
Proceedings Paper:
Penny, Wagner, Palomino, Daniel, Porto, Marcelo et al. (2 more authors) (2019) Performance evaluation of HEVC RCL applications mapped onto NoC-based embedded platforms. In: Proceedings - 32nd Symposium on Integrated Circuits and Systems Design, SBCCI 2019. 32nd Symposium on Integrated Circuits and Systems Design, SBCCI 2019, 26-30 Aug 2019 Proceedings - 32nd Symposium on Integrated Circuits and Systems Design, SBCCI 2019 . Association for Computing Machinery, Inc , BRA .

https://doi.org/10.1145/3338852.3339868

Performance Evaluation of HEVC RCL Applications Mapped onto NoC-Based Embedded Platforms

Wagner Penny
ViTech/PPGC/UFPEL, IFSUL
RTS/University of York
York, United Kingdom
wi.penny@inf.ufpel.edu.br

Daniel Palomino
ViTech/PPGC/UFPEL
Pelotas, Brazil
dpalomino@inf.ufpel.edu.br

Marcelo Porto
ViTech/PPGC/UFPEL
Pelotas, Brazil
porto@inf.ufpel.edu.br

Bruno Zatt
ViTech/PPGC/UFPEL
Pelotas, Brazil
zatt@inf.ufpel.edu.br

Leandro Indrusiak
RTS/University of York
York, United Kingdom
lsi@cs.york.ac.uk

ABSTRACT

Today, several applications running into embedded systems have to fulfill soft or hard timing constraints. Video applications, like the modern High Efficiency Video Coding (HEVC), e.g., most often have soft real-time constraints. However, in specific scenarios, such as in robotic surgeries, the coupling of satellites and so on, harder timing constraints arise, becoming a huge challenge. Although the implementation of such applications in Networks-on-Chip (NoCs) being an alternative to reduce their algorithmic complexity and meet real-time constraints, a performance evaluation of the mapped NoC and the schedulability analysis for a given application are mandatory. In this work we make a performance evaluation of HEVC Residual Coding Loop (RCL) mapped onto a NoC-based embedded platform, considering the encoding of a single 1920x1080 pixels frame. A set of analysis exploring the combination of different NoC sizes and task mapping strategies were performed, showing for the typical and upper-bound workload cases scenarios when the application is schedulable and meets the real-time constraints.

CCS CONCEPTS

• **Information systems** → *Multimedia information systems*; • **Networks** → *Network on chip*; • **Computer systems organization** → *Real-time systems*; Embedded systems.

KEYWORDS

NoC, real-time systems, embedded systems, HEVC

ACM Reference Format:
Wagner Penny, Daniel Palomino, Marcelo Porto, Bruno Zatt, and Leandro Indrusiak. 2019. Performance Evaluation of HEVC RCL Applications Mapped onto NoC-Based Embedded Platforms. In *32nd Symposium on Integrated Circuits and Systems Design (SBCCI '19), August 26–30, 2019, Sao Paulo, Brazil.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3338852.3339868

SBCCI '19, August 26–30, 2019, Sao Paulo, Brazil

ACM ISBN 978-1-4503-6844-5/19/08…$15.00
https://doi.org/10.1145/3338852.3339868

1 INTRODUCTION

Nowadays, digital videos have spread at a huge rate due to the ever-increasing amount of handheld digital devices (according to [20], today, more than 70% of all video visualizations on internet happen at these devices), and the fast popularization of streaming video services, like Youtube, Prime, and Netflix. The crescent demand for higher resolutions and frame rates lead to the development of modern video coding standards, capable of greater compression ratios. Today's state-of-the-art video coding standard is the High Efficiency Video Coding (HEVC), released in 2013 by the Joint Collaborative Team on Video Coding (JCT-VC) [18] to replace its predecessor H.264/AVC (Advanced Video Coding), keeping a similar objective video quality, whereas enhancing the compression ratios almost twice when compared with H.264 [19].

The main problem with HEVC, likewise other modern video coding standards, is the great computational effort required due to the enhanced tools introduced/improved when compared with previous standards. Depending on encoding configurations, HEVC can be up to 500% more complex than its predecessor [14]. E.g., the Residual Coding Loop (RCL) is one of the most time-consuming steps within HEVC, executed several times during the encoding process and responsible for up to 18% of total encoding time [4]. Besides complexity constraints, real-time constraints also arise as a challenge in the context of video coding. Broadly speaking, video applications have soft real-time constraints. E.g., a full high-definition (FHD)(1920x1080 pixels) video requires a minimum frame rate of 30 fps [19] to give a continuous motion sensation, i.e., each frame can spend maximum time of 33 ms to be processed, otherwise, the frame rate won't be reached, missing the deadline. However, in specific video applications, such as in robotic surgeries and coupling of satellites, these constraints are harder and a performance analysis aiming the fulfillment of the deadlines is mandatory.

