Advances in Engineering Software 42 (2011) 868–874

Contents lists available at ScienceDirect

Advances in Engineering Software

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

Cluster-based application mapping method for Network-on-Chip

Suleyman Tosun *

Computer Engineering Department, Ankara University, Besevler, 06500 Ankara, Turkey

ARTICLE INFO

Article history:
Received 29 March 2011
Received in revised form 11 May 2011
Accepted 6 June 2011
Available online 7 July 2011

Keywords:
Network-on-Chip
Mesh topology
Application mapping
Clustering
ILP
Communication

ABSTRACT

Network-on-Chip (NoC) is a newly introduced paradigm to overcome the communication problems of System-on-Chip architectures. Mapping applications onto mesh-based NoC architecture is an NP-hard problem and several heuristic methods have been presented to solve it so far. Scalability is the main problem of the heuristic methods and it is very difficult to conclude that one heuristic is better than the others. Integer Linear Programming (ILP) based methods determine the optimum mappings. However, they take very long execution times. In this paper, we propose a clustering based relaxation for ILP formulations. Our experiments conducted on several multimedia benchmarks and custom graphs show that the proposed method obtains optimal or close to optimal results within tolerable time limits.

1. Introduction

Network-on-Chip (NoC) has been proposed in the beginning of this century as a new communication infrastructure for integrated circuits [1,2]. NoC architectures mimic the traditional interconnection network concepts. Although there are various topologies exist for NoC architectures, the basic and well accepted topology is the mesh topology. Even commercial multi-core architectures have already adopted mesh based topology such as Intel's Teraflops Research Chip [3]. In this chip, 80 processing cores are connected in a 2D mesh network. One of the biggest problems of the mesh-based NoC architecture remains as optimum application mapping.

Application mapping onto mesh topologies has been a well known NP-hard problem [4]. There have been several methods [5–10] proposed so far to solve it, mainly having the energy minimization as an objective criteria. While [2] presents a mapping algorithm called PMAP that supports single-minimum-path routing and split-traffic routing, [6] proposes a fast branch-and-bound algorithm that exploits the routing flexibility and improves the solution quality. MOCA [7] uses slicing tree based task mapping and generates routes on the mapping result. ONYX [8] and CastNet [9] are two heuristic methods that use the symmetric property of the mesh as a starting point. ONYX maps the tasks based on the lozenge-shaped path order, whereas CastNet maps them one by one based on the communication weights between candidate tasks and mapped tasks. CGMAP [10] employs chaos-genetic-based algorithm that obtains close results compared to other meta-heuristic algorithms. However, none of these methods guarantee the optimal mapping onto mesh architectures.

The optimum solutions can be obtained by Integer Linear Programming (ILP) based methods. ILP is a mathematical method for obtaining the best solution among several alternatives. Our mapping problem can be expressed as a linear programming problem having the communication cost minimization as an objective function. However, since the ILP method searches for every possible solution in the huge solution space, it may take very long CPU times to determine the optimum solution. Our earlier work in [11] demonstrates this performance bottleneck of ILP-based method. While one solution to overcome this timing problem can be efficient numerical implementation of Simplex method as suggested in [12] another solution can be decomposing the constraint variables into finite number of polyhedral and solve for each decomposition. In our case, we picked the latter method.

In this paper, we present a cluster-based ILP formulation for application mapping problem for mesh-based NoC architectures. Our method partitions the task graph, representing the given application, and the mesh into smaller sub-graphs and sub-meshes to decompose the given solution space into smaller polyhedral. It then maps each sub-graph onto the corresponding sub-mesh using our ILP-based method. Finally, it merges each mapping to determine the final solution. We implemented our method using commercial ILP tool [13] and tested its effectiveness on several multimedia benchmarks and randomly generated graphs. Our experiments show that the proposed method is very effective to reduce the execution times of ILP method while determining similar results.

The rest of this paper is organized as follows. Section 2 defines the mapping problem. Section 3 presents the ILP formulations.

* Tel.: +90 544 337 6547.
E-mail address: stosun@ankara.edu.tr

0965-9978/$ - see front matter © 2011 Elsevier Ltd. All rights reserved.
doi:10.1016/j.advengsoft.2011.06.005

Section 4 presents our cluster-based mapping method. Section 5 gives the experimental results. Finally, Section 6 concludes this paper.

2. Problem definition

In this section, we formally define the application mapping problem. We use communication task graph (*CTG*) and topology graph (*TG*) to represent the given application and the mesh architecture, respectively.

Definition 1. A *CTG* is a graph $G(V,E)$, where each vertex $v_i \in V$ represents a task in the application and each edge $e_{i,j} \in E$ represents a dependency between two tasks v_i and v_j. The amount of data transfers between v_i and v_j is represented by the weight $w_{i,j}$ for all $e_{i,j}$ and it is given in bits per second.

Fig. 1a shows an example *CTG*. This *CTG* represents the multimedia benchmark 263-enc mp3-dec presented in [16], where the weights of the edges in this *CTG* represent the amount of data transfer between two tasks in kbits/s. In this example, we have twelve vertices meaning that we need at least twelve tiles to map these tasks onto since we bound our ILP formulations to map at most one task to a tile. If one would like to map more than one task to a tile, he/she must add a preprocessing step to determine which tasks could be mapped on the same tile based on the tasks' worst case execution times. Then, the selected tasks can be merged into a single node to be mapped onto a tile.

