Dynamic Task Remapping For Power and Latency Performance Improvement In Priority-Based Non-Preemptive Networks On Chip

James Harbin, Leandro Soares Indrusiak

Real-Time Systems Group, Department of Computer Science, University of York, UK
{james.harbin,leandro.indrusiak}@york.ac.uk

***Abstract*—In dynamic system-on-chip and multicore CPU applications, the communication patterns between tasks are not easy to characterise in advance. Dynamic task mapping is commonly used in Network-On-Chip (NoC) research in order to redistribute tasks around network processing elements at runtime in response to changes in network loading. Dynamic task mapping is anticipated to become more important as general purpose CPUs become massively multicore and system-on-chip (SoC) designs become more reconfigurable in their application usage patterns. Simultaneously, reducing NoC power consumption is a necessary consideration in the development of future scaleable and energy efficient NoC systems. The work illustrated here uses a dynamic metric which combines contention and the power consumption impact of task remapping decisions, in order to produce a non-preemptive NoC that can deliver as good or better latency as a preemptive NoC in a real application scenario, while reducing overall power consumption. The results obtained show a power consumption reduction of approximately 35% in an application case involving an autonomous vehicle application, and significant reductions in the latency of individual flows.**

I. Introduction

Networks-on-chip (NoCs) have been proposed in order to reduce the proliferation of unstructured interconnection wiring in multicore CPUs and system-on-chip (SoC) platforms [1] [2]. NoCs bring the benefits of abstraction and layering obtained in structured networks to the silicon domain, delivering a standard network capable of delivering arbitrary data through a network of on-chip routers. Given that these networks may carry diverse data such as multimedia, memory hierarchy synchronisation and system control messages, the data packets transported can be highly heterogeneous with varying lengths and destinations. Notably, packets may contain different quality of service parameters which may motivate the use of priority levels for transmitted packets [3].

Power consumption is emerging as an important metric for NoC performance. It has been shown that NoCs can consume approximately 30% of overall system power [4]. In order to increase the density of SoC and multicore platforms while delivering adequate cooling, it is important to reduce NoC power consumption. The majority of power consumption in a NoC occurs during the transmission of data upon network links. Therefore, both NoC latency and power consumption are greatly influenced by the mapping of the application tasks to processing elements [5]. A power efficient task mapping can reduce the distance the highest usage tasks have to transmit across the network, thereby reducing the power burden that they impose upon the network.

A static task mapping scheme makes task allocation decisions offline and in advance, which is useful in situations in which traffic patterns can be well characterised in advance. However, in many future applications for reconfigurable SoCs or especially multicore CPUs, task communication patterns would be unknown until runtime or may change dramatically from those anticipated at design time, requiring the system to dynamically adapt its mapping decisions. This paper presents a dynamic mapping scheme that takes into account link contention (free time intervals) and anticipated power consumption in order to produce power-efficient remappings in a case study of a real application model.

In order to deliver low latencies in wormhole NoCs, a typical solution is to use virtual channels and priority preemption at the routers, in order to allow higher priority flows to interrupt ongoing flows. However, preemptive NoCs require additional buffer space at the arbiters, which complicates their design and increases the silicon footprint of the arbiter.

This paper proposes dynamic task mapping as an alternative to preemption, in which tasks are moved around the network to avoid (or where not possible, reduce) contention for network links. The power consumption and latency resulting from applying these new application mappings to the non-preemptive NoC are assessed by simulation and compared to the preemptive network as a baseline. The intent of this paper is to demonstrate that non-preemptive NoCs can increase performance in a real application case study by using task mapping to compensate for the lack of preemption, reducing power consumption while delivering lower overall latencies. The results obtained show that non-preemptive networks with task mapping can achieve a power consumption reduction of approximately 35% compared to the preemptive NoC, in an application case involving an autonomous vehicle application. Results also demonstrate significant reductions in latency, particularly for high priority flows.

II. Related Work

In general, mapping demands for resources (such as NoC tasks) to processing elements is an instance of the quadratic assignment problem which is known to be an NP-complete problem, and therefore infeasible for optimal solution in large systems except by heuristics [6]. If effective mapping decisions can be made from information known at design time, a static task mapping approach can be used [6]. However, the task mapping philosophy most closely associated with the present work is dynamic task mapping, in which nodes use runtime task communication information in making their mapping decisions.

