Low Communication Overhead Dynamic Mapping of Multiple HEVC Video Stream Decoding on NoCs

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

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

ABSTRACT

The High Efficiency Video Coding (HEVC) standard offers several parallelisation tools such as wave-front parallel processing (WPP) and Tiles (independent frame regions) to better manage the computationally expensive workloads on modern multicore/many-core platforms. However, poor allocation of tile-level HEVC decoding tasks to processing elements may result in increased latency and energy consumption due to data-communication overhead between dependent tiles. In this work, we discuss the difficulties in decoding multiple HEVC bitstreams with highly varying resolutions and data-dependency characteristics as seen in HEVC coded video streams with random-access, adaptive group of pictures (GoP) structures. Secondly, in order to address the above challenges, we introduce a runtime tile allocation scheme that help to reduce the energy usage during HEVC decoding. Evaluations against a bin-packing algorithm, show that the proposed workload mapping technique is able to maintain reasonably acceptable latency results, whilst reducing communication overhead (8-10%) and increasing the mean processor idle periods (∼30%) to support dynamic power management.

CCS Concepts

•**Networks** → **Network on chip;** •**Hardware** → **On-chip resource management;** •**Computing methodologies** → *Image compression;*

Keywords

HEVC, Dynamic task mapping, Low-communication, NoC

1 Introduction

The HEVC video compression standard is the successor of the industry leading codec H.264/AVC(Advanced Video Codec), and aims to reduce the bit-rate by 50% while maintaining the same video quality. However, as HEVC decoding complexity is known to be approximately 4 times that of H.264/AVC, significant effort has been made to reduce the latency of HEVC decoding [2, 9]. To overcome the computation complexity of a HEVC codec the standards have introduced native data-parallel mechanisms such as, wave-front parallel processing (WPP) and Tiles [3]. In WPP, lines of coding tree units (CTU) are processed in parallel, where CTUs are essentially 64x64 pixel blocks conceptually similar to macroblocks in H.264/AVC. In tile-based parallel decoding, different rectangular regions of the picture can be processed in parallel without any dependency between different tiles of the same frame [18]. A parallel HEVC decoder obtains the tile offsets in the bitstream by parsing the headers and then distributes the tiles to the different processing elements to be decoded. When decoding multiple streams of live or on-demand video (e.g. multipoint video conferencing, multi-camera video surveillance, multi-user multimedia gateways), the workload characteristics are increasingly dynamic and are difficult to predict beforehand. Therefore, online resource management techniques are required to keep the video decoding latency at a minimum while efficiently utilising the platform resources.

1.1 Problem statement

Advanced coding tools such as the use of hierarchical B-frame structures [18], random-access profiles [4] and adaptive group of picture (GoP) structures for scene-change detection [19], adds to the complexity of the HEVC decoding workloads due to complex inter-task data dependencies. Transmitting large amounts of reference frame data leads to significant increase in on-chip communication energy consumption. Efficient mapping of tiles to processing elements need to take into account the workload and platform characteristics in order to decode the video sequences with minimised data communication energy consumption whilst maintaining acceptable decoding latencies.

Network-on-chip (NoC) communication interconnects, commonly seen in many-core systems, can consume approximately 30% of overall system power [13]. In processing elements, short active working cycles and long idle periods have shown to reduce energy usage when combined with dynamic power and frequency management techniques [12]. Viewer dissatisfaction can scale proportionally to decoding latencies of video streaming applications [6]. Furthermore, the execution overhead of runtime task mapping techniques need to be kept to a minimum such that they do not further impact the response-time of the tasks. Previous work in tile mapping [11, 15] have used first-fit bin-packing heuristics to maximise the utilisation of the cores whilst minimising the number of cores used, during HEVC encoding. However, these heuris-

PARMA-DITAM '16, January 18 2016, Prague, Czech Republic

DOI: http://dx.doi.org/10.1145/2872421.2872422

tics do not attempt to reduce the communication-energy of the platform, neither do they consider inter-task dependencies, varying communication-to-computation ratios (CCR), and resource contention seen in NoC-based multicores when dealing with multimedia workloads.

