Computers and Electrical Engineering 40 (2014) 307–316

Contents lists available at ScienceDirect

Computers and Electrical Engineering

journal homepage: www.elsevier.com/locate/compeleceng

Voltage island based heterogeneous NoC design through constraint programming ☆

Ayhan Demiriz [a,*], Nader Bagherzadeh [b], Ozcan Ozturk [c]

[a] Sakarya University, Sakarya 54187, Turkey
[b] University of California Irvine, Irvine, CA 92697, USA
[c] Bilkent University, Ankara 06800, Turkey

ARTICLE INFO

Article history:
Available online 15 September 2014

Keywords:
Voltage-frequency island
Constraint programming
Application mapping
Scheduling
Heterogeneous network-on-chip

ABSTRACT

This paper discusses heterogeneous Network-on-Chip (NoC) design from a Constraint Programming (CP) perspective and extends the formulation to solving Voltage-Frequency Island (VFI) problem. In general, VFI is a superior design alternative in terms of thermal constraints, power consumption as well as performance considerations. Given a Communication Task Graph (CTG) and subsequent task assignments for cores, cores are allocated to the best possible places on the chip in the first stage to minimize the overall communication cost among cores. We then solve the application scheduling problem to determine the optimum core types from a list of technological alternatives and to minimize the makespan. Moreover, an elegant CP model is proposed to solve VFI problem by mapping and grouping cores at the same time with scheduling the computation tasks as a limited capacity resource allocation model. The paper reports results based on real benchmark datasets from the literature.

© 2014 Elsevier Ltd. All rights reserved.

1. Introduction

NoC design paradigm has been proposed to overcome traditional chip design limitations that have become increasingly eminent under the reality of single chips made of large number of cores. Such technology requires solutions to challenging general problems as highlighted in [1]: application modeling and optimization; NoC communication architecture and optimization; NoC communication architecture evaluation; and NoC design validation and synthesis. Designing an NoC system requires handling persistent optimization that often demands finding the best tradeoff between conflicting objectives and constraints.

Heterogeneous designs may be more desirable for their utility to run various types of computational tasks efficiently by having cores that can dynamically adjust their voltages (i.e. dynamic voltage-frequency scaling – DVFS) and/or can be set to run at certain pre-optimized voltage levels (i.e. Voltage-Frequency Island – VFI) to use more efficiently the resources available such as power, area/surface, bandwidth and other related parameters [1,2]. Core mapping and application scheduling [3] problems are essential optimization problems for designing NoC systems whether they are homogeneous or heterogeneous [4]. In our earlier work [5,6], a Constraint Programming (CP) based approach was proposed successfully to determine optimum core mapping and application scheduling at transmitted packet level. In these studies, CPNoC was proposed to carry out the optimization task for the homogeneous NoC systems.

☆ Reviews processed and recommended for publication to the Editor-in-Chief by Guest Editor Dr. Masoud Daneshtalab.

* Corresponding author.
E-mail addresses: ademiriz@gmail.com (A. Demiriz), nader@uci.edu (N. Bagherzadeh), ozturk@cs.bilkent.edu.tr (O. Ozturk).

http://dx.doi.org/10.1016/j.compeleceng.2014.08.005
0045-7906/© 2014 Elsevier Ltd. All rights reserved.

This paper discusses heterogeneous NoC design from a Constraint Programming (CP) perspective using a two-stage solution as in [5,6]. Given a Communication Task Graph (CTG) and subsequent task assignments for the cores, CPU cores are allocated to the best possible places on the chip in order to minimize the overall communication cost among cores. Then, the application scheduling stage is run to determine the optimum core types from a list of technological alternatives and to minimize the makespan, i.e. time to complete all computation tasks on CTG. Heterogeneous designs may involve optimization problems that have conflicting terms in their objective functions. To accommodate solutions for the heterogeneous designs, our formulation of CPNoC in [5] is extended in this paper to have a multi-objective function, constraints and new decision variables to determine the best core composition, i.e. percentages of various core types on the chip. The main contribution of this paper built upon our work in [7] is to propose a CP based solution to VFI problem by simplifying the two-stage approach in our earlier work [5,6] into a single model which inherits properties from general mapping, scheduling and limited resource allocation problems.

There has been various approaches to reduce power consumption by using voltage islands. These islands are logic and memory areas supplied through separate, dedicated power feed, which provide the flexibility to selectively run different parts of the chip at different voltage/frequency levels. Furthermore, it is possible to shut down these regions to save more energy. Our goal in this paper is to propose a CP model for NoCs which optimally maps and groups cores into voltage islands while scheduling the computation tasks.

The rest of this paper is organized as follows. Section 2 introduces the related literature on heterogeneous NoC designs and VFI implementations. In Section 3, the proposed method and underlying CP model are presented. Section 4 gives the experimental results on real benchmark datasets from [8]. In this section, the results from heterogeneous architecture are presented. Our experiments encompass both 4×4 and 8×8 architectures. The main contribution of this paper which is an extension to solve VFI problem is introduced in Section 5. We also reported related experimental results in Section 5. The paper is concluded in Section 6.

2. Related work

A two-stage solution to core mapping and application scheduling problems was also proposed in [4]. The solution is reached by running iteratively these two consecutive stages (master and sub-problems). In each iteration, a new cut was introduced to the master problem in order to get closer to the optimal solution and satisfy the feasibility of scheduling. In [4], the master problem (core mapping) is modeled by integer programming and sub-problem (scheduling) is modeled by CP. Since there are no task deadlines in our model, it is always feasible to find a solution to the scheduling problem in our case. On the other hand, our scheduling model is more fine-grained than the one proposed in [4].

Recent years have witnessed several efforts at using heterogeneous designs in NoCs at different levels. A heterogeneous NoC design was also proposed in [9] by implementing core mapping as a 2D-packing problem and by finding a heuristic solution to underlying optimization problem. Power usage has also been taken into consideration for the scheduling phase in [9]. Network-bandwidth and latency-sensitive designs can be considered for heterogenous systems [10].

