# Communication-Aware Task Scheduling for Multi-Core Architectures with Segmented Buses

Yuping Zhang
School of Computer
Wuhan University
Wuhan,China
ypzhang88@126.com

Xianbin Xu
School of Computer
Wuhan University
Wuhan,China
xbxu@whu.edu.cn

Yuanhua Yang[1 2]
1 School of Computer
Wuhan University
2 Jianghan Art
Vocational College
yangyuanhua123@163.com

Shuibing He
School of Computer
Wuhan University
Wuhan,China
heshuibing@whu.edu.cn

Zimian Hao
Hubei Urban
Construction
Vocational and
Technical College
Wuhan,China
haozimian@yahoo.com.cn

***Abstract*—As the number of cores on a single chip and their performance continue to increase, the communication architecture plays a major role in the area, performance, and energy consumption of the overall system. This paper presents a mesh-like connected multi-core architecture with segmented buses to meet the requirements of high performance and low energy consumption. Based on the proposed architecture, a communication-aware greedy task scheduling is designed to minimize the communication energy consumption among cores while maintaining the same performance as other scheduling algorithms. We evaluate the algorithm performance through a series of experiments with Gaussian Elimination, and the experimental results confirm the effectiveness of the algorithm.**

***Keywords-Multi-Core Architecture, Segmented Buses, Communication Energy Consumption , Greedy Algorithm, Gaussian Elimination***

## I. INTRODUCTION AND MOTIVATION

As technology scales toward deep sub-micro (DSM), the integration of a complete system consisting of a large number of cores on a single chip is becoming technically feasible. Multi-core or many-core architectures have been widely used in recent times as a viable solution to the increasing chip densities, due to the benefits offered by them with respect to improving system performance, cost, power dissipation and reusability [1] .

With the rising number of cores integrated into a single chip and their performance continue to increase, the communication architecture plays a major role in the area, performance, and energy consumption of the overall system**.** This means that communication, not computation, will be the key performance bottleneck in DSM technologies [2]. To cope with the increasing multi-core performance requirements and power constraints in the DSM era, on-chip communication architectures have undergone an evolution in complexity: from bus-based architecture to Network-on-Chip (NoC). A lot of researches have been done on the on-chip communication architectures [3-5] .

Kumar et al. present that the design choice for the communication have significant effect on the rest of the chip, potentially consuming a significant fraction of the power budget in [3]. Energy consumption is becoming one of the major optimization targets when designing low power multi-core architectures. Therefore, in the past decade there are many works focusing on exploring bus-based communication architecture for energy conservation, especially a variety of shared buses, segmented bus (or splitted, partitioned bus) [1, 5-10].

Long interconnect wires account for a significant fraction (up to 50%) of the energy consumed in an integrated circuit [6]. Segmented buses have obvious energy-related circuit advantages such as reduced capacitive load during bus transfer. A bus-segmentation method to reduce the switched capacitance on the bus is proposed in [8]. The design theory and implementation issues of a bus segmentation method for lowering the energy dissipation on system buses is provided in [11]. It was reported in [10] that the use of a splitting bus architecture yields energy savings of 16%-50% over a monolithic bus. Therefore, in this article, we design a mesh-like connected segmented buses architecture for minimizing the energy consumption of the communication architecture.

An efficient scheduling of a parallel program onto the cores that minimizes the entire execution time is vital for achieving a high performance in multi-core systems. A varieties of scheduling algorithms have been proposed in the literature [12]. All of these schedules share a common characteristic that they try to cluster or duplicate heavily communicating tasks onto the same processor meanwhile to reduce the overall schedule length. None of them considers the relative communication energy consumption among different cores during scheduling. Recently some researchers have turned their attention to consider the communication energy in designing task scheduling algorithms [13, 14]. But all these methods focus on application-specific segmented bus platform. They are not suitable for task schedulings in general purpose multi-core systems. In this paper, we provided a communication-aware greedy task scheduling to minimize the communication energy consumption among cores while maintaining the same performance as other heuristic scheduling algorithms.