1.2 Novel contributions

The energy-efficient workload distribution is facilitated via the following novel contributions:

1. We introduce the challenges in a tile-level parallel HEVC decoding task model, where inter-prediction reference picture data is transmitted via the on-chip interconnect as message flows, by-passing the shared main memory (a communication model commonly seen in NoC-based multi/many-cores).
2. We introduce application-specific, runtime tile-mapping techniques, to decode multiple HEVC video streams in parallel. The mapping scheme takes into account the critical-path, workload CCR and task blocking behaviour when making mapping decisions to reduce the overall communication overhead whilst maintaining reasonable decoding lateness and mapping overheads.

2 Related work

There are arguments for and against using the different parallel tools in the HEVC standard; the scalability of WPP decreases as the number of CTU rows increase due to synchronisation issues whilst Tiles incur high coding losses [3]. Further results by the same author Chi et al. [4], showed that when using a variant of WPP on a many-core, performance saturates as the workload, number of cores and threads increases due to the heavy contention on the shared resources. Similar multi-threaded HEVC decoder experiments on multicore platforms have shown that Tile-level parallelisation offers better speed-ups and scalability over WPP with the increase in thread-count [7]. Bross et al. [2] explore SIMD optimizations and frame-level parallel processing to decode 50Hz 4K/UHD video streams in real-time. Their results show that when only picture-level parallelism is used, inter-prediction limits the speed-up that can be achieved.

Few works in the literature have explored how to distribute HEVC decoding workload on multiple cores in order to simultaneously address power consumption and decoding latency. In [11], monitoring information is fed-back into a centralised HEVC encoding workload allocator to continuously adapt the load and the operating frequency of the processing cores. Extensions to this work have addressed power efficiency issues in tile-based parallel processing by minimising the number of compute cores [15]. Contrary to this work, we do not employ a communication expensive feedback monitoring mechanism and we do not utilise shared memory for inter-core communication.

Attempts to address task allocation issues for decoding random-access streams can be seen in [9]. Static task-to-core configurations for CTU-level decoding are explored to improve the video stream decoding time on a shared memory, non-uniform access system. This work focusses on dynamic task mapping for unpredictable video stream decoding workloads. The state-of-the-art in task mapping primarily use heuristics to allocate unpredictable applications workloads at runtime; whilst, various design-time optimisation strategies are used to map tasks when the workload is known a priori [16]. Communication-aware mapping heuristics first introduced by Carvalho et al. [5] have now been extended to support multiple-tasks in a PE [17] and to balance computation and communication via task-graph clustering [10]. A MPEG2 decoding task-graph aware mapping heuristic that utilises the worst-case slack of the tasks is introduced in [14]; however, they assume a fixed GoP structure and dependency pattern to guarantee hard timing requirements.

3 System model

3.1 Frame-level application model

The application model consists of a number of simultaneous video streams; each video stream having an arbitrary number of independent jobs (Fig. 1). The inter-arrival times of the video streams and their jobs are randomly distributed. A frame-level decoding task τ_i consists of tile-level decoding sub-tasks, denoted as τ_i^j. We denote an HEVC open-GoP as an independent job J_i, and is modelled as a task-graph (TG) with dependency patterns between the frame-level tasks that will differ in subsequent jobs of the same video stream based on the temporal and spatial characteristics (Fig. 2). A tasks' execution can only start *iff* its predecessor(s) have completed execution and their output data is available. We assume the video streams are coded using hierarchical B-frames and adaptive GoP structures with legal inter-frame dependency patterns. A frame-level task τ_i has the following characteristics: priority (p_i), actual execution cost (x_i), worst-case execution cost (c_i), arrival-time (a_i). We assume tasks have a fixed priority and are preemptive. We assume the end-to-end relative deadline of the job (D_{e2e}) is known ($D_{e2e} = |J_i|/framerate$). The lateness of a job is calculated as $D_{e2e} - RT(J_i)$, where $RT(J_i)$ denotes the end-to-end response time of a job. The x_i is derived at the coding unit (CU) level, where the number, type and dimension of the CUs per frame is uniformly distributed.

