Computing (2015) 97:593–609
DOI 10.1007/s00607-013-0378-1

Energy reduction in 3D NoCs through communication optimization

Ozcan Ozturk · Ismail Akturk · Ismail Kadayif · Suleyman Tosun

Received: 27 April 2013 / Accepted: 1 December 2013 / Published online: 12 December 2013

Abstract Network-on-Chip (NoC) architectures and three-dimensional (3D) integrated circuits have been introduced as attractive options for overcoming the barriers in interconnect scaling while increasing the number of cores. Combining these two approaches is expected to yield better performance and higher scalability. This paper explores the possibility of combining these two techniques in a heterogeneity aware fashion. Specifically, on a heterogeneous 3D NoC architecture, we explore how different types of processors can be optimally placed to minimize data access costs. Moreover, we select the optimal set of links with optimal voltage levels. The experimental results indicate significant savings in energy consumption across a wide range of values of our major simulation parameters.

Keywords 3D · NoC · Communication · Energy

This research is supported in part by TUBITAK grants 112E360 and 113E258, by a grant from Intel Corporation.

O. Ozturk (✉)
Computer Engineering Department, Bilkent University, Ankara, Turkey
e-mail: ozturk@cs.bilkent.edu.tr

I. Akturk
Department of Electrical and Computer Engineering, University of Minnesota, Minneapolis, USA
e-mail: iakturk@cs.bilkent.edu.tr

I. Kadayif
Computer Engineering Department, Canakkale Onsekiz Mart University, Çanakkale, Turkey
e-mail: kadayif@comu.edu.tr

S. Tosun
Computer Engineering Department, Ankara University, Ankara, Turkey
e-mail: stosun@ankara.edu.tr

Mathematics Subject Classification 68M01 Computer system organization - General

1 Introduction

International technology roadmap for semiconductors (ITRS) projects that the number of cores will continue to increase [1]. As the number of cores increase, interconnect between these cores becomes a major concern. This is even more pronounced when the number of cores is beyond 16 since buses are no longer an option due to physical limitations. Network-on-Chip (NoC) [2] architectures have been proposed to overcome the limitations by using switches and dedicated links between the nodes.

Similarly, 3D Integration is another way of improving interconnection performance/energy, where multiple device layers are stacked together (3D IC) [3]. This trend is driven mostly by greater density, that is, three-dimensional integrated circuits (3D ICs) is one of the only ways to meet the demand for increased transistor density. In addition to the density, 3D ICs also provide heterogeneous integration, on-chip interconnect length reduction, modular and scalable design. NoC architectures have been extended to the third dimension by the help of through silicon vias (TSVs) [4–6]. 3D NoCs have the potential to achieve better performance with higher scalability and lower power consumption [2,7].

Heterogeneity could be at different levels. We, specifically, consider (within the 3D NoC) different types of cores in terms of performance, area, and energy consumption. Having different kinds of cores within the 3D NoC enables a better match for each application according to its processing needs and memory requirements.

There exist several important problems to consider in a heterogeneous 3D CMP connected via an NoC. However, among these, we focus on the problem of node placement, link shut-down, and power management in a heterogeneous NoC based CMP using a compiler analysis technique combined with integer linear programming (ILP). Our specific contributions are:

- An approach that achieves placement of heterogeneous processor cores within the available chip area to minimize data communication and energy.
- An ILP based approach that decides the set of active and powered down links in a heterogeneous 3D NoC for a given application.
- A technique that decides the voltages/frequencies of the active links if the underlying architecture supports link voltage/frequency scaling.
- An approach that divides the mesh into islands and manages link power consumption at an island granularity.
- An experimental evaluation of the proposed ILP based approach using the applications from various benchmarks.

As technology moves towards heterogeneous chip multiprocessors, one of the challenging problems in the context of 3D NoC systems is the placement of processor cores within the available chip area. Focusing on such a heterogeneous 3D NoC, this paper explores how different types of processors can be placed to minimize data access costs.

The remainder of this paper is structured as follows. Section 2 gives the related work on heterogeneous 3D NoCs. Section 3 discusses the overview of our approach.

The details of our ILP (integer linear programming) based formulation are given in Sect. 4, and an experimental evaluation is presented in Sect. 5. The paper is concluded in Sect. 6.

2 Related work

3D technologies and the motivation for moving from 2D to 3D is explained in [8]. Vivet et al. [9] present 3D NoC as a promising solution for increased modularity and scalability. They show that an efficient implementation can increase the bandwidth while simplifying the assembly process. 3D NoC topologies explored in [2], where they compare 3D NoC to 2D NoC considering physical constraints.

Ebrahimi et al. [10] discuss how 3D architectures with a large number of TSVs can be built using mesh-based topologies. In [11], authors not only reduce the communication delay, but also improve the thermal behavior in a 3D NoC architecture. Coskun et al. [12] propose a dynamic thermally-aware job scheduling technique for 3D architectures to reduce the thermal problems at very low performance cost. AFRA [13] introduces a low cost reliable routing for 3D mesh NoCs. Wu et al. [14] propose a scalable 3D global routing using integer programming. Ozturk et al. [15] explore how processor cores and data blocks can be placed in a 3D architecture.

Our approach is different from these previously proposed techniques since we propose to use heterogeneous processing elements in a 3D NoC architecture and apply voltage scaling both at communication links as well as processing elements. A preliminary version of this work was presented in PDP 2013 [16], where we only target the placement of heterogeneous processors on a 3D NoC architecture. This paper extends the work presented in PDP 2013 by including communication link energies. Specifically, we shutdown some of the links or voltage scale them if voltage scaling is available. Moreover, we selectively form voltage islands to reduce the area overheads introduced by voltage scaling circuitry. To the best of our knowledge, this is the first work which performs communication energy optimization for heterogeneous 3D NoC architectures.