Indeed, video coding has been seen as complex and sophisticated workloads, requiring efficient platform resources management mechanisms besides optimized scheduling of tasks, as a way to optimize performance and meet deadlines. To address such constraints, many approaches consider the mapping of applications onto Systems-on-Chip (SoCs), with multiple processing units, interconnected with Networks-on-Chip (NoCs), which can interconnect tens to hundreds of processing cores by an on-chip packet-switching network that allows data to be transferred between the

local memory of each core and from/to external memory [8]; compounding a complex Multiprocessor System-on-Chip (MPSoC), capable of reducing the computational time and meeting the time constraints of such complex applications [17].

Nevertheless, the mapping of an application, described by parameterized task graphs, onto a given NoC, is seen as a key research problem. In fact, the general NoC's cores mapping problem is NP-complete and its solutions are only allowed based on efficient heuristics [7][16]. Furthermore, the performance of the NoC interconnections also arises as a critical issue regarding time constraints. The choice of the best NoC configuration/application mapping, with a NoC schedulability evaluation, is not a trivial issue. Solutions found through simulation, addressing specific scenarios, are widely used by industry and academia. However, such an approach presents two main limitations: the execution time, which can be prohibitively large; and the pruning of possible scenarios, since only specific scenarios are simulated [8]. Although the simulation analysis could be used in specific situations, analytical methods can be used to deal with its mentioned problems, providing a schedulability evaluation of a NoC-based multicore embedded system, verifying whether or not the system can fulfill all the timing constraints, improving the design space exploration.

In this work, we propose a performance evaluation of the HEVC RCL application mapped onto NoC-based multicore embedded platforms. The application workload is modeled based in Sporadic Task-Chain, mapped onto different NoC sizes. The case study considers the encoding of an FHD frame in the typical and upper-bound workload case scenarios. We propose different task mappings and platform topologies, showing when the tasks and flows are schedulable or not, for each case study.

2 LITERATURE BACKGROUND

2.1 HEVC

The HEVC encoder [18] follows a hybrid coding model, based on the encoding of residues (the difference between the original and the predicted frame). This model is composed of the prediction steps (intra and inter), the transform (T) and quantization (Q) steps, and the entropy coding. As a way to guarantee the same references at encoder and decoder sides, the encoder also contains steps from decoder, like the motion compensation (MC), inverse transform (IT) and inverse quantization (IQ). The set of steps T, Q, IQ, and IT is called Residual Coding Loop (RCL).

In the HEVC, during the encoding process, many decisions must be made. Each frame is divided into basic structures called Coding Tree Units (CTU), a quadtree starting with the size of 64x64 in the root, recursively divided assuming sizes of 32x32 or 16x16 (smallest CTU size). The leaf nodes of a CTU-rooted are called Coding Units (CUs), always-squared shaped blocks presenting the basic information about the blocks being coded, capable of assuming a minimum size of 8x8 (for inter-prediction) or 4x4 (for intra-prediction). See in Fig. 1 a frame recursively divided in CTUs and CUs.

The CUs can be divided into Prediction Units (PUs), during the prediction steps, which inform about prediction modes (intra or inter), and into Transform Units (TUs), during the RCL and after the calculation of the residues (see in Fig. 1 the example of a residual quadtree (RQT) structure starting with a 32x32 root). Each TU

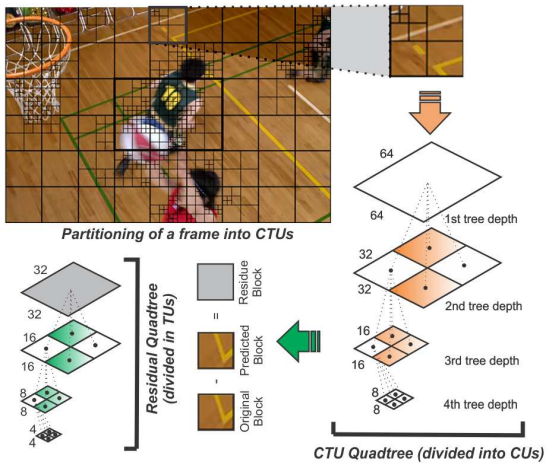

Figure 1: Partitioning of a frame into structured quadtrees (adapted from [21]).

contains information about the transformed and quantized blocks, always squared varying from 32x32 down to 4x4 sizes). Each coding mode (intra or inter), as well as the best partitioning, is determined in HEVC by a Mode Decision (MD) unit, based on the RD (Rate Distortion) cost, which is a trade-off between distortion (objective video quality) and bit-rate, hence, the encoder must test an exhaustive amount of encoding possibilities to make the best decision.