In contrast to DVFS, many implementations aim to solve the static voltage-frequency island problem to optimize design of heterogeneous NoC systems in order to run special applications [2]. Hu et al. [11] present an algorithm for simultaneous voltage island partitioning, voltage level assignment and physical-level floorplanning. Wu et al. [12] present a Voronoi diagram-based method to generate voltage islands that balance the power versus design cost tradeoff under performance requirement, according to the placement proximity of the critical cells. Lee et al. [13] propose a voltage assignment technique based on dynamic programming and perform power-network aware floorplanning for multiple supply voltage design. Li et al. [14] propose a deterministic algorithm to generate a floorplan using voltage islands. Ma and Young [15] present a core-based voltage island driven floorplanning. Specifically, for a given candidate floorplan represented by a normalized Polish expression, they generate the optimal voltage assignment and island partitioning simultaneously. Sengupta and Saleh [16] propose an application-specific voltage partitioning and island creation to reduce the overall SoC power, area, and design time.

David et al. [17] propose a real-time power management algorithm for NoCs that dynamically adjusts the level of voltage islands to reduce power consumption. Along the same lines, in [18], Bogdan et al. implement optimal power management for voltage island platforms where communication happens via an NoC architecture. Ozturk et al. [19] propose a compiler-based mapping of applications on voltage islands. Majzoub et al. [20] propose a voltage-island formation approach for the energy optimization of many-core architectures. They create voltage islands according to intra and inter-island communications and select the voltages accordingly.

Apart from these voltage island improvements, there are various studies that focus on optimization techniques. Specifically, in [21], authors present an optimality study for voltage-islands, where the granularity of voltage islands is on the functional-unit level as opposed to the core level. They formulate the problem as a min-cost network flow problem and reduce it to an integer linear programming (ILP) solution. In [22], authors propose a voltage island aware incremental floorplanning approach to support incremental voltage requests. They optimize the chip area and the wire length using Mixed-Integer Linear Programming (MILP) technique. Lee et al. [23] propose a post-floorplanning stage solution to voltage island generation. Specifically, they implement an ILP formulation to utilize power-network routing resources in the voltage islands.

Our work is different from these studies since our scheme uses constraint programming approach to generate near-optimal mappings/groupings of cores into voltage islands while scheduling the computation tasks. The major advantage of using CP is the clarity and understandability of the models.

3. Proposed optimization model

3.1. Basic assumptions and overview

Assuming that a set of cores is organized as a 2-D mesh of dimensions $n = m \times m$, each core can be labeled according to its position on the mesh (as an x and y coordinate pair) and has the capability of executing several tasks of an application in tandem. The buffer size is assumed to be unconstrained. Any communication link can only be occupied by a single packet at any given time without any limit on the bandwidth capacity. The communication links are bi-directional. In other words, any particular link between two routers can be considered as a separate link (resource) in each direction. Given a CTG, $G(\mathcal{T}, E)$ where E represents the communication requirements and precedence constraints for the tasks $\mathcal{T}$ with corresponding computation times, core mapping and scheduling problems are tackled in order to assign cores to optimum locations on the target network topology and then to schedule tasks to minimize the makespan. The first part of the problem solved in Stage I is an instance of the Quadratic Assignment Problem (QAP) and can be formulated as a CP model as shown in Eq. (1) (see [5]).

$$\begin{aligned} &\underset{\pi}{\text{minimize}} \sum_{i,j=1}^{n} f_{ij} d_{\pi_i \pi_j} \\ &\text{subject to} \\ &\quad \text{alldifferent}(\pi_1, \ldots, \pi_n), \\ &\quad \pi_i \in \{1, \ldots, n\}, \quad i = 1, \ldots, n, \end{aligned} \tag{1}$$

where f and d are flow and distance matrices (parameters) respectively. In other words, the parameter d is the Manhattan distance (cost) between all the possible pair combinations of the cores on the mesh and it is the cost of transferring data among cores which is a function of the routing algorithm. Deterministic routing algorithms can easily be incorporated to the optimization problem on hand and can help in finding the optimal application task schedule by utilizing d. The transfer cost based on the XY routing algorithm is proportional to the Manhattan distance which can be calculated between points a and b on a grid as: $\text{TC}_{ab} = |x_a - x_b| + |y_a - y_b|$. Parameter f represents the amount of data to be transferred between each core which is given as part of CTG. Considering all these parameters, the objective function in Eq. (1) can be considered as a representation of dynamic power, since the power consumption will be directly proportional to the communication distance. In Eq. (1), permutation variables $\pi_i = j$, for $i, j = 1, \ldots, n$. correspond to the location of cores. `alldifferent` constraint enforces unique assignment for each core. In other words, each of the n variables should be assigned a unique value between 1 and n.

3.2. A two-stage model for homogenous case

Stage I of our implementation can be conducted as in our early work [5] by solving Eq. (1). The optimal solution found in Stage I can be considered as the best floor-planning assignment that does not consider any resource and task precedence constraints. Resources in this context could be considered as bandwidth, power, surface area as well as temperature. Such constraints can be introduced to the optimization problem in the second stage that involves task scheduling with routing.

