Shared Memory Aware MPSoC Software Deployment*

Timo Schönwald†, Alexander Viehl†
†FZI Forschungszentrum Informatik
Haid-und-Neu-Str. 10-14
76131 Karlsruhe, Germany
[schoenwald,akoch,viehl]@fzi.de

Oliver Bringmann†‡, Wolfgang Rosenstiel†‡
‡Universität Tübingen
Sand 13
72076 Tübingen, Germany
[bringmann,rosenstiel]@informatik.uni-tuebingen.de

***Abstract*—In this paper we present a novel approach for mapping interconnected software components onto cores of homogenous MPSoC architectures. The analytic mapping process considers shared memory communication as well as the routing algorithm controlling packet-based communication. The software components are mapped with the constraints of avoiding communication conflicts as well as access conflicts to shared memory resources. The core of the elaborated approach consists of an algorithm for software mapping which is inspired by force-directed scheduling from high-level synthesis. Experimental results show that the presented approach increases the overall system performance by 22% while reducing the average communication latency by 35%. For presenting the major advantages of the developed solution, we optimized an advanced driver assistance system on the Tilera TILE*Pro*64 processor.**

I. Introduction

As the computational performance of single microprocessor cores cannot be further increased by higher frequencies due to deep submicron and thermal problems, the design paradigm shifted towards multicore architectures for general purpose and embedded processors. Recent SoC platforms containing more than one core, like the NVIDIA Tegra 3 or the TI OMAP 5 are available and integrated into mainstream products on the market. As the number of cores further increases their interconnection becomes a bottleneck.

The scalability of the communication infrastructure between the cores of a processor declines with their number. Bus-based communication becomes the bottleneck as the number of cores connected to the bus increases. Packet-based communication infrastructures proposed as network-on-chip (NoC) [3] represent a promising approach to handle the growing scalability problems of bus-based communication.

In contrast to bus-based architectures, packet-based communication architectures have the advantage that packets of temporally overlapping communication instances can take different paths through the network for data transmission and thereby access conflicts mainly impacting the communication performance can be reduced or avoided. The routing algorithm e.g. [8] determines the path the data are transmitted between cores. It has a significant impact on the communication and the performance of the entire system.

Besides the communication infrastructure especially shared memory access becomes a bottleneck as the number of cores increases. Further, the local cache size of the cores decline with increasing core numbers. Therefore a lot of data has to be held in off-chip memory and software components have to communicate using shared memories. Because of these implications, the shared memory becomes a significant factor for the performance of multicore and manycore architectures. An explicit consideration of memory properties is getting even more into the task of architecture optimization.

Using novel NoC-based MPSoC architectures, such as the Tilera TILE*Pro*64 [22] and Tile-Gx8000 [21] with separate on-chip-networks for functional and memory communication raises new design challenges for embedded system architects. One of these challenges is the integration of the new platforms into the design flow. In present design flows the mapping of software processes of parallelized algorithms onto the cores of the NoC-based MPSoC architectures and also the architecture and hierarchies of memories are not taken into account sufficiently. Hence new approaches for the mapping of processes onto the cores of NoC-based MPSoC architectures, considering shared memory as well as the communication infrastructure, are needed.

In this paper we present extensions to the so-called force-directed mapping (FDM) [18] [17]. FDM determines the mapping of software processes onto the cores of NoC-based MPSoC architectures with an algorithm inspired by force-directed scheduling (FDS) from high-level synthesis. FDS considers the effects of a scheduling and binding decision for subsequent operations to be scheduled and bound on functional units. Scheduling and binding of operations to functional units have some similarities to the mapping of software processes onto the cores of MPSoC architectures. For this reason a FDS inspired algorithm is well-suited for mapping software processes onto the cores of MPSoC architectures.

The starting point of the design flow is a homogeneous MPSoC architecture with a regular communication structure. This implies that every process of the software application can be computed on every core of the MPSoC architecture. For simplification we assume that respectively one process is mapped on one core of the MPSoC architecture.

The extension we present in this paper is called shared memory aware force-directed mapping (smFDM). Shared memory access is a crucial factor that limits the entire system performance which has not yet been addressed sufficiently by related approaches so far. Therefore it is of particular importance to consider the memory architecture and shared memory during the mapping of software process onto the cores of MPSoCs. The approach presented in this paper is the first one, that takes shared memory in conjunction with the routing algorithm of the communication infrastructure into account. We show, how the extension can be integrated into the FDM algorithm and that the extension has a quite positive impact on the communication latency as well as on the overall system performance. For process mapping, smFDM takes

- the degree of freedom on which core a process is mapped,
- functional communication,
  - the route for functional communication,
  - the communication distance between the cores the software processes are mapped on,
- shared memory communication,
  - the route for shared memory communication,
  - the communication distance between cores an memories according to the architecture,
- the communication network architecture,
  - separate communication networks for functional communication and memory communication

into account. smFDM places the processes in such a way that communication conflicts and memory access conflicts are reduced or even avoided. Thereby the smFDM:

- reduces the communication latency,
- increases the throughput

The paper is structured as follows. In Section II we review existing approaches for the mapping of tasks onto NoC

*This work was partially supported by the BMBF project RESCAR 2.0 under grant 01M3195E and the BMBF project EFA2014/2 under grant 13N11945.

