Journal of Signal Processing Systems (2023) 95:895–907
https://doi.org/10.1007/s11265-022-01819-7

# OpenVVC Decoder Parameterized and Interfaced Synchronous Dataflow (PiSDF) Model: Tile Based Parallelism

**Naouel Haggui[1,2] · Wassim Hamidouche[1] · Fatma Belghith[2] · Nouri Masmoudi[2] · Jean-François Nezan[1]**

Received: 5 May 2022 / Revised: 9 September 2022 / Accepted: 3 October 2022 / Published online: 14 October 2022


**Abstract**
The emergence of the new video coding standard, Versatile Video Coding (VVC), has resulted in a 40-50% coding gain over its predecessor HEVC for the same visual quality. However, this is accompanied by a sharp increase in computational complexity. The emergence of the VVC standard and the increase in video resolution have exceeded the capacity of single-core architectures. This fact has led researchers to use multicore architectures for the implementation of video standards and to use the parallelism of these architectures for real-time applications. With the strong growth in both areas, video coding and multicore architecture, there is a great need for a design methodology that facilitates the exploration of heterogeneous multicore architectures, which automatically generates optimized code for these architectures in order to reduce time to market. In this context, this paper aims to use the methodology based on data flow modeling associated with the PREESM software. This paper shows how the software has been used to model a complete standard VVC video decoder using Parameterized and Interfaced Synchronous Dataflow (PiSDF) model. The proposed model takes advantage of the parallelism strategies of the OpenVVC decoder and in particular the tile-based parallelism. Experimental results show that the speed of the VVC decoder in PiSDF is slightly higher than the OpenVVC decoder handwritten in C/C++ languages, by up to 11% speedup on a 24-core processor. Thus, the proposed decoder outperforms the state-of-the-art dataflow decoders based on the RVC-CAL model.

**Keywords** OpenVVC · Dataflow modeling · Tiles · VVC · PiSDF

## 1 Introduction

According to Cisco statistics, video content now accounts for about 82% of the global Internet traffic [1]. This can be explained by the massive use of social media, YouTube, video streaming sites such as Netflix, and video conferencing that has been used extensively since the appearance of Covid-19 for remote work. Due to this fact, the storage requirement of the video traffic and its energy footprint are increasing. Thus, raising the need for a more efficient codec than High Efficiency Video Coding (HEVC) [2]. In 2020, a new video coding standard called Versatile Video Coding (VVC) [3] has been released. This latter offers between 40 and 50% of bitrate reduction compared to its predecessor HEVC thanks to several enhancement methodologies and new tools including Multiple Transform Selection (MTS), Adaptive Loop Filter (ALF), new intra/inter prediction tools, etc. Those tools improve the efficiency of the encoding process, but come with an increase in computational complexity. In order to overcome this problem, a lot of research has been carried out to reduce the computational complexity on both encoder and decoder sides. Some of the optimizations in the VVC standards were based on the use of machine learning, especially in the encoder parts (intra/inter prediction and partitioning) [4], others explored the parallelism between tasks. In fact, since the advent of Moore's

✉ Naouel Haggui
nawel.hagui@enis.tn

Wassim Hamidouche
Wassim.Hamidouche@insa-rennes.fr

Fatma Belghith
fatmabelghithenis@gmail.com

Nouri Masmoudi
masmoudi123@gmail.com

Jean-François Nezan
Jean-Francois.Nezan@insa-rennes.fr

[1] Univ Rennes, INSA Rennes, CNRS, IETR - UMR 6164, Rennes, 20 Avenue des Buttes de Coesmes, Rennes 35700, France

[2] Electronics and Information Technology Laboratory (LETI) of Sfax, Road of Soukra, Sfax 3038, Tunisia

Law, the industry has been moving towards the creation of multicore hardware using parallelism strategies to cope with the increase in computational complexity of modern applications.

In the VVC standard, there are three main levels of parallelism: the data level, the frame level, and the high-level parallelism including both tile and slice features. The last few years have seen the emergence of new software decoders compliant with the VVC standard, like the Fraunhofer Heinrich Hertz Institute's decoder called VVdec [5] and the open software decoder called OpenVVC [6]. These decoders are developed to offer real time decoding capabilities over different platforms. The OpenVVC decoder use the three levels of parallelism resulting in a decoding performance that overcome both the VTM MPEG reference software and the VVdec decoder.

Implementing complex signal or image processing applications, especially video coding applications, on embedded multicore architectures is challenging, creating the need for design frameworks and methodologies to accelerate the design process. In this context, Dataflow modeling has been used extensively in the generation of efficient multicore implementations for embedded systems. Dataflow is a powerful model to explore data dependencies and to reveal existing parallelism levels. A single dataflow model gives the possibility to target several heterogeneous architectures (multicore, many-core, and Programmable Logic architectures), using efficiently the available parallelism on the hardware and optimizing the memory allocations.

In this context, this paper aims at using the Parallel and Real-time Embedded Executives Scheduling Method (PREESM) [7] tool to create a Parameterized and Interfaced Synchronous Dataflow (PiSDF) [8] model of the VVC decoder in order to automatically generate a multicore algorithm optimizing the execution of tiles on multicore architectures. This work investigates the performance of the parameterized dataflow model (PiSDF) versus the state-of-the-art dynamic dataflow models RVC-CAL. As a matter of fact, RVC-CAL models showed inferior performance compared to C/C++ decoders, which could be explained by either huge time spent on scheduling at runtime or by the data movement overhead. To the best of our knowledge, this is the first work that presents a dataflow implementation for a full decoder compliant with the VVC standard.

The rest of the paper is organized as follows. Section 2 gives an overview of dataflow modeling, the VVC standard and introduces the OpenVVC decoder and its parallelism strategies, Sect. 3 is dedicated to the related work. Section 4 introduces the proposed dataflow model for the OpenVVC decoder. In Sect. 5, a comparison between the OpenVVC decoder and the proposed dataflow model while exploring tile-level parallelism is provided. Finally, Sect. 6 is devoted to the conclusion and future works.

## 2 Background

### 2.1 Dataflow Modeling

A dataflow model is defined as a set of actors exchanging data information through First In First Out (FIFO)s. The interactions between actors are regulated by a Model of Computation (MoC) that specifies which scheduling strategies can be utilized to fire actors. These interactions influence the system behavior of a dataflow model. In reality, actors are defined as a collection of firing rules that specify the circumstances in which an actor may fire.

Dataflow modeling is a methodology that proved to be efficient in computing data dependencies, scheduling and exploring parallelism between the tasks of a process. Moreover, it gives an additional insight into the application and facilitates the detection of any missing items or inadequate details. This last has been used extensively for signal and image processing applications especially while using Digital Signal Processor (DSP) architecture. In addition, Dataflow is widely used for Multiprocessor System-on-Chips (MPSoC) design and programming [9].

There are several toolchains that can be used to create a dataflow model such as SynDEx [10], Open RVC-CAL Compiler (Orcc) [11], and PREESM. In this work, the used tool is PREESM. Compared to the other tools, PREESM allows both the automatic scheduling of tasks and the automatic generation of functional code for heterogeneous multicore embedded systems [7]. In essence, it is an open source rapid prototyping tool that explores the design space of a target system in a way that minimizes its cost and ensures compliance with various constraints, most often latency, throughput, memory and power consumption. In addition, it boosts pluggable features that fit different targets [7]. The PREESM scheduling principle is based on fast scheduling methodology of Kwok [12]. PREESM plugins are generally focused on latency-dominated systems, i.e., systems that ensure throughput constraints are met while fulfilling the latency constraint [13]. Between two application iterations, PREESM uses barriers in order to synchronize all cores of an architecture. Moreover, the code generation is designed to produce self-timed code [14], i.e. static code for each core with automated communication between the cores, cache management and synchronization. PREESM includes different optimizations, one of them is an advanced memory optimization based on a memory exclusion graph [15]. This optimization serves to avoid the preservation of FIFO memory spaces unnecessary for the correct system execution. The simulation of PREESM workflow provides the system designer with a gantt chart that shows the partitioning of the tasks according to the used parallel architecture.