During the HEVC RCL, the direct transform is applied to the residues (typically a Discrete Cosine Transform - DCT), converting the information from space to frequency domain. Direct quantization reduces the magnitude of the transformed residuals, leading most of the quantized transformed equals to zero, which improves the entropy coding efficiency. The inverse quantization and inverse transform are used to perform the inverse operations in order to reconstruct the block, used as a reference in the encoder. The RCL must be executed several times during the encoding of a single frame and has been being the main goal of several works in the literature due to its complexity.

2.2 Performance Analysis for NoCs

Networks-on-chip (NoCs) are common architectural templates for processors with dozens, hundreds or even thousands of cores. In Fig. 2 we show a simplified example of a simple 3x3 NoC architecture. All the nodes are interconnected, and each one has a core *c*, linked to a local cache, which stores local information, and a router *r*, which routes the data packets towards the destinations (it can be another core, the off-chip memory, etc.) [8]. The communication between the cores and the router is made by two unidirectional links (one from *c* to *r* and other from *r* to *c*). In this work, we have applied the widely used 2D-mesh topology [2][8][9], considering wormhole NoCs with priority-preemptive arbitration, widely studied in the literature due to their ability to provide resources for hard real-time guarantees [9][16].

In a wormhole switching network, the data is encapsulated into a packet format, where each packet is divided into a number of

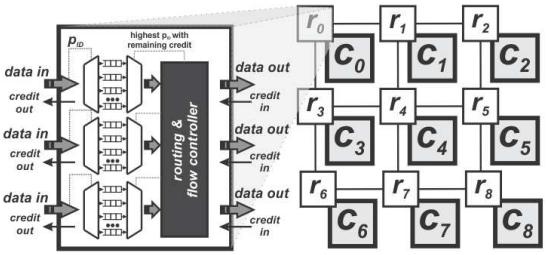

Figure 2: Mesh 2D NoC 3x3 architecture with a router detailing the priority-driven virtual channels (adapted from [16]).

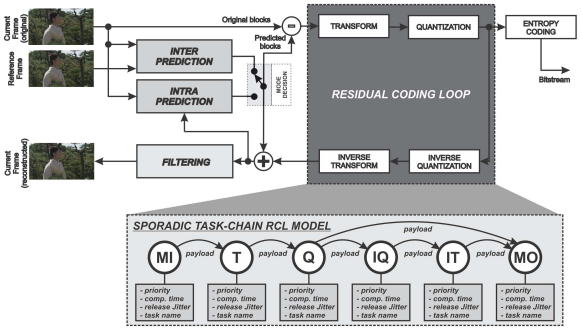

Figure 3: HEVC Simplified Block Diagram with RCL Modeled as Sporadic Task-Chains.

fixed size flits (data words) [16]. The router is based on priority-preemptive virtual channels (VCs) as a way to guarantee more predictability. Each packet has a different priority assigned, thus it is possible to packets with higher priority preempt the ones with lower priorities. See in Fig. 2 that in each input port there is a FIFO buffer storing the incoming flits of packets arriving through different VCs. The routing and flow controller decides the correct output port for each packet, according to its destination. A credit-based approach, first stated in [1], was applied, ensuring the forwarding of the data only when there is sufficient space in the VC of the next router.

In order to determine whether application tasks being executed and communicating over a specific NoC can fulfill the required timing constraints, it is necessary to perform a schedulability analysis. A system is schedulable *iff* all its tasks and communicating flows meet the deadlines. In this paper, we have applied the end-to-end schedulability analysis presented in [8], which ensures all tasks executing over a processing core and their respective packets flow over a NoC will meet their deadlines even in the worst-case scenario. For the analysis of packet flows, we incorporated the improvements from [9], which consider the impact of finite buffers, flow control backpressure, and multi-point progressive blocking in priority-preemptive NoCs. For a comprehensive review on NoC performance analysis, see [10].

3 APPLICATION WORKLOAD MODELING

A co-design flow of embedded systems, which consists of a set of steps starting with the specifications of requirements and ending with the hardware/software integration into silicone chips [6], is of utmost importance nowadays. The co-design modeling phase allows designers to explore the design space, making the best architectural choices in order to meet user requirements, platform and application constraints during the development phase [17].