As in [5], a conventional objective function for the second stage can be represented as following: $\text{minimize}_{t_i \in \mathcal{T}} \max(\text{EndOf}(t_i))$, where $\text{EndOf}(t_i)$ represents the completion time of task $t_i \in \mathcal{T}$. Considering the variable types available in CP related to modeling the scheduling problems, we can incorporate `interval` and `sequence` variables in our formulation of Stage II. Basically, tasks to be scheduled can be represented as `interval` variables. Therefore, they are defined on timeline and they are associated with starting and end times which are practically continuous values. The `sequence` variables, on the other hand, can be utilized to model (finite) resources (e.g. communication channels that transfer data) and they are coupled with related tasks (e.g. transferring the data). To complete our model, we need to incorporate precedence constraint with suitable CP modeling constraints such as `endBeforeStart`, `endBeforeEnd`, `endAtStart`, and `endAtEnd`. To give an example, `endBeforeStart` constraint requires a task t_i to end before the other task t_j starts. Thus, task t_i has been completed before task t_j is started. In order to model (finite) resources by utilizing sequence variables in CP modeling paradigm, we need to associate tasks to be scheduled with (finite) resources (i.e. sequence variables). Considering all these CP modeling tools, we can construct the following CP model to solve the scheduling problem as follows:

$$\begin{aligned} &\underset{t_i \in \mathcal{T}, p_i \in P, \ell \in L}{\text{minimize}} \max(\text{EndOf}(t_i)) \\ &\text{subject to} \\ &\quad \text{sizeOf}(t_i) = \text{JD}(t_i), \quad \text{for all } t_i \in \mathcal{T} \\ &\quad \text{endBeforeStart}(t_i, t_j), \quad \text{for all } t_i \ll t_j \in \mathcal{T} \\ &\quad \text{endBeforeStart}(p_i, p_j), \quad \text{for all } p_i \ll p_j \in P \\ &\quad \text{endBeforeStart}(t_i, p_j), \quad \text{for all } t_i \ll p_j \in \mathcal{T}, P \\ &\quad \text{endBeforeStart}(p_j, t_i), \quad \text{for all } p_j \ll t_i \in \mathcal{T}, P \\ &\quad \text{noOverlap}(L). \end{aligned} \tag{2}$$

where $\ll$ is the precedence operator that indicates precedence relation. Interval variables $t_i \in \mathcal{T}$ and $p_i \in P$ represent respectively computation tasks given in CTG ($\mathcal{T}$ depicts the set of all tasks) and data transfer (communication) tasks that are generated based on the results of core mapping in Stage I and the underlying routing scheme (XY routing with Wormhole switching in our implementation). `sizeOf` (length) constraint enforces computation task times to be the same as the job durations, JD is given to the optimization model as a parameter (i.e. constant) in clock cycles and is determined according to both CTG and the architecture. In addition to precedence constraints constructed based on CTG for computational tasks, there need to be constraints for coupling computational and communication tasks. Basically, data transmission to the next computation node can only start after the current task has been completed. The next computation task can only start after all the necessary data transfer have been completed. The sequence variables, $\ell \in L$, are used for representing the bi-directional data channels that can serve only single flit at any given time. All these particular variables should be associated with related packet transmission tasks in P. `noOverlap` constraint in Eq. (2) practically enforces an orderly job sequencing (packet transmission) of data channels. Therefore, none of the data transmission tasks at flit level overlaps with any other data transmission task on the same data channel excluding opposite directional flows.

3.3. Proposed model for heterogeneous case

The problem aforementioned in Eq. (2) is valid for a homogeneous architecture as computation times of tasks at all cores are considered as same across the chip. One way of modeling heterogeneous architecture is to introduce binary decision variables for each core and technologically alternative core types (e.g. FPGAs, ASICs and GPUs [9,24,25]). Assuming that there are n_c different core types, we can introduce $n \times n_c$ binary decision variables into scheduling model given in Eq. (2) to determine optimum core composition. Notice that any solution would prefer fastest cores by default if the objective function in Eq. (2) is not changed i.e. the maximum task completion time. Even considering only the budget constraint, it may not be feasible to choose all of the cores from the fastest available technology. Therefore, we need to introduce a penalty term to the objective function for the alternative core types as follows.

$$
\begin{aligned}
&\underset{t_i \in \mathcal{T}, p_i \in P, \ell \in L, q \in Q}{\text{minimize}} \ \frac{\max(\text{EndOf}(t_i))}{\lambda} + \sum_{k=2}^{n_c} \frac{\sigma_k}{n} \sum_{j=1}^{n} q_{jk} \\
&\text{subject to} \\
&\quad \sum_{k=1}^{n_c} q_{jk} = 1, \quad j = 1, \ldots, n \\
&\quad \text{sizeOf}(t_i) = \sum_{k=1}^{n_c} \alpha_k q_{\pi_i k} \text{JD}(t_i), \quad \text{for all } t_i \in \mathcal{T} \\
&\quad \text{endBeforeStart}(t_i, t_j), \quad \text{for all } t_i \ll t_j \in \mathcal{T} \\
&\quad \text{endBeforeStart}(p_i, p_j), \quad \text{for all } p_i \ll p_j \in P \\
&\quad \text{endBeforeStart}(t_i, p_j), \quad \text{for all } t_i \ll p_j \in \mathcal{T}, P \\
&\quad \text{endBeforeStart}(p_j, t_i), \quad \text{for all } p_j \ll t_i \in \mathcal{T}, P \\
&\quad \text{noOverlap}(L) \\
&\quad q_{jk} \in \{0, 1\}, \quad j = 1, \ldots, n; \ \ k = 1, \ldots, n_c,
\end{aligned}
\tag{3}
$$

where λ is a normalization coefficient which can be set to the objective value of a solution to Eq. (2), σ_ks are weighting coefficients between 0 and 1 that adjust the preference of certain technologies to others. Notice that q_{jk} is a binary decision variable and takes a value of 1 if kth technological alternative is chosen for the jth core otherwise it is 0. The solutions of Stage I problem, i.e. permutation variables (π) are passed as parameters into Stage II. The objective function of Eq. (3) is a weighted combination of makespan and core compositions, i.e. percentages of various types of cores. Since there are n_c technological alternatives, one of them is linearly dependent to the rest. Alternative 1 can be arbitrarily chosen as dependent, for the sake of argument. Therefore, the index of technological alternatives (k) for decision variable q_{jk} starts at two. This objective function is a form of multi-objective function. Moreover, this multi-objective function enables optimizing the conflicting terms by finding the best tradeoff between normalized makespan and the core composition. Note that it may be undesirable to prefer faster cores in order to reduce the makespan from some design perspectives such as budget, space and power.

