Decentralized Dynamic Task Mapping Approaches

Maier Sebastian
Department of Electrical and Computer Engineering
Technical University of Munich
Munich, Germany
basti.maier@tum.de

***Abstract*—To satisfy QoS requirements of todays multimedia applications MPSoC's have to cope with highly dynamic workload. Task mapping significanty influences the system performance, in modern applications design-time decisions are unable to react to changed conditions. Promising decentralized dynamic system approaches have the ability to manage task mapping at runtime. Plus they can distribute the workload to multiple resoures for a parallel execution of tasks.**
This article gives an overview of current decentralized dynamic task mapping approaches. Therefore this paper summarizes the reviewed papers, analyzes their proposed dynamic mapping approaches with a focus on the general structure, compares the results and points out the advantages and disadvantages of several approaches. The results of the comparison are explained and provided in a clearly structured table. This article helps to decide which approach is useful in a specific scenario.

***Index Terms*—Decentralized, Dynamic, Task Mapping, NoC, Run-Time, MPSoC**

I. Introduction

Modern multi-processor system-on-chip (MPSoC) consist of multiple processing units. They are connected in a communication infrastructure, a Network on Chip (NoC). To satisfy the requirements of complex applications such as multimedia, telecommunication or robotic the MPSoCs have to deal with dynamic workload. Therefore a decentralized dynamic task mapping apporach is neccessary to order the application tasks in the optimal way onto the system resources. An optimized mapping heuristic will be able to improve the computational performance and the energy consumption.
Various articles handle the decentralized dynamic task mapping. They present various approaches considering the proximity of processing elements (PE), the communication volume or power dissipation.
The main goal of this paper is to give an overview of several approaches and to campare them. Therefore we consider the general structure of the approaches and afterwards/later the advantages and disadvantages are represented clearly.
In the field of decentralized dynamic task mapping we make following contributions: We give an overview over various decentralized dynamic task mapping approaches and summarize their characteristics. The general scructure is described in section II. The approaches are then analyzed regarding structure commonalites and differences (section III). Section IV points out the advantages and disadvantages of reviewed approaches. The paper is concluded in section V.

II. General description of dynamic task mapping

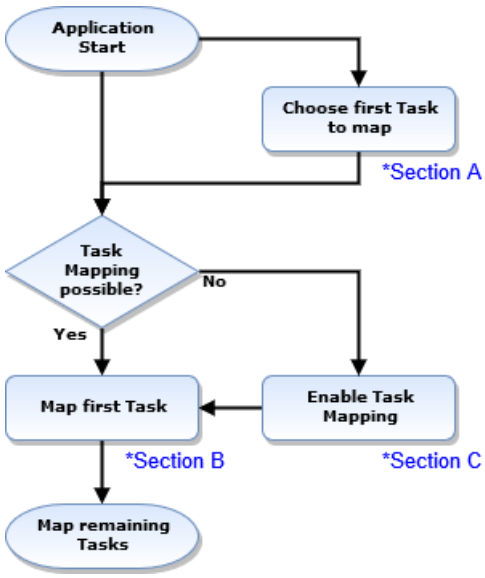

Fig. 1. General structure of dynamic task mapping. The part of choosing a first task to map is optional. The Process to enable task mapping is an additional feature of some approaches.

Task Mapping is a method to coordinate an amount of tasks in an order to meet specific requirements. Tasks are components of applications. During the mapping process, the tasks have to be allocated in an optimal way onto the system. In state-of-the-art multi-core-systems (MPSoC) the regions on which the tasks are mapped communicate with each other through the Network on Chip (NoC). The individual regions are called clusters, nodes or resources. With modern applications generating a dynamic workload, the MPSoCs have to distribute their tasks during runtime. In these scenarios the time to map every task is crucial because it influences the overall system behavior. The main goal of this paper is to optimized metrics such as network congestions, communication latency and energy consumption of MPSoCs [1]. [2].

Since heuristics which improve the communication metrics also have an impact on the power dissipation, mapping approaches are often a cost function for performance and energy. But recent studies also developed heuristics which can improve both. Every approach has its own methods to decide where

to place the first task dependig on proximity, communication volume, number of hops, etc.

The general mapping process basicly follows the same steps as show in Fig. 1:

- After the start of an application a task has to be mapped on the NoC.
- The Decision which task should be mapped at first is an optional feature wich can improve the mapping process.
- The approach searches for the first free node/ cluster/ place/ resource to map the task on and after that allocates the reaining tasks
- If the mapping is not possible the approaches enable it by task migration, reclustering or borrowing resources

If there is no available region to map the tasks on, some approaches have the ability to obtain free nodes from other PEs or restructure themselves. If the number of tasks is bigger than available clusters, they are mapped when available [3].

