3.5 hours, averaging on about 25 minutes. The default simulation parameters used in our experiments are presented in Figure 3. As our base configuration, we assumed a stack of two device layers connected to each other. We also assumed that the chip is composed of 24 blocks, each of which can be a storage block or a processor core. We conservatively assumed that the block-to-block distance is ten times costlier than that of the layer-to-layer distance. This is denoted as the ratio of $\frac{\alpha}{\beta}$ in this paper.

We performed experiments with four different execution models for each benchmark code in our experimental suite:

- *2D-Random:* This is in a sense a conventional topology which uses a single wafer and the storage blocks and the processor cores are placed randomly.
- *2D-Opt:* This is an integer linear programming based strategy, wherein the storage blocks and processor cores are distributed on the die in a way that minimizes the data communication cost of the whole system. Note that, this is an optimal core/storage block placement scheme for 2D.
- *3D-Random:* This is same as the 2D-Random case except that there are possibly multiple device layers.
- *3D-Opt:* This is the integer linear programming based placement strategy for 3D proposed in this paper, wherein the storage blocks and processor cores are placed on several wafers optimally. This scheme represents the optimal placement for 3D.

B. Results

Figure 4 gives our results based on two layers. Note that, these results are obtained using the values given in Figure 3. All results are normalized with respect to those of the 2D-Random scheme. Figure 4(a) shows the improvement brought by our approach for the single-core designs, whereas Figure 4(b) gives the similar results for the multi-core designs. We see that the overall average reduction in data access costs with 2D-Opt is around 63% and 58% for the single-core case and the multi-core case, respectively. On the other hand, the 3D-Opt scheme reduces the costs by about 82% and 69% on average for the single-core case and the multi-core case, respectively. In other words, the best 3D design generates much better results than the best 2D design.

To see the impact of the optimal placement of processor cores/storage blocks, we next compare the optimized and random designs in 3D. In Figure 5, we compare our 3D approach against the randomly-generated placements obtained through 3D-Random. As before, the results are normalized with respect to the 2D-Random scheme. The average data communication cost reductions are 42% and 51% for the single-core case and for the multi-core case, respectively. Overall, the results presented in Figure 4 and Figure 5 clearly show that employing a 3D design with optimal placement is critical for the best results.

Parameter	Value
Number of processor cores (in multi-core designs)	4
Number of blocks	24
Number of layers	2
$\frac{\alpha}{\beta}$	10
Total storage capacity	128KB
Set associativity	2 way
Line size	32 Bytes
Number of lines per block	90
Temperature bound	110°C

Fig. 3. The default simulation parameters.

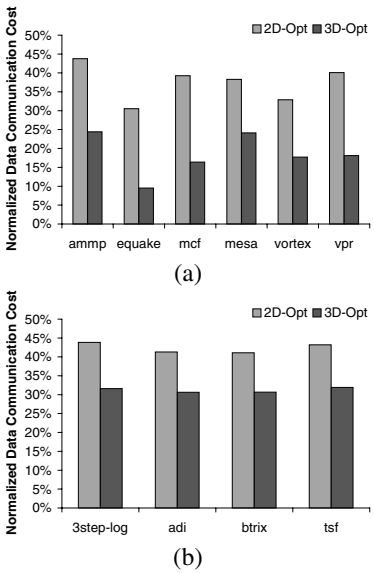

Fig. 4. Data communication costs for 2D-Opt and 3D-Opt normalized with respect to the 2D-Random scheme. (a) Single-core design. (b) Multi-core design (with 4 processor cores).

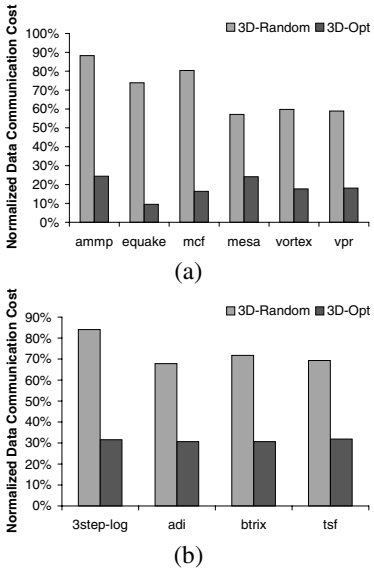

Fig. 5. Data communication costs for 3D-Random and 3D-Opt normalized with respect to the 2D-Random scheme. (a) Single-core design. (b) Multi-core design (with 4 processor cores). The results of 3D-Random are obtained by taking the average over five different experiments with random placement.

VII. Concluding Remarks

Ever-shrinking process technology coupled with increasing data communication requirements of embedded applications make on-chip interconnects an increasing bottleneck in embedded system design. One of the promising solutions to this global interconnect problem is to move towards 3D designs, where direct verical tunneling allows multiple layers to be stacked, one on top of the other. The work described in this paper studies application-specific placement of processor cores and storage blocks in a customized 3D design. We formulated this problem using ILP and solved it using a commercial solver. Our experiments with both the single-core and multi-core designs and with both the 2D and 3D designs indicate that the optimal placement of storage blocks and processor cores (i.e., optimal topology discovery) is very important in a 3D design (i.e., it makes a huge difference in terms of data communication costs) and that the best 3D designs consistently generate better results than the best 2D designs.