Definition 2. A *TG* is a graph $M(P,L)$, where each node $p_i \in P$ denotes a tile (i.e. a processing core) in the topology and each edge denotes a physical link $l_{i,j} \in L$ between p_i and p_j.

Fig. 1b shows an example *TG* with twelve tiles connected in a 4×3 mesh fashion. As illustrated in this figure, each tile has 2D coordinates in the mesh. In Fig. 1c, we give the structure of a tile. A tile contains a router to forward data between tiles, a memory to store program and storage data, and a processing core to process data.

Using the above definitions, the application mapping problem can be formulated as follows:

Given a *CTG* and a *TG* that satisfy

$$|V| \leqslant |P| \tag{1}$$

find a one to one mapping function $F: V \rightarrow P$ from *CTG* to *TG* with

$$MIN\left\{ CommCost = \sum_{\forall e_{i,j} \in E} \eta_{f(v_i)f(v_j)} \times w_{i,j} \right\} \tag{2}$$

such that:

$$\forall v_i \in V, \exists p_k \in P, \quad f(v_i) = p_k \tag{3}$$

$$\forall v_i \neq v_j \in V, \quad f(v_i) \neq f(v_j) \tag{4}$$

where *CommCost* is the total communication (i.e. the number of bits transferred per second between tiles.) on the network and $\eta_{f(v_i)f(v_j)}$ is the minimal path (i.e. the minimum number of hops) between tiles $f(v_i) = p_k$ and $f(v_j) = p_l$. Minimizing *CommCost* is directly proportional to η and our solution aims at minimizing the number of hops, η, resulting in minimized *CommCost*.

In our problem formulation, we do not consider the bandwidth constraints and assume the minimal path routing between every communicating tasks. XY routing algorithm [17] is the commonly accepted deterministic routing algorithm because of its simplicity and very little hardware requirements. However, it may cause network congestion if two communication paths share the same link and this may cause latency overhead. There are several minimal path routing algorithms exist for mesh based NoC's [7,9,14] that can be applied to our mapping result to eliminate congestion. Additionally, virtual channels can be used in router ports for congestion control.

3. ILP formulation

In an ILP problem, problems are formulated using a linear objective function and linear functions as constraints, whereas the solution variables are restricted to be integers. In this work, we use 0–1 ILP formulation and the 0–1 ILP is a smaller subset of the general ILP problem in which each (solution) variable is restricted to be either 0 or 1. In this paper, we used *Xpress − MP* [13], a commercial tool, to formulate and solve our ILP problem, though its choice is orthogonal to the focus of this paper. In our ILP formulation, we view the chip area as a 2D grid and assign tasks to tiles within this grid. Table 1 gives the constant terms and variables used in our formulations. We relist ILP formulations presented in [11] in the following paragraphs to make this paper self contained.

For our formulations, we define a binary variable $\alpha_{i,x,y}$, indicating that task i is mapped to a tile in the coordinate (x,y) if $\alpha_{i,x,y} = 1$, otherwise $\alpha_{i,x,y} = 0$. In the following formulations, Eq. (5) indicates that every task i must be mapped to a tile with the coordinates (x,y) and only one task can be mapped to a single tile. The number of tasks in the *CTG* may be less than the number of available tiles. In this case, there will be some tiles that have no tasks mapped on them. Eq. (6) captures this constraint.

$$\sum_{x=0}^{Xdim} \sum_{y=0}^{Ydim} \alpha_{i,x,y} = 1, \quad \forall i. \tag{5}$$

$$\sum_{i=1}^{n} \alpha_{i,x,y} \leqslant 1, \quad \forall x, y. \tag{6}$$

As indicated in Eq. (2), in order to calculate the total communication of the architecture (i.e. *CommCost*), we must calculate the minimum number of hops $\eta_{f(v_i)f(v_j)}$ between two tasks v_i and v_j mapped on the mesh. The Manhattan distance (i.e. the city block distance) gives the minimum number of hops between two tiles.

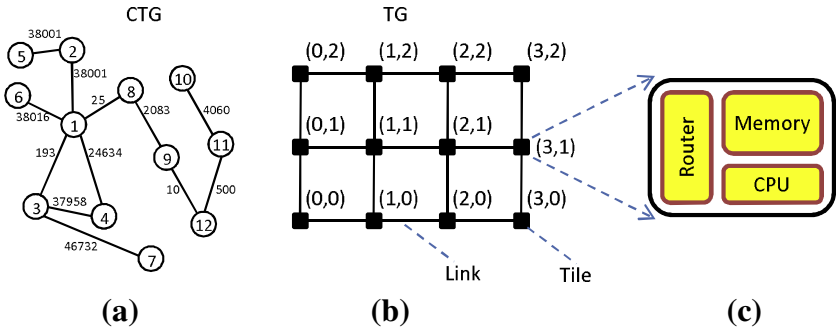

Fig. 1. (a) An example CTG, (b) An example TG with the size of 4 × 3, and (c) The structure of a tile.