978-3-9815370-0-0/DATE13/©2013 EDAA

architectures. Section III presents our approach for mapping processes onto NoC-based MPSoC architectures. A short introduction to the Tilera TILE*Pro*64 architecture which we used for obtaining the results can be found in Section IV-A. In Section IV-B we present the application we used for obtaining the results presented in Section IV-C.

II. Related Work

Approaches used for mapping tasks onto cores of NoCs can be used or adapted for mapping processes onto the cores of MPSoC architectures. The usage of these approaches is possible in the case the MPSoCs uses packet-based communication like NoC architectures. In [4] an approach for the synthesis of NoCs is presented, which uses the SUNMAP tool [16] for mapping. The SUNMAP tool maps a core graph with annotated edge weights onto the NoC architecture. In a first step, an initial mapping is found using a greedy algorithm by placing the tasks with the highest communication demand. In a second step, a pair wise swapping of all placed tasks is performed and the costs for the mappings are calculated. In [15] different optimization functions for the calculation of the mapping costs in the SUNMAP tool are presented.

The approach presented in [10] maps tasks on mesh-based NoC architectures. For this, the approach uses an application characterization graph to map the tasks onto an architecture characterization graph. The goal of that approach is the reduction of the communication energy consumption while preserving the performance of the NoC. Another approach, that performs a power-aware mapping concerning bandwidth and latency constraints is presented in [24]. In [11] an ILP approach is used for the reduction of the energy consumption. An approach for multi-objective optimization concerning energy an temperature is presented in [13].

In [14] an approach is presented that uses a branch and bound algorithm for mapping and physical planning of NoC architectures. For the mapping a directed graph of communicating tasks and a NoC topology graph is used. The mapping is done based on bandwidth constraints of the NoC topology graph. Another approach that performs mapping and physical planning is presented in [5]. That approach uses bandwidth information for the mapping of the tasks on the cores of the NoC architectures.

Only a few approaches consider shared memory communication during the mapping determination. One of these mapping algorithms is presented in [12]. This approach optimizes the order of read and write accesses to the shared memory. The limitations of this approach are that only one shared memory bank in the NoC architecture is allowed, as well as the routing algorithm for the communication is not considered. Another approach taking shared memory into account is presented in [7]. This approach is limited due to the fact, that it only considers the distance between communication partners for improving the data locality. That approach do not consider the routing algorithm of the used architecture.

The drawback of all previously mentioned approaches is that they do not take the routing algorithm into account sufficiently. Either they do not take the routing algorithm into account or they only consider deterministic routing algorithms. The approaches, which take the routing algorithm into account, do not consider that packets routed through the network can conflict with each other by using the same link at the same time. Our approach is feasible for different kinds of routing algorithms as well as for different kinds of topologies. Our approach is more precise, it considers shared memory communication also as the routing algorithm and it is feasible for different kinds of routing algorithms.

In [9] a force-directed approach for optimizing the dynamic power is presented. Another force-directed approach for power optimization is presented in [2]. In [19] a force-directed approach for architecture optimization, using multi local port routers, is presented. This approach tries to optimize the communication architecture by connecting more than one processing element to one router. The optimization goals of these approaches differ from the goal of the approach presented in this paper. The presented approach reduces communication conflicts and thereby the communication latency.

III. Mapping Algorithm

In this section we present the developed extension to the FDM [18] [17] algorithm called shared memory aware force-directed mapping (smFDM). The starting point of the developed approach is a homogeneous NoC-based MPSoC architecture with a specification of the memory architecture. This means that every process can be computed on every processor core resulting in the same execution time characteristics. As the processes of the envisaged application domain are typically strongly connected and have huge amounts of inter-process data communication, the considered architecture class is communication centric. The pipeline of the concurrent functional software processes is dominated by the computation characteristics of the slowest process in the pipeline. For simplification of the mapping problem we abstract from the computation characteristic of the processes. All communication instances between the concurrent software processes are pessimistically considered as potentially simultaneous and hence as overlapping which might lead to access conflicts. However the general approach can be simply extended to consider the computation characteristics using approaches like [20] and [1].

Major adaptions and extensions were made to the original FDS algorithm for aligning the basic principles with the setting of mapping software processes to memories and cores. The developed mapping algorithm uses a *distribution graph* as a representation of conflicting communication on links of the MPSoC topology. *Self forces* are calculated from the values of the distribution graph regarding that links are connected to a route and that a data communication can last some timeslots. The degree of freedom for mapping a process onto a core is huge. However, mapping a process to a particular core can effect the mapping of other processes and thereby the communication distance. Therefore our approach calculates *successor forces* and *predecessor forces* for processes effected by the current mapping. The *total force* is calculated out of the self forces for all data communication instances assigned to a valid route by mapping a process to a core and the successor and predecessor forces. The mapping with the smallest total force is chosen and the distribution graph is updated.