## 2.2 Overview of VVC Decoder

The decoder side of VVC standard has seen a raise in computing complexity approximately two times higher than HEVC standard in All Intra (AI) configuration. Figure 1 presents the structure of the VVC decoder. A decoder takes a bitstream as input and provides a decoded video sequence as output. The VVC decoder is mainly composed by three main processes: entropy decoding, block reconstruction, and the in-loop filters. The entropy decoding is based on the use of Context Adaptive Binary Arithmetic Coding (CABAC) [16]. This latter serves at decoding the binary syntax into syntax elements that feed all other decoder parts. The block reconstruction that is composed from inter/intra prediction, inverse transform and quantization is responsible for constructing the different regions of a frame. In fact, on the encoder side, a frame is divided into different blocks based on the use of tools included in the partitioning part of the encoder. The final step in the decoding process is the in-loop filters. The filters used in the VVC decoder are the inverse Luma Mapping with Chroma Scaling (LMCS), the Deblocking Filter (DBF), the Sample Adaptive Offset (SAO) filter and the ALF filter. Once a picture is entirely decoded it will be stored in the Decoded Picture Buffer (DPB) to be used if needed in the decoding of the next picture (case of inter prediction). More details about VVC standard could be found in [17]. There are three possible parallelism levels in the decoding process, they consist in data level, frame level and high level parallelism: slices/tiles and Wavefront Parallel Processing (WPP). Data level is based on the use of Single Instruction Multiple Data (SIMD) instructions. SIMD optimizations are generally used in applications that perform the same operation on multiple data points simultaneously which means operations that use vectors and matrices. In video coding field, there are several tasks that could take benefits from the SIMD instructions such as ALF filter, the transforms, etc. Data level parallelism has shown its efficiency in minimizing the computational complexity of the video coding process. Authors in [18] have leveraged the use of SIMD instructions to minimize the decoding time of the HEVC decoder. Effectively, the experimental results showed that the optimized decoder performs more than 4 times faster than the HM 4.0 decoder. Frame-level parallelism relies on frames that can be independently encoded/decoded. In this case, each frame is associated with a thread with a synchronisation module to manage the inter dependencies. In video coding standards, a frame can be divided into different regions. Some of these regions are called tiles and slices. The partitioning of an image into tiles and slices was first introduced in HEVC [19]. Tiles are rectangular regions of a frame containing entire CTUs. The prediction dependencies between tiles boundaries are broken and the entropy decoding is reinitialized for each tile. This fact, enables different large regions of a frame to be processed independently.

## 2.3 OpenVVC Decoder

The OpenVVC decoder has been developed in IETR laboratory at INSA Rennes. The decoder was created based on the VVC standard and with the use of C programming language. It has been compiled as a cross-platform library and implemented as a dynamic library in FFmpeg player [20]. OpenVVC is capable of real time decoding and supports all the parallelism levels previously detailed in Sect. 2.2: data level parallelism, frame level parallelism, and tile parallelism.

The decoding process of the OpenVVC decoder follows the same process of the VVC decoder. It starts with parsing the global parameters sets which are the Sequence Parameter Set (SPS), Picture Parameter Set (PPS), Picture header (PH), and Slice header (SH). This process helps gathering information from the different stages of a frame (slice, tile, sequence, picture) in order to decode it. After that, block reconstruction is applied at the CU level. When all CUs in the CTU are fully reconstructed, the DBF filter is applied at CTU level. Finally, the SAO, ALF and CC-ALF filters are applied at a CTU line level. Applying the filters at the CTU lines

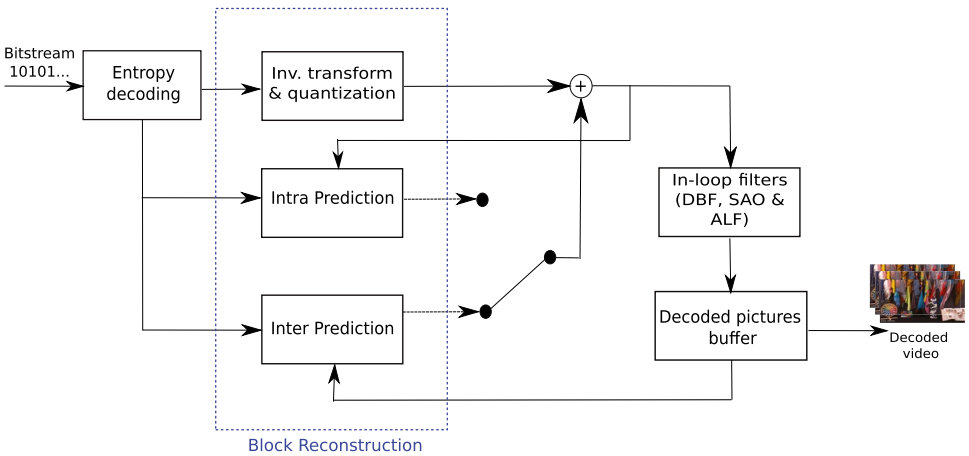


**Figure 1** VVC decoder block diagram.

improves frame-level parallelism in the inter-configuration compared to processing the in-loop filters after the entire frame is reconstructed.

Compared to the other available decoders developed based on the VVC standard such as VTM decoder and the Fraunhofer's decoder VVdeC, OpenVVC offers higher decoding rate with lower memory consumption especially in AI configuration. More detailed information about OpenVVC and the most consuming part of the decoding process could be found in [6].

## 3 Related Work

Usually, the most used approaches in creating a parallel algorithm for multicore processors are OpenMP [21], OpenCL [22] and pthread. The development of decoders using these approaches requires months of algorithmic development and then it is necessary to study the parallelization on each of the target architectures, which can be fast on homogeneous architectures with shared memories with the use of OpenMP for example, but much longer on heterogeneous architectures like FPGA and MPSoC. Besides, these approaches are based on the use of C/C++ programming languages which have many shortcomings and are not suitable for hardware design. As a result, implementing complex signal or image processing applications, especially video coding applications, on embedded multicore architectures has become increasingly difficult. Due to this fact, many researchers have focused on creating a library based on dataflow oriented language CAL [23] for the video coding components. This library is called Reconfigurable Video Coding (RVC) [24]. This framework provides a new specification formalism for the design of video codecs in a way that promotes flexibility and reuse [25]. RVC-CAL [26] actor language has been used in the modeling of several video coding standards such as MPEG-4 [27] and HEVC [28]. Authors in [25] have demonstrated that using a dataflow model facilitates code generation for hardware devices.

Studies in [28] and [29] used RVC-CAL dataflow models in the creation and implementation of a multicore algorithm for the HEVC decoder. Although the proposed dataflow model for the HEVC decoder does not exceed the performance of other HEVC compliant decoders such as OpenHEVC [30], the advantages of RVC-CAL remain undeniable. Indeed, RVC-CAL code is cross-platform, and compilers such as Orcc [11] and OpenDF [31] can automatically produce many languages from a single description, including C, C++, C-HLS, Verilog, etc. Since the creation of the hardware is ultimately complete and the validation of the application can be performed in a software context, this capability is crucial for hardware developers because it speeds up prototyping.

Dataflow modeling has also been tested on some parts of the latest video coding standard VVC and proved to be very efficient. For instance, in [32] a dataflow implementation for MTS concept has been created. The result showed that with choosing coarse grain granularity for x86 architecture, the dataflow model offers a speed-up close to the theoretical result. In [33], a comparison between a dataflow implementation for the MTS concept and the OpenMP method showed that, with the use of the dataflow model, a better speedup was achieved. This evidence proves that, compared to the state-of-the-art approaches used to create a multicore algorithm for embedded systems, dataflow modeling could achieve better performance in a much shorter development time. In addition, dataflow circuits are necessary for efficient C-to-circuit translation of any software program, as they are capable of handling variable latencies and erratic memory dependencies. Many HLS compilers have included dataflow modeling to generate an FPGA design in which all tasks are pipelined and executed concurrently [34, 35].