3.2 Deriving the tile-level application model

When a job (J_i) arrives into the system the resource manager (RM) partitions the frame-level tasks into tile-level sub-tasks, thereby forming a new job structure denoted J_i^T. We assume the tile sub-tasks (denoted τ_i^j) task properties, follow the same notation as the tasks (e.g. x_i^j is the computation cost of the j^{th} tile of task τ_i). Tiles will inherit certain real-time properties of the frames such that $x_i^j{=}x_i$, $p_i^j{=}p_i$, $a_i^j{=}a_i$. x_i^j and c_i^j are calculated proportional to the ratio between the respective tile and frame dimensions. During frame to tile partitioning, the precedence constraints of the TG need to be unaffected. Hence, the number of edges and nodes in the TG will scale proportionally to the number of tiles (N_T) per frame decoding task (number of new edges $= |edges| \times (N_T)^2$). In the example given in Fig. 3, a TG with 4 nodes and 3 edges is transformed into a TG with 8 nodes and 12 edges after tile-partitioning, assuming $N_T{=}2$. We model the J_i^T communication edge (e) volume as per Eq. 1, where $PL(e_i^j)$ refers to the J_i^T edge communication volume and U refers to a uniform random distribution. The resulting increased number of nodes and especially communication edges after tile partitioning, induces more interference to lower priority tasks and flows already existing in the system. The cumulative computation cost and communication volume of the new tile-level TG is still unchanged; however, the degree to which a newly admitted job will disrupt existing lower priority tasks and flows in the system is

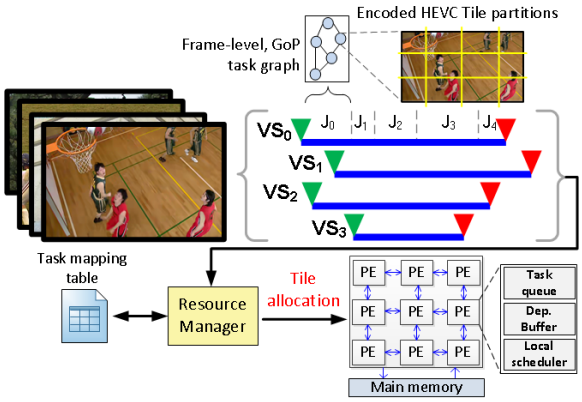

Figure 1: System overview diagram

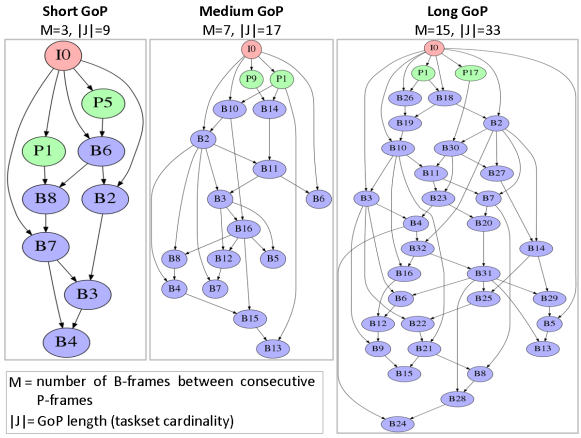

Figure 2: HEVC GoP data precedence and task communication graph (Communication traffic between tasks and main-memory not illustrated)

dependent on how sparsely/densely they are spread across the processing elements. Hence, the tile allocation problem becomes more challenging as the number of tiles increase due to their inter-dependencies. For the sake of clarity, henceforth, *tile-level* subtask and their respective data flows will be commonly referred to as a *tasks* and *flows* respectively. Frames will be referred to as 'frame-level tasks'.

$$PL(e_i^j) \sim U\big(1, PL(e_i)\big); \text{ s.t } \sum_{j=0}^{(N_T)^2} PL(e_i^j) = PL(e_i) \quad (1)$$

3.3 Platform model

We assume a multi-core platform composed of P homogeneous processing elements (PEs) connected by a NoC interconnect (Fig. 1). The PEs are directly connected to the NoC switches which route data packets towards a destination core. The NoC in our platform model uses fixed priority preemptive arbitration, 2D mesh topology and uses XY deterministic routing such as in [1]. Each PE connected to the NoC contains a local memory, local scheduler, task queue and a dependency buffer. All inter-task communication as well as task to main memory (MM) communication occurs via the NoC by passing messages. Tasks transmits reference data used for inter-prediction to the appropriate PEs dependency buffer. If a parent and child tasks are mapped onto the same core, the NoC is not utilised for data transfer and the reference data is saved in the current PE dependency buffer. Higher priority tasks that have all dependencies fulfilled can interrupt already running lower priority tasks. Encoded tile