The rest of this paper is organized as follows. Section II describes the mesh-like connected multi-core architecture and the energy model of the communication architecture. The communication-aware greedy task scheduling is detailed in Section III. We investigate the performance of our solution through simulations and analyze the results in Section IV. Conclusion and future work are shown in Section V.

## II. TARGET ARCHITECTURE AND COMMUNICATION ENERGY MODEL

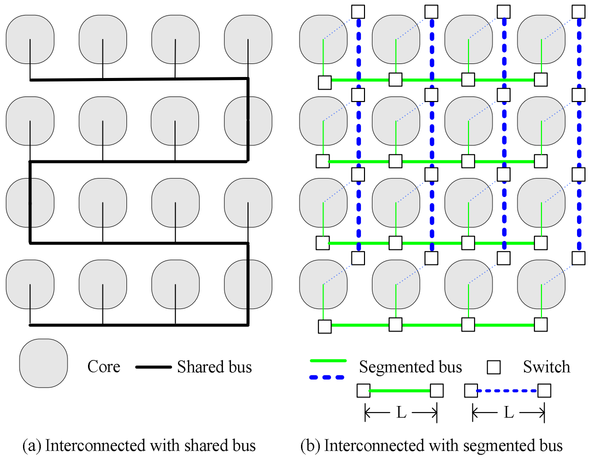
### A. Target Architecture

A traditional shared bus communication architecture is shown in Figure 1(a). All of the cores are connected through one large monolithic shared bus. As the number of the cores increase, the performance of this communication architecture will be significantly decreased because the available bandwidth for each core drops sharply. Moreover, the communication architecture consumes a major fraction of energy in the chip because the whole wire will be activated for every data transfer during the communication of any two cores. To overcome these problems, at the architecture level a lot of studies have been existed for performance improvement and communication energy minimization, especially the segmented bus, such as presented in [1, 5, 6, 8, 10, 11, 14] .

Segmented buses have been proposed in the past for multiple computer architectures [15]. Among various parallel architectures, mesh-connected computers (MCCs) have received considerable attention [16]. Due to its simple and regular interconnection pattern, MCCs are suitable for hardware implementation in VLSI technologies. The mesh, however, has a crucial drawback that its communication diameter is large. To overcome this problem, a variant of the meshes with separable buses have been presented [15-18] . To take advantage of the benefits of meshes and segmented buses, we propose a Mesh-Like Connected Multi-Core Architecture with Segmented Buses (MCMASB), as shown in Figure 1 (b).

In MCMASB architecture, N cores arrange in a $n_1 \times n_2$ grid with a separable bus for each row and each column respectively. For simplicity and clarify, Figure 1 (b) shows a 4×4 mesh with 16 cores. The cores in the same row/column are connected with a separable bus. The horizontal separated solid lines and the vertical separated dotted lines represent the shared buses for the cores on the same row and column respectively. These buses can be segmented by the switches attached on them dynamically during the runtime. Resemble like the cores, the switches are arranged in a grid, and they are placed symmetrically. Each separable bus is partitioned into equal number of segments, and each segment is the same length of "L". The switches are 3 port uni-cast or multi-cast components implemented using tri-state buffer chains (detailed structure seen in [9]). Each switch is configured at runtime to form a path between the source and the sink of the transfer, and the control is light-weight because no handshaking techniques or transfer protocols are needed. The segmented buses in each row and column support multi-cast and concurrent non-overlapping data transfers at the same time. Each segment acts as a normal bus between cores that are connected to it, and is separated from other segmented bus by switches when it is not involved in transactions.