The MMC (minimum and maximum channel) approach [7] takes into account the runtime loading of the most heavily used link (maximum channel rate) on the route to a candidate destination, choosing the destination which minimises this maximum link loading. Also, the PL (path loading) approach [7] averages link rates along a route in order to discover the minimal latency link. The dynamic mapping approach presented here uses a similar approach, however, the structure of the metric considers free time intervals in contention rather than link rates, and also takes into account the hop count and data usage demands of the route in assessing power consumption.

[8] presents a voltage dependent mapping, which takes into account different voltage levels at which tasks must operate in order to meet their deadlines, and uses this in making mapping decisions. This is a useful approach in networks in which voltage scaling is employed to reduce power, although may not be useful in NoC designs in which voltage is globally controlled. [9] takes into account the user's behaviour information in making its mapping decisions, attempting to define which applications are critical for remapping by computing communication rate and cumulative energy ratios and using them to remap tasks within specific regions. Dynamic spiral mapping [10] uses a spiral search outwards from the current location of a task in determining a new remapping destination. An agent based mapping approach [11] uses a set of collaborating agents (both local and global) which interact in order to distribute tasks across the network.

III. Problem Description

A. Introduction

A non-preemptive wormhole NoC is not typically able to deliver low latencies to its high priority flows. Upon contention, higher priority flows are required to wait for the completion of previously admitted lower priority flows. This is illustrated in Figure 1, which illustrates routing nodes within a NoC, the placement of tasks upon their connected processors, and the paths taken by data transmissions. Only two pairs of tasks are shown for the purposes of this example, even though there may be other tasks in the system. Initially, the low priority flow $S_2 \rightarrow D_2$ is admitted and begins transmission. However, a contending higher priority flow from $S_1 \rightarrow D_1$ is then blocked and forced to wait for the shared links to be free

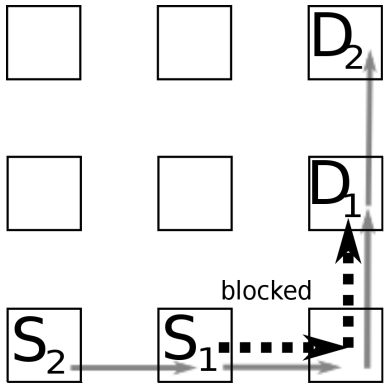

Fig. 1: Blocking of high priority flows by lower priorities in a non-preemptive NoC

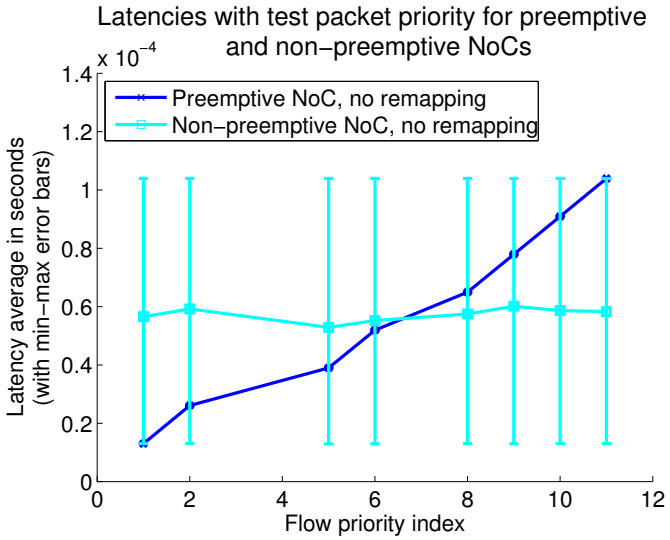

Fig. 2: Flow latencies with priority in preemptive vs non-preemptive NoCs (low priority index represents higher priority flows)

before advancing, even though quality of service goals do not favour this flow waiting for a lower priority flow.