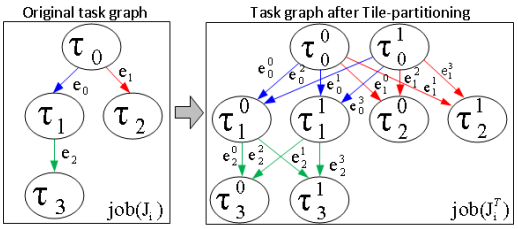

Figure 3: Original HEVC task graph before and after tile partitioning

data is sent to the PE from the MM before execution, and decoded raw data is sent back to the MM after task completion. Four MM controllers are located at the four outer edges of the NoC.

3.4 Runtime task mapping table

The system RM performs dynamic task mapping. It maintains a task-mapping table ($TMTbl$), with information such as: task-to-core mapping, task properties (such as c_i^j, x_i^j, p_i^j). This table is used during the runtime tile mapping process to determine already mapped and active tasks in the system. Compared to a centralised monitoring feedback approach [11], our method would incur lower communication and less scalability issues as the platform size and workload increase [16]. $TMtbl$ is updated after a new task has been mapped to a PE and refreshed before a mapping process, when a new job arrives into the system. During a table refresh, each tasks' *completion status* with respect to the worst-case blocking incurred due to higher priority tasks, is estimated and removed.

4 Clustered tile mapping

The proposed clustered mapping scheme (CL) aims to reduce the data-communication between the tile-tasks by clustering them and mapping them onto the same or neighbouring processing elements. Algorithm 1 describes the pseudo-code of the clustered tile mapper. The primary level of clustering is achieved by grouping the frame-level tasks that lie on the *longest-path* of the TG (line 2 and 6 of Algorithm 1). For fast evaluation ($O(n)$) of this search, we assume nodes and edges are unweighted. In Algorithm 1, τ_0^0 denotes the first tile of the first frame (usually an I-frame) in the job; τ_i^0 denotes the first tile of tasks in the non-clustered set. Task clustering is performed by applying constraints on the hop distances with respect to the job CCR as given in Table 1. NH_T denotes the maximum hop distance of non-clustered tile-tasks to the initial tile of the respective frame-task. NH_{CT} denotes the maximum hop distance between the clustered tile tasks and τ_0^0. NH_{CH} denotes the maximum hop distance from a non-clustered τ_i^0 and their parent with the highest volume data-dependency. HEVC decoding workloads are broadly categorised into three CCR ranges : Low ($CCR < 0.5$), Medium ($1.5 \leq CCR \leq 0.5$) and High ($CCR > 1.5$). The algorithm varies the cluster size based on the job CCR, where CCR is defined as per Eq. 2. We classify jobs using CCR rather than resolution, because CCR takes into account the number of edges (inter-task communication) into account as well as the resolution. A larger cluster is selected for jobs with a low-CCR (e.g. computation-bound decoding of a 4K Ultra-HD stream) in order to exploit maximum data-parallelism. For high-CCR (e.g. communication-bound, 240p video) jobs the cluster

Table 1: CCR specific task mapping hop distance. NOC_W: NoC width, N_T: number of tiles per frame

CCR range	NH_T	NH_{CT}	NH_{CH}
Low	Max. hops	Max. hops	Max. hops
Med	$\min(\frac{NOC_W}{2}, N_T)$	0	$\min(\frac{NOC_W}{2}, N_T)$
High	0	0	1

size is reduced to decrease communication-energy and interference to the NoC traffic. We assume, a very low CCR for compute-intensive, decoding of a 4K Ultra-HD resolution video stream and much higher CCR for decoding a low-resolution (e.g 240p) video stream.

$$CCR_{job} = \frac{\text{Total edges cost}}{\text{Total nodes cost}} = \frac{\sum_{\forall e_i \in TG} C_i}{\sum_{\forall \tau_i \in TG} c_i} \quad (2)$$