3 Overview of our approach

After a parallelization step, we analyze the set of processor nodes that communicate with each other and this information is then forwarded to the ILP solver. ILP solver determines the set of communication links that should be used. Our approach maximizes the number of unused links and reduces the voltages and frequencies of the used links. Figure 1 illustrates the high level view of a heterogeneous 3D NoC based CMP. While different layers of 3D NoC is connected through TSVs, nodes are connected with network switch/router (represented by R). In the same figure, processor is represented by CPU and memory hierarchy is represented by MH. Each node is connected to its north, south, west and east via the network switches.

In this work, we do not focus on the hardware implementation of link shut down and link voltage scaling; rather we assume that link voltage scaling and link shut down are available as indicated by the previous studies. There are potential overheads, such as area and power, associated with implementing such a scheme. However, as indicated

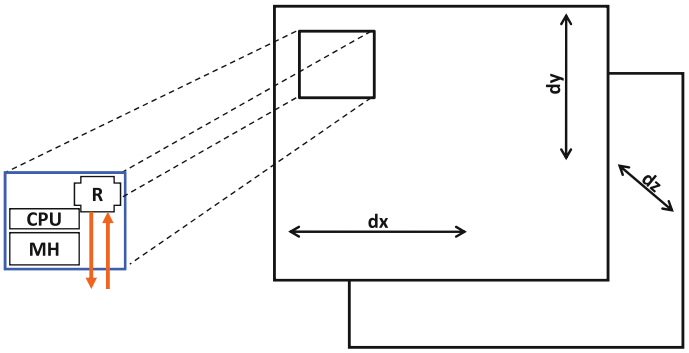


Fig. 1 3D NoC based CMP architecture

by prior studies, the overheads are minimal [17]. We quantify and elaborate these overheads in the experimental evaluation section.

In order to distribute the data among the processors, we need two types of information: the private data accessed by each processor and the amount of data shared across the processors. We obtain these through representing the data accessed by each processor and each processor pair using Presburger formulas and counting them.

4 ILP formulation

Our goal in this section is to present an ILP formulation of the problem of minimizing data communication energy of a given application. This is achieved through optimal placement of nodes in a 3D NoC and through communication link utilization. We use a commercial tool [18] to solve our ILP problem.

Energy consumed can be considered in two main categories: actual communication energy represented with E_{Comm} and link energy represented with E_{Link}. We discuss these in the following subsections.

4.1 Communication energy consumption

In this subsection, we consider the dynamic communication energy required to execute a given application. We use several decision variables to formulate the problem using ILP. For instance, location of a node n is captured by L variable. More specifically, $L^n_{x,y,z}$, indicates whether node n is on the grid location (x, y, z).

We capture the distance between two nodes by using binary variables $dx_{i,j,x}$, $dy_{i,j,y}$, $dz_{i,j,z}$, where they indicate the distances on x-axis, y-axis, and z-axis, respectively.

Distances between nodes can easily be captured using the location binary variables. We express the layer-to-layer distance as:

$$dz_{i,j,z} \geq L^i_{x_1,y_1,z_1} + L^j_{x_2,y_2,z_2} - 1, \quad z = |z_1 - z_2|. \tag{1}$$

Similarly, for in-layer distances dx and dy are also captured. Our cost function is defined as the sum of the data communication loads in both vertical and horizontal

dimensions. More specifically, we denote the total data communication using $Comm_H$ and $Comm_V$ for horizontal, and vertical communication costs, respectively:

$$Comm_H = \sum_{e=1}^{E}\sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{k=1}^{D_X} A_{e,i,j} \times dx_{i,j,k} \times k + \sum_{e=1}^{E}\sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{k=1}^{D_Y} A_{e,i,j} \times dy_{i,j,k} \times k. \quad (2)$$

$$Comm_V = \sum_{e=1}^{E}\sum_{i=1}^{N}\sum_{j=1}^{N}\sum_{k=1}^{D_Z} A_{e,i,j} \times dz_{i,j,k} \times k. \quad (3)$$

Affinity, expressed with $A_{e,i,j}$, indicates the communication load between the nodes i and j during epoch e. Note that, in the above expressions, we sum over all the epochs $(1 \ldots E)$ present in the application. Therefore, our communication energy consumption due to actual communication can be expressed as:

$$E_{Comm} = Comm_H + \theta \times Comm_V. \quad (4)$$

Note that, in expression 4, the difference between horizontal and vertical communication costs is captured by the θ parameter which is conservatively set to 0.2 in our baseline implementation. More specifically, accessing a data from a neighboring node on the same layer is five times costlier than accessing a neighbor on a different layer. While the former requires in-layer communication channels, the latter uses TSVs. Although we use 0.2 as the default value, the θ parameter can be exercised and the most suitable value can be obtained.

4.2 Link energy consumption

This part of the ILP formulation corresponds to the link energy consumption required for keeping links active/inactive. In order to capture the participation of a node in a communication within epoch e, we use $X_{e,i}$, where $X_{e,1}$ denotes the first node, and $X_{e,n}$ denotes the last node. For each communication pattern, we use a different set of $X_{e,i}$ variables. We distinguish these variables by using the signature of the pattern, that is $[a, b]$, representing the communication from node a to node b. More specifically, binary variable $X_{e,i}^{[a,b]}$ indicates whether node i is used for the communication from a to b during epoch e.