## 4 The Proposed Model

The development of a multicore algorithm such as the OpenVVC decoder is time-consuming because it requires a great deal of knowledge of the software and the target architecture. Added to that, when modifying the target architecture, other constraints must be taken into account. To reduce time to market, dataflow modeling is commonly used. It facilitates the mapping of tasks according to the available cores and the fast generation of a multicore algorithm. Indeed, dataflow modeling is widely used to implement an application on a multicore architecture. This stems from the fact that it is efficient for calculating data dependencies and scheduling tasks without having a deep knowledge about the hardware. From this context, this paper aims at studying the efficiency of dataflow modeling in the exploration of tile parallelism.

For the purpose of creating a dataflow model for the OpenVVC decoder that explores the parallelism between tiles with the use of PREESM, the following steps must be performed:

1. Implement the OpenVVC project as an external library.
2. Disable the use of all functions developed using pthread in the OpenVVC project to explore parallelism between tiles and between frames.
3. Provide the necessary inputs for PREESM. Essentially, in order to create the dataflow model and create the multicore algorithm, three elements need to be created first.

    (a) The algorithm graph: a graph presenting a set of actors exchanging data through FIFOs and aims to provide a clear and complete description of the process. The actor description could be hierarchical ( i.e. an actor could contain other actors). The

algorithm graph is created using a type of dataflow model known as PiSDF.

(b) The architecture graph: the kind of architecture graph used in PREESM is called System Level Architecture Model (S-LAM). It is a set of cores linked to a shared memory in order to communicate.

(c) The scenario: an item of PREESM which includes all the information necessary for the execution of the workflow which is a set of tasks responsible for the generation of the code, the display of the Gantt chart and the use of optimizations. The scenario contains the path of the algorithm graph, the path of the architecture graph, and the execution time of each actor.

After implementing the OpenVVC as an external library in PREESM and identifying the functions responsible of decoding the tiles, an algorithm graph (Fig. 2) is created. The Fig. 2 presents the global algorithm graph of the proposed model. This last is composed of 18 FIFOs and 12 actors in total. Some actors are responsible for the duplication or initialization of variables and others for the decoding process. The latter are the following five actors:

- *Attach_Stream*: this actor serves at associating the bitstream to be decoded to the decoder and initializing the decoder and the dumuxer. Then it will send the information about the initialized decoder and dumuxer to the other actors through the variable *ovvc_hdl*.
- *Frame_Informations_Extraction*: this actor is responsible for extracting parts of the bitstream more specifically, the Network Abstraction Layer (NAL) units. The extracted information will be sent to the *Frame_Decoding* actor via *pu* variable which is a vector of NAL units.
- *Frame_Decoding*: this actor passes the sub-stream to the input of the decoder and starts decoding the picture.
- *Frame_Receiving*: this actor polls the decoder in order to see if the next image in the display order is available and can leave the decoder, and places the next image in the display order on the output.
- *Detach_Stream*: this actor closes the file and resets the dumuxer to zero.

As the tile parallelism is the focus of this paper, the decoding of the picture (*Frame_Decoding* actor) itself has been decomposed into several actors. The Fig. 3 represents the algorithm graph associated with the actor *Frame_Decoding*. The latter is composed of 25 FIFOs and 13 actors in total. The main actors in this graph are:

- *Slice_Selection*: selects the slices to be decoded.
- *Tiles_Decoding*: decodes the tiles of a frame.
- *Non_Tiles_Decoding*: decodes other types of NAL units (that don't contain tiles information).

To ensure the tile parallelism, the tiles actor is duplicated based on the information provided by the *tiles_number* parameter. For example, if the *tiles_number* is equal to *N*, the tiles actor will be duplicated *N* times and those *N* tasks will be executed in parallel. In other words, the duplication process is done automatically according to the production and consumption rate of each actor. If an actor A has a production rate equal to *N* for example while actor B has a consumption rate equal to one, PREESM will duplicate actor B *N* times.

## 5 Experimental Results

This section presents the experimental setup and the different results obtained using the proposed dataflow model in AI configuration.

### 5.1 Experimental Setup

The experiments were performed using PREESM version 3.20.0 on Intel(R) Xeon(R) CPU E5-2690 v3 @ 2.60GHz with 24 cores under Ubuntu 18.04 and using various video sequences that are detailed in Tables 1, and 2. The video sequences used are FHD (class B, 1920×1080 pixels) and