A. Basic Definitions

smFDM uses an extension of the *application characterization graph* (APCG) [10] consisting of communicating processes which are mapped onto a MPSoC architecture. The *extended application characterization graph* (EAPCG) is defined as follows:

***Definition** 1:* The $EAPCG\langle P, D\rangle$ is a non-cyclic directed graph, where:

- P is the set of processes p_a in the EAPCG
- D is the set of communications $d_{a,b}$ between the processes $p_a, p_b \in P$
- $time\,(p_a) : P \rightarrow \mathbb{R}^3_+$ denotes the best-, average, and worst-case execution time of a process $p_a \in P$
- $data\,(d_{a,b}) : D \rightarrow \mathbb{R}_+$ denotes the amount of data a communication $d_{a,b} \in D$ transmits ■

For describing the topology of the MPSoC architecture we use an extension of the *architecture characterization graph* (ARCG) [10]. The *extended architecture characterization graph* (EARCG) is defined as follows:

***Definition** 2:* The $EARCG\langle C, M, L_c, L_m, N\rangle$ is a directed graph, where:

- C is the set of cores c_i of the architecture
- M is the set of memories m_j of the architecture
- $L_c \subseteq C \times C$ is the set of links $l_{i,j}$ between cores of the architecture
- $L_m \subseteq C \times M$ is the set of links $l_{i,j}$ between cores and memories

- $L = L_c \cup L_m$ is the set of all links $l_{i,j} \in L_c \cup L_m$ in the architecture
- N is the set of communication networks n_k of the architecture
- $band\,(l_{i,j}) : L \rightarrow \mathbb{R}_+$ denotes the bandwidth of a link $l_{i,j} \in L$
- $net\,(l_{i,j}) : L \rightarrow N$ denotes the association of a link $l_{i,j} \in L$ to a communication network $n_k \in N$
- $cache\,(c_i) : C \rightarrow \mathbb{N}_+$ denotes the cache size of a core $c_i \in C$
- $mem\,(c_i) : C \rightarrow M$ denotes the association of core $c_i \in C$ to a memory $m_j \in M$ ■

***Definition** 3:* The function map, that maps the processes $p_i \in P$ on the cores $c_j \in C$ is defined as follows:

$$map\,(p_i) : P \rightarrow C, |P| \leq |C| \quad \blacksquare$$

***Definition** 4:* The set $P_{mappable}$ contains all processes that are mappable in the current iteration of the algorithm.

$$P_{mappable} = \{p_m \mid p_m \in P \setminus P_{mapped}, \exists d_{a,m} \in D : p_a \in P_{mapped}\}$$

Where P_{mapped} is the set of already mapped processes. ■

The data communication between two processes mapped onto different cores of the MPSoC architecture takes place by packet-based communication. The path between two cores the data is transmitted over is called a *route*. A route $r_{i,j}$ is a non-cyclic end-to-end connection from core c_i to core c_j consisting of links $l_{a,b} \in L$.

The route over which the data of a communication is transmitted through the network is determined by the routing algorithm. Routing algorithms prohibit certain routes for e.g. deadlock avoidance like presented in [8]. Only a subset of all possible routes between two cores are valid.

***Definition** 5:* The set $R_{i,j} = \left\{r_{i,j}^1, ..., r_{i,j}^n\right\}$ contains all routes between the cores c_i and c_j. The subset $V_r\,(R_{i,j}) \subseteq R_{i,j}$ contains all valid routes $r_{i,j}$ between the cores c_i and c_j. In this formulation r is the routing algorithm applied to the MPSoC architecture. ■

The presented approach is also able to handle adaptive routing algorithms, which allow multiple valid routes between two cores. This is taken into account during the computation of the total force.

For complexity reduction coarse grained timeslots instead of clock cycles are used.

***Definition** 6:* A timeslot t is defined as the time interval a link $l_{a,b} \in r_{i,j}$ is used for transmitting data. ■

***Definition** 7:* The probability $Prob$ that a link $l_{a,b}$ is used in timeslot t for a data communication $d_{i,j}$ between the cores c_i and c_j is defined as follows:

$$Prob\,(l_{a,b}, t) = \begin{cases} 0 & \nexists r_{i,j} \in V_r\,(R_{i,j}) : l_{a,b} \in r_{i,j} \\ \frac{1}{|V_r(R_{i,j})| * |C_{free}|} & \exists r_{i,j} \in V_r\,(R_{i,j}) : l_{a,b} \in r_{i,j} \end{cases}$$

Where C_{free} is the set of free cores. ■

The distribution graph shows the concurrency of data communication instances using a particular link in a timeslot. Therefore the distribution graph is defined as follows:

***Definition** 8:* The summation of the probabilities for each link $l_{a,b} \in L$ for each timeslot t is denoted as the distribution graph $DG_{l_{a,b}}$ given by:

$$DG_{l_{a,b}}\,(t) = \sum_{d \in D} Prob\,(l_{a,b}, t) \quad \blacksquare$$

***Definition** 9:* The force f exerted onto a link $l_{a,b} \in L$ in a timeslot t is defined as follows:

$$f\,(l_{a,b}, t) = DG_{l_{a,b}}\,(t) * x\,(l_{a,b}, t)$$

Where $x\,(l_{a,b}, t)$ is the change of the probability that the link $l_{a,b} \in L$ is used in the timeslot t.

$$x\,(l_{a,b}, t) = \begin{cases} 0 - Prob\,(l_{a,b}, t) & l_{a,b} \text{ not used in } t \\ 1 - Prob\,(l_{a,b}, t) & l_{a,b} \text{ used in } t \end{cases} \quad \blacksquare$$

A negative value of force f results, if a link $l_{a,b} \in L$ is not chosen for communication in timeslot t as the probability $x\,(l_{a,b}, t)$ is negative in that case.

