Task Allocation for Decoding Multiple Hard Real-time Video Streams on Homogeneous NoCs

Hashan Roshantha Mendis
Real-time Systems Group
Department of Computer Science
University of York
Email: hrm506@york.ac.uk

Neil C. Audsley
Real-time Systems Group
Department of Computer Science
University of York
Email: neil.audsley@york.ac.uk

Leandro Soares Indrusiak
Real-time Systems Group
Department of Computer Science
University of York
Email: lsi@cs.york.ac.uk

***Abstract*—Hard-real time video systems require deterministic admission control decisions to maintain high levels of predictability. These decisions can be based on the state-of-the-art schedulability analysis of tasks and flows. However, due to the pessimistic behaviour of the schedulability analysis and the uncertainties in the application, the multi-core system resources are usually under-utilised. In this paper we propose two task allocation techniques that exploit application and platform characteristics in order to increase the number of simultaneous, fully schedulable, video streams handled by the system. The first, more generic technique, uses the worst-case remaining slack of the mapped tasks as a heuristic to determine the task to processing core allocation. The paper also investigates a second technique that maps the heavily communicating, critical path tasks of the applications onto same core to reduce the communication overhead. We compare against other heuristic based dynamic mapping techniques in the literature, and show that an overall improvement of up to 10%-15% can be obtained, in admission rates and system utilisation.**

I. Introduction

In multimedia electronics a clear trend can be observed towards using multi/many-core system-on-chip device architectures. The processing load imposed by computation intensive applications such as video decoding can be distributed among the multiple processing cores, to reduce power consumption and to meet timing constraints. Audio-video systems are generally considered soft real-time, as a violation of timing constraints results in degraded quality, but the system can continue to operate. However there exists a range of systems that depend on video streams that need to be processed with hard real-time guarantees. In vision-based robot control systems, information is extracted from the vision sensors and used as feedback to control the motion of a robot. These systems need to process the captured video frames within a strict time period, to function the control system correctly. Another example where hard-real time video processing is required is in the telesurgery industry where a doctor performs surgery on a patient, without physically being in the same location. These safety-critical systems require cutting edge reliable communication technology as well as hard real-time guarantees from the video processing systems to function safely. Furthermore, next generation video surveillance systems will require processing and tracking objects in hundreds of video streams in real-time; missing deadlines in these systems would lead to reduced security and delayed response to threats.

Video decoding systems that can guarantee hard timing requirements for unpredictable workloads are often under-utilised [1]. The intent of this work is to use communication and blocking aware task mapping heuristic, to increase the number of schedulable hard real-time video decoding streams, and thereby increasing system utilisation.

The rest of this paper is organized as follows. Section II presents related work in task mapping and scheduling. Section III introduces the system models. Section IV presents the deterministic admission controller, and Section V describes the proposed heuristic based mapping algorithms. Section VI presents the experimental design and discusses the results. Section VII concludes this paper and provides future directions.

II. Related work

Bamakhrama et al. [2] investigate the applicability of scheduling hard real-time streaming applications modelled as acyclic cyclo-static dataflow graphs. They present an analytical framework to determine the minimum number of processors required to schedule a set of streaming applications with given I/O rates, while guaranteeing the maximum achievable throughput. In [3], the critical paths of the task-graphs are mapped onto the processing cores first to reduce the average end-to-end worst-case execution time of a set of multimedia streaming applications. They use a simple utilisation based feasibility test to determine if a dataflow graph can be allocated to a PE. Ditze et al. [4] present real-time scheduling and admission control of several parallel video decoding streams. They use an extension to the least-laxity-first scheduling algorithm to schedule the MPEG decoding tasks and an admission controller that enforces QoS constraints of the video streams.

Carvalho et al. [5] introduce several communication and congestion-aware runtime mapping heuristics. The hop-distance and path load between cores are exploited, to reduce communication packet latency, energy and channel occupation. This work was later extended by Singh et al. [6] to include processing cores that can accommodate multiple tasks, and grouping tasks to reduce communication energy/traffic. Kaushik et al. [7] extends the work in [6] by balancing both computation and communication using a pre-processing stage to achieve a balanced and reduced task-graph. Chou et al. [8] present an incremental mapping approach which allocates incoming applications to processing elements, such that communication energy is minimised. The mapping also ensures that a PE with minimum operating frequency for meeting the application deadlines is selected. They use a near convex region selection technique to obtain mapping decisions close to optimum ones.