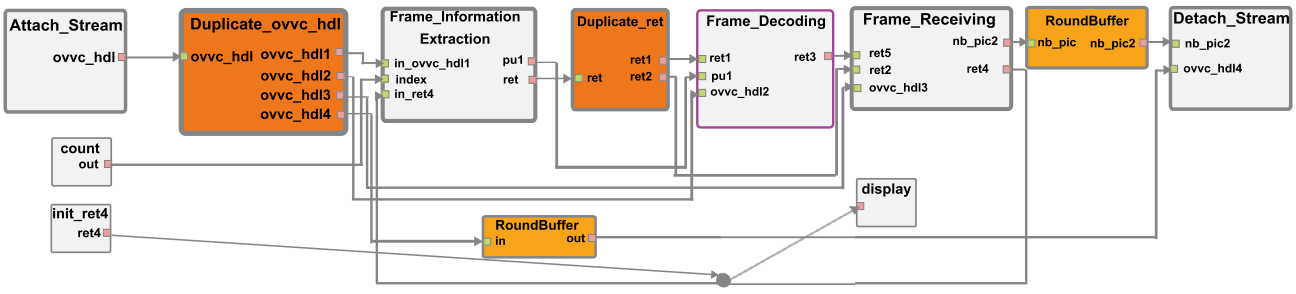


**Figure 2** PiSDF model for the OpenVVC Decoder.

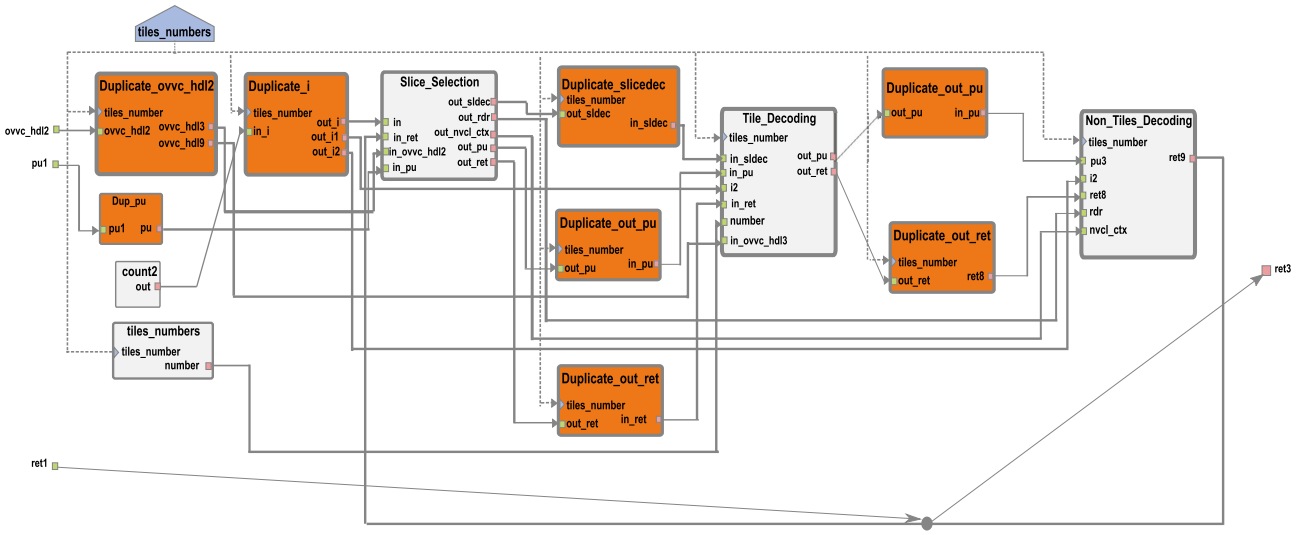


**Figure 3** The inside of the *Frame_Decoding* actor.

UHD (class A, 3840×2160 pixels) included in the Common Test Conditions (CTC) [36], as these are the sequences that have the highest resolution and take a long time in the decoding process. Actually, with the grown complexity of VVC standards there is a raise of critical issue basically for high resolution sequences. Moreover, with high resolution comes high granularity of actors, which is an important factor to achieve a good performance while using x86 architecture, as proven in previous works [32, 33]. To be precise, in [32] a study was made to investigate the effect of the granularity level on the performance of an application using the x86 architecture. The study showed that it is important to use high granularity when using x86 architecture to get a good result. Based on this study, high resolution sequences are used so that each actor performs a large number of computations and the synchronization time becomes negligible compared to the execution time of the actors. In addition, while decoding a tile, each thread requires its own filter buffers and local context structure. The main frame buffer and global context structure, on the other hand, are shared by all threads, which may result in a buffer overhead. But since the sizes of the filter buffers and the local context structure are both small in comparison to the size of the main image buffer, buffer overhead is assumed to be negligible for high-resolution sequences [37]. The used video sequences are encoded with three tile configurations $4 \times 3$ (four horizontal by three vertical splits), $6 \times 2$, and $6 \times 4$. The same encoder configuration, without tiles, is used as a reference to calculate the BD-Rate objective measurement [38]. The BD-Rate is an objective measure applied in video compression to compare the bitrate distortion performance or compression efficiency of two different video codecs or different parameters of the same video codec over a range of bitrate or quality values. Tables 1 and 2 show that as the number of tiles increases, there is an increase in the BD-Rate. Hence, a bitrate overheat at same quality. In fact, the study of [39] showed that the quality of coding decreases with the increase of the number of independent regions in the image. This is due to the fact that intra prediction dependencies across tiles boundaries are broken and that the entropy coding is initialized at each tile, resulting

**Table 1** Benchmarks of UHD (3840×2160) sequences used in the experiment.

| | BD-Rate | |
|---|---|---|
| Sequence | 12 Tiles ($6 \times 2$) | 24 Tiles ($6 \times 4$) |
| DaylightRoad2 | 0.88% | 1.26% |
| Campfire | 0.84% | 1.33% |
| CatRobot | 0.91% | 1.76% |
| ParkRunning | 0.29% | 0.48% |
| Tango2 | 1.79% | 2.84% |
| FoodMarket | 1.32% | 2.13% |

**Table 2** Benchmarks of FHD (1920x1080) sequences used in the experiment.

| | BD-Rate | |
|---|---|---|
| Sequence | 12 Tiles ($4 \times 3$) | 24 Tiles ($6 \times 4$) |
| BQTerrace | 0.77% | 1.27% |
| Cactus | 1.08% | 1.79% |
| BasketballDrive | 1.80% | 3.08% |
| MarketPlace | 1.10% | 1.84% |
| RitualDance | 1.49% | 2.45% |

in a loss of coding quality compared to the unpartitioned sequence coding.

In order to provide a fair comparison between the openVVC decoder and the proposed dataflow model, the frame level parallelism is disabled in the original OpenVVC decoder. The comparison will be made based only on tiles level parallelism. The bitstreams used in this work are generated with VTM12.0 encoder. The input of an encoder is a configuration file that contains various information about the sequence to be encoded such as the number of frames, the type of configuration (AI, Random Access, Low Delay), etc. In this paper, the configuration file has been modified to allow the decomposition of a frame into tiles and to set the number of tiles and the location of their boundaries. Actually, tiles offer flexible classification of CTUs. Besides, tiles provide an excellent coding efficiency and a favored correlation of pixels over slices as they do not have a header information [40]. The performance of both OpenVVC decoder and the proposed model are assessed at various bit-rates, obtained with QP values of {22, 27, 32, 37} following the CTC.

### 5.2 Results Analysis

Figures 4 and 5 showcase the speed-up of both the proposed model and the OpenVVC decoder while using UHD and FHD sequences encoded with 12 tiles respectively. Results showed that speed gains of 6 and 7 are achieved using 12 cores for a bitstream consisting of 12 tiles for FHD and UHD sequences, respectively. For FHD and UHD sequences with 24 tiles, speed gains of 9 and 11 on 24 cores are achieved respectively as illustrated in Figs. 6 and 7). For the results obtained using FHD sequences, the speeds of the proposed model and the OpenVVC decoder shown in Figs. 5 and 7 are not too close using four cores. However, they are closer using UHD sequences, which can be explained by the fact that as the resolution of the sequence increases, the actor has more computational process, so the granularity increases and the synchronization time becomes negligible compared to the actor execution time. Ideally, for a fully parallelizable process, a speedup equal to 12 is expected while using 12-core architecture. However, the decoding process is not 100% parallelizable. In reality, many parts of the decoding process are more complex than decoding a tile and are performed sequentially, making them more time consuming. For example, according to the study presented in [6], loop filters account for more than half of the decoding complexity in an AI configuration. This fact is one of the main reasons that explain the difference between the theoretical and experimental results, since the loop filters are executed sequentially after the decoding of a frame. Another factor that explains the discrepancy between the theoretical case and the obtained result is the fact that the tiles are not of equal sizes. For example, Fig. 8 shows the size inequality between 12 tiles for a class A sequence. An image is a set of CTUs and the details of an image change from one pixel to another. Consequently, the amount of detail in each CTU is different. It is thereby different for each tile. The decoding time is therefore not the same for all tiles as shown in Fig. 9. In fact, this figure details the decoding time in percentage for each tile presented in a frame of a class B sequence (BQTerrace). This decoding time