In the presented approach, the self force is associated with the mapping of a process onto a core and thereby with the assignment of a valid route to a data communication. Therefore the self force is a measure of conflicting communication instances on a route.

***Definition** 10:* The self force sf is associated with each route $r_{i,j} \in V_r\,(R_{i,j})$. The self force is calculated as follows:

$$sf\,(r_{i,j}) = \sum_{\forall l_{a,b} \in r_{i,j}} f\,(l_{a,b}, t) \quad (1) \quad \blacksquare$$

If the mapping of a process onto a core of the NoC-based MPSoC architecture and thereby the assignment of a valid route to a data communication has effects on the mapping costs of other processes and data communications successor forces and predecessor forces have to be calculated.

***Definition** 11:* The predecessor $predf$ force is calculated like the self force (Eq. 1). Predecessor forces are calculated for processes $p_p \in P_{pred}$ which can be mapped in the current iteration of the algorithm and are not equal to the currently mapped process p_m.

$$P_{pred} = \{p_p \mid p_p \in P_{mappable} \setminus \{p_m\}\} \quad (2) \quad \blacksquare$$

***Definition** 12:* The successor force $succf$ is calculated like the self force (Eq. 1). Successor forces are calculated for processes $p_s \in P_{succ}$ that have not been mapped before. The processes in P_{succ} have a data communication $d_{m,s} \in D$ with the currently mapped process p_m.

$$P_{succ} = \{p_s \mid p_s \in P \setminus \{P_{mapped} \cup P_{mappable}\} : \exists d_{a,s} \in D : p_a \in P_{mapped} \cup \{p_m\}\} \quad (3) \quad \blacksquare$$

The total force tf is the summation of all self forces sf, all successor forces $succf$ and all predecessor forces $predf$, where p_m is the process currently mapped on a core c of the MPSoC architecture, $p_a \in P_{mapped}$ is a process already mapped on a core, p_s is a process a successor force is exerted on (Eq. 3) and p_p is a process a predecessor force is exerted on (Eq. 2).

***Definition** 13:* The total force tf is defined as follows:

$$tf\,(p_m, c) = \sum_{\forall d_{i,j} \in D_a} \sum_{\forall r_{i,j} \in V_r(R_{i,j})} sf\,(r_{i,j}) + \sum_{\forall d_{i,j} \in D_s} \sum_{\forall r_{i,j} \in V_r(R_{i,j})} succf\,(r_{i,j}) + \sum_{\forall d_{i,j} \in D_p} \sum_{\forall r_{i,j} \in V_r(R_{i,j})} predf\,(r_{i,j})$$

Where the sets D_a, D_s and D_p are defined as follows.

$$D_a = \{d \mid d_{a,m}, d_{m,a} \in D : p_a \in P_{mapped}\}$$
$$D_s = \{d \mid d_{s,m}, d_{m,s} \in D : p_s \in P_{succ}\}$$
$$D_p = \{d \mid d_{p,m}, d_{m,p} \in D : p_p \in P_{pred}, p_a \in P_{mapped}\} \quad \blacksquare$$

B. Algorithm

The smFDM algorithm computes the mapping $map : P \rightarrow C$ of the processes of the EAPCG onto the cores of the EARCG. Therefore the mappable processes are iteratively mapped onto the free cores of the architecture. smFDM tries to reduce the concurrency of communications on all links of the architecture as well as the concurrency for the access to the off-chip memory. The mapping of a process to a core that results in the lowest communication concurrency on the links and the lowest access concurrency to the off-chip memory is chosen as mapping for that process.

In Fig. 1 the pseudo code for the smFDM algorithm can be found. In the initial step (lines 2 to 4 in Fig. 1) a heuristics for the mapping of the first process is used. Afterwards the

Algorithm 1 smFDM

```
1: begin
2:    map first process p_first on core c_first
3:    C_free = C \ {c_first}
4:    P_mapped = {p_first}
5:    while P \ P_mapped ≠ ∅ do
6:       compute P_mappable
7:       for all p_m ∈ P_mappable do
8:          compute C_dev
9:          for all c_m ∈ C_dev do
10:            map p_m on c_m
11:            compute all valid routes
12:            compute distribution graph
13:         end for
14:      end for
15:      compute all total forces
16:      choose mapping p_map onto c_map with smallest total force
17:      update distribution graph
18:      C_free = C_free \ {c_map}
19:      P_mapped = P_mapped ∪ {p_map}
20:   end while
21: end begin
```

Figure 1: Algorithm for smFDM

used core c_{first} is removed from the set of free cores C_{free} as well as the mapped process p_{first} is inserted into the set of already mapped processes P_{mapped}.

