Congestion-Aware Task Mapping in NoC-based MPSoCs with Dynamic Workload

Ewerson Carvalho Ney Calazans Fernando Moraes
PUCRS - FACIN - Av. Ipiranga 6681- Porto Alegre - 90619-900 - Brasil
{ecarvalho, calazans, moraes}@inf.pucrs.br

Applications running in heterogeneous MPSoCs, as multimedia and networking, normally contain a dynamic workload of tasks. This implies a varying number of tasks simultaneously running, with their number possibly exceeding the available resources. This may require the execution of task mapping at run time, to meet real-time constraints.

Most works in the literature propose the use of *static mapping* [1][2], where the best placement of tasks is defined at design time. Consequently, such methods are not appropriate for dynamical workloads. *Task migration* [3][4] has also been used in heterogeneous MPSoCs to optimize the performance at run-time. It consists either in relocating tasks when a performance bottleneck is detected, or to distribute the workload more homogeneously among the MPSoC processors. Differently from task migration, *dynamic mapping* can insert new tasks into the system at run time.

This work investigates the performance of different mapping algorithms in NoC-based MPSoCs with dynamic workload. The main cost function in mapping algorithms is to optimize the occupation of the NoC links. It is possible to achieve performance gains if the mapping algorithm is able to minimize NoC congestion.

Without loss of generality, heterogeneous MPSoC architectures may be represented, as a set of processing nodes that interact via a communication network. Processing nodes may support either hardware or software task execution. Hardware tasks execute in reconfigurable logic (*reconfigurable areas*) or dedicated IPs. If reconfigurable logic is used, the hardware presents flexibility similar to software. It becomes possible to load the hardware tasks on-the-fly using dynamic reconfiguration. Software tasks execute in *instruction set processor, ISPs*.

The number of tasks may exceed the MPSoC resources. One processor (the Manager Processor, *MP*) may be reserved to manage the system resources When the MPSoC starts its execution, only the initially needed tasks are allocated into the system.

New tasks are allocated when a given task tries to communicate with a task not yet present. The *MP* is responsible for resource control, task binding, task mapping, task relocation/migration and to control the reconfiguration process. Once *tasks* start their execution, communication requests are first transmitted to the *MP*. If the destination is not present, it is necessary that the *MP* execute a dynamic mapping heuristic. Five such heuristics are evaluated here to map tasks at run-time. All discussion of heuristics assume a 2D mesh NoC infrastructure using XY routing.

The ***first free*** (FF) method is the reference mapping, used for comparison purposes only. This approach selects the first free node able to execute the requested task (task binding), searching from the NoC address (0,0) on. Clearly, there is no congestion cost evaluation.

Nearest Neighbor (NN) mapping is similar to the FF strategy, also with no congestion cost evaluation. The NN mapping starts searching a free node able to execute the requested task (task binding) around the node address which makes the request. The search procedure tests all *n-hops* neighbors of the current node *n* varying between 1 to the maximum possible.

Minimum Maximum Channel Load (MMC) congestion-aware mapping heuristic tries to reduce the maximum occupation of the NoC links. The goal of this heuristic is to avoid congestions in the NoC, and consequently improve the overall performance. MMC computes the cost of each mapping *k* according to Equation 1.

$$cost_k = \max(rate_{l(i,j)}) \mid \ 0 \le i < lx;\ 0 \le j < ly \qquad (1)$$

The $rate_{l(i,j)}$ denotes the total rate of each NoC link, and *lx* and *ly* are the NoC dimensions. The selected mapping is the one that has the minimum cost.

Minimum Average Channel Load (MAC) congestion-aware heuristic aims at reducing the average occupation of the NoC links. This heuristic is similar to the MMC, replacing the **max** function by the *avg* (average) function. While the MMC heuristic minimizes the peak link usage, the MAC heuristic tries to homogenously distributes the communication load into the NoC. Equation 2 presents the MAC cost function. Links not used for communication ($rate_{l(i,j)}$=0) are not considered in the heuristic. The selected mapping is the one that has the minimum cost.

$$cost_k = avg(rate_{l(i,j)}) \ \forall\, rate_{l(i,j)} > 0 \mid \ 0 \le i < lx;\ 0 \le j < ly \qquad (2)$$

MMC and MAC consider all links of the NoC while mapping a new slave task. If a given task does not increase the total cost, any mapping may be accepted. To overcome this problem, the ***Path Load*** (*PL*) congestion-aware heuristic considers only the links that will be used by the task being mapped. *PL* computes the cost of each mapping *k* according to Equation 3.

$$cost_k = \sum rate_{c(i,j)} + \sum rate_{c(j,i)} \qquad (3)$$

Where $rate_{c(i,j)}$ and $rate_{c(j,i)}$ are the rates in the individual channels from the master to the new slave and the rates of the channels in the opposite direction. This is due to the asymmetric nature of the XY routing algorithm.

Figure 1 graphs present the execution time and link occupation (avg, max and min values) for the different mapping heuristics. Examples use an 8*x*8 Mesh NoC, to support large systems, with dozens of processors, expected to become a reality in a near future. In addition, small NoCs would make

IEEE Computer Society Annual Symposium on VLSI(ISVLSI'07)
0-7695-2896-1/07 $20.00 © 2007 IEEE

evaluation harder. Graphs are plotted for 13 applications.

The average NoC occupation is higher (± 80%) for the FF mapping, since no congestion-aware heuristic is used. The heuristics MAC and MMC have similar average NoC occupation. The NN mapping, a very simple heuristic, and the PL mapping present the lower NoC occupation. Note that the execution time of the applications running in the system, for NN and PL mappings, is smaller, corroborating the fact that reducing congestion the execution time is also reduced.