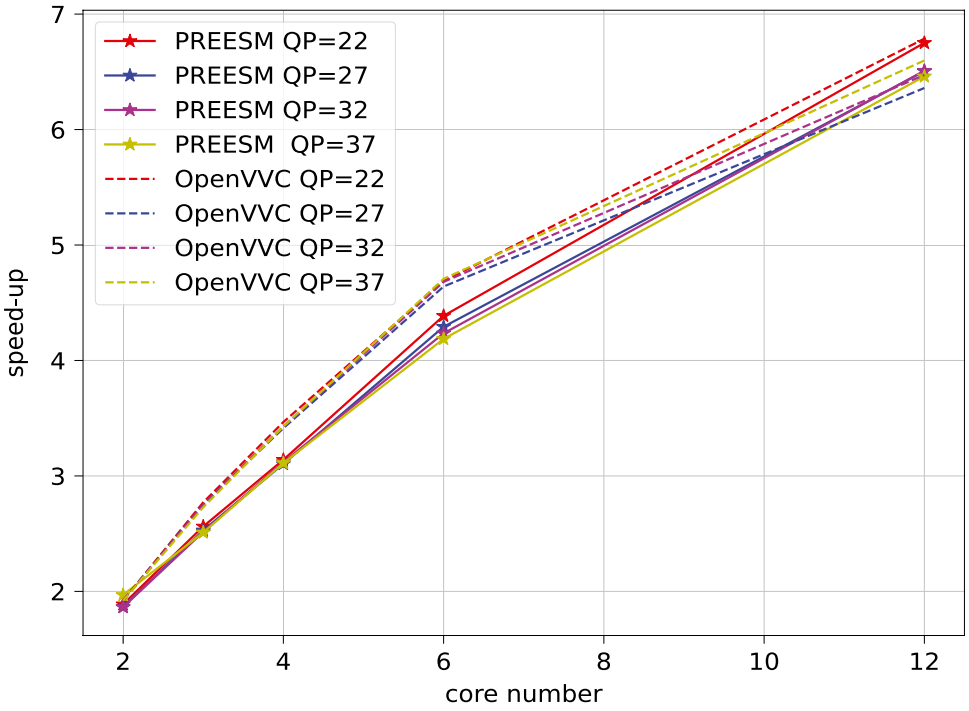


**Figure 4** Speed-up of the proposed model and of the OpenVVC decoder while using UHD sequences encoded with 12 tiles.

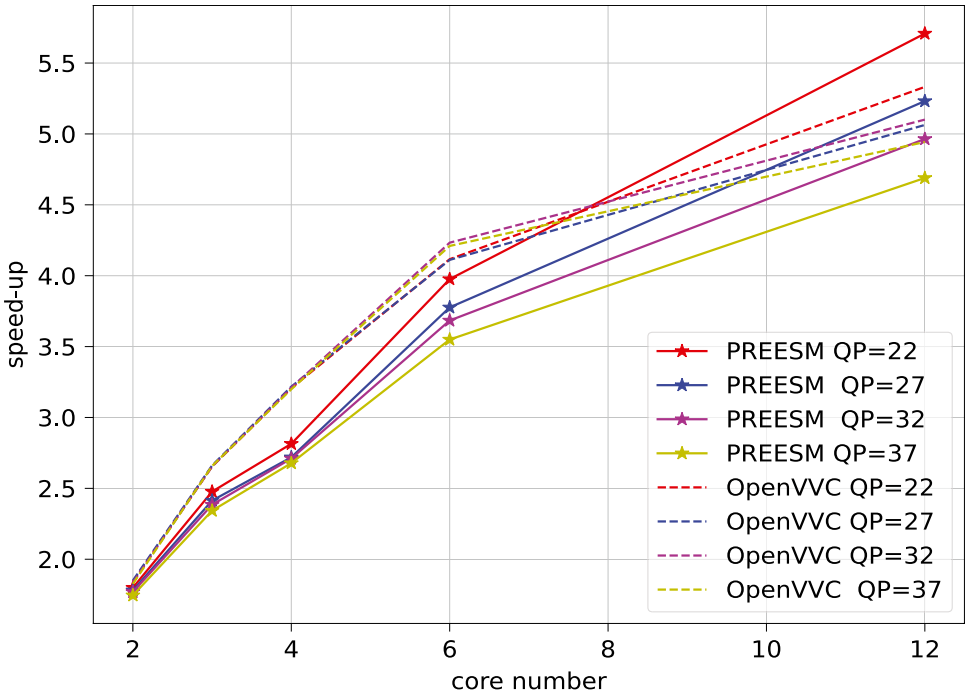


**Figure 5** Speed-up of the proposed model and of the OpenVVC decoder while using FHD sequences encoded with 12 tiles.

imbalance between tiles has less effect on UHD sequence than on FHD sequence as the former's frame size is larger and it grows with the number of tile contained in the frame.

Although the experimental result of the proposed dataflow model is very close to that of the OpenVVC decoder and does not significantly outperform it, dataflow modeling remains a design methodology of great importance. The state-of-the-art approaches used for the creation of a multicore algorithm necessitate a lot of development time compared to dataflow modeling as they require deep knowledge of both hardware and software. For example, as real-time codecs typically offer fine-grain parallelism, architectures like Field Programmable Gate Arrays (FPGA) are perfect targets for implementation. However,

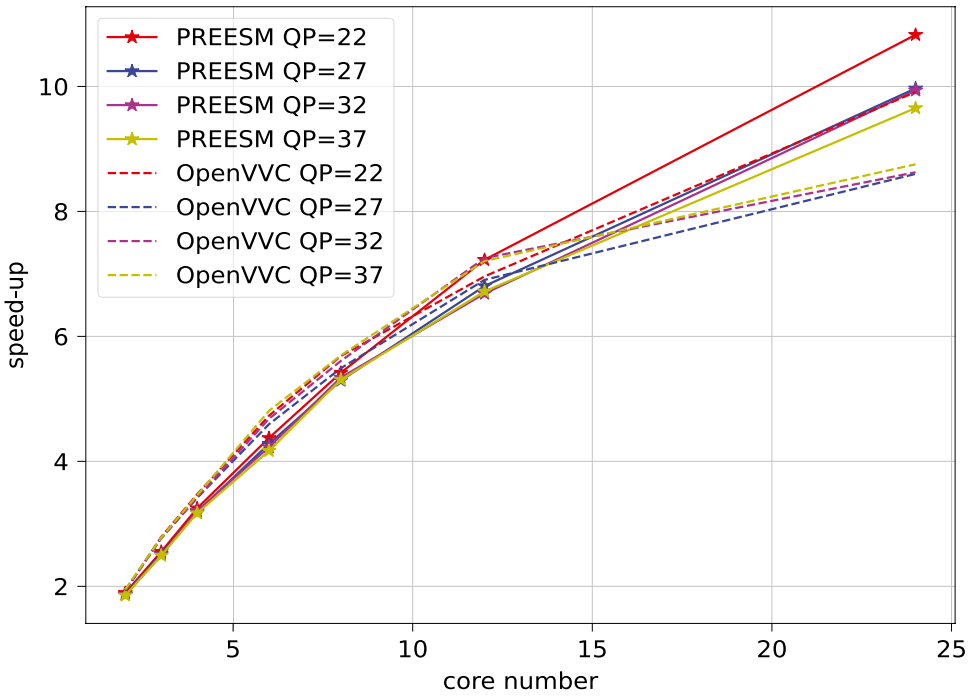


**Figure 6** Speed-up of the proposed model and of the OpenVVC decoder while using UHD sequences encoded with 24 tiles.

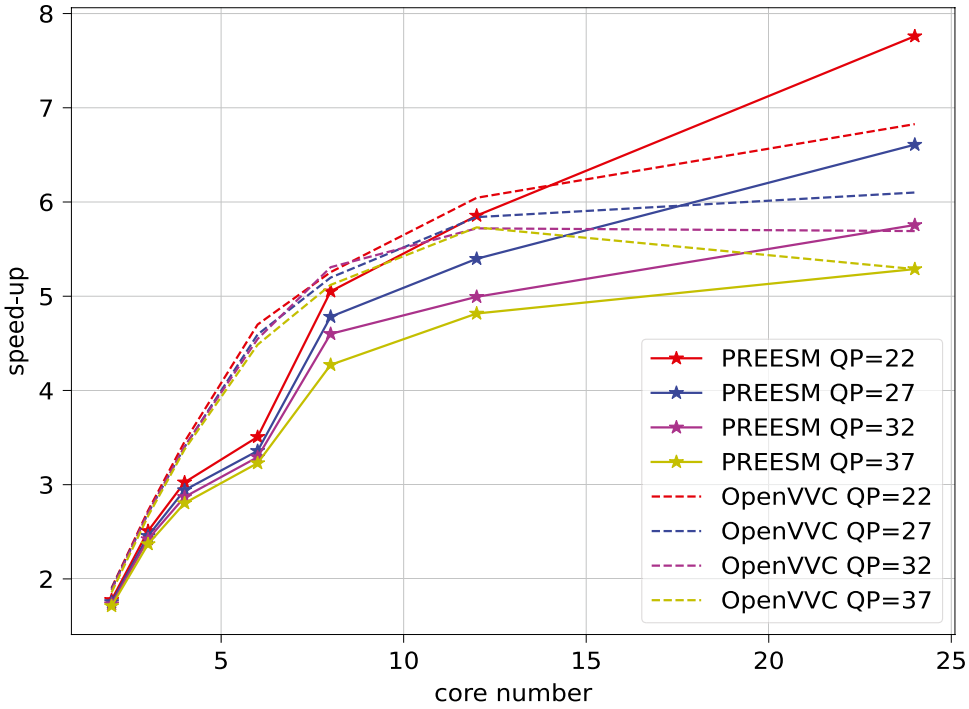


**Figure 7** Speed-up of the proposed model and of the OpenVVC decoder while using FHD sequences encoded with 24 tiles.

the programming languages used to program FPGA, such as VHDL or Verilog, require digital design expertise and result in an increase in time to market [41]. In a nutshell, using dataflow modeling saves development time. Also, compared to other approaches, dataflow modeling presents a suitable framework that allows switching from one architecture to another in a flexible way, as a dataflow model can target different architectures. Indeed, a dataflow model could target both shared and distributed memory architectures without changing the developed model. However, in the case of the OpenVVC decoder that was developed using pthread and with the knowledge of software and hardware, there is a possibility of presenting memory storage problems when switching from one architecture to another. Indeed,