First constraint in Eq. (3) enables the assignment of only one core type to a single place (unit) on chip. Second constraint determines the computation time of the task t_i based on coefficients α_ks. Depending on technological properties of core types (α_k), computation times may vary (below or above) from a baseline time (JD(t_i)). In other words, duration of a task depends on which core on the mesh it is going to run on and its technological type (i.e. fast, normal, and slow). This is inline with the results from [24,25] since the speeds of various alternative technologies can be summarized as a function of power and area. The remaining constraints are the same as in Eq. (2) except binary value constraints for the decision variables q_{jk}.

Table 1
Completion times in clock cycles for heterogeneous architectures.

Application	λ	4×4 Completion times		8×8 Completion times	
		Eq. (3)	Eq. (3) w/area	Eq. (3)	Eq. (3) w/area
R-S32ENC	1800	1452	1452	1466	1466
R-S32DEC	3000	2138	2138	2151	2151
ROBOT	92,000	66,675	66,675	65,082	65,057
FPPPP	60,000	59,853	59,853	59,783	59,773
SPARSE	20,000	14,290	14,259	14,294	14,289

3.4. Power model

Power consumption in the semiconductor technologies consists of static and dynamic power. Static power consumption becomes a critical factor in the total power as the transistors become smaller and faster. The leakage current is a significant factor when the technology is scaled down. On the other hand, dynamic power consumption is produced by the switching activities in the semiconductor circuits. In CMOS technology, it is caused by the capacitive load charge of the transistor gates as well as the capacitance and resistance of wires. NoC systems consist of routers and links that connect routers. Flits travel from to the designated routers until they reach their final destinations. The router logic processes flit header information to select the corresponding output link for delivering the data to the next router. Therefore, the flit energy can be expressed as follows: $E_{flit} = H \times E_{router} + (H-1) \times E_{link}$ where E represents the energy and H represents the number of hops.

The header flit consumes higher energy since it takes extra clock cycles to be evaluated. The packet energy is expressed by the following equation: $E_{packet} = H \times (E_{Hflit} + t \times E_{Bflit}) + (H-1) \times E_{link}$ where t is the number of data flits. E_{Hflit} and E_{Bflit} represent the energy consumption of header and data (body) flits, respectively and furthermore can be expressed as follows: $E_{Hflit} = E_{buffer} + E_{crossbar} + E_{arbiter}$ and $E_{Bflit} = E_{buffer} + E_{crossbar}$.

Dynamic power consumption is reduced when the energy of the packet is minimized. This can be achieved by minimizing communication energy between tasks. Task mapping plays an essential role in determining the path that packets follow. On the other hand, task scheduling affects the buffer requirements for packets.

Communication requirements of an NoC-based multicore platform affects both performance and energy consumption. Specifically, the energy consumed by the NoC itself is 28% of the per-tile power in the Intel TeraFlop [26] and 36% of the total chip power in MIT RAW [27]. Therefore, it is critical to reduce dynamic energy consumption in the NoC channels and routers through diminished switching activity.

Prior research [28,29] has shown that links are responsible for a considerable amount of overall energy consumption. As the distance between the routers is increased, the link energy consumption linearly increases. This is due to the fact that long wires require large driving gates and repeaters, thereby increasing the switching activities and energy consumption. In order to capture these characteristic, we used Manhattan distance as the energy consumption metric for the NoC links. As indicated in our packet energy consumption, our model assumes (in accordance with the literature [28]) that the energy consumption for the transfer of one flit through an NoC is linearly dependent on the number of hops required. As Ye et al. [30] have shown, in a switch fabric circuit, the power is dissipated on three components: (1) the internal switches, (2) the internal buffers, and (3) the interconnect wires that connect node switches. Thus, our energy model captures all the parameters influencing energy consumption at the low implementation levels through respective constraints and objective function. These models can therefore be used to estimate the energy consumption of a complete NoC.

4. Application on real benchmark datasets

We have employed real application benchmark datasets to evaluate the mapping and the scheduling algorithm in this section. Multi-Constraint System-Level (MCSL) benchmark suite [8] provides a set of real applications where each application composes multiple tasks and traffic data patterns between these tasks. MCSL benchmark records the data traffic for different mesh network sizes and measures the execution time for each task in the application. Most of the architectural settings are borrowed from [5], exceptions are specified as needed. Results from heterogeneous architectures are presented in this section. The CP models are implemented by using IBM ILOG OPL Studio, which is available free of charge to the academicians at IBM Academic Initiative web site.[1]

Some characteristics of MCSL benchmark can be found in Table 1 of [7]. For each different application in MCSL Benchmark, we generated 10 different random sets[2] of the execution times and the traffic patterns according to distributional parameters provided in the benchmark data specifically the files with 'STP' extension [8]. Two different sizes of the mesh architecture were utilized in our experiments: 4×4 and 8×8. The packet size was set to eight for all the applications except H.264 which was set to 64 due to the computational complexity caused by the small packet size.

[1] http://tinyurl.com/cu5txlg.
[2] all the model and data files are available at http://tinyurl.com/orbzny2.

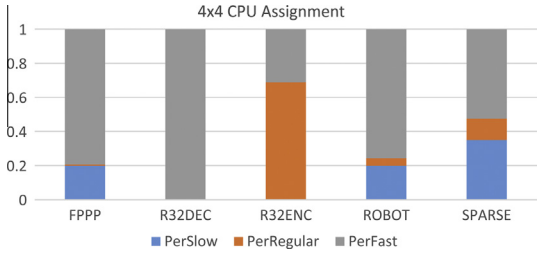

Fig. 1. Composition of core assignment on 4×4 architecture.

A new set of experiments has been conducted to test the applicability of Eq. (3). All the benchmark applications in MCSL (including H.264) have been used this time. Compared to Eqs. (2), (3) has additional parameters such as λ, σ and α.