In the next steps (lines 5 to 20 in Fig. 1) the remaining processes are iteratively mapped onto free cores of the MPSoC architecture until all processes are mapped. First the set $P_{mappable}$ of all mappable processes for the current iteration of smFDM are computed. For each mappable process $p_m \in P_{mappable}$ the set $C_{dev} \subseteq C_{free}$, containing all cores c_n with the shortest communication distance for a process p_m and cores with a specified deviation from the shortest communication distance, is computed. The cores in the set C_{dev} are sorted ascending to the communication distance, starting with the shortest communication distance. The details on the calculation of the set C_{dev} are out of scope of this paper due to the page limitation, but can be found in [18] [17]. Then the process p_m is iteratively mapped onto each free core in the set C_{dev}. For each mapping all valid routes are computed using the algorithm in Fig. 2.

Algorithm 2 Valid Routes

```
1: begin
2:    for all d_{m,a} do
3:       if data_{m,a} > cache(c_m) then
4:          compute valid routes R_{mem,m} from mem(c_m) to c_m
5:       end if
6:       compute valid routes R_{m,a} between c_m and c_a
7:       if data_{m,a} > cache(c_a) then
8:          compute valid routes R_{a,mem} from c_a to mem(c_a)
9:       end if
10:   end for
11:   for all d_{a,m} do
12:      if data_{a,m} > cache(c_a) then
13:         compute valid routes R_{mem,a} from mem(c_a) to c_a
14:      end if
15:      compute valid routes R_{a,m} between c_a and c_m
16:      if data_{a,m} > cache(c_m) then
17:         compute valid routes R_{m,mem} from c_m to mem(c_m)
18:      end if
19:   end for
20: end begin
```

Figure 2: Algorithm for computation of valid routes

The number of timeslots is calculated based on the duration of each single communication. The duration of a communication is calculated using the amount of data transmitted and the bandwidth of the links used. The point in time a communication is started is calculated using a max-plus formulation [6]. If the sending process of a communication received all subsequent communication instances from directly precedent processes, the communication data can be sent. For the valid routes and timeslots the distribution graph is computed.

After all routes for the processes $p_m \in P_{mappable}$ are computed, the self forces, successor forces, predecessor forces and the total forces for the assignment of a process to a core and thereby a data communication to one of the valid routes are computed. The mapping with the smallest total force is chosen. For that mapping the distribution graph is updated. The mapped process is inserted into the set P_{mapped} of mapped processes and the used core is removed from the set C_{free} of the free cores.

smFDM can further be used for mapping software processes on heterogeneous MPSoC architectures. For this purpose the mapping algorithm has to be adapted slightly. The algorithm needs to maintain different lists for the different kinds of free cores (e.g. DSP, GPU, ...). The algorithm then uses the corresponding list of free cores on that a process can be executed for the calculation (lines 8 to 9 Fig. 1).

IV. Results

As homogeneous MPSoC architecture we used the Tilera TILE*Pro*64 processor [22] placed on a TILExpress*Pro*-64 Card [23] for obtaining the results. The chosen architecture is highly suitable for streaming applications processing data in a computation pipeline. The industrial application from the automotive domain we used for obtaining the results is parallelized using concurrent processes and pipelining. Therefore this applications can benefit from the advantages of the Tilera architecture.

A. System Architecture

The Tilera TILE*Pro*64 architecture shown in Fig. 3a consists of 64 identical tiles arranged in a regular 8×8 mesh topology. Each tile consists of a processor, 16 kB L1 cache, 64 kB L2 cache and a router. Some of the tiles are reserved for special purpose functionality like I/O or the ethernet interface. The reserved tiles are shaded red in Fig. 3a. As the Tilera TILE*Pro*64 architecture does not support native floating point operations, floating point operations are emulated using fixed point operations instead. For communication the TILE*Pro*64 architecture has six packet-based communication networks.