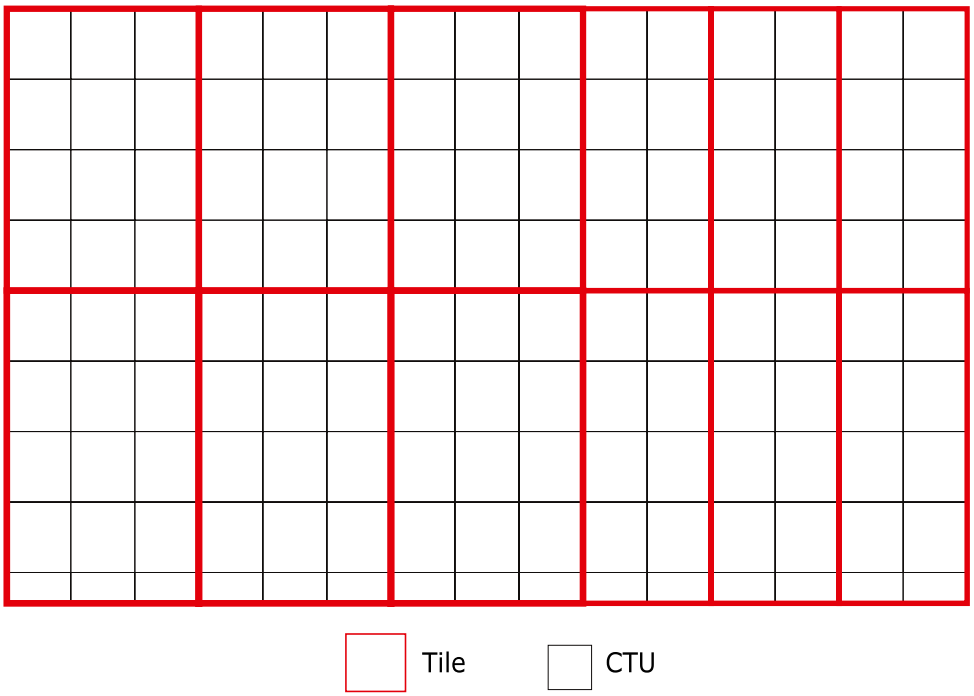


**Figure 8** $6 \times 2$ Tiles partitioning for Class A ($3840 \times 2160$).

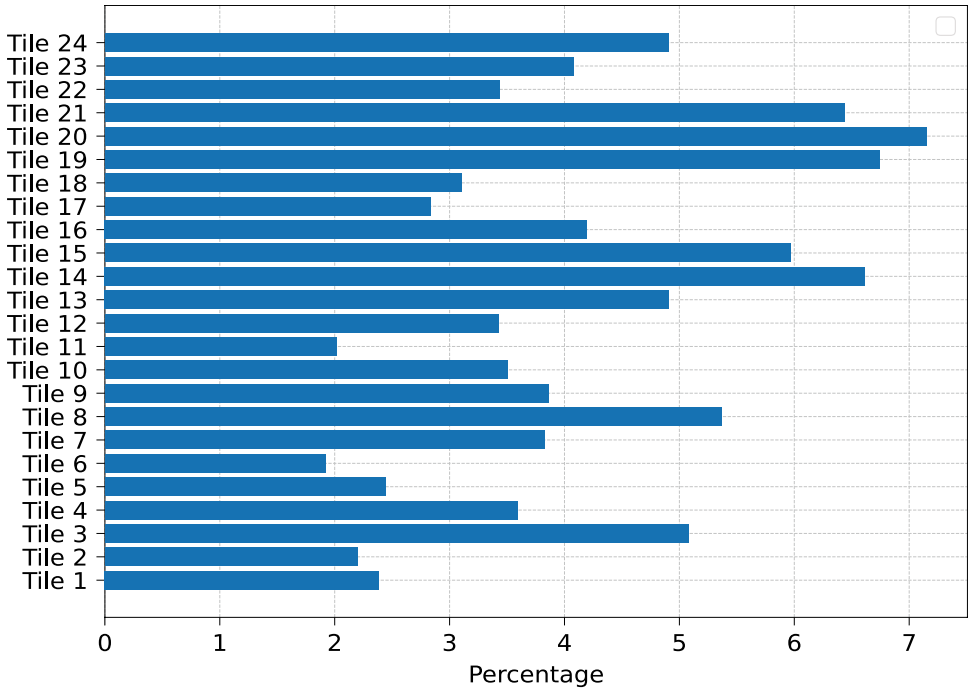


**Figure 9** Per tile decoding time in percentage for a frame of BQTerrace sequence.

due to the large number of memory allocations included in OpenVVC, portability on systems with heterogeneous architecture can be challenging.

One other major result up for debate is the time spent by the dataflow decoder in the decoding process on a single core. In fact, previous dataflow models for decoders have shown promising performance on multicore, but on a single core, the performance is degraded compared to the reference decoders. In [29] a DSP based decoder conforming to the HEVC standard has been implemented using the RVC-CAL model. The presented RVC-CAL decoder is 50% less efficient than HM9.0 and five times less efficient than OpenHEVC, while using a single core. Similarly, in [28], a dataflow model for a full decoder that explores parallelism between the YUV component showed a decrease in framerate (FPS) compared to the reference, and the OpenHEVC decoder outperformed both the reference and the presented model while using a single core architecture. Nevertheless, the PiSDF model presented for the OpenVVC decoder outperformed the reference software decoder (VTM12.0) and has a similar decoding time as OpenVVC on monocore as shown in Table 3. This result proves the efficiency of the PiSDF model compared to the decoder models that have been implemented using RVC-CAL. In fact, dynamic dataflow models such as RVC-CAL perform scheduling at runtime which slows down the decoding process. However, parameterized models such as PiSDF used in this work are predictable, allowing optimizations at compile time. Consequently, the scheduling time is negligible compared to the decoding process.

**Table 3** Decoding time on single core of UHD sequences, QP = 22.

| Sequence | Decoding time (s) | | |
|---|---|---|---|
| | VTM12.0 | OpenVVC | Dataflow model |
| DaylightRoad2 | 42.62 ± 0.205 | 15.39 ± 0.375 | 15.10 ± 0.042 |
| Campfire | 34.32 ± 0.571 | 11.55 ± 0.405 | 11.65 ± 0.67 |
| CatRobot | 34.65 ± 0.265 | 11.15 ± 0.091 | 10.95 ± 0.04 |
| ParkRunning | 35.86 ± 0.078 | 12.97 ± 0.078 | 12.71 ± 0.046 |
| Tango2 | 23.66 ± 0.054 | 8.32 ± 0.053 | 8.07 ± 0.082 |
| FoodMarket | 21.49 ± 0.06 | 7.32 ± 0.059 | 7.19 ± 0.037 |

## 6 Conclusion

This paper investigates the efficiency of using dataflow modeling in creating a model for a complete decoder based on the VVC standard. The tool used to create the dataflow model is called PREESM. This tool allows the automatic scheduling of tasks according to the number of used cores and the automatic generation of multicore algorithms. The proposed model is created to make explicit and explore the parallelism between tiles that are a rectangular region of a frame encompassing entire CTUs. Parallel tile decoding is allowed due to the fact that prediction dependencies between tile boundaries are broken and entropy decoding is initialized for each tile. The results showed that, unlike the state-of-the-art dataflow decoders based on the RVC-CAL model, the proposed model still performs

well on a single-core architecture compared to decoders developed with C/C++ languages. In addition, it achieves a competitive speed-up to that of the OpenVVC decoder. Whereas the speed improvement is not very significant, the relevance dataflow comes from the fact that one model can target different architectures and dataflow modeling is an effective framework that facilitates the transition from software to hardware. Moreover, creating a multicore algorithm using dataflow modeling requires a shorter development time compared to state-of-the-art approaches such as OpenMP and OpenCL. In the near future, we aim to adopt task-based code generation and execution, a method proposed in [42], to automatically balance the differences of tile decoding times and improve our proposed model. It is also planned to improve the proposed model to support frame-level parallelism. Furthermore, it is planned to implement the proposed dataflow model on heterogeneous architectures such as ARM BIGlittle, FPGA, and Kalray.