λ is used for normalizing the objective term related to makespan, i.e. the completion time of all tasks on CTG. The parameter σ is used for penalizing composition of the cores. Assuming that one of the core types is more preferable than the rest, we can practically penalize the usage of the remaining core types in the composition of the chip. Notice that σ_k for $k = 2, \ldots, n_c$ should be specified for each technological alternative in our formulation. By using a σ_k closer to 1, we can practically penalize the usage of that particular core type within the composition of chip more. In our experiments, three different technological alternatives have been used. Hypothetically, we can think of these core types as slow, regular, and fast. In our experiments, σ_2 and σ_3 were set to 0.25 and 0.5 respectively. Technically, budgetary constraints can be easily used for determining the σ parameter. Parameter α is used for assessing the job duration times of technological alternatives. $\alpha = [1.4, 1, 0.7]$ is chosen for our experiments. More specifically, it takes 1.4 times more execution time for running computation tasks on a slow core type than the regular core type and it takes 30% less time to run tasks on fast core type than the regular core type. Alternatively Job Duration times (JD) could have been picked differently for each task on different core types, but the α parameter is used for simplicity.

Figs. 1 and 3 report composition of cores yielded by experiments to solve Eq. (3) for 4×4 and 8×8 architectures respectively. These figures practically show percentage of each core type in CPU core composition. Basically, fast cores are preferred for the 4×4 architecture in almost all datasets except R-S32ENC. This indicates that bandwidth is responsive and level of parallelization responds positively in increasing speed of the cores. On the other hand, slow cores are preferred for the 8×8 architecture. This underlines that there is no need to utilize faster cores as the level of parallelization is very high and these cores can wait for the data transmissions to be completed in most cases. We can conclude that this particular architecture is network-bandwidth sensitive [10].

Table 1 reports completion times in clock cycles for all applications in MCSL benchmark. The results are averaged on 10 realizations of the datasets (applications). The values of parameter λ used in the experiments are also listed in Table 1. Notice that parameter λ is chosen by considering the results of homogeneous design given in Tables 2 and 3 of our earlier work [6] except for application FPPPP. As a bandwidth parameter, 64 bit packet size is used for FPPPP to reduce the number of decision variables and constraints for the CP model to solve Eq. (3) – instead of 8 bit used for rest of the applications (see [5] for architectural parameters). It can be observed from Table 1 that completion times are shortened significantly in most cases for both 4×4 and 8×8 architectures. Even if most of the cores are formed by slow ones, the remaining fast cores on the chip can speed up the computational tasks significantly. According to results from [24,25], the functional relationship between cores' speed and area usage can be constructed easily. However, we can introduce a parametric constraint to represent surface/area usage on the chip for simplicity. In most of the design problems, surface/area usage is an important factor and Eq. (4) defines a constraint to consider the area limitation. By picking ω_k for $k = 1, \ldots, n_c$ and Ω appropriately, we can define a form-factor for various core types. $\omega = [0.9, 0.8, 0.65]$ and $\Omega = 0.8$ were chosen in our experiments. Experiments by solving Eq. (3) with area constraint (i.e. Eq. (4)) were also conducted on MSCL benchmark datasets. Results are reported in Figs. 2 and 4 and Table 1. When compositions of the chips within each architecture are compared, there is almost no change for 4×4 architecture (see Figs. 1 and 2). This means that solutions to Eq. (3) already satisfy the area constraint for the 4×4 architecture. However, introducing area constraint changes the solutions for 8×8 architecture.

$$\sum_{k=1}^{n_c} \frac{\omega_k}{n} \sum_{j=1}^{n} q_{jk} \leqslant \Omega \tag{4}$$

5. A practical CP based VFI model

Although the two-stage model introduced in Section 3 for heterogeneous designs has some preferable aspects such as ability of modeling scheduling problem at packet transmission level, separating mapping and scheduling problems into two different sub-problems raises some issues such as optimality of task scheduling. To alleviate this concern, we can run each step consecutively as in expectation maximization (EM) algorithm [31] until the convergence of the results. However, one can easily be tempted by the challenging nature of solving mapping and scheduling problems by a single optimization

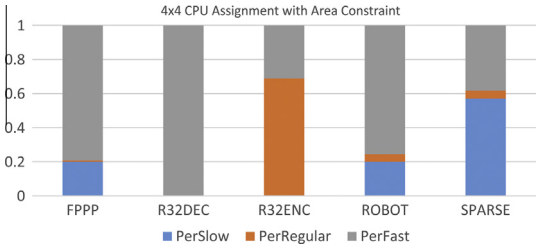

Fig. 2. Composition of core assignment on 4×4 architecture with area constraint.

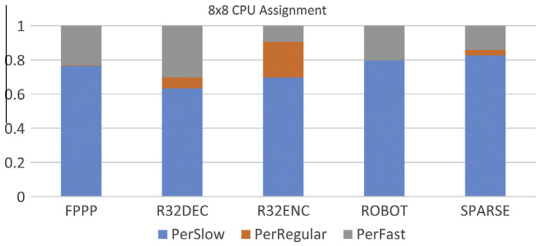

Fig. 3. Composition of core assignment on 8×8 architecture.

Table 2
MCSL VFI average completion times.

Application	VFI composition					
	3-3	3-4	3-5	4-3	4-4	5-3
ROBOT	73,923	67,368	65,872	73,714	71,006	73,279
RS32DEC	2321	2219	2232	2298	2296	2328
RS32ENC	1561	1471	1486	1558	1632	1598
SPARSE	17,510	15,669	14,907	16,544	16,283	16,482
FPPP	65,240	62,418	61,439	65,307	63,300	65,342
H264	13,644,522	13,642,991	13,640,795	13,643,616	13,643,792	13,642,571

Table 3
MCSL VFI minimum (best case) completion times.