Many of the existing mapping heuristics also do not consider the temporal behaviour of the tasks which is necessary when dealing with applications that require strict timing guarantees. In contrast to these existing approaches, our mapping approach exploits the task timing and communication properties as well as the architecture characteristics.

III. System Model and Problem Formulation

A. Application model

We consider an application model of M independent stream based work-flows, each containing video streams that require decoding. Each video decoding stream consists of an arbitrary number of N independent, sporadic, jobs, as shown in Figure 1. We assume the number of video streams and their start/end times are completely arbitrary, and therefore at a given time the system will be decoding multiple video streams simultaneously which are contained within the different parallel workflows. Each job denoted as J_i, represents a group of dependant MPEG frame decoding tasks (also known as an MPEG group of pictures - MPEG GoP). Hence each job contains a chain of tasks (Figure 2) with certain dependency/precedence constraints such that a tasks' execution can only start *iff* its predecessor(s) have completed execution and their output data is available. All tasks read/write to main memory via the on-chip network interconnect. Each task reads data (i.e. encoded MPEG frame) from the main memory before starting execution and writes back after execution completion (i.e. decoded MPEG frame). A task τ_i is characterised by the following tuple: (p_i, t_i, x_i, c_i, a_i); where p_i is the fixed priority, t_i is the period, x_i is the actual computation cost in terms of execution time, c_i is the worst-case computation cost and a_i is the arrival time of the task τ_i. Tasks are preemptive and have a fixed priority. Tasks within a job are assigned fixed mapping and priorities at the start of the video stream; these exact assignments are used for all tasks of all jobs in a video stream, thereby enabling us to analyse the application worst-case timing properties. Tasks of low resolution video streams are given higher priority over high-resolution video streams, to ensure a lower response-time for low-resolution video streams.

We consider MPEG-2 encoded video streams, with, I (Intra), P (Predictive) and B (Bi-directional) picture frames. We assume a MPEG GoP structure, with a fixed task precedence and communication pattern, as shown in Figure 2. The spatial resolution of a video stream will correspond directly to the computation cost of the task and the payload of the message flow. The exact execution time of the tasks are unknown in advance; however, we assume the worst-case computation cost can be estimated via execution profiling [9]. Subtask deadlines are not known however, each job is considered schedulable if it completes execution on/before its end-to-end deadline (D_{e2e}). Once a task has completed execution, its output (i.e. the decoded frame data) is immediately sent as a message flow to the processing element executing its successor child tasks. Message flows inherit the priority of their source tasks, with an added offset to maintain unique message flow priorities. A message flow, denoted by Msg_i is characterised by the following tuple: (P_i, T_i, PL_i, C_i); where P_i is the priority, T_i is the period, PL_i is the payload and C_i is the basic latency of message flow Msg_i.

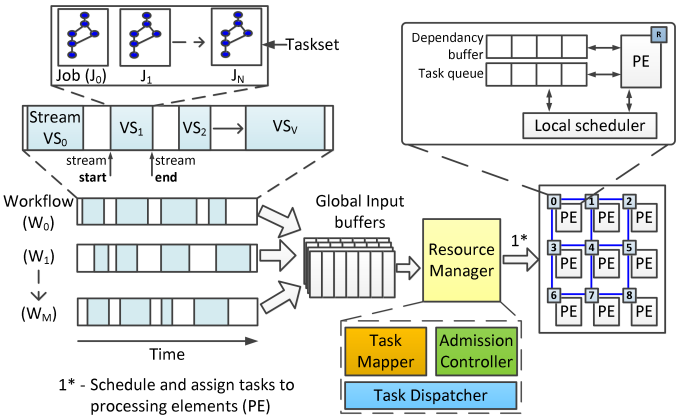


Fig. 1: System overview diagram

B. Platform model

The multi-core platform we are focusing on is composed of P homogeneous processing elements (PEs) connected by a network-on-chip (NoC) interconnect. The PEs are directly connected to the NoC switches which route data packets towards a destination core. We assume the NoC in our platform model uses fixed priority preemptive arbitration, has a 2D mesh topology and uses the XY deterministic algorithm for routing such as in [10]. We assume that the NoC link arbiters can preempt packets when higher-priority packets request the output link they are using. This makes it easier to predict the outcome of network contention for specific scenarios.