In Figure 1, each core is assumed to contain multiple functional units and a cache. The functional units have direct access to its cache at any moment with negligible communication cost. The data is needed to transfer through the segmented buses only when a functional unit requests the information stored in a remote cache. Each core is attached two



Core
Shared bus
Segmented bus
Switch
L
L
(a) Interconnected with shared bus
(b) Interconnected with segmented bus

Figure 1. Mesh-Like Connected Multi-Core Architecture with Segmented Buses

switches that are used for connecting the core to the horizontal and vertical shared bus respectively. The configurations of the switches are controlled by their corresponding cores at runtime. In the following we compare the architecture of MCMASB with the NoC from several aspects:

- Structure: MCMASB and NoC are regular mesh–based connection architecture, thus both them are feasible for VLSI implementation.
- Scalability: Except for supporting concurrent data transfer on non-overlapping segmented buses, MCMASB can provide multi-cast on all or partial separated row and column buses, therefore the performance scales as well as the NoC.
- Energy: There are two important disadvantages for NoC design [19]. First, a direction-based protocol inherently suffers from indirection, requiring multiple messages for every coherence transaction. Second, the messages traverse multiple routers require more clock cycles. All of these mean more execution time and energy consumption for the operation to complete. In addition, the routers are energy-hungry devices. However, the segmented buses are presented for energy conservation, and the cost of driving the switches of the segmented bus can be negligible compared with the gain obtained from the segmentation [9].

Except for the above discussions, at the present, NoC do not always provide the huge predicted impact on the design process [5]. However, the segmented bus architecture have been successfully been employed in some implementation of multi-core systems. Therefore the proposed MCMASB architecture is completely feasible to meet the requirements of the high performance and low energy dissipation.

### B. Communication Energy Model

A well-known communication energy model is as follows:

$$\mathrm{E} = \alpha \times C \times V_{DD}^2 \qquad (1)$$

Where $\alpha$ is the switching activity of the signal that is being transmitted, $C$ is the physical capacitance switched during signal transitions, including the total capacitance of interconnects and the capacitance of driving and driven gates, and $V_{DD}$ is the supply voltage.

The energy cost of the segmented bus depends directly on the wire length and the switching activity of each segment. Based on (1) and the MCMASB architecture, the energy model of segmented buses can be computed as follows:

$$E = V_{DD}^2 \times C_L \times \sum_{i,j \in N} \alpha(i,j) \times Num_{seg}(i,j) \times L \qquad (2)$$

Where $C_L$ is the capacitance introduced by each segmented bus, $\alpha(i, j)$ is the communication frequency of the data that are transmitted between $\mathrm{Core}_i$ and $\mathrm{Core}_j$. $Num_{seg}(i, j)$ is the number of segmented bus from $\mathrm{Core}_i$ to $\mathrm{Core}_j$. L is the length of each segmented bus.

Minimizing the value of (2) is the objective of this paper for reducing the communication energy consumption. For a specific technology, both $V_{DD}$ and $C_L$ are constants, therefore the (2) is simplified to (3).

$$E = \sum_{i,j \in N} \alpha(i,j) \times Num_{seg}(i,j) \times L \qquad (3)$$

## III. COMMUNICATION-AWARE GREEDY TASK SCHEDULING

In this section, we propose a Communication-Aware Greedy Task Scheduling (CAGTS) algorithm, which is an extension to the concept of traditional list scheduling, for reducing the communication energy consumption. The target system is the MCMASB described in Section II. The proposed CAGTS algorithm is presented in Algorithm 1.

The scheduling parallel program can be modeled as a Directed Acyclic Graph (DAG), in which the nodes and edges represent tasks and messages, respectively. The detailed DAGs can be found in [12]. Given the DAGs and the MCMASB architecture, the proposed CAGTS algorithm is implemented in three steps. Step 1 is to construct a task list according to tasks' priorities. Then, in step 2, the task, $T_i$ with the highest priority is allocated to a core that allows the earliest start time for minimizing the schedule length. Repeat step 2 until all the tasks have been allocated to cores. In the last step calls the greedy algorithm() for reducing communication energy consumption.