In this work, we follow modeling based on Sporadic Task Model [8], where an application can be modeled as a taskset $\Gamma = \{\tau_1, \tau_2, ..., \tau_n\}$, where each task τ_i is a 6-tuple $\tau_i = \{C_i, T_i, D_i, J_i, P_i, \phi_i\}$, which are respectively the worst-case computation time, the period (minimum inter-release time interval), the deadline, the release jitter (time between the request of a task and it starts to be processed), the priority, and a message, defined as a 3-tuple $\phi_i = \{\tau_d, Z_i, K_i\}$, representing the destination task, the message's size and the maximum release jitter (total time the packet takes to reach the destination, including preemption and interference). A sporadic task-chain $X = \{\tau_1, \tau_2, ..., \tau_x\}$ is an ordered subset of Γ, where a task sends a message to a subsequent task in X. In this case, all tasks within X have the same period and deadline. The final task of every task-chain must be the empty set Ø.

In this work, the evaluated application is the HEVC RCL. See in Fig. 3 a simplified block diagram of the HEVC encoder and the RCL modeled as a sporadic task-chain model. Each step of the RCL is considered as a task: direct transform (T), direct quantization (Q), inverse quantization (IQ), and inverse transform (IT). In addition, the first and last tasks of the chain are memory-related tasks: MI (Memory Input) and MO (Memory Output), respectively. The first one represents the modeling of a block being read from off-chip memory until reaches T. The final task is an empty set, treated as a sink (i.e., a place holder with computation time equals to zero), where the necessary information is stored for the forward steps of the encoder (outside the scope of our evaluation). The payload, given in flits, are packets containing information about the data being processed, varying the size according to the TU size (4x4, 8x8, 16x16 or 32x32), NoC flit size (depends on NoC topology), bit word size, and adopted subsampling (relation between number of luminance and chrominance samples, in this work we have adopted the widely used 4:2:0).

As a way to simplify the modeling, we have made some assumptions. We considered that all blocks of a single frame can be independently processed, allowing the exploration of maximum parallelism. In order to do that, we must consider only the processing of inter-predicted blocks in the modeled RCL, since intra-predicted blocks have strong data dependency, which narrows the full parallelism exploration. Since the majority of blocks in a real encoding are inter-predicted such assumption does not incur in loss of importance of the proposed evaluation. Besides that, we also have considered that the inter-prediction is giving to RCL always the final decision in terms of prediction modes and CU size, i.e., only the evaluation of the best TU is carried out (see that in this case we are not caring about which kind of PU decision was made, we just need to know what was the selected CU sizes, which will be the roots of the RQTs in the RCL).

To model the HEVC RCL, we have first analyzed the computation time of each RCL step in HEVC. We encoded 64 frames of five video sequences (*BasketballDrive*, *BQTerrace*, *Cactus*, *Kimono*, and *ParkScene*) from class B (1920x1080 pixels), considering four Quantization Parameters (QPs - 22, 27, 32, and 37), according to the

Table 1: RCL Task's Computation Time and Payload

TU Size	*MI*	*T*	*Q*	*IQ*	*IT*	*MO*	*Payload*
4x4	42	96	535	99	122	0	26
8x8	81	270	2072	159	375	0	98
16x16	223	1444	9889	394	1465	0	386
32x32	716	9365	42017	1249	9000	0	1538

HEVC Common Test Conditions (CTCs) [3], using the reference software HM 16.18 (HEVC Test Model) [5], running isolated into an *i7* core with a fixed frequency of 3.0 GHz. The computation times of RCL tasks were obtained for each TU size (4x4, 8x8, 16x16, and 32x32) and are presented in Table 1. Note that we found the average computation time, obtaining a Gaussian distribution with the measured times. The computation times applied in the modeling (showed in Table 1 in nanoseconds) were the upper quartiles of the distribution since we want to model worst-case scenarios. In Table 1 we also show the payload for each TU size, considering a packet header of two flits. Furthermore, the period of the tasks into a chain is the same and equal to the inverse of frame rate (e.g., we considered 30 fps in this work, thus the period is 33 ms). The deadline for all tasks in the chain is equal to the period.

Each task was conceived with a different priority and must have given different names, ensuring that each task node is different from the others. The developed model is modular, a task-chain for each TU size can be created, and further replicated and instantiated to compound any sort of RQT distribution (e.g., the evaluation of a block 16x16 must use one chain 16x16 to analyze the 16x16 TU, four chains 8x8 to analyze the four 8x8 TUs compounding the 16x16 TU, and so on). This approach is better detailed in the next section.