**Acknowledgements** This work is supported by the France Campus, and within a co-supervised thesis between the Institue of Electronics and Telecommunications (IETR) of Rennes, France, and the Laboratory of Electronics and Information Technology (LETI) of Sfax, Tunisia.

**Author Contributions** N. Haggui designed, coordinated this research, drafted the manuscript and conducted the experiments and data analysis. W.Hamidouche participated in the conceptualization, methodology, writing, revision, editing, assisted in data analysis and participated in the coordination of the research, F.Belghith, N.Masmoudi and J.F.Nezan assisted in the data analysis, participated in the coordination of the research, supervision, writing, revision and editing. The authors read and approved the final manuscript.

**Funding** This work was supported by the MEAE, MESRI (France), MESRS (Tunisia), MESRS (Algeria), MEN, CNRST (Morocco), through the Hubert Curien Partnerships (PHC) Maghreb 2021, No 45988WG (Eco-VVC project).

**Data Availability** The data will be made publicly available upon acceptance of the article.

## Declarations

**Ethics Approval and Consent to Participate** Not applicable.

**Conflicts of Interest** The authors declare that they have no conflict of interest.

## References

1. Cisco, U. (2020). *Cisco annual internet report (2018–2023) white paper*. Cisco: San Jose, CA, USA.
2. Sullivan, G. J., Ohm, J.-R., Han, W.-J., & Wiegand, T. (2012). Overview of the high efficiency video coding (HEVC) standard. *IEEE Transactions on Circuits and Systems for Video Technology, 22*(12), 1649–1668.
3. Hamidouche, W., Biatek, T., Abdoli, M., Francois, E., Pescador, F., Radosavljevic, M., Menard, D., & Raulet, M. (2022). Versatile video coding standard: A review from coding tools to consumers deployment. *IEEE Consumer Electronics Magazine*.
4. Li, T., Xu, M., Tang, R., Chen, Y., & Xing, Q. (2021). DeepQTMT: A deep learning approach for fast QTMT-based CU partition of intra-mode VVC. *IEEE Transactions on Image Processing, 30*, 5377–5390.
5. Wieckowski, A., Hege, G., Bartnik, C., Lehmann, C., Stoffers, C., Bross, B., & Marpe, D. (2020). Towards a live software decoder implementation for the upcoming versatile video coding (VVC) codec. In *2020 IEEE International Conference on Image Processing (ICIP)* (pp. 3124–3128). IEEE.
6. Amestoy, T., Cabarat, P.-l., Gautier, G., Hamidouche, W., & Menard, D. (2022). OpenVVC: A lightweight software decoder for the versatile video coding standard. Preprint retrieved from http://arxiv.org/abs/2205.12217
7. Pelcat, M., Desnos, K., Heulot, J., Guy, C., Nezan, J.-F., & Aridhi, S. (2014). PREESM: A dataflow-based rapid prototyping framework for simplifying multicore DSP programming. In *2014 6th European Embedded Design in Education and Research Conference (EDERC)* (pp. 36–40). IEEE.
8. Desnos, K. & Heulot, J. (2014). PISDF: Parameterized & interfaced synchronous dataflow for MPSoCs runtime reconfiguration. In *1st Workshop on Methods and Tools for Dataflow Programming (METODO)*.
9. Aguilar, M. A., Leupers, R., Ascheid, G., & Murillo, L. G. (2016). Automatic parallelization and accelerator offloading for embedded applications on heterogeneous MPSoCs. In *Proceedings of the 53rd Annual Design Automation Conference* (pp. 1–6).
10. Grandpierre, T., Lavarenne, C., & Sorel, Y. (1999). Optimized rapid prototyping for real-time embedded heterogeneous multiprocessors. In *Proceedings of the Seventh International Workshop on Hardware/Software Codesign*, CODES '99 (pp. 74–78). New York, NY, USA. Association for Computing Machinery.
11. Yviquel, H., Lorence, A., Jerbi, K., Cocherel, G., Sanchez, A., & Raulet, M. (2013). ORCC: Multimedia development made easy. In *Proceedings of the 21st ACM International Conference on Multimedia* (pp. 863–866).
12. Yu-Kwong, K. (1997). High-performance algorithms for compile-time scheduling of parallel processors. *The Hong Kong University of Science and Technology in Partial Fulfillment of the Requirements for the Degree of Doctor of Philosophy in Computer Science Hong Kong*.
13. Ghamarian, A. H., Geilen, M. C., Stuijk, S., Basten, T., Theelen, B. D., Mousavi, M. R., Moonen, A. J., & Bekooij, M. J. (2006). Throughput analysis of synchronous data flow graphs. In *Sixth International Conference on Application of Concurrency to System Design (ACSD'06)* (pp. 25–36). IEEE.
14. Sriram, S., & Bhattacharyya, S. S. (2018). *Embedded multiprocessors: Scheduling and synchronization*. CRC Press.
15. Desnos, K., Pelcat, M., Nezan, J.-F., & Aridhi, S. (2013). Pre- and post-scheduling memory allocation strategies on MPSoCs. In *Proceedings of the 2013 Electronic System Level Synthesis Conference (ESLsyn)* (pp. 1–6). IEEE.
16. Sze, V., & Budagavi, M. (2012). High throughput CABAC entropy coding in HEVC. *IEEE Transactions on Circuits and Systems for Video Technology, 22*(12), 1778–1791.
17. Bossen, F., Sühring, K., Wieckowski, A., & Liu, S. (2021). VVC complexity and software implementation analysis. *IEEE Transactions on Circuits and Systems for Video Technology, 31*(10), 3765–3778.
18. Yan, L., Duan, Y., Sun, J., & Guo, Z. (2012). Implementation of HEVC decoder on x86 processors with simd optimization. In *2012 Visual Communications and Image Processing* (pp. 1–6). IEEE.
19. Misra, K., Segall, A., Horowitz, M., Xu, S., Fuldseth, A., & Zhou, M. (2013). An overview of tiles in HEVC. *IEEE Journal of Selected Topics in Signal Processing, 7*(6), 969–977.