As described in Algorithm 1, Step 1 and 2 are two basic steps of traditional list scheduling, so these two steps can be implemented with any list scheduling reported in the literature. The extension of the traditional list scheduling is step 3. The greedy algorithm() is listed in Algorithm 2.

The input of the greedy algorithm is the result, i.e., the task clusters in different cores, which is produced by the traditional list scheduling. The t-th population is represented as P(t). P(t) is an array consisting of M elements, where M is the size of the population. The label $X_i(t)$ indicates that the i-th individual of the t-th population. Each individual is a mapping of task clusters to cores. For example, $X_3(t)=(2,0,1,3)$ shows that this

**Algorithm 1** Task scheduling with CAGTS algorithm

```
Require: DAGs, MCMASB architecture with N cores
Ensure: Optimal communication energy task scheduling
1. Determine the tasks' priorities. Order the tasks into a list
   according to their priorities, respecting their precedence
   constraints.
2. Repeat
   (1) Remove the first task $T_i$ from the list.
   (2) Schedule $T_i$ to the core that allows the earliest start
       time while satisfying the tasks' precedence
       constraints.
   Until the list is empty.
3. Call Greedy algorithm() for reducing communication
   energy consumption.
```

individual is the third element of the t-th population, and the task clusters from 0 to 3 are mapped to 2, 0, 1, and 3 cores respectively (4 cores in the system). The initialization population is generated randomly.

The symbol $f(X_i(t))$ represents that the fitness value of thei-th individual of the t-th population. The greedy algorithm is to minimize the communication energy consumption, thus according to (3), we can draw fitness function as follows:

$$f(X_i(t)) = \sum_{i=1}^{N} \left( \sum_{k=i+1}^{N} CommCost[i,k] \times Dist[i,k] \right) \qquad (4)$$

where $CommCost[i, k]$ denotes the communication frequency of data transfer between $\mathrm{Core}_i$ and $\mathrm{Core}_j$. The distance of these two cores is represented as $Dist[i, k]$, and in the MCMASB architecture, it can be calculated as follows:

$$\begin{aligned} \mathrm{Dist}[i,k] &= \left| Row(Core_i) - Row(Core_j) \right| \\ &+ \left| Col(Core_i) - Col(Core_j) \right| \end{aligned} \qquad (5)$$

In (5), Row() and Col() denote the row and column locations of the core lying in the two-dimensional architecture, respectively.

The next generation is produced by choosing M individuals with low fitness value from the last population and the new individuals generated by the greedy operator, as shown from Line 4 to 9 in Algorithm 2. The greedy operator is described in the following.

The first element of the individual is generated randomly, $X_r(t)=[\mathrm{Core}_1]$, then produces the rest $\mathrm{Core}_j(j=2, \ldots, N)$ by using greedy operator.

The front α elements of individual $X_r(t)$ are assumed to be known, the (α +1)-th element will be choosed according to the following criterion.

$$\min_{Core_{\alpha+1}\in\theta}(f(X_r(t))) = \min_{Core_{\alpha+1}\in\theta}\left(\sum_{i=1}^{\alpha}\left(\sum_{k=i+1}^{\alpha+1} CommCost[i,k]\times Dist[i,k]\right)\right) \tag{6}$$

where $\theta = \Omega - \sigma$, represents the set of the un-assigned core number. There, $\Omega$ is the set of all cores number, and $\sigma$ is the set of cores number which have been allocated to the front $\alpha$ elements of the individual. Repeat this step until all the rest elements are completely produced.

```
Algorithms 2  Greedy algorithm()
Require: The task clusters produced by list scheduling
Ensure:  Optimal communication energy task scheduling
Begin
1.    t:=0
2.    initialize P(t);  P(t) = {X_1(t), X_2(t),…, X_N(t)}
3.    evaluate P(t);  f (P(t )) = {f (X_1(t)), f (X_2(t)),…,f (X_N(t))}
4.    while (not termination condition) do
5.         Pg(t) = greedy {P(t)}
6.         evaluate [Pg(t)]
7.         P(t+1) = select [Pg(t) U P(t) ]
8.         t:=t+1
9.    od
10. print X_best , f (X_best )
end
```

A schedule with the best communication energy consumption is generated when the termination condition or the number of iterations is met.