One solution to this problem of the blocking of high priority flows is priority preemption: the design of a NoC arbiter that assesses the priority of waiting and requesting flows, and allows the higher priority flow to interrupt an ongoing lower priority flow. However, a priority preemptive NoC requires additional buffer space in order to support the buffering necessary on virtual channel endpoints (with buffering requirements scaling with the number of priority levels), imposing additional implementation costs both in silicon area and in arbiter design and development complexity.

An example of the effects of blocking upon the latencies of high priority flows is illustrated in Figure 2. It compares the latencies delivered by non-preemptive and priority preemptive NoCs, for packets of identical size generated simultaneously towards the same destination. Release jitter is applied, consisting of a very small randomised timing offset which ensures

that the temporal order of arrival of packets at the NoC is effectively random. Without the presence of release jitter, the value of preemption would not be fully evident, since the non-preemptive NoC may perform as well due to packets being generated from the application model in priority order.

By convention in this paper lower priority index values dominate higher priority index values, therefore packets with priority 1 represent the highest priority flows. Notably, the distribution of latencies for the non-preemptive system is close to uniform across priorities, due to blocking of newly introduced high priority flows by ongoing flows. By contrast, the preemptive NoC is able to deliver a smooth gradient of latency with priority, illustrating its ability to deliver faster communication to high-priority flows by preemption of the lower priority ones.

The intent of this paper is to assess whether a non-preemptive NoC can use task remapping to deliver lower latencies, while delivering also reductions in overall power consumption. This is illustrated in Figure 3, which shows a possible outcome of the task remapping process in Figure 1. The remapping decisions which have been made seperated the tasks of the contending flows and therefore removed the contention, allowing both flows to transmit simultaneously.

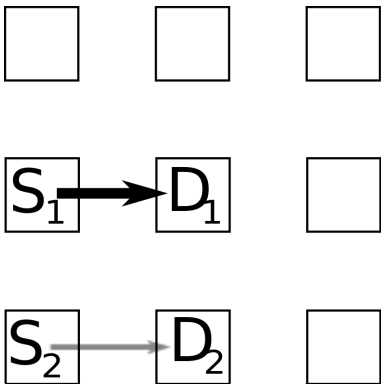

Fig. 3: Task mapping adjustments allowing both flows to proceed simultaneously

In dynamic power consumption models, power consumption can be modelled by approximating the bus as a series of parallel capacitors which have to be charged and discharged every time bit transitions occur over the relevant link [12]. This power consumption scales in proportion to the size of packets transmitted multiplied by the number of hops from source to destination. Therefore, remapping also leads to reduced power consumption by moving each source closer to its respective destination. Solving the problem through resource management (task mapping) rather than hardware (arbiter decision making complexity) will enable a reduction in the complexity of NoC routers and therefore an increase in overall performance under dynamic application models in which the inter-core communication patterns cannot be accurately predicted in advance.

B. Scenario Definition

The scenario assumed for this paper is a priority preemptive NoC with virtual channels, versus a non-preemptive NoC. Both NoCs operate using wormhole switching which allows data to be propagated through the network without the need to completely buffer the packet at an intermediate router. Both alternative NoC designs consist of a 4x4 grid topology in which XY routing [13] is used to form the routes between communicating cores, although the remapping algorithm presented is sufficiently general as to allow extension to NoCs incorporating other routing algorithms. The difference between the NoC architectures studied is that the preemptive NoC allows a flow in progress to be interrupted by a higher priority flow, while in the non-preemptive NoC a flow which has claimed a link cannot be interrupted at that link, and all other flows that contend have to be blocked. The dynamic remapping process is applied in the non-preemptive NoC, but in the preemptive NoC, which is used as a baseline case without remapping, the original static mapping is used throughout.

The application considered in the results case study is an autonomous vehicle application. As opposed to the use of synthetic traffic, this full application is a highly heterogeneous data set of 38 tasks which incorporates the video processing, navigation and control information for an autonomous vehicle. A manually tuned default initial mapping is used as a starting point in all simulations as defined in our previous work [14].

C. Simulation Implementation

The simulations performed in this paper are implemented as transaction level models (TLM) in the Ptolemy II simulator. The TLM concepts used in this paper generate simulator events only upon flow admission, completion, or upon contention with another flow which shares at least one link with the flow. Using these coarser transaction level models which issue transactions only upon flow entry, exit or contention boundaries enables considerable simulation performance improvement (approximately three orders of magnitude) over cycle-accurate models, or TLM schemes which model the progression of every flit through the network [15].