We decide whether a node participates at the communication by using $X_{e,i}^{[a,b]}$ variable. More specifically, a node participates at the communication if a neighbor is part of this communication and the link connecting these two nodes is active (not shutdown). We use a different 0–1 variable to capture the activity of a link. In mathematical terms, for all neighboring nodes in the 3D NoC, we have:

$$E_{e,i\rightarrow j} = \begin{cases} 1, & \text{if link between } X_{e,i} \text{ and } X_{e,j} \\ & \text{is active during epoch } e \\ 0, & \text{otherwise.} \end{cases}$$

Recall that, in our 3D NoC based CMP architecture, there are bi-directional links between neighboring nodes denoted by $i \rightarrow j$. These links can be active in one epoch of the program and inactive during the next one. This enables us to control the 3D NoC according to different epoch communication patterns.

As explained earlier, in order to include a node in a communication activity, we need to ensure that a neighbor is part of the communication and the connecting link is active. These constraints can be expressed as follows:

$$X_{e,i}^{[a,b]} \geq X_{e,j}^{[a,b]} + E_{e,i,\rightarrow j} - 1, \quad \forall e, i, j, a, b, \\ \text{where } i \text{ and } j \text{ are neighbors.} \tag{5}$$

Nodes that are on the borders of the 3D NoC have a subset of these constraints (depending on their specific locations).

We also need to capture the link state transitions (i.e., shutdowns and startups). It might be possible to hide the performance overhead due to these activations/deactivations by using a *preactivation* strategy (i.e., by activating a link slightly ahead of time before it is really needed so that it will be ready when it is needed). However, the energy overheads occurred due to such activities cannot be hidden. To capture this overhead, we use $AC_{e,i\rightarrow j}$ and $DC_{e,i\rightarrow j}$ for activation (startup) and deactivation (shutdown), respectively.

Since we shut down some of the communication links in the mesh based 3D NoC architecture and try to reduce the number of links used, our ILP based technique may return a solution that demands a higher link sharing than the original case. This, in turn, can affect the performance of the application and, eventually, result in higher energy consumption. Therefore, we limit the number of communications exercising the same communication link (within a given epoch).

In order to count the number of communication patterns that exercise a given link, we need to capture the links that are used for implementing each communication pattern. To achieve this, we define:

$$UE_{e,i\rightarrow j}^{[a,b]} = \begin{cases} 1, & \text{if link } X_{e,i,j}^{[a,b]} \text{ is used} \\ & \text{for communication}(e, [a, b]) \\ 0, & \text{otherwise} \end{cases}$$

We next give our link energy cost function as the sum of the active communication links, which are captured by the $E_{e,i\rightarrow j}$ binary variables. More specifically, our link energy function can be expressed as follows:

$$E_{link} = \sum_{e=1}^{E} \sum_{i=1}^{n} \sum_{j=1}^{n} \sum_{w=1}^{W_{max}} E_{e,i\rightarrow j} + \alpha \times AC_{e,i\rightarrow j} \\ + \beta \times DC_{e,i\rightarrow j} \tag{6}$$

In the above expression, α and β capture the weights of link activation and deactivation, respectively (i.e., architecture specific transition costs).

Using both communication energy and link energy, we can express the objective function as

$$min \quad E_{comm} + E_{link}. \tag{7}$$

4.3 Discussion

Recall that, our goal so far is to shut down as many communication links as possible to reduce energy consumption (under performance bounds). We can further increase energy savings by scaling voltages and frequencies of the active links whenever it is possible to do so. Voltage/frequency scaling of communication links should be done with care though, as it can increase communication latencies significantly. However, our observation is that, the communication patterns in a given epoch of a communication graph can be divided into two groups: *critical communication*(*s*) and *non-critical communications.* The former represents the communications that last the longest, whereas the latter represents the remaining ones. In fact, in most communication epochs, there is only a single critical communication. A key observation is that; non-critical communications can be delayed as long as their latency do not exceed that of the critical communication(s). Therefore, the energy consumption of the active links, onto which non-critical communications are mapped, can be reduced through voltage/frequency scaling. With this observation, we modify our problem formulation to include voltage/frequency scaling in the communication links.

This part of our approach starts with the communication link information returned by the link shutdown approach explained in the previous section (i.e., the set of used/unused links). Using this input, we identify the critical communication patterns and scale voltages and frequencies of the links that are used for non-critical communications.

We formulated this approach by modifying the variables captured by binary variable $E_{e,i\rightarrow j}$. Recall that, earlier, this variable is used to indicate whether the associated link is on or off. To formulate voltage scaling, we expand this binary variable to $E_{e,v,i\rightarrow j}$ to indicate whether the given link is assigned voltage level v (and the corresponding frequency). In other words, we have:

$$E_{e,v,i\rightarrow j} = \begin{cases} 1, & \text{if link between } X_{e,i} \text{ and } X_{e,j} \\ & \text{is at voltage level} v \text{during epoch } e \\ 0, & \text{otherwise} \end{cases}$$

We are assuming that there are V discrete voltage/frequency levels, and at any point during the course of execution, a communication link can have a single voltage level. In order to enforce this last requirement, we need to include the following constraint:

$$\sum_{v=1}^{V} E_{e,v,i\rightarrow j} = 1, \quad \forall e, i, j. \tag{8}$$

In addition to variable $E_{v,i\to j}$, we need to modify $UE^{[a,b]}_{e,i\to j}$ to include different voltage levels, for different neighboring nodes:

$$UE^{[a,b]}_{e,v,i\to j} = \begin{cases} 1, & \text{if link between } X^{[a,b]}_{e,i} - X^{[a,b]}_{e,j} \text{ is used for} \\ & \text{communication } (e,[a,b]) \text{ and this link} \\ & \text{is at voltage level } v \\ 0, & \text{otherwise} \end{cases}$$