III. Decentralized Dynamic Task Mapping

In this section we analyze the various decentralized dynamic task mapping approaches. The subsections represent the

- A first step to achieve an efficient task mapping is to choose the right task to be mapped at first III-A. This is an optional process, which most of the heuristic do not consider.
- The subsection III-B - Task Mapping - describes the different ways to map the task on the system.
- Some approaches provide methods to enable task mapping if insufficient nodes are available. These optimizations are considered in section III-C.

A. Choose first task to map

Choosing which task should be mapped first is a method to improve the efficiency of a mapping algorithm. But most of the approaches don't consider this step to avoid increasing compexity.

The contiguous neighborhood alocation (CoNA) presents a method by selecting the task with the biggest communication volume to be mapped on the first node. This assures the largest possible number of available resources next to the task with the most intense communication. The process to select the first task is only dependent on the strucute and not influenced by the system [3].

B. Task Mapping

The challenge of mapping the first task respectively choosing the first node to map on, is one step most of the approaches have in common. The method used to select first node of the application has a significant impact on the performance of the mapping algorithm [4].

One way to allocate the tasks is the first free (FF) method. The FF searches for a free resource by scanning the network ascending column by column from. It doesn't consider any other metrics [1].

Similar to FF, the nearest neighbor (NN) also doesn't consider any metrics like channel load and commmunication volume. But NN differs because it searches for the next free node near to the source task. Therefore it spirally counts the hops between the neighbors and takes the nearest free resource. Because of it's uncomplex structure this approach has evolved to an efficient task mapping method [5] [1].

The contiguous neighborhood allocation (CoNA) does not only decide which task should be maped at first, it also provides an approach to find the first node to map the task on. The strategy is similar to NN but CoNA considers the status of neighboring nodes. This should prevent fragmentation of the tasks and decrease the congestion probability. By monitoring the possible available neighbors of each node, the algorithm chooses the first node with the largest number of free neighbors and the nearest to the managing node. Each node has between two and four neighbors. The managing node is located on the resource $n_{0,0}$ and manages the system. The remaining tasks are mapped onto a rectangular region while keeping all tasks in a close neighborhood CoNA does not inrease external congestion [3].

Since the mapping of the first node is an important step to achieve a contiguous mapping the Smart Hill Climbing (SHiC) algorithm improves the CoNA approach by a more effective search for a first resource. The optimum area for application mapping is contiguous and in a square shape of nodes. SHiC provides a method to find the optimum area. It calculates the square factor, which is the approximated number of free square-shaped nodes. The SHiC starts from a random node and searches the network in the direction of the optimum square factor until the best node is found [6].

To improve system performance heuristics, which also consider the channel load, are developed in several works. Their main goal is the reduction of channel congestion. Therefore the minimum average channel load (MACL) and minimum maximum channel load (MMCL) approaches consider all channels inside the NoC. They choose the task mapping according to the lowest average channel usage (MACL) and respectively to avoid channel usage peaks (MMCL) [1].

As searching a free cluster in the whole system causes a lot of time, a more constructive approach is to only consider the segments which are used by the task. For each path-segment, defined as a set of network channels, the cost of each mapping has to be calculated. Afterwards the Path Load Algorithm (PL) chooses the mapping with the minimum communication cost [1] [7] [5].

To find a suitable first node cluster for the first task the Agent-based Distributed Application Mapping (ADAM) approach searches for the PE with the lowes energy consumption. The run-time mapping decisions of the ADAM heuristic are also based on computation resource usage. The

selection of a tile/node to map a task on is contiguent on a particular cost function. However the ADAM approach has additional features which are considered in section III-C [2].
Related to ADAM, the Decentralized Agent Based Re-clustering for Task Mapping (DARTM) and Distributed Resource Management with Dynamic Cluster Sizes (DRMDC) also regard the MPSoC as a set of clusters and manage the system by the use of agents or masters. Because of the constant resource monitoring, DARTM and DRMDC are always aware of the number of free tiles in each resource. Thus the heuristics select the cluster with the highest number of available resources. The only difference is that in DARTM the actual mapping is performed in awareness of energy consumption. This is achieved by a power aware cost function. And in DRMDC the algorithm additionally considers the proximity of resources to each other [8] [9].
Another agent based algorithm is provided by Distributed Resource Management (DistRM). The mapping flow is managed by agents, as in other distributed methods (e.g. ADAM). If an application starts in DistRM, this approach chooses a random core to map the agent on, which searches for a suitable set of resourses by itself. The agent may change it's startpoint if it finds a better cluster. During the process, the agents send requests to other agents to obtain resources. Requested agents offer cores by considering the loss and gain of each agent in a greedy algorithm [10].