Power consumption modelling has been added to the simulations following the methodology presented in [14]. The power consumption is approximated by tracking the number of bit transitions occurring between adjacent flits upon the same link, upon the assumption that the majority of power consumption in a CMOS circuit results from charging and discharging the capacitances represented by the bus wires of the NoC links [12]. The simulator therefore registers the contents of flits applied to each link in temporal order during TLM processing and performs an XOR operation upon adjacent flits, keeping a running counter of bit transitions and therefore power consumption per link. This allows the total power consumption during simulation to be computed.

Two different simulation models are contrasted in the results, with the preemptive NoC without any remapping being the baseline or control case. When flows in the simulator share at least one link, only one is active and allowed to

proceed simultaneously. The logic upon flow addition chooses the highest priority flow to activate when multiple flows are admitted simultaneously. If another higher priority flow is admitted, the original is preempted and must wait for the new one to complete before advancing further. The preemptive NoC architecture is as presented fully in [14].

The non-preemptive simulation model does not permit preemption by a higher priority flow. Upon flow admission, the distance to its first contention point is computed. The flow which will reach the contention point first is allowed to claim it and is set to active. If two simultaneously admitted flows are an equal distance from the contention point, then the highest priority of the two will claim it. Once a flow has passed through the contention point, then the model assumes that it cannot be preempted and will flow until completion.

IV. Remapping Algorithm Description

The intent of the remapping algorithm considered here is to improve power consumption performance by intelligent reallocation of the tasks upon the processing cores of the system. Accordingly, the remapping process first identifies the flows that are most in need of remapping. These are the flows upon congested routes, or with a high impact upon power consumption due to carrying a large amount of data a relatively long distance over the NoC. Then the process constructs a listing of possible endpoints, namely possible source-destination node pairings. The remapping process then relocates the tasks generating these needy flows to closer and less congested processing elements, in which the power and contention impact of these flows will be reduced.

The remapping algorithm is triggered periodically upon the expiry of a timer known as the sampling interval. This reduces the number of remapping events and therefore the operating system and communication rate required to perform the remappings. Throughout the sampling interval, each NoC router examines the headers of blocked packets to keep track of those experiencing contention, and communicates properties of these contended flows to the core executing the operating system (or other component responsible for management of task mappings).

Upon the expiry of the sampling interval timer, the remapping matching process is performed according to the algorithm in Listing 1. The remapping matching process employs two priority queues. The first priority queue $flowNeeds$, contains the flows that experienced contention, sorted in descending order of their need metric RN (Equation 1). The second priority queue $endpoints$ is constructed using all possible source destination pairings (including those with source and destination values the same, if design constraints permit multiple processes being assigned to a single core).

In sorting the $endpoints$ priority queue, the primary criterion is the hopcount in the NoC between the endpoints, and the secondary criterion is the total number of tasks mapped at source and destination endpoints, divided by a mapping density factor M_f and truncated to an integer. Therefore, the general preference of the algorithm will be for choosing source-destination pairs that are close together and lightly loaded. However, the algorithm will treat endpoints as having identical loading when their loading is below the mapping density factor M_f. This motivates the algorithm to cluster tasks together in one region, rather than dispersing them too widely.

Listing 1: Pseudocode for the remapping matching process triggered during execution each sampling interval

```
1  remappingAlgorithm() {
2  foreach flow_i in contendingFlows() {
3    flowNeeds.insert(flow_i, RN_flow_i);
4  }
5
6  foreach endpoint in possibleEndpoints() {
7    endpoints.insert(endpoint);
8  }
9
10 flowsToRemap = int(flowNeeds.size *
       remapProportion);
11
12 remapCount = 0;
13 remappedTasks = {};
14 while (!flowNeeds.empty() && remapCount
       < flowsToRemap) {
15   target = flowNeeds.removeHighest();
16   TASK_S = getSourceTask(target);
17   TASK_D = getDestinationTask(target);
18   if (!remappedTask.containsEither(
         TASK_S, TASK_D)) {
19     chosen = endpoints.getLowest();
20     remap(TASK_S, chosen.source);
21     remap(TASK_D, chosen.dest);
22     remapCount = remapCount + 1;
23     remappedTasks.add(TASK_S, TASK_D);
24   }
25 }
26 }
```