4 PERFORMANCE EVALUATION

The task-chains were conceived in a modular way (the explanation of a single RCL task-chain modelling was done in the previous section). The modules are presented in Fig. 4. We follow a cluster-based approach to instantiate any number of task-chains as wished. To perform the RCL of a 4x4 TU a simple task-chain is necessary, called *Cluster 4x4*. To analyze an 8x8 TU is necessary a task-chain for the 8x8 TU plus four task-chains to 4x4 TUs (which compound the 8x8). This structure is called *Cluster 8x8*. When analyzing a 16x16 TU, a task-chain for 16x16 is necessary, besides four *Cluster 8x8* (which already contain resources to evaluate the 4x4 TUs). Such a structure is called *Cluster 16x16*. Finally, to perform the evaluation of a 32x32 TU, *Cluster 32x32* is built by using a sporadic task-chain for 32x32 TU, in addition to four *Cluster 16x16*. As explained before, 64x64 TUs are not allowed, but the CUs can assume such size. In this case, the block must be split into four 32x32 TUs, requiring four *Cluster 32x32*. See that in order to cover all possibilities of TU splitting (starting from a given CU size), the divisions were also considered to the maximum limit, always going to the lowest level of the RQT (leaves 4x4).

The performance evaluation was based on two main case studies: the building of an upper-bound workload model and the building of a typical workload model. The upper-bound workload model considers a single FHD frame, equally divided into CTUs 64x64.

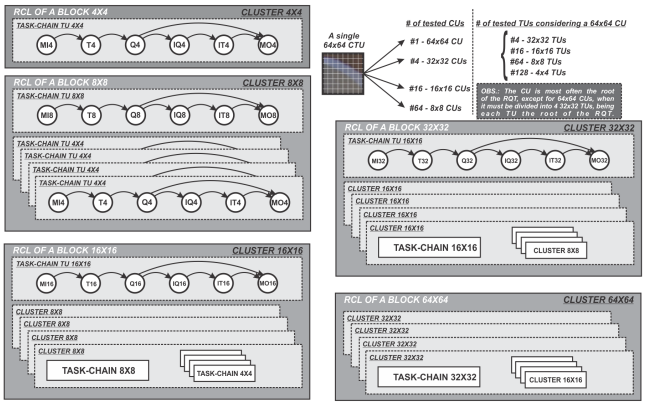

Figure 4: Modular Distribution of HEVC RCL Modeled as Sporadic Task-Chains.

Such a pessimist scenario consists of the assumption that the inter-prediction selected only 64x64 CUs. These CUs must be partitioned into 32x32 TUs and the RQT can reach the smallest value of 4x4 TUs, resulting in the processing of 506 64x64 blocks, recursively divided until 4x4 TU sizes. Note that, regarding processing effort, no other situation is worse than this one. A schedulability test for this workload, ensuring the meeting of timing constraints, resulting in a schedulable system, automatically means that any other workload scenario will be also schedulable. However, such a pessimist scenario is not real, in a practical way would never happen.

A typical workload scenario, also considering an FHD frame, must take into account the occurrence of skips and the average CU size selection. A skip occurs when the predicted block is equal to the original. In this case, the generated residue will be equal to zero. Therefore, the RCL can be skipped since the resulting transformed and quantized coefficients will also be equal to zero. This way, all effort demanded to process the RCL of this block will be avoided. The average skip occurrence in FHD videos is about 73.5% [11]. The percentual of selection of each CU in HEVC was obtained using results presented in [15] combined to in-house experiments using the reference software: 64x64 (5.86%), 32x32 (19.49%), 16x16 (35.21%), and 8x8 (39.44%). In terms of processing effort, the combining of skip occurrence and the percentual of CU selection results in a scenario (in terms of required processing), equivalent to process 136 64x64 blocks, recursively divided until 4x4 TU sizes.

The task mapping step is a critical part of the development. This process defines where each task of the application is mapped onto which processing core, i.e., where each task is executed. A first approach, poorly efficient, which can be applied is a Random Task Mapping. In this mapping, the tasks (even from the same task-chain), are mapped in aleatory cores. Besides being straightforward to implement, the distribution of the tasks may imply in high-level of interference among data flows.