For each voltage level v, we use a predetermined communication latency overhead associated with that specific voltage level due to the decrease in the frequency. In our formulation, this latency overhead is given by $O(v)$ for voltage level v. The $O(v)$ variables depend on the discrete voltage levels available and their corresponding frequencies. Based on this variable, the latency overhead associated with a link between two nodes for a specific pattern and epoch (given as input) can be obtained as follows:

$$T(e,p) = \sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{v=1}^{V} UE^{[a,b]}_{e,v,i\to j} \times L_{e,a,b} \times K \times O(v), \quad \forall(e,a,b). \tag{9}$$

In the above expression $T(e,p)$ indicates the latency incurred to perform the communication for pattern $[a,b]$ in epoch e. On the other hand, K represents the latency of communicating a unit (of data) through a communication link, whereas $UE^{[a,b]}_{e,v,i\to j}$ indicates if link between $X^{[a,b]}_{e,i}$–$X^{[a,b]}_{e,j}$ is running on voltage level v for pattern $[a,b]$ in epoch e. Our goal here is to limit the latency of each and every communication pattern occurring in an epoch with the original latency of that epoch (T_e). Note that this last value is dictated by the critical communication pattern as explained earlier. This constraint enables us to voltage scale only the (links used by the) communication patterns that are not in the critical path within an epoch. Since we do not voltage scale for the patterns that are in the critical path, we will not incur any additional performance overhead due to voltage scaling. It is important to note that, if desired, this formulation can easily be modified to work under a performance degradation bound; that is, allowing a certain increase in the latency of the critical communication pattern.

$$T^{[a,b]}(e) \le T_e, \quad \forall(e,a,b). \tag{10}$$

A potential overhead introduced by link voltage scaling is the area overhead incurred by the voltage/frequency scaling circuit required for every link in the 3D NoC. In order to reduce this area overhead, recent architectures define voltage regions (also called voltage islands [19]), which use the same voltage level for a certain region of the 3D NoC. This way, instead of using one voltage/frequency scaling circuit per link, we are able to use one scaling circuit per region, thereby saving considerable amount of die area. We can extend our baseline formulation by introducing a new voltage level variable for each region, $R_{e,v,r}$.

$$R_{e,v,r} = \begin{cases} 1, & \text{if region } r \text{ is at voltage} \\ & \text{level } v \text{ during epoch } e \\ 0, & \text{otherwise} \end{cases}$$

We assume that the region information (partitioning) is given as input to our ILP formulation, which basically specifies the links and nodes that are part of each region. For each region, we need to assign a unique voltage level as in the case of a link (in our baseline formulation). We can write:

$$\sum_{v=1}^{V} R_{e,v,r} = 1, \quad \forall e, r. \tag{11}$$

The next question to address is how to select the voltage level for a specific region. In our implementation, we conservatively select the highest voltage level (when all the links within that region are considered) in order not to introduce any performance overhead which could be caused by a critical path in a given pattern.

$$R_{e,v_1,r} \times v_1 \geq E_{e,v_2,i\to j} \times v_2, \quad \forall e, v_1, v_2, r, i, j \quad \text{where} \quad v_1 \geq v_2 \quad \text{and} \quad i \to j \in r. \tag{12}$$

Above formulation assigns the highest voltage level (among those demanded by the links in the region) to the region. Additionally, we need to modify the constraint given before as follows:

$$UE_{e,v,i\to j}^{[a,b]} \geq X_{e,i}^{[a,b]} + X_{e,j}^{[a,b]} + R_{e,v,r} - 2, \quad \forall e, r, i, j, a, b, v \text{ such that } \quad i \to j \in r. \tag{13}$$

This constraint ensures that $UE_{e,v,i\to j}^{[a,b]}$ values are assigned such that the region's voltage level is used for that specific link given by $i \to j$.

Note that, in our ILP formulation, we employ area and distance as two main constraints, whereas performance, energy, and communication bandwidth and other possible constraints are left out. For example, depending on the switch present in a node, bandwidth available to the connected links will be limited. Our ILP formulation, in its current form, does not cover this constraint. However, our formulation can easily be modified to include such constraints. In addition to additional constraints, our ILP formulation can also be modified to optimize for a different objective function instead of data communication energy.

5 Experimental evaluation

To test the effectiveness of our ILP-based approach, we performed experiments using a set of array-based applications. The compiler part of our approach (interprocessor communication extraction) has been implemented using SUIF [20]. We used Xpress-MP [18] as our ILP solver. *The solution times we experienced varied between 27.4 s and 4.6 min* (*over 90 % of our compilation times are spent within the ILP solver*).