From line 14 onwards, the remapping process then processes a fixed proportion of the flows deemed to be in greatest need of remapping. Firstly, the tasks associated with the flow are looked up. If they have not yet been remapped during this run of the algorithm, their remapping this time is recorded (to prevent these tasks being remapped later by other flows with a lower need). The tasks are then remapped to the best endpoints (source and destination processing cores) chosen. Finally, any endpoints which have been updated (by either a task entering or leaving them) have their metrics recomputed, and the priority queue $endpoints$ is resorted to ensure the best endpoint is chosen if necessary.

$$RN_{flow} = \left[\sum_{links} (1 - F_{link}) + \frac{D}{D_{max}} H \right] C_{count} \quad (1)$$

The terms employed in Equation 1 are defined as follows. F_{link} refers to the *free proportion* of the link, namely the longest proportion of the sampling interval in which the link was unused. $links$ is the set of links used by the flow. H is the hop count of the route (number of links). D is the total amount of data contained in the flow, and D_{max} is the largest amount of data contained in any flow yet encountered during remapping. C_{count} is the total number of times in which another flow caused contention with this flow during the sampling interval.

The first part of Equation 1 favours the selection of flows which are upon heavily used links for remapping, since flows which are heavily contended will tend to experience much greater latencies, as other flows will have to wait for their completion in a non-preemptive network. The second part of the equation selects flows based upon their approximate power consumption, using the number of links traversed and the data demand of a flow to approximate its power consumption. Although actual power consumption during transmission will be unknown without knowledge of the precise sequence of bit transitions that occur upon the link, it can be expected that reducing the product of end-to-end flit sent and hop count will reduce power consumption on average.

Therefore, the intent of the remapping need equation is to combine an incentive for both latency reduction (by prioritising for remapping flows upon heavily congested links, as exhibited in the first term in the equation) and power consumption reduction as exhibited by the second term. Of course, with traffic pattern changes it may be possible to achieve both objectives simultaneously, for example by remapping tasks to a closer destination thereby reducing both the contention encountered and the power consumption on intermediate links (as illustrated in Figure 3). However, in the autonomous vehicle application used in the results, 38 tasks are assigned amongst a total of 16 processors, so the task remapping algorithm is frequently faced with the tradeoff of reducing the latency of one source-destination task pair at the cost of increasing another. This therefore provides a test of its adaptability to real scenarios in which processors are heavily contended.

V. Simulation Results

The default parameters used in the simulation are presented in Table I.

Parameter	Value
Default flit size (bits)	32
NoC Clock speed (MHz)	100
Buffer capacity (flits)	7
Number of virtual channels	40
Sampling interval	0.5s
Proportion of flows remapped	0.5
Mapping density factor M_f	6
Simulation run time	10s