A next step to improve dynamic task mapping is to combine communication cost functions and progressive search strategies. The Best Neighbor (BN) approach has evolved to be an effective mapping strategy by combining the NN and PL heuristic. During spiral searches BN selects the nearest neighbor with the best cost function regarding routing and communication. Additionaly the BN approach can be executed in a parallel tree structure and thereby map more efficiently [7] [1] [5].
Another combination of approaches is presented by lower energy consumption based on dependencies-neighborhood (LEC-DN). If the algorithm has only one commmunicating task, it searches for the nearest neighbor in a spiral way by just considering the number of hops between the tasks. But if the algorithm detects more than one communicating task, it takes into account the communication volume and maps the new task closer to the one with the higher volume [5].

C. Improve Mapping Potential

Some algorithms present approaches to enable dynamic task mapping, even if no suitable cluster is found. The ADAM heuristic uses task migration to make sure every task gets mapped in time. Therefore it freezes all connected tiles and performs the migration. A further way to improve QoS is re-clustering used by ADAM and DARTM. Their algorithm borrows processing elements from neighboring clusters to execute the task. Afterwards the re-clustered tiles return to their owner.
To enable borrowing processing elements from other clusters, task migration, as used in ADAM, DARTM or DRMDC, can be performed. Task migration is performed to create space next to the required nodes. Therefore the tasks in affected clusters are interrupted, transmitted to a new PE and restarted. As a result the task mapping is possible because the algorithm now has enough available nodes [2] [8] [9].
The DistRM approach accomplishes optimization without monitoring the global communication traffic. DistRM uses the features of it's decentralized agent approach to continuously optimize task mapping. Because the agents do not stop requesting resources they are able to recieve additional cores if the gain in speedup is significantly higher than the loss of the giving application [10].

IV. Evaluation

To give an overview over the presented methods we discuss advantages and disadvantages of the analyzed approaches.

The table I below gives a short overview of the best approaches. The FF, MMCL and MACL heuristics are not considered because they are not competitive to the other methods.

As shown in I, the NN presents a simple task mapping scheme, which leads to fast execution time and less channel load. On the other hand, the absence of cost evaluation could lead to impractical system occupancy. Because this approach accomplishes good mapping results, it is often used to measure other heuristics [1].

The PL approach is also often used to evaluate the performance mapping algorithms. By considering the communication cost and choosing the minimum cost for the mapping, the method reduces the channel load and packet latency as intended. In [1] this approach gets the best results but [7] criticizes that PL does not consider the topology of the application-graph and has no multitasking.

By restricting the search of the minimum communication cost to the nearest nodes, the BN combines the PL and NN approaches. Thereby it achieves similar congestion reduction and packet latency as PL is able to reduce the execution time [1].

An even faster execution time and smaller number of hops as BN is provided by the LEC-DN approach. This is possible by considering the communication volume if multiple tasks have to be mapped, otherwise it just applies the NN method. Although the heuristic is able to reduce energy consumption against other dynamic approaches, static solutions provide even less [5].

The structure of the ADAM approach and its possibility to enable task mapping if avaiable resources are rare, reduce the computational effort and has less monitoring congestion as centralized approaches. On the downside it has additional monitoring traffic compared to uncomplex approaches [2].

To gain better results as ADAM the DARTM uses a similar methods like agents and reclustering. This as well leads to

TABLE I
EVALUATION OF DECENTRALIZED DYNAMIC TASK MAPPING APPROACHES. APPROACHES MAY OFFER AN ADITIONAL FEATURE TO OPTIMIZE MAPPING POTENTIAL. SOME PAPERS FOCUS ON COMMUNICATION PERFORMANCE AND OTHER ON ENERGY CONSUMPTION.