The algorithm iteratively assigns tiles to PEs in topological order (line 1). At each iteration a constrained set of possible PEs (sPE_N) are obtained according to the maximum hop-distance specified in Table 1. Higher number of hops will result in larger PEs to be considered. A hop number of zero forces the task to be mapped on to the same PE as its parent or τ_i^0. sPE_N is then searched to obtain the PE with the minimum number of active lower-priority tasks with respect to the target task τ_i^j, such that the task blocking introduced to already active lower priority tasks in the system can be minimised. This is done via the *getPELowBlocking* helper function (lines 8,11,17,20). This function returns the PE in set sPE_N, which has the least number of low-priority tasks with respect to target task τ_i^j. The utility function *getNeighbours* returns the neighbouring PEs of a target PE, within a specified hop distance. The mapping example illustrated in Fig. 4, shows the distribution of HEVC video decoding tasks over a 4x4 NoC. In both mapping schemes, the high resolution (low CCR) 2160p video tasks are scattered over the PEs to reduce utilisation hotspots. On the other hand, the 1080p video (medium CCR) on the clustered mapper has constrained the tasks mainly to node (3,2), and to a few neighbouring nodes, attempting to balance clustering vs. spreading the tasks. The 240p video has a high CCR and hence to minimise the communication, the tasks are tightly grouped to node (3,1) while the least utilised mapper produces a fair distribution, causing increased communication on the NoC.

The algorithm complexity is dependent on the number of tasks in a job $|J_i^T|$, number of processing elements in the NoC $|PE|$ and number of tasks in a task queue $|TQ|$. Hence, under heavy workload conditions, the worst-case complexity would be $O(|J_i^T| \times |PE| \times |TQ|) \equiv O(n^3)$.

5 Evaluation

Experimental evaluation is performed through a discrete-event, abstract simulation of a 6x6 NoC platform with the characteristics described in Section 3. Further details of the NoC simulation implementation and its level of accuracy can be found in [8]. Three increasing workload levels are explored - 7, 14 and 28 parallel video streams (denoted as WL1, WL2, WL3) with uniformly distributed video resolutions ranging from 240p to 2160p resolutions and 10 GoPs each. Number of tile partitions per frame is proportional to resolution. Low resolution videos are assigned higher priorities than higher resolution videos. We measure the job lateness, data communication overhead (volume in GB), mean node idle period (longer idle gaps in the PE schedule can support more low-power/sleep states) and mapping overhead (as profiled via the simulator). Varying arrival rates, job dependency patterns and task execution costs are tested via 30 uniquely seeded simulation runs per experimental treatment.

Algorithm 1: CL: CCR and blocking-aware clustered tile mapping pseudo-code

```
   /* get pre-requisites                                          */
1  J_i^T = Topologically sorted tile-level taskset of J_i;
2  C_Tasks = getLongestPath(TJ_i);
3  {NH_T, NH_CT, NH_CH} = ccr_specific_hops(TJ_i);
4  TMtbl = runtime task mapping table;
   /* map each task iteratively                                   */
5  foreach τ_i^j ∈ J_i^T do
6      if τ_i^j ∈ C_Tasks then
           /* if task is part of clustered tasks                  */
7          if τ_i^j = τ_0^0 then
8              PE_i = getPELowBlocking(TMtbl, τ_0^0, PE_{i∈P});
9          else
10             sPE_N = getNeighbours(PE(τ_0^0), NH_CT );
11             PE_i = getPELowBlocking(TMtbl, τ_i^j, sPE_N);
12         end
13     else
           /* if task not part of clustered tasks                 */
14         if τ_i^j = τ_i^0 then
15             P_τ_i^j = parent task with largest dependency;
16             sPE_N = getNeighbours(PE(P_τ_i^j), NH_CH );
17             PE_i = getPELowBlocking(TMtbl, τ_i^j, sPE_N);
18         else
19             sPE_N = getNeighbours(PE(τ_i^0), NH_T );
20             PE_i = getPELowBlocking(TMtbl, τ_i^j, sPE_N);
21         end
22     end
23     Map τ_i^j to PE_i
24     update TMtbl{PE_i, τ_i^T}
25 end
```

5.1 Evaluated mapping techniques

We evaluate the proposed CL mapper against different variations of CL and two non-clustered, heuristic based mappers using PE utilisation as a metric. We hypothesise that