We assume all inter-PE communication occurs via the NoC by passing messages. Once a task is released from a global input buffer, it is sent to the task queue of the assigned PE. The PE upon completing a tasks execution, transmits its output to the appropriate PEs dependency buffer. Once a task has completed, the local scheduler picks the next task with the highest priority with dependencies fulfilled, to be executed next. The resource manager (RM) of the system (Figure 1), performs task mapping and priority assignment, admission control (AC) and task dispatching to the PEs. It also maintains a task-to-PE mapping table of the jobs of every admitted and active video stream in the system.

C. Problem statement

Deterministic AC tests that use worst-case response time of jobs can be used to guarantee that all admitted video decoding jobs do not miss their deadlines; however, they result in under-utilised system resources [1]. With proper task to node mapping approaches the admission rates of a deterministic AC can be increased and in turn the system utilisation also can be improved. This is challenging as the workload characteristics such as execution time and arrival patterns are not known beforehand. We present heuristic based runtime mapping approaches that consider the current state of the platform nodes, the task and flow blocking behaviour in order to minimise communication and computation load of the processing elements.

IV. Deterministic Admission Control

The deterministic admission controller (D-AC) performs worst-case timing analysis based schedulability tests on the video streams to determine whether to admit or reject a video

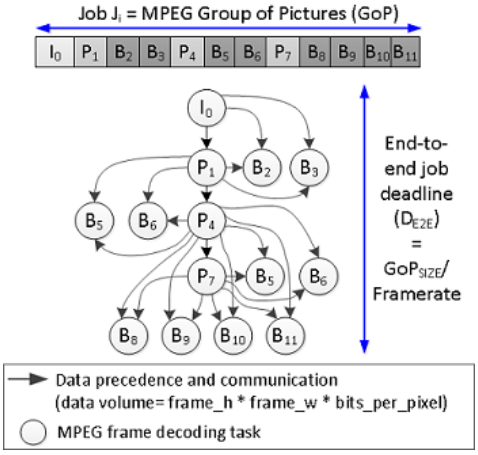


Fig. 2: MPEG GoP data precedence and task communication graph. (Communication traffic between tasks and main-memory not illustrated)

stream. In [1], we are shown via simulation that a D-AC can be used to guarantee that none of the admitted video streams will miss their deadlines; hence a D-AC can be used in a hard real-time video streaming system.

Each video stream is composed of a set of sporadic tasks and flows, and due to the fixed-priority preemptive nature of the NoC arbiters and the PE local schedulers, tasks and flows can be blocked by higher priority tasks and flows of other active video streams. We use a fixed mapping scheme for all the jobs within a video stream, hence classical schedulability analysis [11], [12] can be applied to obtain the worst-case response-time of the tasks and flows of each active video stream. In [11], we are given a method to calculate the worst-case response time (WCRT) of tasks r_i taking into account the blocking introduced by higher priority tasks (denoted as $hp(i)$) mapped on the same PE; this equation is shown in Eq.1. This work is then adapted in [12] to derive an upper bound for the worst-case network latency of each traffic flow in wormhole switching, fixed priority preemptive NoCs. In [13], the response time r_i of the task τ_i that releases the message-flow is considered to be the release jitter J_i^R of Msg_i - Eq.3. In this equation, J_i^I is the interference-jitter and C_i is the basic latency of the message flow Msg_i as described in Eq.2. Direct-interferers (denoted as S_{id}) are higher priority traffic-flows that have at least one physical link in common with the observed traffic-flow.

$$r_i^{n+1} = c_i + \sum_{\forall \tau_j \in hp(i)} \left\lceil \frac{r_i^n}{t_j} \right\rceil c_j \tag{1}$$

$$C_i = (numHops \times arbitrationCost) + numFlits \tag{2}$$

$$R_i^{n+1} = C_i + \sum_{\forall Msg_j \in S_{id}} \left\lceil \frac{R_i^n + r_j + J_j^I}{T_j} \right\rceil C_j \tag{3}$$