Even if two mapping heuristics present similar average occupation, it is important to analyze the maximum NoC occupation. The worst behavior is the FF mapping (peak occupation is 280%). The simple NN mapping leads to a small value of average occupation (peak occupation is 200%).

The MAC mapping is not effective to avoid congestion. The reason was advanced before: when a new mapping does not reduce the average link load, the algorithm selects the first mapping option available. The MMC has the same deficiency of the MAC, but with smaller values of maximum NoC occupation. The PL mapping presents the smallest values for the maximum NoC occupation. This arrives because this heuristic minimizes, for each mapping, the load added by the new task, in an opposite way to MAC and MMC which try to globally minimize system congestion.

Compared to FF, PL reduces 19.3 % the total execution time in average. Note that the simple NN mapping also reduces the execution time (18.7%). The advantage of the PL mapping is congestion reduction.

The clock-cycle simulation of a small MPSoC implementation (NoC with Plasma processors) generated the values for the NoC model, configuration delays, and execution times. A larger system, with dozens of processors, should be simulated in order to tune the models, allowing deeper analysis of the link congestion, the communication delay and the power consumption.

As mentioned before, multi-task processors increase the number of the software tasks available in the system. Future works includes extending the mapping heuristics to multi-task processors. Since these have a microkernel to manage tasks execution, processors may send their load, as well the links loads, to the MP. In this way, the mapping heuristic could take decisions based in the actual system load, instead of the load furnished by applications and link load estimations. Finally, if the manager processor knows the actual load of processors and links, task migration algorithms can be implemented to ensure QoS to applications.

REFERENCES

[1] Hu, J.; Marculescu, R. *Energy- and Performance-Aware Mapping for Regular NoC Architectures*. IEEE Transaction on Computer-Aided Design of Integrated Circuits and Systems, v.24(4), 2005, pp. 551-562.

[2] Marcon, C.; Borin, A.; Susin, A.; Carro, L.; Wagner, F. *Time and Energy Efficient Mapping of Embedded Applications onto NoCs*. In: ASP-DAC, 2005. pp. 33-38.

[3] Nollet, V.; Marescaux, T.; Avasare, P.; Mignolet, J-Y. *Centralized Run-Time Resource Management in a Network-on-Chip Containing Reconfigurable Hardware Tiles*. In:DATE, 2005, pp. 234-239.

[4] Bertozzi, S.; Acquaviva, A.; Bertozzi, D.; Poggiali, A. *Supporting task migration in multi-processor systems-on-chip: a feasibility study*. In:DATE, 2006, pp. 1-6.

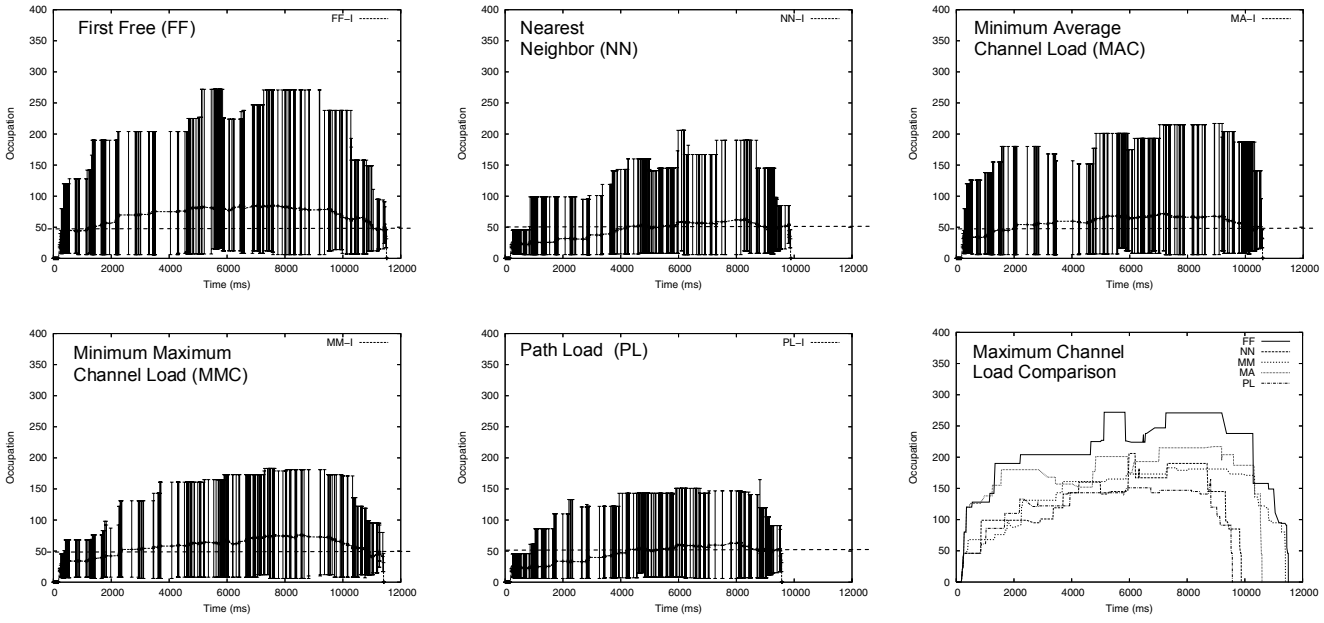

Figure 1 - Execution time versus link load (occupation) for the dynamic mapping strategies (initial mapping: clustering), for 13 applications. The continuous line in the graphs denotes the average link load.