Approach	Advantages	Disadvantages	Optimization Method	Communication Aware	Energy Aware	Reference
NN	Low complexity, reduced channel load, fast execution time	no cost evaluation		✓		[1]
CoNA	reduced congestion	more hops than NN		✓		[3]
SHiC	better network latency & power dissipation	only an add-on no independent approach		✓		[6]
PL	reduce channel load, low packet latency	very complex, no multitasking error-prone		✓		[1]
ADAM	lower computational effort	additional monitoring traffic	✓	✓		[2]
DARTM	less monitoring traffic (than ADAM) reduced power density	slower re-clustering (than ADAM)	✓	✓	✓	[8]
DRMDC	less execution time & smaller hop number	monitoring overhead is not considered	✓	✓		[9]
BN	reduce channel load, low packet latency	monitoring costs		✓		[1], [7]
LEC-DN	fast execution time, small number of hops low energy consumption	higher energy than static mapping (slower than static mapping)			✓	[5]
DistRM	parallelism and avoid computational bottlenecks	huge negotiation traffic fragmentation	✓	✓		[10]

improvements regarding less monitoring traffic than ADAM and a reduced power density. But the re-clustering algorithm of DARTM is slower than the one used for in ADAM [8].

Another decentralized approach offers also a good total execution time and a smaller number of hops. DRMDC uses distributed masters - agents in ADAM and DARTM - to manage and monitor it's task mapping procedure [9].

DistRM presents agents which are negotiating resources during the whole mapping process. This can improve execution time because tasks with high computational effort achiev more processing elements. Thereby it avoids computational bottlenecks and proviedes a parallel architecture [10]. On the downside the approach creats huge negotiation traffic and lead to fragmentation on the chip [6].

One big advantage of ADAM, DARTM, DRMDC and DistRM is their possibility to enable task mapping if resources are limited throug methods like task migration and reclustering.

V. CONCLUSION

In this paper we gave an overview of current decentralized dynamic task mapping approaches. The work analyzes the structure of several approaches and points their out advantages and disadvantages. All presented heuristics were able to reduce communication traffic or energy consumption. Some provide a method to enable task mapping even if the available resources are rare. The decentralized approaches with that kind of feature look very promising.

REFERENCES

[1] E. L. d. S. Carvalho, N. L. V. Calazans, and F. G. Moraes, "Dynamic task mapping for mpsocs," *IEEE Design Test of Computers*, vol. 27, no. 5, pp. 26–35, Sep. 2010.

[2] Mohammad Abdullah Al Faruque, R. Krist, and J. Henkel, "Adam: Run-time agent-based distributed application mapping for on-chip communication," in *2008 45th ACM/IEEE Design Automation Conference*, June 2008, pp. 760–765.

[3] M. Fattah, M. Ramirez, M. Daneshtalab, P. Liljeberg, and J. Plosila, "Cona: Dynamic application mapping for congestion reduction in many-core systems," in *2012 IEEE 30th International Conference on Computer Design (ICCD)*, Sep. 2012, pp. 364–370.

[4] E. Carvalho, N. Calazans, and F. Moraes, "Heuristics for dynamic task mapping in noc-based heterogeneous mpsocs," in *18th IEEE/IFIP International Workshop on Rapid System Prototyping (RSP '07)*, May 2007, pp. 34–40.

[5] M. Mandelli, L. Ost, E. Carara, G. Guindani, T. Gouvea, G. Medeiros, and F. G. Moraes, "Energy-aware dynamic task mapping for noc-based mpsocs," in *2011 IEEE International Symposium of Circuits and Systems (ISCAS)*, May 2011, pp. 1676–1679.

[6] M. Fattah, M. Daneshtalab, P. Liljeberg, and J. Plosila, "Smart hill climbing for agile dynamic mapping in many-core systems," in *2013 50th ACM/EDAC/IEEE Design Automation Conference (DAC)*, May 2013, pp. 1–6.

[7] A. Weichslgartner, S. Wildermann, and J. Teich, "Dynamic decentralized mapping of tree-structured applications on noc architectures," in *Proceedings of the Fifth ACM/IEEE International Symposium*, May 2011, pp. 201–208.

[8] Y. Cui, W. Zhang, and H. Yu, "Decentralized agent based re-clustering for task mapping of tera-scale network-on-chip system," in *2012 IEEE International Symposium on Circuits and Systems (ISCAS)*, May 2012, pp. 2437–2440.

[9] G. Castilhos, M. Mandelli, G. Madalozzo, and F. Moraes, "Distributed resource management in noc-based mpsocs with dynamic cluster sizes," in *2013 IEEE Computer Society Annual Symposium on VLSI (ISVLSI)*, Aug 2013, pp. 153–158.

[10] S. Kobbe, L. Bauer, D. Lohmann, W. Schrder-Preikschat, and J. Henkel, "Distrm: Distributed resource management for on-chip many-core systems," in *2011 Proceedings of the Ninth IEEE/ACM/IFIP International Conference on Hardware/Software Codesign and System Synthesis (CODES+ISSS)*, Oct 2011, pp. 119–128.