$$WCRT(J_i^{CP}) \leq D_{e2e} \tag{4}$$

Since we are only concerned with the end-to-end deadline D_{e2e}, the schedulability test should check if the worst-case response-time of the *critical path* of a job, denoted $WCRT(J_i^{CP})$ is less than or equal to the end-to-end deadline of the job D_{e2e}, as defined in Eq. 4. The critical path of a job (denoted J_i^{CP}) is the path with the maximum accumulated cost, where *cost* refers to the WCRT of tasks and flows within a job. For example in the task graph shown in Figure 2 the critical path of the job would start with the root node I_0 and end in any one of the leaf nodes (i.e. B-frames). For every received video stream decoding request the RM first performs an initial task-to-PE mapping. Secondly the D-AC then calculates the response time of every path in the task-graph to determine J_i^{CP} and thereby evaluating $WCRT(J_i^{CP})$ for that particular task-to-PE mapping. A video stream is granted admission, *iff* the expression given in Eq. 4 is true for the new and all active video streams in the system; this guarantees that the worst-case timing requirements of all existing and new video streams will be successfully met. The J_i^{CP} and $WCRT(J_i^{CP})$ are properties of a given task-to-PE assignment, hence task mapping is integral to the D-AC decision.

V. Proposed Mapping Approaches

A. Least worst-case remaining slack (LWCRS)

The utility function given in Algorithm 1 is used to find the task-to-PE mapping which will give the least worst-case remaining slack (LWCRS) for the target task. The worst-case slack of a task is the difference between the task deadline and worst-case computation cost. This function is integral to both mapping strategies introduced in this paper. It takes into account the blocking behaviour of the tasks already mapped onto a given PE to approximate the slack a PE has to accommodate a new task.

The function iterates through the provided *PE_list* and calculates the following for each task-to-PE mapping: $RemSlack_t$ - the worst-case remaining slack (WCRS) for the given target task - blocking incurred by higher-priority tasks already mapped on the PE (line 7) are taken into account; and $RemSlack_{lp}$ - the WCRS for each of the lower-priority tasks already mapped on the PE, taking into account the worst-case execution cost (c_i) of the target task (line 10-11). We use the worst-case execution cost of higher priority tasks to determine the amount of blocking for a target task, hence the slack calculation may be pessimistic but safe. A weight is then assigned to each PE in *PE_list*, which is equal to the sum of $RemSlack_t$ and $RemSlack_{lp}$, for each of the searched PEs (line 14). The PE with the lowest weight is selected as the PE with the least worst case remaining slack (line 19). If no PE is found with positive worst-case slack the PE with the lowest utilisation (Section VI-B1) is selected (line 22). This attempts to find the PE mapping that will result in the tightest temporal-fit, without missing the deadlines of the target task nor any of the already mapped tasks. Since we do not know the deadline of each individual task in the job, we use the sub-task deadline assignment given by Kao and Garcia-Molina [14], which divides the total remaining slack among the subtasks in proportion to their estimated execution times.

B. LWCRS-aware mapping

The behaviour of the LWCRS aware mapping algorithm is illustrated in Algorithm 2, and makes use of the LWCRS utility function explained in Section V-A as well as tries to minimise distance between communicating tasks. The primary

Algorithm 1 Utility function: _get_PE_with_least_slack pseudo-code

Input: τ_i : target task; *copy_tm_tbl* : copy of the runtime task mapping table; *PE_list* : list of PEs to search
Output: tuple : (result PE, search result (boolean))
1: $PE\_packing = \{\}$
2: **for all** $PE_i \in PE_{list}$ **do**
3: // obtain following from *copy_tm_tbl*
4: Get $MPTasks(PE_i)$: tasks already mapped on PE_i
5: Get $hp(\tau_i)$: higher priority tasks from $MPTasks(PE_i)$
6: Get $lp(\tau_i)$: lower priority tasks from $MPTasks(PE_i)$
// get worst-case remaining slack (WCRS) to target task
7: τ_i^{slack} : task slack w.r.t estimated sub-task deadline
8: $RemSlack_t = \tau_i^{slack} - \sum_{\forall \tau_j \in hp(\tau_i)} c_j$
// get WCRS on low-pri mapped tasks
9: $RemSlack_{lp} = \{\}$
10: **for all** $\tau_j \in lp(\tau_i)$ **do**
11: $RemSlack_j = \tau_j^{slack} - \sum_{\forall \tau_k \in hp(\tau_j)} c_k$
12: Insert $RemSlack_j$ to $RemSlack_{lp}$
13: **end for**
// populate local data structure
14: **if** $RemSlack_t > 0$ and $\forall x \in RemSlack_{lp} | x > 0$ **then**
15: $PE\_packing\,[PE_i] = RemSlack_t + \sum RemSlack_{lp}$
16: **end if**
17: **end for**
// if none of the PEs in the list will provide a positive WCRS, then choose the PE with min. utilisation
18: **if** $\forall x \in PE\_packing | x > 0$ **then**
19: $found = TRUE$
20: PE_j = index of MIN(PE_packing)
21: **else**
22: $found = FALSE$
23: PE_j = *_get_PE_with_min_util*(*copy_tm_tbl*, *PE_list*)
24: **end if**
25: **return** $(PE_j, found)$