TABLE I: Simulation parameters

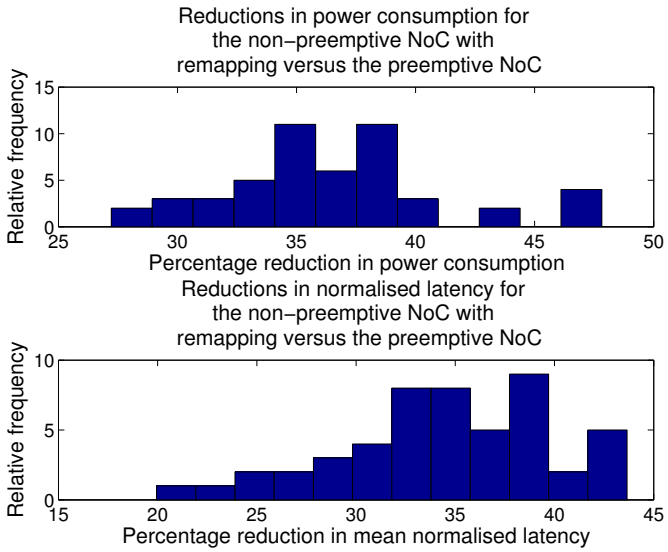

Fig. 4: Power consumption and mean normalised latency reductions as histograms

Figure 4 presents a histogram of power consumption and mean normalised latency with application remapping, and in particular the reduction in power consumption with jitter. The maximum jitter in flow arrival at the NoC is one tenth of the clock period. The power consumption reduction is calculated relative to the baseline of the preemptive system. (The baseline power consumption results for both the preemptive and non-preemptive NoC without remapping are virtually identical. The impact of preemption itself on power consumption is minimal given that flit transitions occurring on a preemption boundary are outnumbered greatly by those within a non-preempted packet). Power consumption is mainly a feature of the application mapping and how closely communicating flows can be sited. The results show that the mean power reduction is a saving of approximately 35%, and that this has a standard deviation of around 4.5%. There are two outliers delivering a power consumption reduction of over 42%, which occured when the jitter of flow arrival causes some remappings to be performed in one sampling interval rather than the next, causing particularly advantageous remapping decisions.

Figure 5 demonstrates the normalised latencies per flit (in seconds per flit) delivered for the preemptive vs a non-preemptive NoC using dynamic application remapping. An initial mapping was used which was developed from our previous research [15] [14]. The left graph contrasts the latencies in the final three seconds of simulation runtime (following stabilisation of the system under the new application) showing that the non-preemptive NoC with application remapping is capable of delivering considerably lower latencies than the preemptive NoC. In particular a number of flows receive effectively zero latency since their source and destination pairs are placed upon the same endpoint, which means they do not have to cross the NoC. The flow with priority index 7 is the only outlier, which is one of the video processing flows.