Application	VFI Composition					
	3-3	3-4	3-5	4-3	4-4	5-3
ROBOT	70,141	63,574	63,702	70,398	66,748	69,952
RS32DEC	2112	2048	2101	2167	2140	2162
RS32ENC	1433	1301	1303	1,350	1400	1428
SPARSE	16,561	14,724	14,065	15,287	14,995	15,199
FPPP	57,931	56,101	55,044	57,801	56,633	57,843
H264	13,112,993	13,112,042	13,111,049	13,111,835	13,111,810	13,112,407

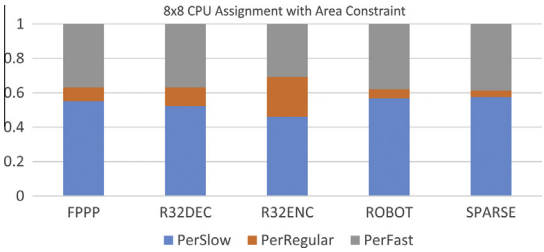

Fig. 4. Composition of core assignment on 8×8 architecture with area constraint.

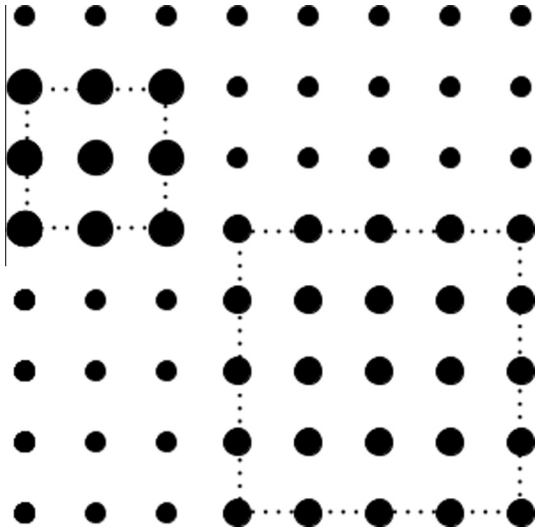

Fig. 5. A sample VFI layout.

model. Thus, we propose an efficient and practical CP model in this section to solve VFI problem by simplifying some of the constraints in our two-stage solution for the heterogeneous NoC designs.

In order to build a practical optimization model to solve the problems of mapping and grouping cores and then scheduling the tasks, we need to simplify the second stage of the optimization model defined in Eq. (3) and combine this with the first stage. Instead of scheduling data transmission at the packet level, we can simply model the data transmission between computation tasks as a whole by adding appropriate delays between tasks. By taking this approach, we no longer consider the data channels as resources and remove them from our model. Of course, another implication of relaxing the optimization model in Eq. (3) is that the data transfer capacity between cores becomes unbounded.

VFI problem requires clustering (grouping) cores into some regions that each region consists of cores running at the same voltage and frequency levels. Forming these regions is generally not arbitrary but it requires some assumptions regarding the shapes of these regions. VFIs are usually formed as rectangular or square blocks (regions) [2,11,15,16,23]. The size and the number of such blocks may depend on particular design constraints. We can assume that clustering the cores may be considered as placing some number of smaller blocks on a mesh. To further simplify the problem, one can assume that sizes and number of these blocks are set ahead of the optimization. For the sake of our argument, recall that there are n_c core types in our heterogeneous design and let's assume that we set the number of blocks (islands) as $n_c - 1$. Considering that one particular core type is the default type across the mesh, we can place one block for each of the remaining $n_c - 1$ core (CPU) types. Fig. 5 depicts such a VFI layout. In Fig. 5, three core types are depicted as in our experimental setup in the previous section. Dotted lines represent the borders of the VFI regions.

Once the number of blocks and their sizes are fixed, problem of placing them to appropriate positions becomes equivalent to limited resource allocation problem. Assume that a 2-D mesh, like the one in Fig. 5, is represented by x and y axes. Since there are limited capacities along both x and y axes, we can model the area usage along each axis by interval decision variables (borrowed from CP scheduling representation) that have a dimension equal to the number of blocks. In other words, we need to define two interval decision variables for each block, i.e. one for each axis. These decision variables can represent the positions of where each block starts and ends along the x and y axes. Considering the south-west corner of Fig. 5 corresponds to origin (1, 1), the smaller block has starting and ending points of 1 and 3 respectively along the x axis and has starting and ending points of 5 and 7 respectively along the y axis. We can determine the north-west and south-east corners of each block and then force the cores that fall into this area to be a particular type in our optimization model.

A practical and efficient CP based model of VFI problem can be given as in Eq. (5). Notice that permutation variables are also used in Eq. (5) by borrowing the idea from Eq. (1). This eliminates the Stage I in our previous optimization model. Flow and distance matrices (f and d) in Eq. (1) are incorporated into VFI model in the form of delay time (ϕ) between individual tasks where Ł is the set of all links in CTG. Same as in Eq. (3), we can utilize the decision variables, q, to determine the core types from a pool of n_c types. A new constraint type, `"alwaysIn"`, is used to ensure that blocks (VFI regions) are placed within boundaries of the mesh along each axis. Although not shown here for clarity, two more logical constraints are also used in our formulation to prevent overlapping of the blocks and to assign the proper core type within each block.[3] As in Eq. (3), the objective of our VFI model is to minimize completion time of all tasks. A practical CP model is proposed in Eq. (5) to concurrently solve three different problems: (1) mapping cores, (2) grouping cores, and (3) scheduling the tasks.

[3] A working model and a sample data file are provided at http://bit.ly/MeBkRI.

Table 4
MCSL VFI maximum (worst case) completion times.