objective of the algorithm is to tightly pack (i.e in the temporal domain) each task of the job, into the PEs, in order to leave room for tasks of future video streams. Parallel video streams will be pipelined on the PEs, such that the initial PEs in the NoC will be heavily utilised before selecting the next available PE. We do this in order to increase the number of simultaneous video streams that the system can handle without missing any deadlines. LWCRS-aware mapping is a generic technique, that can be applied to map other types of applications that have dependency/communication patterns and use fixed priority-preemptive scheduling.

The algorithm takes the task-graph (TG) given in Figure 2, and a copy of the runtime task mapping table *copy_tm_tbl* (maintained by the RM) as input. Each entry of the task mapping table includes the task-to-PE mapping and the characteristics of each mapped task of every active video stream. The utility function explained in Section V-A is used to find a PE with the least amount of task slack. Each non-root node in the TG can have up to two parents. We define the parent with the largest route cost from the root node to be the closest-parent τ_i^{PARENT}. For example, in Figure 2, $\tau_{B3}^{PARENT} = P1$. For each node in the TG the algorithm obtains the PE that has the closest-parent mapped onto it - PE_i^P (line 6). Then the closest PE to PE_i^P with the least-remaining worst-case slack

Algorithm 2 LWCRS-aware mapping heuristic algorithm pseudo-code

Input: all tasks in the job (J_0), *copy_tm_tbl* : copy of the runtime task mapping table
Output: task to processing element mapping : MPG ($\tau_i \rightarrow PE_i$)
1: **for all** unmapped tasks : $\tau_i \in J_0$ **do**
2: **if** τ_i is *root_node* **then**
3: $(PE_i, found)$ = ***_get_PE_with_least_slack***(τ_i, *copy_tm_tbl*, all_PEs)
4: Map ($\tau_i \rightarrow PE_i$); Update *copy_tm_tbl*;
5: **else**
6: // obtain following from *copy_tm_tbl*
7: Get τ_i^{PARENT} : parent task with max route cost.
8: Get PE_i^P : PE that τ_i^{PARENT} was mapped onto
// get neighbours of increasing hop_counts
9: **for** $hc = 1$ to MAX_HOPS **do**
10: $N\_PE_i^P$ = *_get_neighbours*(PE_i^P, hc)
11: Append PE_i^P into $N\_PE_i^P$
12: $(PE_i, found)$ = ***_get_PE_with_least_slack***(τ_i, *copy_tm_tbl*, $N\_PE_i^P$)
13: **if** $found$ is *TRUE* **then**
14: Map ($\tau_i \rightarrow PE_i$); Update *copy_tm_tbl*;
15: break loop; Go to step 1;
16: **end if**
17: **end for**
// if no suitable PE is found, select closest min. util. PE
18: **if** a suitable PE is NOT found **then**
19: $N\_PE_i^P$ = *_get_neighbours*(PE_i^P, $hop\_count = 1$)
20: PE_i = *_get_PE_with_min_util*(*copy_tm_tbl*, $N\_PE_i^P$)
21: Map ($\tau_i \rightarrow PE_i$); Update *copy_tm_tbl*
22: **end if**
23: **end if**
24: **end for**
25: **return** *copy_tm_tbl*

is searched (lines 8-15). It is important to note that the PE_i^P is included in the PE list that is to be searched (line 10). Once a suitable PE is found, the *copy_tm_tbl* is updated with the selected PE (line 4, 13). If no suitable PE is found, the PE with lowest utilisation (Section VI-B1) is selected (line 17-20).