## IV. EXPERIMENTS AND RESULTS

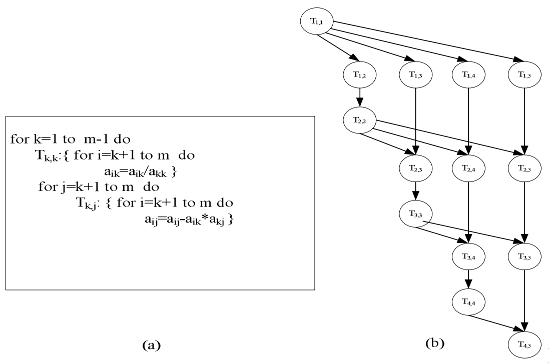
### A. Gaussian Elimination

In our study, we present the comparative evaluation of our algorithm based on gaussian elimination. It is a well known method used in mathematics for solving a system of equations, and represents a real world problems.

Figure 2 (a) shows the sequential program for the gaussian elimination [20]. The DAG for the special case of m=5, where m is the dimension of the matrix, is given in Figure 2 (b). The number of tasks in the task graph of this algorithms can be roughly estimated as $\frac{m^2+m-2}{2}$. $T_{k,k}$ and $T_{k,j}$ represent a pivot column and an update operation respectively. The computation cost function for any task $T_{k,j}$ is equal to $W_{k,j}=2\times(n-k)\times w$, where $1\le k\le j\le m$, and w is the average execution time of either an addition and multiplication, or of a division. The communication cost function of edge between dependent tasks is equal to $C_{k,j}=(n-k)\times b$, where b is the transmission rate. In our experiment, the value of b is set by tuning the Communication-to-Computation Ration (CCR).

### B. Performance Comparison

For the first comparison, we present the communication energy consumption produced by the traditional list scheduling and the proposed CAGTS for various CCR of gaussian elimination. It is noteworthy that all the list schedulings can be used in the first two steps of our algorithm. The proposed technique reduces the communication energy consumption and in the meantime retains the same schedule length as the list scheduling. Therefore, we implement the list scheduling according to the techniques presented in [21] because of its low complexity and efficient solution.



```
for k=1 to  m-1 do
     T_{k,k}:{ for i=k+1 to m  do
                    a_ik=a_ik/a_kk }
      for j=k+1 to m  do
           T_{k,j}: { for i=k+1 to m do
                          a_ij=a_ij-a_ik*a_kj }
```

Figure 2. (a) Gaussian Elimination Algorithm($k_{ji}$ version) (b) DAG for a Matrix of Size 5

Figure 3 shows the communication energy consumption generated by the proposed CAGTS algorithm at various CCR. It is normalized to that of the traditional list scheduling. The results are achieved through a set of gaussian elimination DAGs with the matrix size varying from 5 to 20, and the number of cores in the system is 16. As can be observed from Figure 3, the communication energy dissipation is reduced about 34% at coarse-grain (CCR<1), 25% at fine-grain (CCR>1). The reduction is the least for CCR=1, about 18%. This is because the communication load are distributed uniformly among the cores, the adjustments of task clusters to the cores are more difficult achieved for reducing the communication energy dissipation. The coarse-grain graph means less communication cost among cores, so it is easy to redistribute the task clusters to cores for achieving the best energy reduction. However, fine-grain applications mean heavy communication traffic, the energy reduction less than that of the coarse-grain is reasonable. In addition, it can be observed from the experiments that the energy reductions are constant both for the fine-grain and coarse-grain DAGs. It further confirms the scalability and effectiveness of the proposed algorithm in this article.