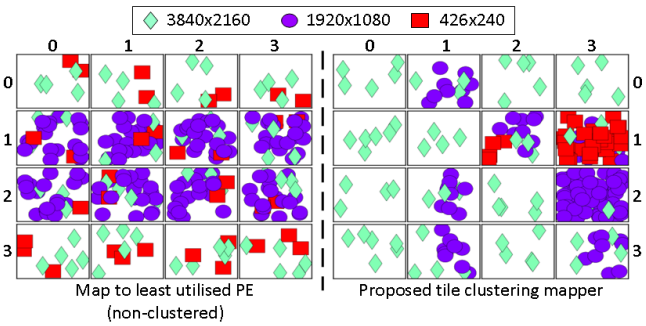

Figure 4: Task allocation example of least-utilised PE mapper (LU) vs Cluster based mapper. 4x4 NoC, 3 video streams, 1 GoP each.

the proposed clustered mapper can maintain reasonable job lateness levels whilst significantly reducing NoC data communication, when compared to a non-clustered approach.

CL: The clustered tile mapper as proposed in Section 4.

CL-FFI: Fast, $O(n)$ implementation of CL. PEs are sorted in increasing order of number of lowest priority tasks, only once at the start (unlike CL). Tasks are iteratively assigned to each PE starting from the first PE, cycling through the sorted list, wrapping back to the top of the list when the last PE has been reached.

CL-FO: clustering according to the TG fan-outs. TG nodes with a higher number of outgoing edges than the mean number of outgoing edges per node are clustered together. Similar to CL in other aspects.

CL-NoCCR: CCR based hop-distance is replaced by fixed hop distances, such that NH_T=N_T; NH_{CT}=0; NH_{CH}=N_T. Similar to CL in other aspects.

LU: worst-fit bin-packing based, *non-clustered* mapper using the PE utilisation as a metric. Tasks are sorted and mapped in topological order to the least utilised PE. Utilisations are calculated and searched at each iteration to find lowest utilised PE.

LU-FFI: Fast, $O(n)$ implementation of LU. PEs sorted according to increasing utilisation, only once at the start (unlike LU), then assigned sequentially.

5.2 Discussion of experimental results

Fig. 5 shows that the cluster based mapping approaches (CL, CL-FO, CL-FFI, CL-NoCCR) have a relatively lower communication overhead when compared to the non-clustered mappers (LU and LU-FFI). The improvement of CL over LU increases slightly as the workload level increasing (overall 8-10%). CL-NoCCR significantly outperform all the other mappers in reducing the data communication overhead as it maps a large amount of communicating tasks on to the same PE (irrespective of their CCR). In Fig. 6, the clustered mappers (especially the CL-NoCCR variant) show significantly higher mean idle periods compared to the non-clustered mapping schemes. CL shows 30-35% longer mean idle period durations compared to LU in high workloads. Longer idle periods can be exploited by dynamic power management techniques to put the PEs in low-power/temporary-sleep modes to save overall system power.

On average the proposed mapper (CL) shows much lower mapping overhead distribution range when compared to LU (Fig. 7). This is because the LU searches *all* PEs at every task mapping iteration to obtain the lowest utilised PE. Unlike LU, CL only searches a limited PE set (sPE_N in Algorithm 1) and this set would only be large for mapping very high resolution video streams (i.e. low CCR). However, CL shows a marginally higher worst-case execution time (i.e. higher outliers) than LU. The fast implementations (LU-FFI and CL-FFI) are an order of magnitude faster than the LU and CL mappers as they do not repeatedly search the PEs in each task mapping iteration. Even though faster, the CL-FFI mapper has limitations in accuracy because it does not take into account the tasks mapped in previous iterations of the mapping loop. However, surprisingly, its performance in terms of communication overhead is comparable to CL and only marginally worse in the mean PE idle period metric.

