Journal of Systems Architecture 57 (2011) 69–78

Contents lists available at ScienceDirect

Journal of Systems Architecture

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

New heuristic algorithms for energy aware application mapping and routing on mesh-based NoCs

Suleyman Tosun

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

ARTICLE INFO

Article history:
Received 26 March 2010
Received in revised form 22 July 2010
Accepted 1 October 2010
Available online 25 October 2010

Keywords:
Network-on-Chip
Application mapping
Routing
Mesh topology

ABSTRACT

Ever shrinking technologies in VLSI era made it possible to place several IP (Intellectual Property) blocks onto a single die. This technology improvement also brought the challenge of inventing new communication methods since traditional bus-based systems suffer from signal propagation delays, signal integrity, and scalability on these large platforms. Network-on-Chip (NoC) is the biggest step towards the solution of this communication bottleneck of System-on-Chip (SoC) architectures. Topology selection is the very first step of designing NoC systems and mesh topology is the commonly accepted topology type for NoCs. However, mapping applications represented by the weighted task graphs onto the mesh architectures is an NP-hard problem. In this paper, we present a new low complexity heuristic algorithm, CastNet, for the application mapping and bandwidth constrained routing algorithm for mesh-based NoC architectures aiming to minimize the energy consumption. We compared the presented approach with the one that generates the optimal solutions and two existing heuristic methods. Our experiments on multimedia benchmarks show that the proposed mapping algorithm obtains optimal or, in worst case, within 6% to the optimal solutions in very short times.

1. Introduction

Rapid improvements in VLSI technology sizes made it possible to integrate multiple components of an electronic system on a single chip. Such a system that contains several components (e.g. processors, memory blocks, analog interfaces, DSPs, etc.) is called System-on-Chip (SoC) [1]. The increase in the number of IP (Intellectual Property) blocks on a SoC platform and high throughput demand of today's applications forced the designers to investigate new communication methods as a superior alternative to the traditional bus-based or point-to-point based communication structures. In the beginning of this millennium, Network-on-Chip (NoC) technology, also known as on-chip interconnection network (OCIN), has been proposed [2] for parallel execution of system components to meet the system needs for performance, power consumption, and high throughput.

NoC inherits the traditional computer network concepts and mimics it on the SoC platforms for on-chip communication. Several studies and real industrial implementations demonstrated the substantial performance gain of the NoC systems over conventional bus based systems. However, NoC system design methodologies are still in their early phases and there are several problems ahead to be solved such as the need for synthesis tools for topology generation or selection; for optimal, fault-tolerant, deadlock free routing mechanisms; and for mapping and scheduling algorithms [3].

Selecting the most suitable topology for the given application is the first step in designing a NoC architecture. This synthesis step is not trivial since the optimization parameters such as energy, cost, and performance may vary from one topology to another. Additionally, the flow control and the routing mechanisms depend heavily on the selected topology. The topologies for NoC architectures can be categorized into two main groups based on their architectural structures; namely, regular topologies and irregular topologies. Mesh, torus, fat tree, butterfly topologies are the examples for the regular topologies [4]. The irregular topologies are custom generated topologies based on the communication structures of the application [5].

Irregular topologies demonstrate better performance characteristics than their regular counterparts since they are costumed for the given application allowing more optimization area. However, they lack the reusability. Additionally, the design procedures of the regular topologies are simpler than irregular topology generation. Existing industrial interconnection networks use different regular topologies such as ring [6,7], crossbar [8], and mesh [9]. One important point to note is that the final product of the design procedure will be a 2D VLSI implementation of the network. Thus, we must avoid topologies that results in die stacking, which introduces need for 3D networks. A simple, regular mapping to 2D VLSI implementation, and most commonly used topology for the NoC

E-mail address: tosun@eng.ankara.edu.tr

1383-7621/$ - see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.sysarc.2010.10.001

systems is the mesh topology. In a mesh topology, the cores (or the components of the system) are connected to each other in a grid fashion. Intel's teraflops chip is one good example that uses mesh topology for its 80 cores that are connected in a 8×10 grid. The leading researchers from academia and industry proposed the next generation chip multiprocessor (CMP) of the year 2015, which is also build on a mesh topology of size 16×16 [10]. After selecting the topology and its flow control mechanism for the application, the next step is mapping the tasks of the application onto the cores of the mesh and selecting a deadlock free routing path between communicating tasks.

Mapping an application onto the mesh architecture is known to be NP-hard. If the number of cores on the target architecture has enough cores for the given application, $n!$ different solutions can be found for the application with n tasks. There have been several algorithms presented so far [11–17] for the problem of application mapping onto the mesh architectures. These methods use different optimization techniques such as integer linear programming (ILP) [11], heuristic methods [12,13], simulated annealing [16], and genetic algorithms [17] to minimize the total communication cost or the total energy consumption of the system. Some of these methods [11–13,15] have high complexities resulting in huge increase in computation times. Some of them [16,17] do not guarantee optimal results and may not obtain desirable solutions.