A more efficient mapping must consider all tasks from a chain onto the same processing core, except its memory-related tasks. These tasks need to be mapped onto cores that directly access the off-chip memory through DMAs (Direct Memory Accesses). Based on such information, we have developed two approaches to map the

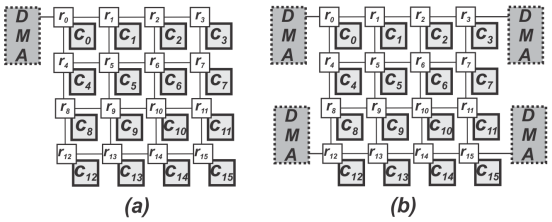

Figure 5: HEVC RCL Workload Mapping Strategies onto NoCs 4x4 (a) Heuristic 0 and (b) Heuristic 1.

workload onto NoCs, presented in Fig. 5. In Fig. 5 (a) is presented the *Mapping Heuristic 0* (MH0) and in Fig. 5 (b) is presented the *Mapping Heuristic 1* (MH1). MH0 consists of mapping all memory-related tasks in core 0, whose router is connected to the DMA. In this approach, there is an intensive traffic flow from the other cores towards core 0. On the other hand, MH1 tries to decrease the data flow intensity, strategically allocating DMAs in all NoC corners. In such mapping, the traffic flows are kept into smaller quadrants from the NoC, flowing from the other cores to the corners. For MH1 onto NoCs 2x2 and 3x3 are considered only two DMAs, in opposite corners (routers 0 and 3, 0 and 8, respectively). Furthermore, a third heuristic that is not showed in the figure, called *Mapping Heuristic 2* (MH2), is also proposed. MH2 is a non-realistic scenario, applied just to verify the upper limit of tasks that can be mapped into a single core. In this case, it is considered that each router in the NoC is able to connect directly the DMA. In such a specific scenario there are no flows running inside the NoC, only from the core to router, and from the router to off-chip memory.

Schedulability analysis for the proposed workload scenarios, combining different NoC sizes and mapping approaches were performed and presented in the next section.

5 EXPERIMENTAL SETUP AND RELATED WORKS

In order to perform the schedulability evaluation of the HEVC RCL modeling, an experimental setup proposing 24 different scenarios is presented in Table 2. Each experiment has an identification number (*Exp. ID*), mapped onto NoCs 2x2, 3x3, 4x4, and 5x5. The considered workloads are the ones mentioned in section 4 (upper-bound and typical). Note that upper-bound was analyzed only for 4x4 and 5x5 NoCs due to the fact that, according to the great number of tasks, there are no available cores to map all tasks from this workload onto a NoC smaller than 4x4. The mapping strategies are four: a random mapping (but keeping memory-related tasks onto core 0), MH0, MH1, and MH2. We also considered two types of core distribution: a maximum core utilization (MCU), where we map a number of tasks near to the supported limit of the core (which can imply in sub utilization of cores sometimes, or even cores without mapped tasks), and an uniform core utilization (with equal task distribution among all the cores of the NoC).

The platform follows the architecture stated in section 2, with homogeneous cores running priority-preemptive task schedulers, 2D-mesh NoC interconnected with XY squared dimension routing, distributed memory, 8 virtual channels with priority-preemptive

Table 2: Experimental Setup

Exp. ID	*NoC Size*	*Workload*	*Mapping*	*Core Util.*
01	2x2	typical	MH0	uniform
02	2x2	typical	MH1	uniform
03	2x2	typical	random	none
04	3x3	typical	MH0	uniform
05	3x3	typical	MH0	MCU
06	3x3	typical	MH1	uniform
07	3x3	typical	MH1	MCU
08	3x3	typical	random	none
09	4x4	typical	MH0	uniform
10	4x4	typical	MH0	MCU
11	4x4	typical	MH1	uniform
12	4x4	typical	MH1	MCU
13	4x4	typical	ramdom	none
14	4x4	upper-bound	MH0	uniform
15	4x4	upper-bound	MH0	MCU
16	4x4	upper-bound	MH1	uniform
17	4x4	upper-bound	MH1	MCU
18	4x4	upper-bound	random	none
19	4x4	upper-bound	MH2	uniform
20	5x5	upper-bound	MH0	uniform
21	5x5	upper-bound	MH0	MCU
22	5x5	upper-bound	MH1	uniform
23	5x5	upper-bound	MH1	MCU
24	5x5	upper-bound	random	none

link arbitration. The schedulability analysis is based on [8] and [9], and the results are presented in Fig. 6.

In Fig. 6 we present the response times regarding all flows within the proposed 24 experiments. The average response time is plotted in orange triangles, the worst-case response time is represented as blue crosses, and the deadline is showed as a gray dashed line. Response times greater than the deadline are presented above its dashed line and their values, considerable high, are not represented in the graph for simplicity. Note that even though many experiments have been presented worst-case time responses smaller than the deadline, the whole system may not be fully schedulable since the time response of the tasks also must be taken into account.