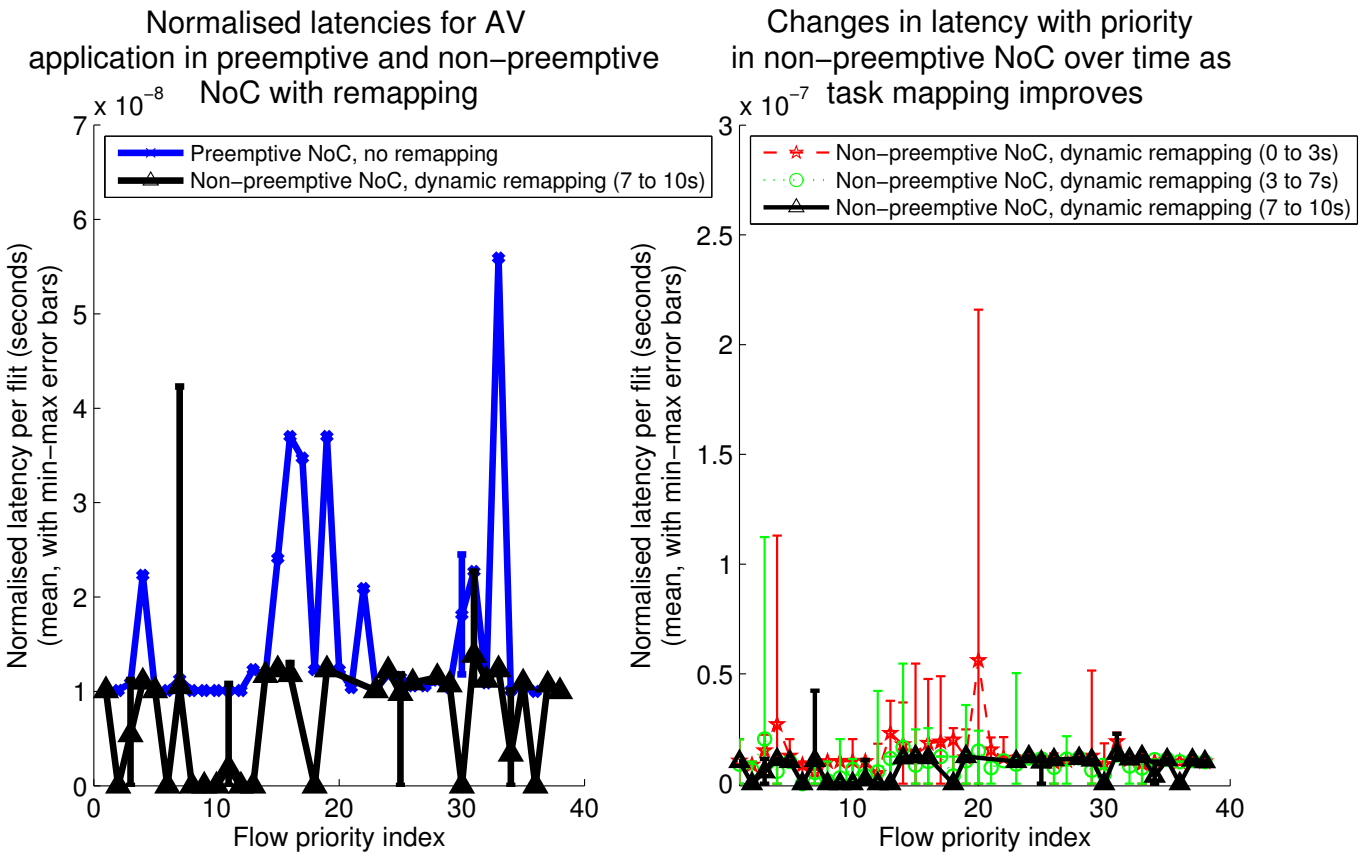

Fig. 5: Latencies with priority index values for the autonomous vehicle application. Low priority values represent high priority flows.

Since it is issued simultaneously with other flows of higher priority that transfer more data, it therefore experiences a higher latency than the others as part of the cost of giving them a better mapping.

In the rightmost graph of Figure 5, the three series show the latencies experienced during three stages of the dynamic mapping process over the first ten seconds of execution, during which the system adapts to the application traffic pattern. During the exploratory phase within which the first remapping decisions are made, the latencies of some flows increase considerably as early remapping decisions are taken, leading to additional contention and longer delays for communication to occur. In particular, an outlier is flow 20. This is partially due to the need to adapt to the loading provided by the application, which develops in the early phases of execution historical data such as D_{max}. Since the power impact of flows is ranked relative to the largest ever yet encountered in the NoC, the earliest remapping decisions may not be able to correctly discriminate between small and large flows. However, from 3 to 7 seconds mean normalised latencies for most flows decline considerably, although some flows in the range 9 to 15 are showing increased latency from more exploratory remapping decisions.

VI. Further Work

Although the remapping need metric used in this paper combines consideration of the impact of link free intervals and power impact in choosing the flows most in need of remapping, it has combined both these aspects uniformly. Future work will consider varying this by the use of additional parameters to allow the end user to manage the trade off between these potentially conflicting objectives.

Additional further work will include a consideration of the hardware implementation of the features necessary to support flow tracking at the NoC routers, and the implementation costs of these features and any control messages that have to be added. The granularity of the sampling interval and remapping algorithm execution (currently at 0.5s) is relatively long compared to the overall NoC timescale. Therefore, the impact of the power consumption of moving tasks across the network is expected to be limited, compared to the power savings achieved in regular data transmission. One potential extension for future work is to bound the maximum remapping rate of a particular task, and determine the impact of this constraint on power consumption performance. Although the granularity of the sampling interval is sufficiently long in this example case study that remapping decisions will be relatively

infrequent, a real hardware implementation will be very useful to verify the concepts.

In future work, it will also be important to take into account the impact of link coding on power consumption. Using low power coding (e.g. [16]) it is possible to reduce the power consumption of particular links by encoding data so as to reduce its bit transition activity. In this case, the algorithm proposed for dynamic remapping would have to be extended to take into account that the power consumption impact of particular links differs depending on the codec selected.

VII. Conclusions

This paper has demonstrated that dynamic application mapping can produce both power consumption and latency benefits in a non-preemptive network, equivalent to and frequently exceeding those of obtained from using a preemptive NoC. The latency impact upon priority levels has been considered in order to determine the impact of the remapping process upon individual flows, showing that the majority of flows experience significantly reduced latencies. These results have been generated in a real case study incorporating heterogeneous data in a complete autonomous vehicle application. The results obtained showed a power consumption reduction of approximately 35% in an application case involving an autonomous vehicle application, and significant reductions in latency particularly for high priority flows.