C. I and P frames combined (IPC) mapping

Unlike in LWCRS aware mapping, for the IPC task allocation strategy, we exploit the known application-specific dependency and communication patterns of the TG. Figure 2 shows there are many message flows being sent out by I and P frame decoding tasks. Also the path $I_0 \rightarrow P_1 \rightarrow P_4 \rightarrow P_7$ is usually the critical path of the job, assuming the B-frame decoding tasks do not get blocked too heavily. Therefore by combining the I and P frames we attempt to: (a) reduce the number of communication flows active in the NoC; (b) attempt to reduce the end-to-end job response time by making sure a majority of the TG nodes in the potential critical path is executed as soon as possible. By reducing the number of potential communication flows active in the NoC we reduce the probability of other lower-priority message flows in the network being blocked. We map all the I and P frame decoding tasks of the job to the currently lowest-utilised PE on the platform. The B-frame decoding tasks are mapped as close to their parent tasks; the hop-distance is limited to 2 hops. The B-frame decoding tasks can be executed in parallel. Also

their computation cost is usually smaller than I/P frames, they have a higher chance of occupying temporal gaps in the PE task queues. The utility function *_get_PE_with_least_slack*, explained in Section V-A, is used to find the task-to-PE mapping that will result in the least worst-case remaining slack from the 2-hop neighbours of the parent task.

VI. Evaluation

A. Experiment design

A discrete-event, abstract simulator with a light-weight NoC simulation component as explained in [1] has been adopted to perform all experiments. A 3x3 NoC and video stream workload is modelled as described by the characteristics explained in Section III. The D-AC in Section IV is used for all experiments, to guarantee all admitted video streams meet their deadlines. The total workload introduced into the system can be defined as a summation of the resolutions of all the video streams (VS) admitted and active in the system, as shown in Eq.5. We evaluate our mapping approaches for a range of workload values; starting from a single video stream with 230×180 resolution to 9 parallel video streams with 720×576 resolution. The relationship between video stream spatial resolution and task computation cost was implemented at a MPEG-block level as described in [15]. The inter-arrival time of the video stream jobs were uniformly distributed between $1.0\times D_{e2e}$ and $1.3\times D_{e2e}$.

$$Workload = \sum_{\forall v_i \in VS} [frame\_h(v_i) \times frame\_w(v_i)] \quad (5)$$

We measure the video stream *admission-rates* and *PE utilisation*. Admission rate is calculated as a ratio between the admitted video streams over the total video stream decoding requests. The PE utilisation is the ratio between the total active (busy) time of all PEs in the system over the total simulation time.

B. Baseline mapping heuristics

1) PE with lowest utilisation (LU) and lowest mapped (LM) tasks: We compare against two computation load-balancing, heuristic based mappers. The LU heuristic maps each task to the lowest utilised PE. Utilisation of a PE is measured as given in Eq.6, where $MPTasks(PE_i)$ denote all tasks mapped on PE_i. The LM heuristic maps the target task to the PE with the least amount of mappings.

$$U = \sum_{\forall \tau_i \in MPTasks(PE_i)} \left\lceil \frac{c_i}{t_i} \right\rceil \quad (6)$$

2) Communication-aware mapping: The path-load based best-neighbour (BN) heuristic defined in [5], and the pre-processing (PP) based algorithm defined in [7] are used as baselines. The original BN algorithm was adapted to support multiple tasks and have used PE utilisation to determine available nodes, while maintaining path-load as the main heuristic.

C. Experimental results

Lower worst case end-to-end job response-time will lead to better admission rates, by the D-AC. Figure 3 shows the calculated analytical worst-case end-to-end job response time distributions for different mapping types. The distribution

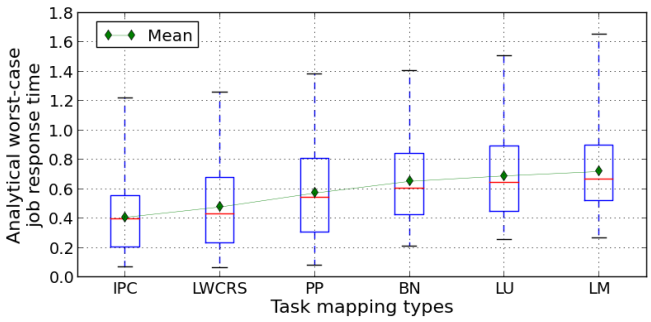
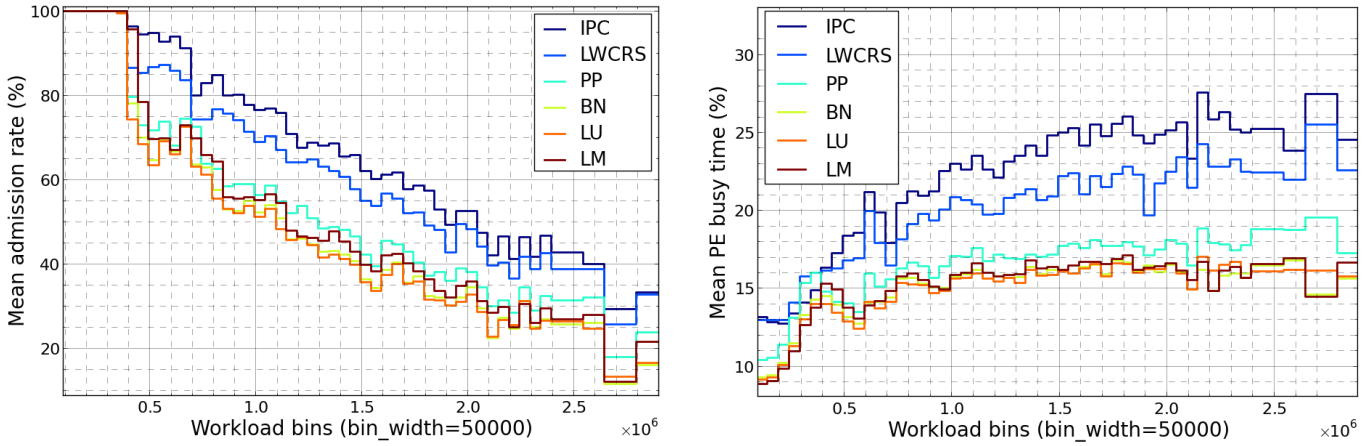