Figure 4 illustrates the communication energy reductions at various number of cores. In the simulations, the matrix size of gaussian elimination varies from 5 to 20, CCRs are between 0.2 and 10 (increment is 0.2). The experiments on various number of cores make use of the same set of gaussian elimination. As shown in Figure 4, the reduction of communication energy dissipation is about zero on 4 cores. This is because the diameter is 1 when there are only 4 cores in the system, and the result is consistent with the MCMASB architecture proposed in Section II. From Figure 4 we can see that the reduction of the communication energy dissipation increases as the number of cores in the system increases. For example, the reduction is about 24%, 27%, and 37% for 8, 16, and 32 cores, respectively. We believe that more cores exist in the system, more reduction of communication energy dissipation.

## V. CONCLUSION AND FUTURE WORK

In this paper, we have presented a Mesh-Like Connected Multi-Core architecture with Segmented Buses. The proposed architecture combines the benefits of both the shared bus and the Network-on-Chip. Based on this architecture, an extension of the traditional list scheduling algorithm is proposed for reducing the communication energy consumption. The mapping of task clusters to cores that produced by the traditional list scheduling is adjusted by the proposed communication-aware greedy algorithm in order to minimize the length of the data transferred, the short wire length results in the reduction of the communication energy consumption. We have performed experiments with a variety of gaussian elimination graphs. The results confirm the effectiveness of the algorithm.

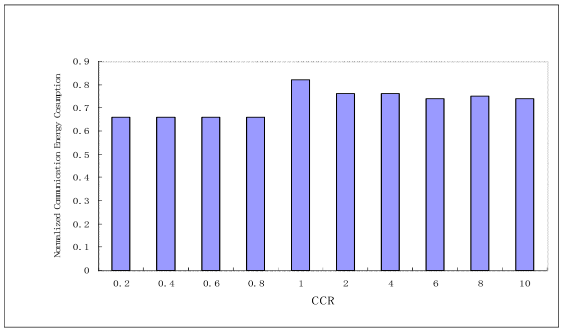


Figure 3. Communication Energy Consumption of CAGTS Normalized to Traditional List Scheduling at Various CCR

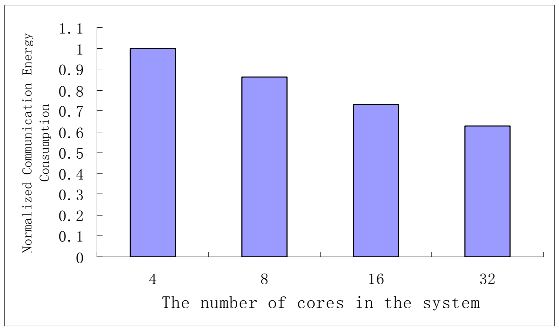


Figure 4. Communication Energy Consumption of CAGTS Normalized to Traditional List Scheduling at Various Cores

In future work, we attempt to implement this algorithm on the systems with heterogeneous multiple cores and the switches placed asymmetrically.

## REFERENCES

[1] Srinivasan S., Li L., and Vijaykrishnan N., *Simultaneous Partitioning and Frequency Assignment for on-Chip Bus Architectures*, in *Proceedings of the conference on Design, Automation and Test in Europe - Volume 1*. 2005, IEEE Computer Society.

[2] Pasricha S. and Dutt N., *Trends in Emerging on-Chip Interconnect Technologies.* IPSJ Transactions on Systems LSI Design Methodology, 2008. **1**: pp. 2-17.

[3] Kumar R., Zyuban V., and Tullsen D. M. *Interconnections in Multi-Core Architecture:Understanding Mechanisms,Overheads and Scaling*. in *Proceeding of the 32nd International Symposium on Computer Architecture*. 2005. pp. 408-419.