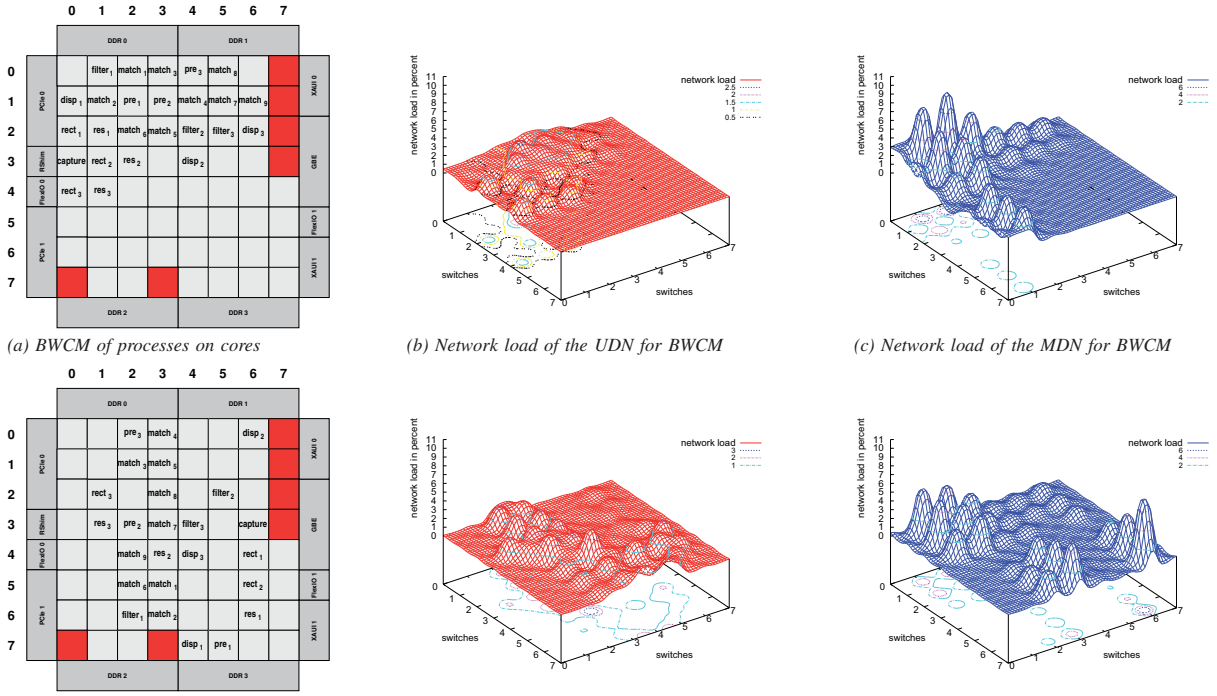
- **STN** Static Network
- **UDN** User Dynamic Network
- **TDN** Tile Dynamic Network
- **MDN** Memory Dynamic Network
- **IDN** I/O Dynamic Network
- **CDN** Coherence Dynamic Network

Only the STN and the UDN are accessible for the programmer, the other communication networks are managed by the processor itself. The UDN is used for communication between the tiles, the MDN is used for the communication between the tiles and the off-chip memory. The 64 tiles are divided in four quadrants (top-left, top-right, bottom-left, bottom-right). The cores in these quadrants use the corresponding off-chip DDR memory bank for shared memory communication.

The router transmits the data received from the local processor or from other routers using dimensional ordered XY-routing. XY-routing transmits packets first in the X direction and then in Y direction until the packet reaches the destination. For each cycle 32 bit data are transmitted and the switching frequency of the communication network is 700 MHz. Therefore the bandwidth between two routers is about 2.6 GBytes per second.

B. Application

For obtaining the experimental results we used an application from the automotive domain, a stereo camera based depth map computation as it is typically used in advanced driver assistance systems like predictive brake and adaptive cruise control for object recognition. Latency is a very crucial factor because of safety-relevant hard real-time constraints which have to be fulfilled by the object recognition.



(a) BWCM of processes on cores

(b) Network load of the UDN for BWCM

(c) Network load of the MDN for BWCM

(d) smFDM of processes on cores

(e) Network load of the UDN for smFDM

(f) Network load of the MDN for smFDM

Figure 3: Different mapping alternatives and network load for the stereo depth map computation

The stereo depth map computation uses images taken from two ethernet cameras with a resolution of 656×492 pixels. For the rectify, resize, preprocess, filter and display stages three concurrent processes were used. For the match stage nine concurrent processes are used and for the capture stage one process is used. The corresponding EAPCG with the data communication used for mapping is depicted in Fig. 4.

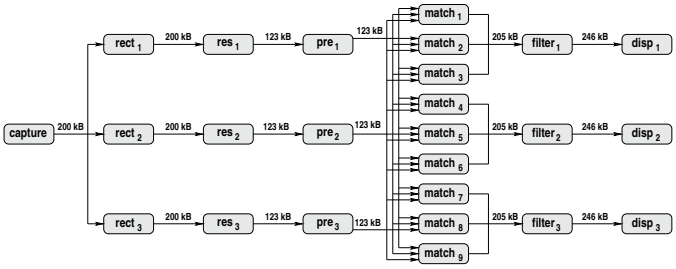


Figure 4: EAPCG of depth map computation

The concurrent processes of the depth map computation use buffered channels for packet-based data communication over the user dynamic network (UDN) of the Tilera TILE*Pro*64 architecture. Due to the fact that all communication data of the EAPCG (Fig. 4) are larger than the local caches of all cores, the communication between the concurrent processes mapped on theses cores are shared memory communication using the memory dynamic network (MDN) of the Tilera TILE*Pro*64 architecture.

C. Results

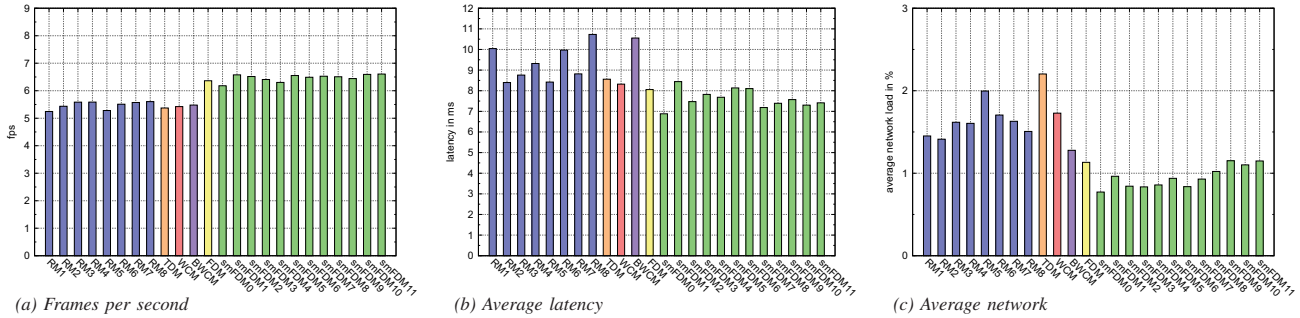
We used the presented approach for mapping the processes of the EACPG shown in Fig. 4 onto the Tilera TILE*Pro*64 architecture. As the Tilera TILE*Pro*64 architecture only supports the XY-routing algorithm this was considered during the computation of the valid routes $V_r\,(R_{i,j})$.