In the considered experiments, the tasks were most often schedulable since we have previously analyzed the capacity of each core. Only on the random mapping of experiments 02 and 18 some tasks

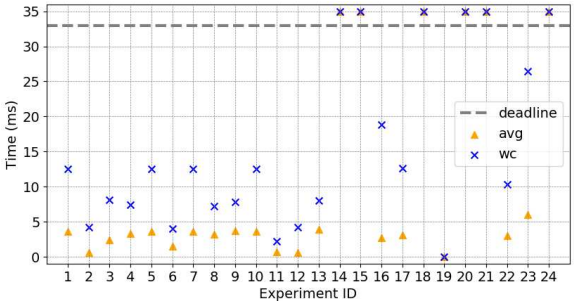

Figure 6: Flow's Response Times of 24 Proposed Experiments.

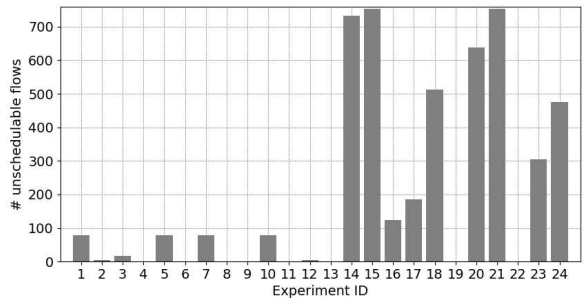

Figure 7: Number of Unschedulable Flows within the 24 Proposed Experiments.

were unschedulable. On the other hand, the flows depend on NoC resources and the path they need to cross until reaching the destination. We have also analyzed the number of unschedulable flows within the experiments and they are presented in Fig. 7, thus verifying which experiments give a fully schedulable system.

When analyzing Fig. 7, we can see that only experiments 04, 06, 08, 09, 11, 13, 19, and 22 have zero unschedulable flows. Among them, experiments from 04 to 13 consider a typical workload. Note that the system is fully schedulable with a 3x3 NoC, connected to off-chip memory using only one DMA and with uniform task core distribution (experiment 04). Although other configurations also being schedulable, they spend more resources, which makes experiment 04 the best configuration when mapping a typical workload. Experiment 19 is fully schedulable but represents a non-realistic scenario, where all routers can connect to DMA. Finally, regarding upper-bound workload, the only configuration that is fully schedulable is experiment 22. Therefore, the system will be fully schedulable when the HEVC RCL is mapped onto a 5x5 NoC, connected with four DMAs and uniform task core distribution. In this case, we can guarantee that the system will be always schedulable since the time constraints were met even for the worst case.

Many works in the literature aim the modeling of HEVC in higher abstraction levels. In [17], besides modeling the HEVC video decoder with Synchronous Data Flow in order to solve problems of placing and scheduling the application onto an embedded platform, they consider only the decoder and apply simulation methods to analyze the system. Works [13] and [12] also propose HEVC modeling in higher levels of abstraction. In [13] is developed a synthetic workload generation of broadcast-related HEVC stream decoding in constrained systems. Work [12] proposes a dynamic and static task allocation for hard real-time video stream decoding on NoCs. Although modeling the HEVC in higher levels of abstraction, these works consider only the decoder and they do not model specific steps of the encoder, like RCL. To the best of the author's knowledge, it is the first work in the literature proposing the modeling of the HEVC RCL onto embedded platforms based on NoCs, performing a schedulability evaluation to meet the real-time constraints of the application.

6 CONCLUSIONS

In this work, we presented a performance evaluation of HEVC RCL applications mapped onto NoC-based embedded platforms. The application was modeled based on Sporadic Task Model, following a cluster-based approach. Two workloads were generated (upper-bound and typical), regarding characteristics of the HEVC encoder. A set of 24 experiments was built, combining workloads, NoC configurations, and different task mapping strategies. For each configuration, a schedulability analysis was applied in order to verify if the system meets the real-time constraints posed by the application. We found out that, for typical workloads, a 3x3 NoC with only one DMA and uniform core distribution is schedulable. On the other hand, for upper-bound workload scenarios, the system is only schedulable using 5x5 NoCs, with four DMAs and uniform core distribution.

ACKNOWLEDGMENTS

This study was financed in part by the Coordenação de Aperfeiçoamento de Pessoal de Nível Superior - Brasil (CAPES) – Finance code 001 (bolsista da Capes/PDSE/processo nº88881.188774/2018-01), the CNPq, and the FAPERGS.