Application	VFI composition					
	3-3	3-4	3-5	4-3	4-4	5-3
ROBOT	78,073	73,898	68,592	76,255	73,863	76,198
RS32DEC	2463	2383	2384	2478	2429	2499
RS32ENC	1659	1632	1639	1713	1803	1765
SPARSE	18,935	16,625	15,941	17,776	17,085	18,067
FPPP	69,917	66,877	67,537	71,172	67,506	72,427
H264	14,535,231	14,532,841	14,532,645	14,533,715	14,533,528	14,533,888

$$
\begin{aligned}
&\underset{t_i\in\mathcal{T},q\in Q,\pi\in\Pi,\phi\in\Phi,x,y}{\text{minimize}} \ \max(\text{EndOf}(t_i))\\
&\text{subject to}\\
&\quad \text{alldifferent}(\pi_1,\ldots,\pi_n),\\
&\quad \sum_{k=1}^{n_c} q_{jk}=1, \quad j=1,\ldots,n,\\
&\quad \text{sizeOf}(t_i)=\sum_{k=1}^{n_c}\alpha_k\sum_{j=1}^{n} q_{jk}\ \text{JD}(t_i), \quad \text{for all } t_i\in\mathcal{T}\\
&\quad \text{endBeforeStart}(t_i,t_j,\phi_{ij}), \quad \text{for all } t_i\ll t_j\in\mathcal{T}\\
&\quad \phi_{ij}=f_{ij}d_{\pi_i\pi_j}, \quad \text{for all } i,j\in Ł\\
&\quad \text{alwaysIn}(x_{i-1},0,m), \quad i=2,\ldots,n_c\\
&\quad \text{alwaysIn}(y_{i-1},0,m), \quad i=2,\ldots,n_c\\
&\quad q_{jk}\in\{0,1\}, \quad j=1,\ldots,n;\ \ k=1,\ldots,n_c\\
&\quad \pi_i\in\{1,\ldots,n\}, \quad i=1,\ldots,n.
\end{aligned}
\tag{5}
$$

Although the number of blocks and their sizes are provided to our model as parameters, we can easily utilize an iterative approach to find the optimal layouts of various sizes and shapes (both rectangular and square) of VFI compositions. Experimental study can be conducted to determine optimal VFI layouts from a pool of predetermined VFI compositions. For this purpose, we utilize the same benchmark sets and the same assumptions regarding heterogeneous design (such as core types and their performances) as in the previous section. Recall that three types of cores are used in our experiments: namely slow, regular and fast. We set the slow core type as default in our VFI experiments. Types 2 and 3 (i.e. regular and fast) are tested in various square formations. Considering that an 8×8 architecture is used in our experiments, we vary the side length of the square blocks between 3 and 5 units. For example, the VFI layout given in Fig. 5 corresponds to 4×4 and 3×3 blocks of core types 2 and 3, respectively. All the benchmark sets are used in VFI experiments for 8×8 architecture.

We use following VFI layouts for types 2 and 3 cores respectively: 3-3, 3-4, 3-5, 4-3, 4-4, and 5-3. Square blocks are considered in our experiments for their simplicity but rectangular shapes are also implementable in our formulation. Table 2 depicts average completion times over ten realizations of corresponding application. They are consistent with the results from heterogeneous design experiments (see Table 1). Complexity of VFI model is much simpler than two-stage model for the heterogeneous design in the previous section. The largest application, H264 was used in VFI experiments. We set the optimization runtimes of FPPPP and H264 applications as 1200 s and the remaining runtimes were set to 300 s, allowing iterative runs on various VFI compositions were possible. In most of the VFI experiments 3-5 VFI composition was preferable except for RS32DEC and RS32ENC applications in which 3-4 VFI composition was the best. Tables 3 and 4 show the completion times (i.e. objective value of Eq. (5)) of all the applications. They present minimum and maximum completion times across the VFI compositions used in our experiments.

6. Concluding remarks

A CP-based two-stage model is proposed to solve the core mapping and the application scheduling problems for heterogeneous NoC architectures. The major advantage of using CP is the clarity and understandability of models. We successfully experimented our model on various MCSL benchmark datasets. Surface/area constraint has been introduced to our formulation for the heterogeneous architecture. It has been shown that similar constraints for power and temperature can easily be implemented in the CP framework of heterogeneous NoC architectures. We also implement this approach in a VFI-aware fashion, where cores are mapped according to the optimal voltage levels while maintaining satisfactory task schedules. Our CP approach for VFI problem is composed of a single-stage model which requires prior size and shape assumptions regarding islands. Under these assumptions, single-stage model becomes possible via an elegant model.