In this paper, we propose a new low complexity heuristic algorithm called CastNet for the application mapping onto the mesh based Network-on-Chip architectures. Our algorithm aims to map highly communicating tasks close to each other aiming to minimize the path the data travel through. To do this, it first analyzes the mesh architecture based on its dimensions and determines the initial mapping positions for the first task. Then, it gives a priority to each task on the application graph. After the first task is mapped on to the initial core, our algorithm maps remaining tasks one by one based on their communication weight between mapped tasks. We next present a routing table-based deterministic routing method that runs under bandwidth constraints. The routing algorithm uses the mapping result from our previous step and aims to minimize the required bandwidth of the system. We tested our CastNet algorithm on several real benchmarks and custom generated graphs and in the worst case, we obtained energy consumption results within 6% to the optimal solutions. We also evaluated the performance of our routing algorithm by comparing it with XY, odd–even, and west-first algorithms.

The remaining of this paper is organized as follows. We give the problem definition and a motivational example in Section 2. We explain the details of our mapping and routing algorithms in Sections 3 and 4, respectively. We present the complexity analysis of our algorithms in Section 5. We give our experimental results illustrating the effectiveness of our approach in Section 6. Finally, we conclude this paper in Section 7.

2. Problem formulation

In this section, we formally define the application mapping and routing problems and illustrate the effects of different mappings on the total energy consumption with an example.

2.1. Energy model

In this study, our goal is minimizing the energy consumed by the network resources of the NoC architecture. The energy consumption of the network is directly proportional to the amount of bit transitions on the network. In order to estimate the energy consumption of NoC architecture, we should use an energy model based on the total bit transitions. This work uses a well-accepted energy model given in [13] as,

$$E_{T_{bit}} = E_{S_{bit}} + E_{B_{bit}} + E_{W_{bit}} + E_{L_{bit}} \quad (1)$$

where $E_{S_{bit}}$, $E_{B_{bit}}$, $E_{W_{bit}}$, and $E_{L_{bit}}$ represent the energy consumed by the switches, buffers, interconnection wires inside the fabric, and the links, respectively. Since the energy consumption of the buffering ($E_{B_{bit}}$) and internal wires ($E_{W_{bit}}$) are negligible, Eq. (1) can be reduced to:

$$E_{T_{bit}} = E_{S_{bit}} + E_{L_{bit}} \quad (2)$$

Then, the average energy consumption of sending one bit data from core i to core j can be calculated by:

$$E^{i,j}_{T_{bit}} = (\eta_{i,j} + 1) \times E_{S_{bit}} + \eta_{i,j} \times E_{L_{bit}} \quad (3)$$

where $\eta_{i,j} + 1$ is the number of routers the bit passes through. If a bit passes through $\eta_{i,j} + 1$ routers, clearly, it also passes through $\eta_{i,j}$ links, which is called the number of hop distances between the cores i and j.

Eq. (3) shows that, minimizing the number of hop distances between the communicating cores gives us the minimized energy consumption of sending one bit of data from core i to core j. For 2D mesh networks, the minimal hop distance $\eta_{i,j}$ is determined by the Manhattan distance between core $i(x_i, y_i)$ and core $j(x_j, y_j)$ as follows:

$$\eta_{i,j} = |x_i - x_j| + |y_i - y_j| \quad (4)$$

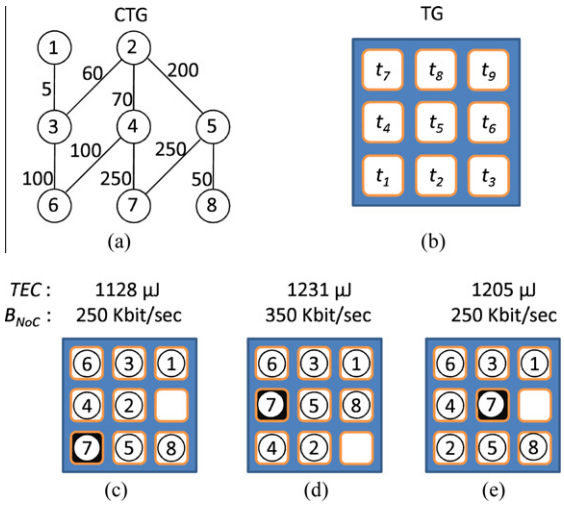

Fig. 1. (a) An example *CTG*, (b) an example *TG* with the size 3×3, and (c–e) different mappings of *CTG* onto *TG* generated by CastNet.

2.2. Problem formulation

Before elaborating the details of the mapping and routing problems, we first give the necessary definitions.

Definition 1. A *CTG* is a graph $G(V, E)$, where each vertex $v_i \in V$ represents a task (i.e. a node[1]) 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. 1(a) shows an example *CTG* taken from [11]. The weights of the edges in this *CTG* represent the amount of data transfer between two tasks in Kbits/s.

[1] We use the words *task* and *node* interchangeably throughout the paper.