REFERENCES

[1] T. Bjerregaard and S. Mahadevan. 2006. A Survey of Research and Practices of Network-on-chip. *ACM Comput. Surv.* 38, 1, Article 1 (June 2006).
[2] E. Bolotin, I. Cidon, R. Ginosar, and A. Kolodny. 2004. QNoC: QoS architecture and design process for network on chip. *JSA* 50, 2 (2004), 105 – 128. Special issue on networks on chip.
[3] F. Bossen. 2011. Common test conditions and software reference configurations. JCTVC-L1100, Geneva.
[4] F. Bossen, B. Bross, K. Suhring, and D. Flynn. 2012. HEVC Complexity and Implementation Analysis. *IEEE TCSVT* 22, 12 (Dec 2012), 1685–1696.
[5] J. Boyce. 2014. HM16: High Efficiency Video Coding Test Model (HM16) Encoder Description. JCTVC-R1002, Sapporo.
[6] P. Ehrlich and S. Radke. 2013. Energy-aware software development for embedded systems in HW/SW co-design. In *2013 IEEE DDECS*. 232–235.
[7] W. Hu, C. Du, L. Yan, and C. Tianzhou. 2009. A fast algorithm for energy-aware mapping of cores onto WK-recursive NoC under performance constraints. In *2009 HiPC*. 359–367.
[8] L. Indrusiak. 2014. End-to-end schedulability tests for multiprocessor embedded systems based on networks-on-chip with priority-preemptive arbitration. *JSA* 60, 7 (2014), 553 – 561.
[9] L. Indrusiak, A. Burns, and B. Nikolić. 2018. Buffer-aware bounds to multi-point progressive blocking in priority-preemptive NoCs. In *2018 DATE*. 219–224.
[10] A. Kiasari, A. Jantsch, and Z. Lu. 2013. Mathematical Formalisms for Performance Evaluation of Networks-on-chip. *ACM Comput. Surv.* 45, 3, Article 38 (July 2013).
[11] J. Kim, J. Yang, K. Won, and B. Jeon. 2012. Early determination of mode decision for HEVC. In *2012 PCS*. 449–452.
[12] H. Mendis, N. Audsley, and L. Indrusiak. 2017. Dynamic and Static Task Allocation for Hard Real-Time Video Stream Decoding on NoCs. *Leibniz Transactions on Embedded Systems* 4, 2 (2017), 01–1–01:25.
[13] H. Mendis and L. Indrusiak. 2016. Synthetic Workload Generation of Broadcast Related HEVC Stream Decoding for Resource Constrained Systems. In *2016 ICETE*. SCITEPRESS - Science and Technology Publications, Lda, Portugal, 52–64.
[14] L. Mengzhe, J. Xiuhua, and L. Xiaohua. 2015. Analysis of H.265/HEVC, H.264 and VP9 coding efficiency based on video content complexity. In *IEEE ICCC*.
[15] W. Penny, G. Paim, M. Porto, L. Agostini, and B. Zatt. 2015. Real-Time Architecture for HEVC Motion Compensation Sample Interpolator for UHD Videos. In *28th SBCCI*. ACM, New York, NY, USA, Article 12, 6 pages.
[16] Z. Shi and A. Burns. 2008. Real-Time Communication Analysis for On-Chip Networks with Wormhole Switching. In *2008 ACM/IEEE NOCS*. 161–170.
[17] H. Smei, A. Jemai, and K. Smiri. 2017. Performance Estimation of HEVC/h.265 Decoder in a Co-Design Flow with SADF-FSM Graphs. *IJCNS* 10 (2017), 261 – 281.
[18] G. Sullivan, J. Ohm, W. Han, and T. Wiegand. 2012. Overview of the High Efficiency Video Coding (HEVC) Standard. *IEEE TCSVT* 22, 12 (2012), 1649–1668.
[19] J. Vanne, M. Viitanen, T. Hamalainen, and A. Hallapuro. 2012. Comparative Rate-Distortion-Complexity Analysis of HEVC and AVC Video Codecs. *IEEE TCSVT* 22, 12 (2012), 1885–1898.
[20] Youtube. 2019. Youtube Statistics. Retrieved Jan 26, 2019 from https://www.youtube.com/intl/pt-BR/yt/about/press/
[21] C. Zhou, F. Zhou, and Y. Chen. 2013. Spatio-temporal correlation-based fast coding unit depth decision for high efficiency video coding. *JEI* 22, 4 (2013).