Table 1 Our simulation parameters and their default values

Parameter	Value
Types of processor cores	4
Number of blocks	24
Number of layers	2
Temperature bound	110
θ	0.2
3D NoC size	2 layers of 5 × 5 Mesh
Mesh node	2-issue CPU / 16 KB local memory
Number of voltage levels	4
Voltage range	[0.8 V, 1.4 V]
Voltage switching latency	200 ns
Voltage switching energy	2.2 nJ
Link activation latency	650 ns
Link activation energy	9.0 nJ
Packet header size	3 flits
Flit size	39 bits

We used Orion [21], an architectural level dynamic power simulator that is capable of providing detailed power estimates and performance results, to simulate link power consumption behavior. We enhanced the simulator with static (leakage) power models. While our main focus in this work is on reducing communication energy consumption through accurate node placement, link shutdown, and link voltage/frequency scaling, it is also important to consider the chip-wide energy impact, including not only communication energy but also computation and memory access energies as well. Using the default simulation parameters shown in Table 1 and a complete simulation environment that uses both SIMICS (for processor simulation) and Orion (for network simulation), we found that the 3D NoC energy consumption constitutes about 29 % of total on-chip energy consumption for our benchmarks, assuming that each CPU can issue two instructions at each cycle, and has 32KB software-managed local memory. As a result, one can expect decent chip-wide savings if the 3D NoC energy consumption can be cut significantly. Table 2 gives important statistics on our benchmark codes. In this table, the third and fourth columns show the total number of communication messages issued at the source code level and total inter-processor communication volume, respectively. The next column lists the link energy consumption when *no* power management scheme is employed, and the last column gives the total execution cycles for the benchmarks. The numbers under the fifth and sixth columns are obtained using the default values of the simulation parameters listed in Table 1.

Figure 2 gives the energy consumption results when our approaches are used. Unless otherwise stated, in this and all other graphs that show energy, the results are given as *normalized values,* with respect to the case where no node placement and no link energy optimization is applied. Also, as our performance bound, we set S_{limit} to the average number of messages that share a link in the original case (without any rerouting). The first bar in Fig. 2 shows the results when only link shutdown

Table 2 Benchmarks used in our experimental evaluation

Name	Benchmark description	Communication			Execution cycles ($\times 10^6$)
		Messages count	Volume (KB)	Energy (mJ)	
go	go-playing program	1,084	137.53	0.53	96.52
m88ksim	a chip simulator	3,351	489.22	2.36	324.82
compress	in-memory version of the UNIX utility	1,627	228.52	0.79	106.08
li	Xlisp interpreter	1,283	178.94	0.69	296.13
ijpeg	image compression/decompression	3,427	449.07	1.58	239.49
vortex	an object oriented database	4,108	486.71	1.63	279.73
tomcatv	vectorized mesh generation	1,786	235.75	0.85	114.56
swim	shallow water equations	2,025	256.34	0.92	186.62
su2cor	Monte-Carlo method	3,654	479.89	1.73	249.03
hydro2d	Navier Stokes equations	1,606	203.26	0.73	189.14
mgrid	3D potential field	1,928	243.77	0.88	171.91
applu	partial differential equations	3,218	424.92	1.53	236.58
turb3d	turbulence modeling	4,457	536.29	1.98	301.47
apsi	Weather prediction	1,196	157.15	0.57	261.84
fpppp	Gaussian series for quantum chemistry	4,056	487.89	1.67	254.70
wave5	Maxwell's equations	4,362	505.45	1.82	273.16

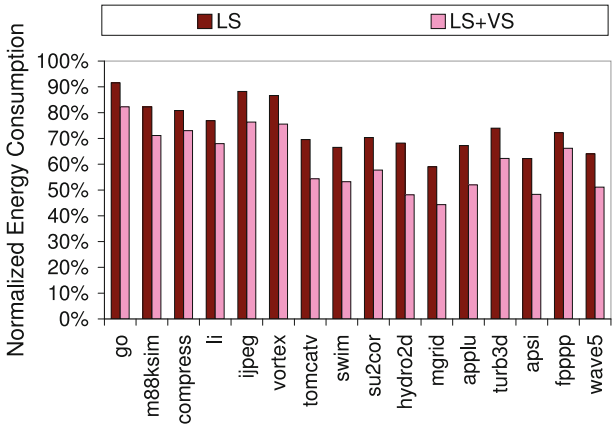


Fig. 2 Energy consumption results with the default simulation parameters. LS: link shutdown; LS + VS: link shutdown followed by voltage scaling

(Sect. 4) and node placement is used, and the second bar gives the results when both link shutdown and voltage/frequency scaling are used (Sect. 4) on top of an optimal node placement. As mentioned earlier, these results include the overheads incurred by our schemes. We see that the average energy savings with link shutdown only and

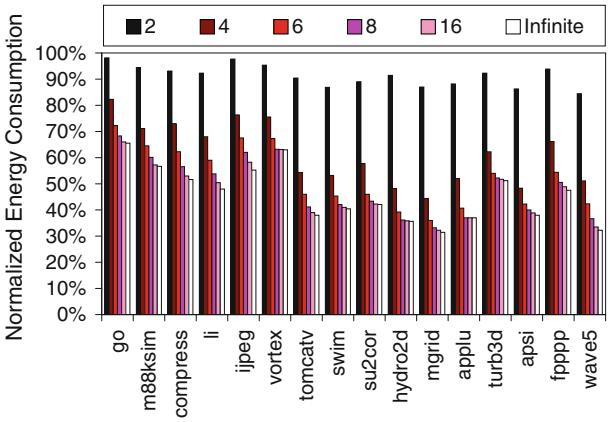


Fig. 3 Impact of the number of voltage/frequency levels. The last *bar*, for each benchmark, captures continuous scaling

combined scaling/shutdown are 26.27 and 38.50 %, respectively. Although we do not present detailed performance results, with the performance bound mentioned above, our approach incurred only around 1.5 % performance degradation on average; the leakage impact of this is also included in our energy results. Even when preactivation is not employed, the average performance overhead was only 1.7 %.