In Fig. 3 two different mapping alternatives for the processes of the depth map computation are shown. The first alternative (Fig. 3a) is gained using the bandwidth-constrained mapping (BWCM) presented in [15]. The processes are placed corresponding to their bandwidth requirements. Due to the fact that implementations of the mapping algorithms discussed in section II are not freely available and that details of the algorithms are missing in the corresponding papers, we used the BWCM mapping for comparison, because it could be implemented on the details in [15]. In Fig. 3d the mapping is done using the presented shared memory aware force-directed mapping (smFDM) with a deviation $dev = -1$. Here the average distance between communicating processes is minimized and conflicting communication instances are reduced.

The network loads of the UDN and the MDN for the different mapping alternatives for the stereo depth map computation, determined using the profiling capability of the Tilera Multicore Development Environment, are shown in Fig. 3e, 3f, 3b and 3c. Fig. 3b and Fig. 3c show the network load for the links of the UDN and the MDN for the mapping alternative determined using the BWCM algorithm. It can be seen that the network load for the links is nearly two times higher than the network load of the links in the UDN and the MDN for the mapping alternative determined using the presented smFDM algorithm. This is due to the fact that smFDM reduces the amount of conflicting communications on a link by distributing the communications over different link in the communication infrastructure.

For the results shown in Fig. 5a to 5c the Tilera default mapping (TDM), a possible worst-case mapping (WCM), the BWCM, the FDM [18], the smFDM with different values for the deviation dev (line 8 in Fig. 1), and eight synthetic random mappings (RM1 - RM8) were used. The results are based on time measurements on the Tilera TILE*Pro*64 architecture using special purpose registers containing the cycle count of the cores.

In Fig. 5a the results for the total system performance of the stereo depth map computation for the different mappings are shown. As it can be seen the overall frame rate is maximal for the mapping that was determined using smFDM. The frame rate can be increased in contrast to BWCM by 22%. The justification for this is that smFDM maps the processes in such a way that conflicting communication instances and memory access conflicts are reduced. Therefore the overall performance



(a) Frames per second *(b) Average latency* *(c) Average network*

Figure 5: System performance for the different mapping alternatives of the stereo depth map computation

of the system is enhanced. As the stereo depth map computation uses a lot of floating point operations and floating point is emulated by the Tilera TILE*Pro*64 architecture using fixed point operations, the frame rate is not very high. This is negligible because the absolute differences between the frame rates are of interest. The new Tilera Tile-Gx8000 [21] architecture support native floating point operations.

Fig. 5b shows that the average latency can be reduced when the smFDM is used for mapping the processes onto the cores of the Tilera architecture. The latency reduction is achieved by reducing conflicting communication and memory access conflicts by distributing the communication uniformly across the two communication networks. In contrast to BWCM the average latency for the stereo depth map computation can be reduced by 35%.

In Fig. 5c it can be seen that the average network load of the communication network can be decreased by 65% when smFDM is used.

V. Conclusion

In this paper we presented a novel approach for an automated mapping of software processes on homogeneous MPSoC architectures, taking shared memory communication as well as the routing algorithm into account. The presented approach can increase the overall system performance by mapping the processes in such a way that communication conflicts and memory access conflicts are reduced or even avoided.

The developed shared memory aware force-directed mapping (smFDM) algorithm is the first approach taking

- the route communication instances are routed through the network,
- and shared memory communication,

in conjunction with each other into account during the mapping of software processes onto the cores of a homogenous NoC-based MPSoC architecture.

The argumentation that the overall system performance can be increased is demonstrated by the application of the smFDM algorithm to the processes of a depth map computation for a stereo camera system. The presented approach was compared to the BWCM algorithm presented in [15]. The presented results show that the smFDM can reduce the latency and increase the throughput while reducing the network load in comparison to BWCM.

Our next steps are the extension of the smFDM algorithm for taking the computational characteristics of the processes into account. Additionally we are working on extension for mapping more than one process onto a core of the MPSoC architecture by using schedulability analysis. Furthermore we are working on an extension taking cache coherency into account.

References

[1] AbsInt Angewandte Informatik GmbH, "aiT WCET Analyzer," www.absint.com/ait/.

[2] A. K. Allam and J. Ramanujam, "Modified Force-Directed Scheduling for Peak and Average Power Optimization using Multiple Supply-Voltages," in *ICICDT '06: Proceedings of the 2006 IEEE International Conference on Integrated Circuit Design and Technology*, 2006.