Fig. 3: Worst-case end-to-end job response times

represents data from 40 uniquely seeded simulation runs of 5 fixed parallel video streams of different resolutions. The arrival patterns and execution costs of the tasks will differ depending on the randomiser seed used. The proposed mapping techniques, IPC and LWCRS shows lower analytical worst-case job response times compared to the baselines. IPC performs slightly better than LWCRS because the critical path of the job is mapped on the same processing element, hence the job is completed quicker. The LU, LM, and BN mappings show very similar response-time distributions, while the PP mapping has a lower mean job response time than the other baselines, but show a similar maximum to BN.

The admission rates for different workload levels are shown in Figure 4a. Each data point obtained represents the percentage of admitted streams in a particular simulation run. The x-axis represents the different workload levels, and the measured data is separated into equal width bins; the step-plot represents the mean of the data within each bin. Admission rates decrease as the workload increase, because for high workloads the AC cannot guarantee the timing requirements will be met, and hence more rejections will be made. Around the mid-high workloads the proposed mapping techniques (IPC and LWCRS) show an improvement of about 10% - 15% over the baseline mappings. Since the worst-case response times of the jobs are lower, when these two mappers are used, the AC will admit more video streams. We can observe that overall IPC performs better than LWCRS for all workloads, showing a 2-8% improvement. However, LWCRS is application independent, and can be applied to map tasks of different applications that have similar type of dependency based task graphs. Under very high workload conditions, unlike IPC, LWCRS has an advantage because tasks can be more spread out across the PEs, reducing the communication contention. The improvement of IPC and LWCRS over the baselines drop slightly during certain higher workloads (e.g. 2.25×10^6), but still show a distinct advantage over the baselines. PP shows slightly better admission rates than the other baselines for all workload levels because it reduces communication by grouping certain tasks in the task-graph together.

Higher admission rates result in more tasks being processed by the system, leading to increased PE utilisation as depicted in Figure 4b. We can see that as the workload increases, the PE utilisation also increase up to a certain level (approx. 1.0×10^6), after which the PE busy level does not improve significantly, due to the decline of the admission rates (Figure 4a). The IPC and LWCRS mappings show a 5%-10% improvement in utilisation over the LM, LU, BN and PP mappings for



(a) (b)

Fig. 4: Varying admission rates and PE utilisation for a range of workload levels, for different mapping techniques. (a) Workload vs. Admission rate (mean values plotted for equal bin widths) (b) Workload vs. PE busy time (mean values plotted for equal bin widths)

workloads over 1.0×10^6. IPC shows a higher PE busy time than LWCRS for workloads higher than 0.5×10^6. PP seems to offer better PE utilisation than the other compared baselines, specially at higher workload levels (i.e. over 2.3×10^6). Both IPC and LWCRS are designed to reduce task communication; hence the mappings that they perform result in higher PE utilisation while the baseline mappings have a higher NoC utilisation due to more communication. For example around the lowest workload level (i.e. $0-0.25\times10^6$), the admission rate for all mapping schemes are at 100% but the proposed LWCRS and IPC mappings show higher PE busy values than the baselines.