In our first set of sensitivity experiments, we study the impact of the number of voltage/frequency levels on our energy savings. Recall that the default number of voltage/frequency levels was 4. Figure 3 gives the results when the number of voltage/frequency levels is varied between 2 and 16. In obtaining the results with k voltage/frequency levels, we divided the [0.8 V, 1.4 V] voltage range into k and, for each voltage value, we used the corresponding frequency value. The graph in Fig. 3 also shows the results of a hypothetical case with continuous voltage scaling in the range [0.8 V, 1.4 V], i.e., when we have an infinite number of voltages (with their corresponding frequencies) available in that range. We see from these results that, while increasing the number of voltage/frequency levels initially brings substantial benefits, the additional savings start to diminish dramatically when we reach a certain number (of levels). In fact, the results with 16 voltage/frequency levels and infinite number of levels are almost the same (with the average energy savings of 53.22 and 54.14 %, respectively).

Our next set of experiments is designed to investigate the impact of the 3D NoC size on our results. The original mesh size used in our experiments so far was two layers of 5 × 5. First, we only change the NoC size in a layer while keeping the number of layers fixed at 2. The results shown in Fig. 4 indicate that the savings from both link shutdown and voltage scaling increase as we increase the mesh size. This is because an increased mesh size results in more sparse use of communication links (i.e., the links usage frequency gets reduced), thereby creating better opportunities for both link shutdown and voltage scaling. We see from these results that the average energy savings with a 7 × 7 mesh 3D NoC are 34.53 and 45.56 % for link shutdown and voltage scaling, respectively.

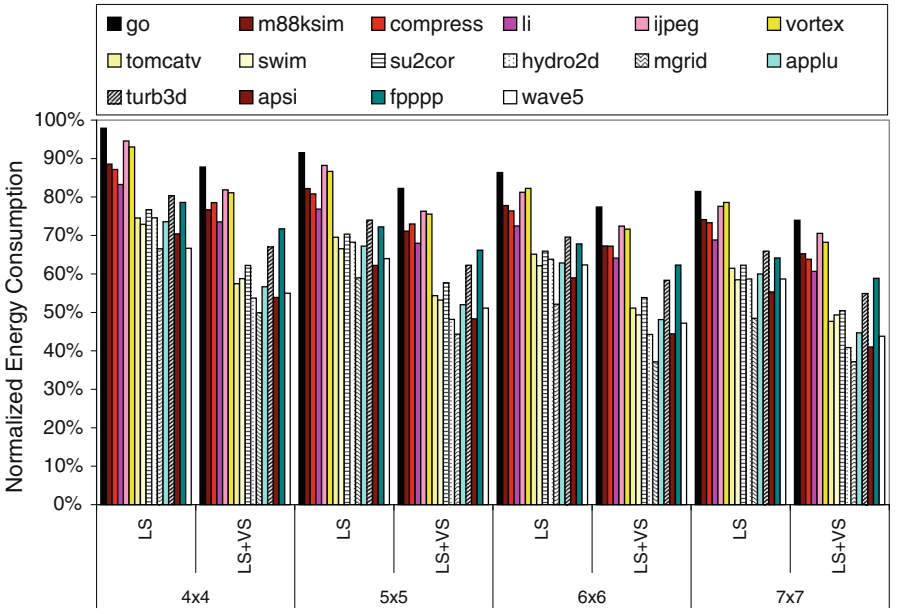
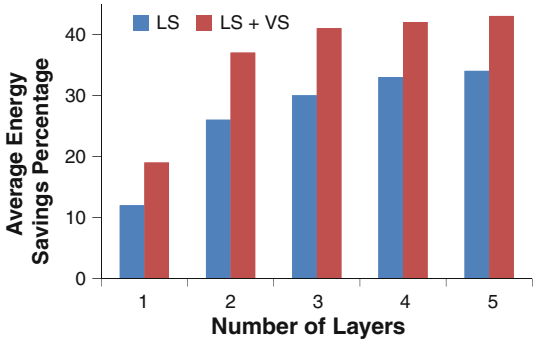


Fig. 4 Impact of the mesh size

Fig. 5 Impact of the number of layers in 3D NoC



Next, we explore the effect of number of layers, while keeping the NoC size within a layer as 5×5. Figure 5 shows the energy savings when the number of layers vary from 1 to 5. As can be seen from this figure, with a single layer (a 2D NoC), there are still energy savings but it is a lot less compared to higher number of layers. As the number of layers increase energy savings also increase, however the energy reductions brought by an additional layer keeps dropping. Note that, we do not apply any temperature bound in implementing these multiple layers which can potentially drop these savings further.

We now study the impact of dividing the mesh into islands and managing link power consumption at an island granularity (as discussed at the end of Sect. 4). That is, the links in each island are assigned a single voltage/frequency, in order to minimize the potential area/placement overheads of having too many voltage/frequency scaling

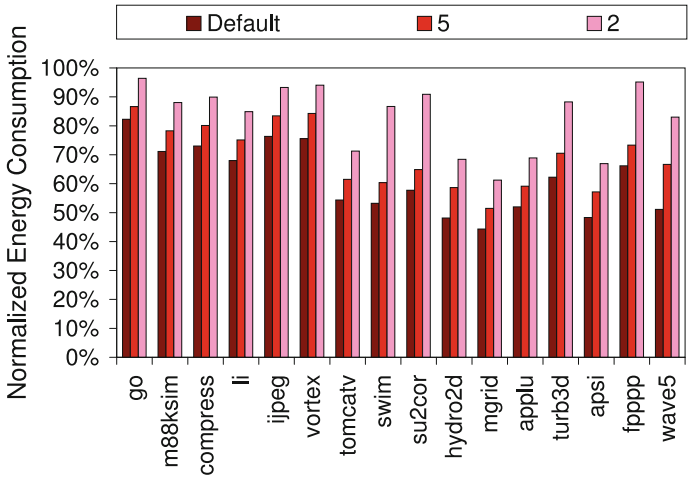


Fig. 6 Impact of the number of voltage islands

circuits (when we have a separate circuit for each and every link). The graph in Fig. 6 shows the results when we have only two islands (i.e., the mesh area is divided into two parts, and the links in a given island are controlled together using a single scaling circuit) and five islands (i.e., each row in the 5×5 mesh has its own scaling circuit). The graph also reproduces the results with our default configuration where each link is controlled independently. We see from these results that, as expected, a smaller number of islands translate to lower energy savings. However, we also see that even with two islands, the average energy savings obtained through voltage scaling is about 17 %; that is, link voltage scaling is still quite effective.