[3] L. Benini and G. D. Micheli, "Network on Chips: A New SoC Paradigm," *IEEE Computer*, vol. 35, no. 1, 2002.

[4] D. Bertozzi, A. Jalabert, S. Murali, R. Tamhankar, S. Stergiou, L. Benini, and G. D. Micheli, "NoC Synthesis Flow for Customized Domain Specific Multiprocessor Systems-on-Chip," *IEEE Transactions on Parallel and Distributed Systems*, vol. 16, no. 2, 2005.

[5] E. Bolotin, A. Morgenshtein, I. Cidon, R. Ginosar, and A. Kolodny, "Automatic Hardware-Efficient SoC Integration by QoS Network on Chip," in *ICECS '04: Proceedings of the 2004 IEEE International Conference on Electronics, Circuits and Systems*, 2004.

[6] C. G. Cassandras and S. Lafortune, *Introduction to Discrete Event Systems*. Springer Science and Business Media Inc., 1999.

[7] G. Chen, F. Li, S. S.W., and K. M., "Application Mapping for Chip Multiprocessors," in *DAC '08: Proceedings of the 45th annual conference on Design automation*, 2008.

[8] C. J. Glass and L. M. Ni, "The Turn Model for Adaptive Routing," in *ISCA '92: Proceedings of the 19th annual international symposium on Computer architecture*.

[9] S. Gupta and S. Katkoori, "Force-Directed Scheduling for Dynamic Power Optimization," in *ISVLSI '02: Proceedings of the IEEE Computer Society Annual Symposium on VLSI*, 2002.

[10] J. Hu and R. Marculescu, "Exploiting the Routing Flexibility for Energy/Performance Aware Mapping of Regular NoC Architectures," in *DATE '03: Proceedings of the conference on Design, Automation and Test in Europe*, 2003.

[11] J. Huang, C. Buckl, A. Raabe, and A. Knoll, "Energy-Aware Task Allocation for Network-on-Chip Based Heterogeneous Multiprocessor Systems," in *PDP '11: Proceedings of the 2011 International Conference on Parallel, Distributed and Network-Based Computing*, 2011.

[12] X. Jin, N. Guan, Q. Deng, and W. Yi, "Memory Access Aware Mapping for Networks-on-Chip," in *RTCSA '11: Proceedings of the 2011 IEEE International Conference on Embedded and Real-Time Computing Systems and Applications*, 2011.

[13] Y. Liu, Y. Ruan, Z. Lai, and W. Jing, "Energy and Thermal Aware Mapping for Mesh-based NoC Architectures using Multi-objective Ant Colony Algorithm," in *ICCRD '11: Proceedings of the 2011 International Conference on Computer Research and Development*, 2011.

[14] S. Murali, L. Benini, and G. D. Micheli, "Mapping and Physical Planning of Networks-on-Chip Architectures with Quality-of-Service Guarantees," in *ASP-DAC '05: Proceedings of the 2005 Asia and South Pacific Design Automation Conference*, 2005.

[15] S. Murali and G. D. Micheli, "Bandwidth-Constrained Mapping of Cores onto NoC Architectures," in *DATE '04: Proceedings of the conference on Design, automation and test in Europe*, 2004.

[16] S. Murali and G. D. Micheli, "SUNMAP: A Tool for Automatic Topology Selection and Generation for NoCs," in *DAC '04: Proceedings of the 41st annual conference on Design automation*, 2004.

[17] T. Schönwald, A. Viehl, O. Bringmann, and W. Rosenstiel, "Distance-Constrained Force-Directed Process Mapping for MPSoC Architectures," in *Proceedings of the International Conference on Digital System Design (DSD)*, 2012.

[18] T. Schönwald, A. Viehl, O. Bringmann, and W. Rosenstiel, "Optimized Software Mapping for Advanced Driver Assistance Systems," in *Proceedings of the International Symposium on Industrial Embedded Systems (SIES)*, 2012.

[19] B. Sethuraman and R. Vemuri, "A Force-directed Approach for Fast Generation of Efficient Multi-Port NoC Architectures," in *VLSID '07: Proceedings of the 20th International Conference on VLSI Design*, 2007.

[20] A. Siebenborn, A. Viehl, O. Bringmann, and W. Rosenstiel, "Control-Flow Aware Communication and Conflict Analysis of Parallel Processes," in *ASP-DAC '07: Proceedings of the 12th Asia and South Pacific Design Automation Conference*, 2007.

[21] Tilera Corporation, "Tilera TILE-Gx8000," http://www.tilera.com/products/processors/TILE-Gx-8000.

[22] Tilera Corporation, "Tilera TILE*Pro*64," www.tilera.com/products/processors/TILEPRO64.

[23] Tilera Corporation, "Tilera TILExpress*Pro*-64 Card," www.tilera.com/products/platforms/TILExpressPro64card.

[24] X. Wang, M. Yang, Y. Jiang, and P. Liu, "Power-Aware Mapping for Network-on-Chip Architectures under Bandwidth and Latency Constraints," in *EM-Com '09: Proceedings of the 4th International Conference on Embedded and Multimedia Computing*, 2009.