VII. Conclusion

This paper describes two application and platform aware mapping strategies that attempt to decrease the end-to-end response-time of the video stream decoding jobs. The first (LWCRS) technique attempts to tightly pack tasks in the temporal domain by using the worst-case remaining slack as a heuristic. The second technique (IPC) combines the critical-path tasks and maps them to a single PE, and the remaining tasks according to LWCRS. Our mapping techniques improves the admission rates of a hard real-time deterministic admission controller and thereby increasing system utilisation. Potential further work in this area will be to explore better combined priority assignment and mapping techniques and evaluate against a hard real-time mapper.

Acknowledgement

We would like to thank the LSCITS program (EP/F501374/1) and DreamCloud project (611411), for funding this research and RheonMedia Ltd. for providing industrial case studies.

References

[1] H. R. Mendis, L. S. Indrusiak, and N. C. Audsley, "Predictability and utilisation trade-off in the dynamic management of multiple video stream decoding on network-on-chip based homogeneous embedded multi-cores," in *Proc. of the 22nd Int. Conf. on Real-Time Networks and Sys.*, 2014, pp. 161–170.

[2] M. A. Bamakhrama and T. P. Stefanov, "On the hard-real-time scheduling of embedded streaming applications," *Design Automation for Embedded Sys.*, vol. 17, pp. 221–249, 2013.

[3] H. I. A. A. Ali, L. M. Pinho, and B. Akesson, "Critical-path-first based allocation of real-time streaming applications on 2d mesh-type multi-cores." in *RTCSA*, 2013, pp. 201–208.

[4] M. Ditze, P. Altenbernd, and C. Loeser, "Improving resource utilization for MPEG decoding in embedded end-devices," in *Proc. of the 27th Australasian Conf. on Computer Science*, 2004, pp. 133–142.

[5] E. de Souza Carvalho, N. Calazans, and F. Moraes, "Dynamic task mapping for MPSoCs," *IEEE Design Test of Computers*, vol. 27, pp. 26–35, 2010.

[6] A. K. Singh, T. Srikanthan, A. Kumar, and W. Jigang, "Communication-aware heuristics for run-time task mapping on NoC-based MPSoC platforms," *Journal of Sys. Arch.*, vol. 56, pp. 242–255, 2010.

[7] S. Kaushik, A. Singh, and T. Srikanthan, "Computation and communication aware run-time mapping for NoC-based MPSoC platforms," in *SOC Conference*, 2011, pp. 185–190.

[8] C.-L. Chou, U. Ogras, and R. Marculescu, "Energy- and performance-aware incremental mapping for networks on chip with multiple voltage levels," *IEEE Trans. on Computer-Aided Design of Integrated Circuits and Sys.*, vol. 27, pp. 1866 –1879, 2008.

[9] M. Roitzsch and M. Pohlack, "Principles for the prediction of video decoding times applied to MPEG-1/2 and MPEG-4 part 2 video," in *Real-Time Sys. Symp. (RTSS)*, 2006, pp. 271–280.

[10] E. Bolotin, I. Cidon, R. Ginosar, and A. Kolodny, "QNoC: QoS architecture and design process for network on chip," *Journal of Sys. Arch.*, vol. 50, pp. 105–128, 2004.

[11] N. Audsley, A. Burns, M. Richardson, K. Tindell, and A. J. Wellings, "Applying new scheduling theory to static priority pre-emptive scheduling," *Software Eng. Journal*, vol. 8, pp. 284–292, 1993.

[12] Z. Shi, A. Burns, and L. S. Indrusiak, "Schedulability analysis for real time on-chip communication with wormhole switching:," *Int. Journal of Embedded and Real-Time Comms. Systems*, vol. 1, pp. 1–22, 2010.

[13] L. S. Indrusiak, "End-to-end schedulability tests for multiprocessor embedded systems based on networks-on-chip with priority-preemptive arbitration," *Journal of Sys. Arch.*, vol. 60, pp. 553–561, 2014.

[14] B. Kao and H. Garcia-Molina, "Deadline assignment in a distributed soft real-time system," *IEEE Trans. on Parallel and Distributed Systems*, vol. 8, pp. 1268–1274, 1997.

[15] Y. Tan, P. Malani, Q. Qiu, and Q. Wu, "Workload prediction and dynamic voltage scaling for mpeg decoding," in *Proc. of the 2006 Asia and South Pacific Design Automation Conf.* IEEE Press, 2006, pp. 911–916.