The savings achieved by our results also depend on the code and data mapping scheme used. Recall that our default mapping schemes have been explained earlier in Sect. 3. To study the impact of a different mapping, we also performed experiments when the data manipulated by the application are mapped to the memories of the CMP nodes in a fashion different from our default mapping (the computation mapping though still used the owner-computes rule). We found that the results with these new mappings are within (2 %) of those obtained using our original mapping. Also, since our approach uses profile data for integer benchmarks, we performed experiments with different inputs for these benchmarks and found that our results are consistent (within 1 %) across different input sets (we tried three different input sets for each integer benchmark).

We, next, compare the savings obtained using our approach to those obtained using two heuristic schemes. Figure 7 presents the average energy savings (over all benchmark codes in our experimental suite) using the default values of our simulation parameters. The specific link shutdown heuristic used is from [22] and the specific voltage scaling heuristic is from [23]. We see from this figure that, while the heuristic schemes perform very well, there is still a gap between them and the ILP based schemes, motivating for further research on developing better heuristics. This gap increases with the heterogeneous case as opposed to the homogeneous case.

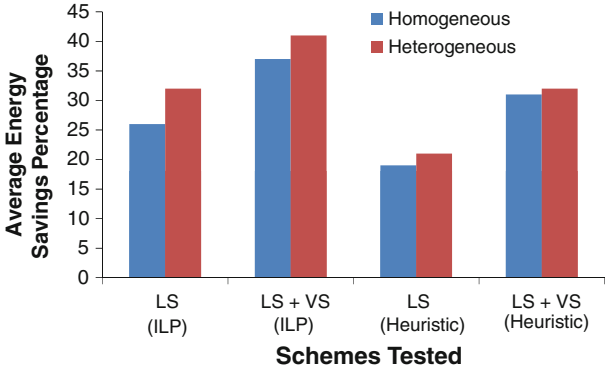


Fig. 7 Comparison of the ILP based schemes with heuristic approaches

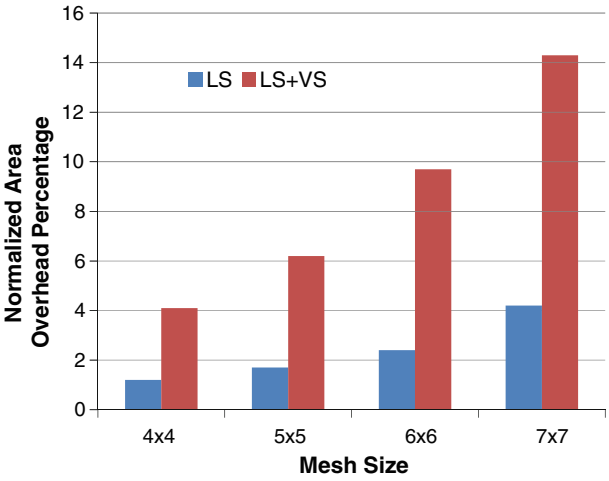


Fig. 8 Area overheads normalized with respect to no energy optimization case. LS: link shutdown; LS + VS: link shutdown followed by voltage scaling

We evaluate the effect of link energy optimization on area with varying mesh sizes in Fig. 8. The area overheads are estimated using an implementation on an FPGA. As can be seen from this graph, voltage scaling require substantial circuits to support voltage and frequency changes. Therefore, area overhead of link shutdown and voltage scaling is much more compared to only applying link shutdown (4.2 versus 14.3 % for a 7×7 mesh).

Our last set of experiments compare the performance degradation caused by the proposed schemes. Figure 9 shows the performance overhead normalized with respect to no energy optimization case. Performance overhead (increase in the execution latency) due to link shut down (LS) is higher compared to voltage scaling (LS + VS) due to delays inherent in state transitions between on and off states. Our schemes are accompanied with minimal impact on performance, ranging from 6.2 to 19.7 % for LS and

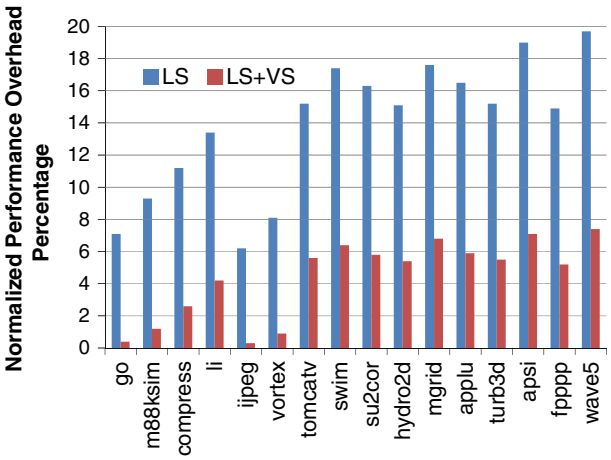


Fig. 9 Performance overheads normalized with respect to no energy optimization case. LS: link shutdown; LS + VS: link shutdown followed by voltage scaling

from 0.3 to 7.4 % for LS + VS. Note that, we obtain the aforementioned energy savings with no degradation in network throughput.