References

[1] Marculescu R, Ogras ÜY, Peh L-S, Jerger NDE, Hoskote YV. Outstanding research problems in noc design: system, microarchitecture, and circuit perspectives. IEEE Trans CAD Integr Circ Syst 2009;28(1):3–21.
[2] Ogras ÜY, Marculescu R, Marculescu D, Jung EG. Design and management of voltage-frequency island partitioned networks-on-chip. IEEE Trans VLSI Syst 2009;17(3):330–41.
[3] Lombardi M, Milano M. Optimal methods for resource allocation and scheduling: a cross-disciplinary survey. Constraints 2012;17(1):51–85.
[4] Ruggiero M, Bertozzi D, Benini L, Milano M, Andrei A. Reducing the abstraction and optimality gaps in the allocation and scheduling for variable voltage/frequency mpsoc platforms. IEEE Trans Comput-Aid Des Integr Circ Syst 2009;28(3):378–91.
[5] Demiriz A, Bagherzadeh N, Alhussein A. Cpnoc: on using constraint programming in design of network-on-chip architecture. In: 2013 21st Euromicro international conference on parallel, distributed and network-based processing (PDP); 2013. p. 486–93.
[6] Demiriz A, Bagherzadeh N, Alhussein A. Using constraint programming for the design of network-on-chip architectures. Computing 2013:1–14.
[7] Demiriz A, Bagherzadeh N. On heterogeneous network-on-chip design based on constraint programming. In: Proceedings of the sixth international workshop on network on chip architectures. ser. NoCArc '13. New York, NY, USA: ACM; 2013. p. 29–34.
[8] Liu W, Xu J, Wu X, Ye Y, Wang X, Zhang W, et al. A noc traffic suite based on real applications. In: 2011 IEEE computer society annual symposium on VLSI (ISVLSI); 2011. p. 66–71.
[9] Demirbas D, Akturk I, Ozturk O, Gudukbay U. Application-specific heterogeneous network-on-chip design. Comput J 2013.
[10] Mishra AK, Mutlu O, Das CR. A heterogeneous multiple network-on-chip design: an application-aware approach. In: Proceedings of the 50th annual design automation conference. ser. DAC '13. New York, NY, USA: ACM; 2013. p. 36:1–36:10.
[11] Hu J, Shin Y, Dhanwada N, Marculescu R. Architecting voltage islands in core-based system-on-a-chip designs. In: Proceedings of the 2004 international symposium on low power electronics and design, 2004. ISLPED '04; August 2004. p. 180–5.
[12] Wu H, Wona M, Liu I-M. Timing-constrained and voltage-island-aware voltage assignment. In: Design automation conference, 2006 43rd ACM/IEEE; 2006. p. 429–32.
[13] W.-P. Lee, H.-Y. Liu, and Y.-W. Chang, "Voltage island aware floorplanning for power and timing optimization. In: IEEE/ACM international conference on computer-aided design, 2006. ICCAD '06; November 2006. p. 389–94.
[14] Li H-Y, Jiang I-R, Chen H-M. Simultaneous voltage island generation and floorplanning. In: SOC conference (SOCC), 2010 IEEE international; September 2010. p. 219–23.
[15] Ma Q, Young E. Voltage island-driven floorplanning. In: IEEE/ACM international conference on computer-aided design, 2007. ICCAD 2007; November 2007. p. 644–9.
[16] Sengupta D, Saleh R. Application-driven voltage-island partitioning for low-power system-on-chip design. IEEE Trans Comput-Aid Des Integr Circ Syst 2009;28(3):316–26.
[17] David R, Bogdan P, Marculescu R, Ogras U. Dynamic power management of voltage-frequency island partitioned networks-on-chip using intel's single-chip cloud computer. In: 2011 Fifth IEEE/ACM international symposium on networks on chip (NoCS); May 2011. p. 257–8.
[18] Bogdan P, Marculescu R, Jain S, Gavila R. An optimal control approach to power management for multi-voltage and frequency islands multiprocessor platforms under highly variable workloads. In: 2012 Sixth IEEE/ACM international symposium on networks on chip (NoCS); May 2012. p. 35–42.
[19] Ozturk O, Kandemir M, Chen G. Compiler-directed energy reduction using dynamic voltage scaling and voltage islands for embedded systems. IEEE Trans Comput 2013;62(2):268–78.
[20] Majzoub S, Saleh R, Wilton SJE, Ward R. Energy optimization for many-core platforms: communication and pvt aware voltage-island formation and voltage selection algorithm. IEEE Trans Comput-Aid Des Integr Circ Syst 2010;29(5):816–29.
[21] Chen D, Cong J, Fan Y, Xu J. Optimality study of resource binding with multi-vdds. In: Design automation conference, 2006 43rd ACM/IEEE; 2006. p. 580–5.
[22] Qiu X, Ma Y, He X, Hong X. Voltage island aware incremental floorplanning algorithm based on milp formulation. In: 9th international conference on solid-state and integrated-circuit technology, 2008. ICSICT 2008; October 2008. p. 2264–7.
[23] Lee W-P, Liu H-Y, Chang Y-W. An ilp algorithm for post-floorplanning voltage-island generation considering power-network planning. In: Proceedings of the 2007 IEEE/ACM international conference on computer-aided design, ser. ICCAD '07; 2007. p. 650–5.
[24] Chung ES, Milder PA, Hoe JC, Mai K. Single-chip heterogeneous computing: Does the future include custom logic, fpgas, and gpus? In: Proceedings of the 2010 43rd annual IEEE/ACM international symposium on microarchitecture. ser. MICRO '43. Washington, DC, USA: IEEE Computer Society; 2010. p. 225–36.
[25] Zidenberg T, Keslassy I, Weiser U. Multiamdahl: how should I divide my heterogenous chip? IEEE Comput Architec Lett 2012;11(2):65–8.
[26] Hoskote Y, Vangal S, Singh A, Borkar N, Borkar S. A 5-ghz mesh interconnect for a teraflops processor. IEEE Micro 2007;27(5):51–61.
[27] Taylor M, Lee W, Amarasinghe S, Agarwal A. Scalar operand networks: on-chip interconnect for ilp in partitioned architectures. In: The ninth international symposium on high-performance computer architecture, 2003. HPCA-9 2003. Proceedings; February 2003. p. 341–53.
[28] Bhat S. Energy models for network-on-chip components; 2005.
[29] Kim H, Ghoshal P, Grot B, Gratz P, Jimenez D. Reducing network-on-chip energy consumption through spatial locality speculation. In: 2011 Fifth IEEE/ACM international symposium on networks on chip (NoCS); May 2011. p. 233–40.
[30] Ye T, Benini L, De Micheli G. Analysis of power consumption on switch fabrics in network routers. In: Design automation conference, 2002. Proceedings. 39th; 2002. p. 524–9.
[31] Dempster AP, Laird NM, Rubin DB. Maximum likelihood from incomplete data via the em algorithm. J Roy Stat Soc Ser B (Methodol) 1977;39(1):1–38.

Ayhan Demiriz is an Associate Professor in the Department of Industrial Engineering at Sakarya University. His Ph.D. research was primarily on Semi-Supervised Learning, Support Vector Machines and Boosting problems. Prior to joining Sakarya, he worked for Verizon on Click-Stream Analysis and Product Recommendation problems. His current research topics are applications of constraint programming and location intelligence.

Nader Bagherzadeh is a Professor in the Department of EECS at the University of California, Irvine, where he served as a chair from 1998 to 2003. Dr. Bagherzadeh has been involved in research and development in the areas of: computer architecture, reconfigurable computing, VLSI chip design, network-on-chip, 3D chips, sensor networks, and computer graphics.

Ozcan Ozturk is an Associate Professor in the Department of Computer Engineering at Bilkent University. His research interests are in the areas of manycore accelerators, on-chip multiprocessing, and computer architecture. Prior to joining Bilkent, he worked in Cellular and Handheld Group at Intel and Marvell. His research has been recognized by Fulbright, Turk Telekom, IBM, Intel, and EC FP7.