[4] Lee H. G., et al., *On-Chip Communication Architecture Exploration: A Quantitative Evaluation of Point-to-Point, Bus, and Network-on-Chip Approaches. ACM Trans. Des. Autom. Electron. Syst., 2007.* ***12****(3): pp. 1-20.*

[5] Seceleanu T., et al., *Improving the Performance of Bus Platforms Bymeans of Segmentation and Optimized Resource Allocation.* EURASIP J. Embedded Syst., 2009. pp. 1-14.

[6] Raghunathan V., Srivastava M. B., and Gupta R. K., *A Survey of Techniques for Energy Efficient on-Chip Communication*, in *Proceedings of the 40th annual Design Automation Conference*. 2003, ACM: Anaheim, CA, USA.

[7] Loghi M., et al., *Analyzing on-Chip Communication in a Mpsoc Environment*, in *Proceedings of the conference on Design, automation and test in Europe - Volume 2*. 2004, IEEE Computer Society.

[8] J.Y.Chen, et al., *Segmented Bus Design for Low-Power Systems.* IEEE TRANSACTIONS ON VLSI SYSTEMS, 1999. **7**(1): pp. 25-29.

[9] Heyrman K., et al. *Energy Costs of Transporting Switch Control Bits for a Segmented Bus*. in *the 16th Annual Wsh. on Circuits, Systems and Signal Processing (ProRisc)* 2005. pp. 359--364.

[10] Hsieh C.-T. and Pedram M., *Architecture Energy Optimization by Bus Splitting.* Proceedings of IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems(IEEE TCAS), 2002. **21**(4): pp. 408-414.

[11] Jone W.-B., et al., *Design Theory and Implementation for Low-Power Segmented Bus Systems.* ACM Trans. Des. Autom. Electron. Syst., 2003. **8**(1): pp. 38-54.

[12] Kwok Y.-K. and Ahmad I., *Static Scheduling Algorithms for Allocating Directed Task Graphs to Multiprocessors.* ACM Comput. Surv., 1999. **31**(4): pp. 406-471.

[13] Korthikanti V. A. and Agha G., *Analysis of Parallel Algorithms for Energy Conservation in Scalable Multicore Architectures*, in *Proceedings of the 2009 International Conference on Parallel Processing*. 2009, IEEE Computer Society.

[14] Guo J., et al. *Physical Design Implementation of Segmented Buses to Reduce Communication Energy*. in *Asia and South Pacific Conference on Design Automation*. 2006. pp. 42-47.

[15] Katsinis C., *A Multicomputer Architecture with a Segmented Shared Bus* Computers & Electrical Engineering, 1995. **21**(1): pp. 33-46.

[16] Pan Y., et al., *An Improved Generalization of Mesh-Connected Computers with Multiple Buses.* IEEE Trans. on Parallel and Distributed Systems, 2001. **12**(3): pp. 293--305.

[17] Chung K.-L., *Prefix Computations on a Generalized Mesh-Connected Computer with Multiple Buses.* IEEE Transactions on Parallel and Distributed Systems, 1995. **6**(2): pp. 196-199.

[18] *Semigroup and Prefix Computations on Improved Generalized Mesh-Connected Computers with Multiple Buses*, in *Proceedings of the 14th International Symposium on Parallel and Distributed Processing*. 2000, IEEE Computer Society.

[19] Loghi M., et al. *Towards Scalable,Energy-Efficient,Bus-Based on-Chip Networks*. in *Proceedings of the Disign,Automation and Test in Europe Conference and Exhibition*. 2004. pp. 752-757.

[20] Topcuoglu H., Hariri S., and Wu M.-Y., *Task Scheduling Algorithms for Heterogeneous Processors*, in *Proceedings of the Eighth Heterogeneous Computing Workshop*. 1999, IEEE Computer Society.

[21] Kwok Y.-K., Ahmad I., and Gu J., *Fast: A Low-Complexity Algorithm for Efficient Scheduling of Dags on Parallel Processors*, in *the 1996 International Conference on Parallel Processing*. 1996, IEEE Computer Society. pp. 150 - 157