6 Conclusion

Global interconnect problem has become more important with the increase in the number of processor cores in chip multiprocessing. 3D designs and NoC architectures have been unified as 3D NoCs to overcome the interconnect scaling bottleneck. We try to map heterogeneous processors onto the given 3D chip area with minimal data communication costs. Moreover, we try to achieve link power reduction through communication link shutdown and voltage/frequency scaling. We formulated this problem using ILP and solved it using a commercial solver. Our experiments show that the ILP based schemes generate very impressive results as far as energy reduction is concerned. Our approach also provides a bar against heuristic schemes.

References

1. ITRS (2013) International technology roadmap for semiconductors
2. Pavlidis V, Friedman E (2007) 3-d Topologies for networks-on-chip. Very large scale integration (VLSI) systems. IEEE Trans On 15(10):1081–1090
3. Davis W, Wilson J, Mick S, Xu J, Hua H, Mineo C, Sule A, Steer M, Franzon P (2005) Demystifying 3d ics: the pros and cons of going vertical. Design Test Comput IEEE 22(6):498–510
4. Li F, Nicopoulos C, Richardson T, Xie Y, Narayanan V, Kandemir M (2006) Design and management of 3d chip multiprocessors using network-in-memory. In: Computer architecture, 2006. ISCA '06. 33rd international symposium on, pp 130–141
5. Murali S, Benini L, De Micheli G (2010) Design of networks on chips for 3d ics. In: Proceedings of the 2010 Asia and South Pacific design automation conference, pp 167–168
6. Park D, Eachempati S, Das R, Mishra AK, Xie Y, Vijaykrishnan N, Das CR (2008) Mira: a multi-layered on-chip interconnect router architecture. SIGARCH Comput Archit News 36:251–261

7. Loi I, Angiolini F, Benini L (2008) Developing mesochronous synchronizers to enable 3d nocs. In: Design, automation and test in Europe, 2008. DATE '08, pp 1414–1419
8. Borkar S (2011) 3d integration for energy efficient system design. In: design automation conference (DAC), 2011 48th ACM/EDAC/IEEE, pp 214–219
9. Vivet P, Dutoit D, Thonnart Y, Clermidy F (2011) 3d NoC using through silicon via: an asynchronous implementation. In: VLSI and system-on-chip (VLSI-SoC), 2011 IEEE/IFIP 19th international conference on, pp 232–237
10. Ebrahimi M, Daneshtalab M, Liljeberg P, Plosila J, Tenhunen H (2013) Cluster-based topologies for 3d networks-on-chip using advanced inter-layer bus architecture. J Comput Syst Sci 79(4):475–491
11. Daneshtalab M, Ebrahimi M, Plosila J (2012) Hibs: novel inter-layer bus structure for stacked architectures. In: 3D systems integration conference (3DIC), 2011 IEEE, international, pp 1–7
12. Coskun AK, Ayala JL, Atienza D, Rosing TS, Leblebici Y (2009) Dynamic thermal management in 3d multicore architectures. DATE 1410–1415
13. Akbari S, Shafiee A, Fathy M, Berangi R (2012) Afra: a low cost high performance reliable routing for 3d mesh nocs. DATE 332–337
14. Wu TH, Davoodi A, Linderoth JT (2009) Grip: scalable 3d global routing using integer programming. DAC 320–325
15. Ozturk O, Wang F, Kandemir M, Xie Y (2006) Optimal topology exploration for application-specific 3d architectures. In: Design automation, 2006. Asia and South Pacific conference on (2006)
16. Akturk I, Ozturk O (2013) Ilp-based communication reduction for heterogeneous 3d network-on-chips. PDP 514–518
17. Soteriou V, Peh LS (2004) Design-space exploration of power-aware on/off interconnection networks. In: Computer design: VLSI in computers and processors, 2004. ICCD 2004. Proceedings. IEEE international conference on, pp 510–517
18. XPressMP (2008) FICO XPress optimization suite, http://www.fico.com/en/Products/DMTools/Pages/FICO-Xpress-Optimization-Suite.aspx
19. Hu J, Shin Y, Dhanwada N, Marculescu R (2004) Architecting voltage islands in core-based system-on-a-chip designs. In: Low power electronics and design, 2004. ISLPED '04. Proceedings of the 2004 international symposium on, pp 180–185
20. Amarasinghe SP, Anderson JM, Lam MS, wen Tseng C (1993) An overview of the suif compiler for scalable parallel machines. In: Proceedings of the seventh SIAM conference on parallel processing for scientific computing, pp 662–667
21. Wang HS, Zhu X, Peh LS, Malik S (2002) Orion: a power-performance simulator for interconnection networks. In: Microarchitecture, 2002. (MICRO-35). Proceedings 35th annual IEEE/ACM international symposium on, pp 294–305. doi:10.1109/MICRO.2002.1176258
22. Chen G, Li F, Kandemir M (2006) Compiler-directed channel allocation for saving power in on-chip networks. SIGPLAN Not 41(1):194–205. doi:10.1145/1111320.1111055
23. Chen G, Li F, Kandemir M, Irwin MJ (2006) Reducing NoC energy consumption through compiler-directed channel voltage scaling. SIGPLAN Not 41(6):193–203. doi:10.1145/1133255.1134004