The cluster mappers show relatively poor results in terms of job lateness (Fig. 8), especially in the case of CL-NoCCR. All mappers show a similar inter-quartile range of the distributions, but the outliers represent the low-priority jobs that were significantly delayed by higher-priority task/flow interference. In the tests, the maximum/worst-case job lateness of CL was seen to be at most 5 seconds more than LU, even though average lateness results are comparable. However, these delays maybe acceptable for soft/non-real time application where video buffering can be employed. Furthermore, initial video playback delays of up to 30s have shown to be acceptable by most users [6]. Further investigation show, that the main cause for the increase in job lateness is the high memory communication latencies (Fig. 9). The latencies of memory traffic flows are shown in Fig. 9; these correspond to the memory read/write traffic before and after task execution. The non-clustered mappers (especially LU-FFI) show lower memory flow latencies due to the sparse distribution of tasks. Due to clustering of the tasks, flow congestion related to memory traffic (especially blocking due to high priority memory-reads) results in longer end-to-end job response times.

6 Conclusion

This work presented a runtime, low-communication overhead, clustering based HEVC tile to processing element mapping scheme (CL) which takes into account the workload CCR and task blocking behaviour. We illustrated how a frame-level task graph can polynomially grow due to tile partitioning and inter-prediction. The proposed task mapper can be used to reduce the inter-task data communication by clustering dependent tasks together on to the same or neighbouring PEs with respect to the job CCR. Results show significant reduction in data communication overhead and increased mean PE idle periods (resulting in reduced energy consumption) when compared to a greedy mapper that evenly distributes tasks based on PE utilisation. Comparisons against non-greedy mappers and fast first-fit-increasing mappers were made. Memory traffic congestion in the NoC limits the performance of the cluster-base mappers; however the proposed mapping techniques can still be highly beneficial if energy saving is of more importance than latency (e.g. video on-demand on multicore mobile phones). Future work, will involve investigating interconnect memory traffic aware mapping techniques.

Acknowledgement

We would like to thank the LSCITS program (EP/F501374/1), DreamCloud project (EU FP7-611411) and RheonMedia Ltd.

7 References

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

[2] B. Bross, M. Alvarez-Mesa, V. George, C. C. Chi, T. Mayer, B. Juurlink, and T. Schierl. HEVC real-time decoding. *SPIE XXXVI journal*, 8856, 2013.

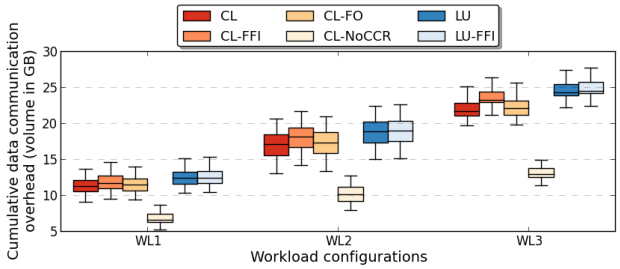

Figure 5: Distribution of cumulative data communication overhead volume for the evaluated mappers