ACKNOWLEDGEMENTS

The authors are grateful to the EPSRC for providing financial support, under project 'LowPowNoC', contract EP/J003662/1.

References

[1] S. Pasricha and N. Dutt, *On-chip communication architectures: system on chip interconnect*. Morgan Kaufmann, 2008.

[2] R. Marculescu, U. Y. Ogras, L.-S. Peh, N. E. Jerger, and Y. Hoskote, "Outstanding research problems in NoC design: System, microarchitecture, and circuit perspectives," *IEEE Trans. Computer-Aided Design of Integrated Circuits and Systems*, vol. 28, no. 1, pp. 3 –21, Jan. 2009.

[3] E. Bolotin, I. Cidon, R. Ginosar, and A. Kolodny, "QNoC: QoS architecture and design process for network on chip," *J. Syst. Archit.*, vol. 50, no. 2-3, pp. 105–128, Feb. 2004.

[4] K. Lee, S.-J. Lee, and H.-J. Yoo, "Low-power network-on-chip for high-performance SoC design," *IEEE Trans. VLSI Syst.*, vol. 14, no. 2, pp. 148 –160, Feb. 2006.

[5] P. K. Sahu and S. Chattopadhyay, "A survey on application mapping strategies for network-on-chip design," *Journal of Systems Architecture*, vol. 59, no. 1, pp. 60 – 76, 2013.

[6] A. Hansson, K. Goossens, and A. Rădulescu, "A unified approach to constrained mapping and routing on network-on-chip architectures," in *Proceedings of the 3rd IEEE/ACM/IFIP international conference on Hardware/Software codesign and system synthesis*, ser. CODES+ISSS '05. New York, NY, USA: ACM, 2005, pp. 75–80.

[7] 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*, 2007, pp. 34–40.

[8] C.-L. Chou and R. Marculescu, "Incremental run-time application mapping for homogeneous nocs with multiple voltage levels," in *Hardware/Software Codesign and System Synthesis (CODES+ISSS), 2007 5th IEEE/ACM/IFIP International Conference on*, 2007, pp. 161–166.

[9] ——, "User-aware dynamic task allocation in networks-on-chip," in *Design, Automation and Test in Europe, 2008. DATE '08*, 2008, pp. 1232–1237.

[10] A. Mehran, A. Khademzadeh, and S. Saeidi, "Dsm: A heuristic dynamic spiral mapping algorithm for network on chip," *IEICE Electronics Express*, vol. 5, no. 13, pp. 464–471, 2008.

[11] M. A. Al Faruque, R. Krist, and J. Henkel, "Adam: run-time agent-based distributed application mapping for on-chip communication," in *Proceedings of the 45th annual Design Automation Conference*, ser. DAC '08. New York, NY, USA: ACM, 2008, pp. 760–765.

[12] T. Mudge, "Power: a first-class architectural design constraint," *Computer*, vol. 34, no. 4, pp. 52 –58, Apr. 2001.

[13] V. Rantala, T. Lehtonen, and J. Plosila, "Network on chip routing algorithms," University of Turku, Turku Centre for Computer Science, Tech. Rep. 226, 2006.

[14] J. Harbin and L. S. Indrusiak, "Fast transaction-level dynamic power consumption modelling in priority preemptive wormhole switching networks on chip," in *SAMOS: International Conference on Embedded Computer Systems Architectures Modelling and Simulation*, Jul. 2013.

[15] L. S. Indrusiak and O. M. dos Santos, "Fast and accurate transaction-level model of a wormhole network-on-chip with priority preemptive virtual channel arbitration," in *DATE 2011: Design, Automation Test in Europe Conf.*, Mar. 2011, pp. 1–6.

[16] A. Garcia-Ortiz, L. S. Indrusiak, T. Murgan, and M. Glesner, "Low-power coding for networks-on-chip with virtual channels," *Journal of Low Power Electronics*, vol. 5, no. 1, pp. 77–84, 2009.