References

[1] C. Ababei, H. Mogal, and K. Bazargan, *Three-dimensional Place and Route for FPGAs,* In Proc. of Asia South-Pacific Design Automation Conference (ASP-DAC), 2005.

[2] S. G. Abraham and S. A. Mahlke, *Automatic and Efficient Evaluation of Memory Hierarchies for Embedded Systems,* In Proc. of the 32nd Annual International Symposium on Microarchitecture, Haifa, Israel, November 1999.

[3] A. J. Alexander, et al, *Three-Dimensional Field-Programmable Gate Arrays,* In Proc. of The Eighth Annual IEEE International Application Specific Integrated Circuits Conference, pp. 253-256, 1995.

[4] S. P. Amarasinghe, J. M. Anderson, M. S. Lam, and C. W. Tseng. *The SUIF compiler for scalable parallel machines,* In Proc. SIAM Conference on Parallel Processing for Scientific Computing, February, 1995.

[5] F. Angiolini, L. Benini, and A. Caprara, *Polynomial-Time Algorithm for On-Chip Scratch-Pad Memory Partitioning,* In Proc. of the International Conference on Compilers, Architectures and Synthesis for Embedded Systems, San Jose, CA, 2003.

[6] B. Black, et al, *3D Processing technology and Its Impact on IA32 Microprocessors*, In Proc. of ICCD, 2004.

[7] D. Brooks, V. Tiwari, and M. Martonosi, *Wattch: A framework for architectural-level power analysis and optimizations,* In Proc. of the 27th Annual International Symposium on Computer Architecture, June 2000.

[8] Y. Cao, H. Tomiyama, T. Okuma, and H. Yasuura, *Data Memory Design Considering Effective Bit-width for Low-Energy Embedded Systems,* In Proc. of the 15th International Symposium on System Synthesis, Kyoto, Japan, October 2002.

[9] F. Catthoor, et al, *Custom Memory Management Methodology – Exploration of Memory Organization for Embedded Multimedia System Design,* Kluwer Academic Publishers, 1998.

[10] S. Das, *Timing, energy, and Thermal performance of Three-Dimensional Integrated Circuits*, In Proc. of GLSVLSI, 2004.

[11] S. Das, et al, *Technology, Performance, and computer-aided Design of Three-Dimensional integrated Circuits*, In Proc. of ISPD, 2004.

[12] Y. Deng, et al, *2.5D System Integration: A Design Driven System Implementation Schema*, In Proc. of ASP-DAC, 2004.

[13] F. Gharsalli, S. Meftali, F. Rousseau, and A. A. Jerraya, *Automatic Generation of Embedded Memory Wrapper for Multiprocessor SoC,* In Proc. of the 39th Design Automation Conference, New Orleans, Louisiana, 1999.

[14] S. Im and K. Banerjee, *Full Chip Thermal Analysis of Planar (2D) and Vertically Integrated (3D) High Performance ICs*, Tech. Digest IEDM 2000, pp.727-730.

[15] International Technology Roadmap for Semiconductors, *http://www.itrs.net/Common/2004Update/2004Update.htm.*

[16] S. Meftali, F. Gharsalli, F. Rousseau, and A. A. Jerraya. *An Optimal Memory Allocation for Application-Specific Multiprocessor System-on-Chip,* In Proc. of the International Symposium on Systems Synthesis, Montreal, Canada, 2001.

[17] G. Nemhauser, L. Wolsey. *Integer and Combinatorial Optimization*, Wiley-Interscience Publications, 1988.

[18] P. R. Panda and L. Chitturi, *An Energy-Conscious Algorithm for Memory Port Allocation,* In Proc. of the 2002 IEEE/ACM International Conference on Computer-Aided Design, San Jose, California, November 2002.

[19] A. Ramachandran and M. F. Jacome, *Xtream-Fit: An Energy-Delay Efficient Data Memory Subsystem for Embedded Media Processing,* In Proc. of the 40th Design Automation Conference, Anaheim, CA, June 2003.

[20] W.-T. Shiue and C. Chakrabarti, *Memory Exploration for Low-Power Embedded Systems,* In Proc. of the 36th Design Automation Conferences, New Orleans, LA, 1999.

[21] K. Skadron, et al, *HotSpot Thermal Modeling Simulator*, http://lava.cs.virginia.edu/HotSpot/

[22] C. Tsai and S. Kang, *Cell-Level Placement for Improving Substrate Thermal Distribution*, IEEE Transactions On Computer-Aided Design of Integrated Circuits and Systems, vol. 19, pp. 253-266, Feb. 2000.

[23] Xpress-MP, http://www.dashoptimization.com/pdf/Mosel1.pdf, 2002.