20. FFMPEG: Open source and cross-platform multimedia library. Retrieved January 2022, from http://www.ffmpeg.org
21. Dagum, L., & Menon, R. (1998). OPENMP: An industry standard API for shared-memory programming. *IEEE Computational Science and Engineering, 5*(1), 46–55.
22. Stone, J. E., Gohara, D., & Shi, G. (2010). OpenCL: A parallel programming standard for heterogeneous computing systems. *Computing in Science & Engineering, 12*(3), 66.
23. Eker, J., & Janneck, J. (2003). *CAL language report: Specification of the CAL actor language*. December.
24. Bhattacharyya, S. S., Eker, J., Janneck, J. W., Lucarz, C., Mattavelli, M., & Raulet, M. (2011). Overview of the MPEG reconfigurable video coding framework. *Journal of Signal Processing Systems, 63*(2), 251–263.
25. Abid, M., Jerbi, K., Raulet, M., Déforges, O., & Abid, M. (2013). System level synthesis of dataflow programs: HEVC decoder case study. In *Proceedings of the 2013 Electronic System Level Synthesis Conference (ESLsyn)* (pp. 1–6). IEEE.
26. Wipliez, M., Roquier, G., & Nezan, J.-F. (2011). Software code generation for the RVC-CAL language. *Journal of Signal Processing Systems, 63*(2), 203–213.
27. Bezati, E., Mattavelli, M., & Raulet, M. (2010). RVC-CAL dataflow implementations of MPEG AVC/H. 264 CABAC decoding. In *2010 Conference on Design and Architectures for Signal and Image Processing (DASIP)* (pp. 207–213). IEEE.
28. Jerbi, K., Yviquel, H., Sanchez, A., Renzi, D., De Saint Jorre, D., Alberti, C., Mattavelli, M., & Raulet, M. (2017). On the development and optimization of HEVC video decoders using high-level dataflow modeling. *Journal of Signal Processing Systems, 87*(1), 127–138.
29. Chavarrías, M., Pescador, F., Garrido, M. J., Juarez, E., & Raulet, M. (2013). A DSP-based HEVC decoder implementation using an actor language dataflow model. *IEEE Transactions on Consumer Electronics, 59*(4), 839–847.
30. IETR/VAADER (2016). Open source HEVC decoder (OpenHEVC). Retrieved December 2021, from https://github.com/OpenHEVC
31. Bhattacharyya, S. S., Brebner, G., Janneck, J. W., Eker, J., Von Platen, C., Mattavelli, M., & Raulet, M. (2009). OpenDF: a dataflow toolset for reconfigurable hardware and multicore systems. *ACM SIGARCH Computer Architecture News, 36*(5), 29–35.
32. Haggui, N., Belghith, F., Hamidouche, W., Masmoudi, N., & Nezan, J.-F. (2021). Multiple transform selection concept modeling and implementation using interface based SDF graphs. In *Workshop on Design and Architectures for Signal and Image Processing (14th edition)* (pp. 60–67).
33. Haggui, N., Belghith, F., Hamidouche, W., Masmoudi, N., & Nezan, J.-F. (2022). Multiple transform selection concept modeling and implementation using dynamic and parameterized dataflow graphs. *Journal of Signal Processing Systems*, 1–12.
34. Amiri, P., Pérard-Gayot, A., Membarth, R., Slusallek, P., Leißa, R., & Hack, S. (2021). Flower: A comprehensive dataflow compiler for high-level synthesis. In *2021 International Conference on Field-Programmable Technology (ICFPT)* (pp. 1–9). IEEE.
35. Josipović, L., Sheikhha, S., Guerrieri, A., Ienne, P., & Cortadella, J. (2021). Buffer placement and sizing for high-performance dataflow circuits. *ACM Transactions on Reconfigurable Technology and Systems (TRETS), 15*(1), 1–32.
36. Boyce, J., Suehring, K., Li, X., & Seregin, V. (2018). JVET common test conditions and software reference configurations. In *Document JVET-J1010*.
37. Amestoy, T. (2021). Optimisation du codec VVC basé sur la réduction de complexité et le traitement parallèle.
38. Bjontegaard, G. (2001). Calculation of average PSNR differences between RD-curves. *VCEG-M33*.
39. Chi, C. C., Alvarez-Mesa, M., Juurlink, B., Clare, G., Henry, F., Pateux, S., & Schierl, T. (2012). Parallel scalability and efficiency of HEVC parallelization approaches. *EEE Transactions on Circuits and Systems for Video Technology, 22*(12), 1827–1838.
40. Abu Taha, M., Hamidouche, W., Sidaty, N., Viitanen, M., Vanne, J., El Assad, S., & Déforges, O. (2020). Privacy protection in real time HEVC standard using chaotic system. *Cryptography, 4*(2), 18.
41. Abid, M., Jerbi, K., Raulet, M., Déforges, O., & Abid, M. (2018). Efficient system-level hardware synthesis of dataflow programs using shared memory based FIFO. *Journal of Signal Processing Systems, 90*(1), 127–144.
42. Georgakarakos, G., Kanur, S., Lilius, J., & Desnos, K. (2017). Task-based execution of synchronous dataflow graphs for scalable multicore computing. In *2017 IEEE International Workshop on Signal Processing Systems (SiPS)* (pp. 1–6). IEEE.




**Naouel Haggui** was born in Kasserine, Tunisia, in 1995. She received the electrical engineering degree from the National Engineering School of Sfax (ENIS), Tunisia in 2019 Since 2020, she has joined the Electronics and Information Technology Laboratory (LETI), Sfax and the Institute of Electronics and Telecommunications of Rennes (IETR) where she is currently a PhD student. Her research interests include video coding, dataflow modelisation and hardware implementation using embedded multi-core platforms.

**Wassim Hamidouche** received Master's and Ph.D. degrees both in Image Processing from the University of Poitiers (France) in 2007 and 2010, respectively. From 2011 to 2013, he was a junior scientist in the video coding team of Canon Research Center in Rennes (France). He was a post-doctoral researcher from Apr. 2013 to Aug. 2015 with VAADER team of IETR where he worked under collaborative project on HEVC video standardisation. Since Sept. 2015 he is an Associate Professor at INSA Rennes and a member of the VAADER team of IETR Lab. He has joined the Advanced Media Content Lab of b-com IRT Research Institute as an academic member in Sept. 2017. His research interests focus on video coding and multimedia security. He is the author/coauthor of more than one hundred and forty papers at journals and conferences in image processing, two MPEG standards, three patents, several MPEG contributions, public datasets and open source software projects.

**Fatma Belghith** was born in Sfax, Tunisia, in 1988. She received her degree in Electrical Engineering from the National School of Engineering (ENIS), Sfax, Tunisia, in 2012. She received her ph.D degree in Electronic Engineering in 2016. She is currently an assistant professor at the faculty of sciences and techniques of Sidi Bouzid (Tunisia) Her current research interests include video coding with emphasis on HEVC standard and beyond, hardware implementation using FPGA and embedded systems technology.

**Nouri Masmoudi** was born in Sfax, Tunisia, in 1955. He received electrical engineering degree from the Faculty of Sciences and Techniques - Sfax, Tunisia, in 1982, the DEA degree from the National Institute of Applied Sciences-Lyon and University Claude Bernard-Lyon, France in 1984. From 1986 to 1990, he prepared his thesis at the laboratory of Power Electronics (LEP) at the National School Engineering of Sfax (ENIS). He received his PhD degree from the National School Engineering of Tunis (ENIT), Tunisia in 1990. From 1990 to 2000, he was an assistant professor at the electrical engineering department -ENIS. Since 2000, he has been an associate professor and head of the group 'Circuits and Systems' in the Laboratory of Electronics and Information Technology. Since 2003, He is responsible for the Electronic Master Program at ENIS. His research activities have been devoted to several topics: Design, Telecommunication, Embedded systems and Information technology. Video Coding (Motion Estimation, Mode Decision, H.264 Standard, complexity reduction of the VVC standard using deep learning, Implementation of the VVC transform unit), Image Processing (Wavelet Image Compression, Subband Image Coding, Image Interpolation, Denoising).

**Pr. Jean-François Nezan** is a Professor at the National Institute of Applied Sciences (INSA), Rennes Scientific and Technical University and the "Institut d'Electronique et des Technologies du numéRique de Rennes" (IETR). He is the leader of the VAADER team (Video Analysis and Architecture Design for Embedded Resources). He is coauthor or coeditor of more than 100 technical articles including 1 Book, 2 Book chapters, 23 publications in International Journals and 2 patents. He supervised 17 defended PhDs and supervises 5 PhDs. J-F. NEZAN is involved in the French research society "GDR ISIS" in the C theme entitled "Adequation Algorithm Architecture" and the European Network of Excellence on High Performance and Embedded Architecture and Compilation (HiPEAC). He participated to the setting-up and the management of 5 founded national projects including the COMPA project he led, 1 European project and 10 research agreements with industrial partners. His research focuses on new hardware/software codesign methodologies involving dataflow models in the computing continuum, from embedded to heterogeneous low-power high-performance computing systems. In that context, he studies algorithms and implementations of Signal Processing and Machine Learning algorithms (SPML) in several domains including video compression, computer vision and astronomy.