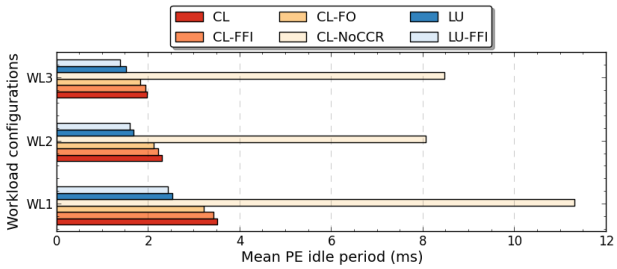

Figure 6: Mean node idle period for clustered and non-clustered mappers

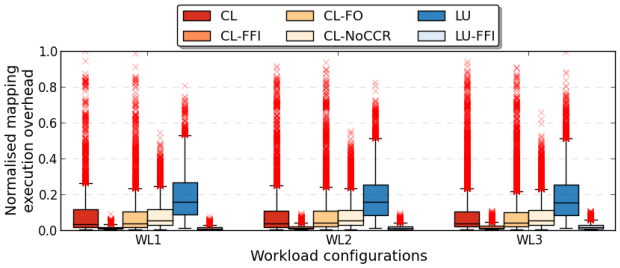

Figure 7: Normalised mapping execution overhead for the evaluated mappers

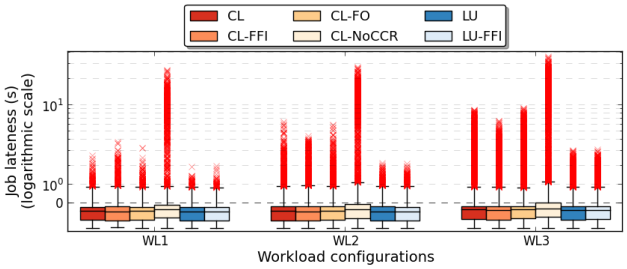

Figure 8: Job lateness distribution for the evaluated mappers

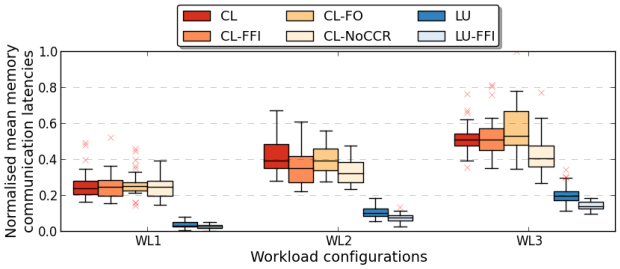

Figure 9: Normalised memory communication latencies for the evaluated mappers

[3] C. C. Chi, M. Alvarez-Mesa, B. Juurlink, G. Clare, F. Henry, S. Pateux, and T. Schierl. Parallel Scalability and Efficiency of HEVC Parallelization Approaches. *IEEE TCST*, 22:1827–1838, 2012.

[4] C. C. Chi, M. Alvarez-Mesa, J. Lucas, B. Juurlink, and T. Schierl. Parallel HEVC Decoding on Multi- and Many-core Architectures: A Power and Performance Analysis. *Journal of Signal Processing Systems*, 71:247–260, 2013.

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

[6] S. Egger, T. Hossfeld, R. Schatz, and M. Fiedler. Waiting times in quality of experience for web based services. In *IEEE QoMEX workshop*, 2012.

[7] G. Georgakarakos, L. Tsiopoulos, J. Lillius, J. Haldin, and U. Falk. Performance evaluation of parallel HEVC strategies. In *Euromicro PDP conf.*, 2015.

[8] L. S. Indrusiak, J. Harbin, and O. M. Dos Santos. Fast simulation of networks-on-chip with priority preemptive arbitration. *ACM TODAES*, 20:56:1–56:22, 2015.

[9] J. Jeong, J. Choi, and S. Ha. Parallelization and performance prediction for HEVC UHD real-time software decoding. In *IEEE ESTIMedia conf.*, 2014.

[10] S. Kaushik, A. Singh, and T. Srikanthan. Computation and communication aware run-time mapping for NoC-based MPSoC platforms. In *SOC conf.*, pages 185–190, 2011.

[11] M. U. K. Khan, M. Shafique, and J. Henkel. Software architecture of High Efficiency Video Coding for many-core systems with power-efficient workload balancing. In *DATE conf.*, 2014.

[12] S. Kim, H. Kim, J. Kim, J. Lee, and E. Seo. Empirical analysis of power management schemes for multi-core smartphones. In *ACM ICUIMC conf.*, 2013.

[13] K. Lee, S.-J. Lee, and H.-J. Yoo. Low-power network-on-chip for high-performance soc design. *IEEE TVLSIS journal*, 14, 2006.

[14] H. R. Mendis, N. C. Audsley, and L. S. Indrusiak. Task allocation for decoding multiple hard real-time video streams on homogeneous nocs. In *INDIN conf.*, 2015.

[15] M. Shafique, M. U. K. Khan, and J. Henkel. Power efficient and workload balanced tiling for parallelized high efficiency video coding. In *IEEE ICIP conf.*, 2014.

[16] A. K. Singh, M. Shafique, A. Kumar, and J. Henkel. Mapping on multi/many-core systems: survey of current and emerging trends. In *DAC conf.*, 2013.

[17] 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.*, 56:242–255, 2010.

[18] G. Sullivan, J. Ohm, W.-J. Han, and T. Wiegand. Overview of the High Efficiency Video Coding (HEVC) Standard. *IEEE TCSVT journal*, 22(12):1649–1668, Dec. 2012.

[19] B. Zatt, M. Porto, J. Scharcanski, and S. Bampi. GOP structure adaptive to the video content for efficient H. 264/AVC encoding. In *IEEE ICIP conf.*, 2010.