Multiple Transform Selection Concept Modeling and Implementation Using Dynamic and Parameterized Dataflow Graphs

Naouel Haggui, Fatma Belghith, Wassim Hamidouche, Nouri Masmoudi, Jean-François Nezan

Noname manuscript No.
(will be inserted by the editor)

Multiple Transform Selection concept modeling and implementation using Dynamic and Parameterized Dataflow Graphs

Naouel HAGGUI · Fatma BELGHITH · Wassim HAMIDOUCHE · Nouri MASMOUDI · Jean-François NEZAN

Received: date / Accepted: date

Abstract The new video coding standard, Versatile Video Coding (VVC), released by the end of 2020 has increased the coding complexity both at encoder and decoder sides. This complexity increase is due to several coding tools proposed to enhance the coding efficiency. One of these tools is the Multiple Transform Selection (MTS) concept, a new approach for the transform unit. This paper aims at providing a new optimization of the MTS based on dataflow modeling. The proposed approach takes benefit of the different parallelism levels of the MTS in order to create an optimized multicore implementation. Also, this paper study the impact of the dataflow model granularity and the dynamic reconfiguration on the implementation efficiency on x86 multicore architectures. The PREESM tool is used in this study to develop the proposed dataflow models and for the granularity analysis. The dynamic reconfiguration study is here performed using the SPIDER runtime optimized for the multicore execution of applications modeled using Parameterized and Interfaced Synchronous Dataflow (PiSDF) dataflow graphs. Two architectures were used in this work: an x86 architecture with 4 cores and an x86 architecture with 24 cores. The results show that the SPIDER overhead time is almost negligible (0.05%) compared to the execution time of the application. Furthermore, a speed-up of 3.9 and up to 22 for all block sizes was achieved using a 4-core and 24-core machine, respectively.

Keywords Versatile Video Coding (VVC) · Multiple Transform Selection (MTS) · PREESM · SPIDER

N. HAGGUI
Electronics and Information Technology Laboratory (LETI) of Sfax and Univ Rennes, INSA Rennes, CNRS, IETR - UMR 6164, Rennes Campus de Rennes,5 Avenue de la Boulaie, 35510 Cesson-Sévigné
Tel.: +33753154885
E-mail: nawel.hagui@enis.tn

F. BELGHITH
Electronics and Information Technology Laboratory (LETI) of Sfax, Road of Soukra, Sfax, Tunisia, 3038

W. HAMIDOUCHE
Univ Rennes, INSA Rennes, CNRS, IETR - UMR 6164, Rennes

N.MASMOUDI
Electronics and Information Technology Laboratory (LETI) of Sfax, Road of Soukra, Sfax, Tunisia, 3038

J.NEZAN
Univ Rennes, INSA Rennes, CNRS, IETR - UMR 6164, Rennes

1 Introduction

Since 1988, the appearance of the first MPEG-1 standard, a variety of MPEG video coding standards has been developed. The process of standardization has always focused on providing efficient coding tools for a wide and easy deployment. The increase of video traffic, which is expected to reach 82% of global Internet traffic by 2022, has led to the appearance of a new video coding standard called Versatile Video Coding (VVC) [11]. This latter saves 40% of the bitrate compared to the previous standard High Efficiency Video Coding (HEVC) for the same visual quality [5], [37]. VVC offers many advantages dealing with higher resolutions, larger color gamut and a variety of different applications. However, the decoding complexity has been doubled compared to HEVC. This increase in complexity is due to the adoption of new efficient coding tools. One of this coding tools is the Multiple Transform Selection (MTS) concept. The MTS concept is based on three transform types (DCT type II (DCT-II),DCT type VIII (DCT-VIII), and DST type VII (DST-VII)).

The encoder executes the three types of transformation and then applies the one resulting with the lower rate distortion for each block to encode. The MTS concept has significantly improved the compression ratio, but it raises the computational complexity for the encoder and the decoder as well [14]. Unfortunately, due to the additional computing complexity, both the execution time and the hardware cost are increased. These facts have led several researchers to propose optimizations for the MTS concept. The majority of the optimizations have focused on reducing the number of required operations (addition, multiplication) or on deducing one transform type from another.
This paper introduces an extended version of [17]. Both papers aim at providing a method for optimizing the MTS concept based on dataflow modeling and studying its efficiency. Dataflow modeling has proven its efficiency many times in the past, especially in the field of signal and image processing. Dataflow modeling is a powerful tool to study data dependencies and to reveal different levels of parallelism. Whereas [17] investigates the use of static dataflow models to optimize and evaluate the MTS at compile time, this paper extends the results using dynamic dataflow models and extends the analysis at runtime. The tool used to create the dataflow models for the MTS concept is called Parallel and Real-time Embedded Executives Scheduling Method (PREESM). PREESM is also used to analyse the graphs and to automatically generate an optimized multi-core algorithm for the MTS. PREESM allows to implement a multicore algorithm characterized by an optimal use of available computing resources (memory, computing units) in days whereas traditional methods take months. Dataflow modeling has been used on previous video coding standards, for example in [38]-[29] a proposal of dataflow models for Mpeg-4 and Advanced Video Coding (AVC) using SynDEx and Orcc has been introduced. To the best of our knowledge, this is the first dataflow model for the MTS concept presented in the VVC standard. The proposed model takes advantage of PREESM features such as memory optimization and automatic pipelining. Three points summarize the contributions of this work:

1. The proposition of an efficient Interface-Based Synchronous Dataflow (IBSDF) graph for the MTS concept.
2. The transformation of the IBSDF graph into a Parameterized and Interfaced Synchronous Dataflow (PiSDF) graph in order to study the different challenges of dynamic behaviour on the application performance, also details the extra benefits that a PiSDF graph provides compared to an IBSDF graph.
3. The generation of a multicore optimized algorithm for the MTS concept.

The rest of this paper is organized as follows. Section 2 presents a background on the design of the MTS concept and the state of the art of its hardware and software optimizations. It also introduces the importance of using dataflow modeling in embedded systems and gives an overview of the PREESM framework. Section 3 studies the impact of granularity on application performance while using an x86 architecture and based on the IBSDF graph. A discussion on the parallelization possibilities of the MTS concept was also provided. In the Section 4, a proposal for a PiSDF graph is introduced. Moreover, a study of its efficiency using x86 architecture is detailed. Finally, Section 5 concludes the paper.

2 Background

This section encompasses the different optimizations made for the MTS concept in a first part and, in a second part, it introduces the benefits of dataflow modeling on the application performance, as well as the different dataflow models (IBSDF and PiSDF) and the different tools used to create these models.

2.1 Multiple Transform Selection Concept

The transform unit has been the focus of different researchers because of the computing complexity it contains. Several optimizations have been proposed, both in terms of software and hardware. This section concentrates on the optimizations of the MTS concept. The MTS has been introduced in the VVC standard [22]. The reason to use several types of transforms is based on an awareness that using a single transform to model the different statistical variations that might be contained in an intra prediction residual block is not efficient [40]. MTS allows the VVC encoder to choose the transform that minimizes the rate distortion among predefined trigonometric transforms, including DCT-II, DCT-VIII and DST-VII. The MTS concept is based on three features which are the butterfly decomposition for DCT-II, the derivation of DCT-VIII from DST-VII and vice versa, and the zeroing-out of high frequency coefficients for block of size 64×64 for DCT-II and of size 32×32 for DST-VII and DCT-VIII.

The majority of the improvements made to the MTS concept, whether hardware or software, are based on minimizing the computational complexity (addition, multiplications) or on the derivation of one transform type

from another. In [32], a link between the two types of transformation DCT-II and DST-VII was found by Reznik. Authors in [18], provides a detailed study of two approximations; the first for DCT-VIII and the second for DST-VII. The first approximation explored the link between DCT-VIII and DST-VII. In fact, DCT-VIII can be inferred from the DST-VII without causing additional computational complexity, based on the vector reflection matrix Γ and the sign change matrix Λ as depicted in the following equation 1

$$C_8 = \Lambda \cdot S_7 \cdot \Gamma, \tag{1}$$

where C_8 and S_7 present respectively the transforms coefficient matrices of DCT-VIII and DST-VII. The vector reflection matrix Γ and the sign change matrix Λ are derived using the equations 2 and 3, respectively.

$$\Gamma_{i,j} = \begin{cases} 1 \text{ if } j = N - i + 1 \\ 0 \text{ otherwise.} \end{cases} \tag{2}$$

$$\Lambda_{i,j} = \begin{cases} (-1)^{i-1} \text{ if } j = i \\ 0 \qquad\quad \text{otherwise.} \end{cases} \tag{3}$$

with $i, j \in \{1, 2, \ldots, N\}$ and N is the transform size. The second approximation is based on the link between the DST-VII and the DCT-II. In fact, the estimated DST-VII can be obtained using DCT-II as the expression 4 shows.

$$S_7 = \Lambda \cdot C_2^T \cdot \Gamma \cdot A, \tag{4}$$

where $S_7 = \Lambda \cdot C_2^T$ is equivalent to the DST-III transform, the matrices Λ and Γ are calculated by the equations (2) and (3), respectively, and A being a sparse banded matrix. Due to this approximation, the reduction of computational complexity is ensured as it is based on the DCT-II algorithm which is known by fast computation. Actually, DCT-II forms the base of several approximations due to the symmetry and recursion properties that it offers [36]-[2]. In [39], a method based on the factorization of fast implementation algorithms was adopted in order to minimize the number of required operations by the matrix multiplication. There are three properties of the DST-VII. These properties make it possible to factorize many multiplications, which have the same absolute value coefficient in the same baseline, in a single multiplication operation. In [6] a $2D$ implementation of the HEVC DCT transformation has been introduced. The proposed design takes advantage of multiple hardware resources, such as multipliers, DSP blocks and memory blocks, to reduce the logic utilization. This reconfigurable architecture can handle blocks of size up to 32×32. The proposed architecture has been implemented in diverse Field Programmable Gate Arrays (FPGA) platforms. The result showed that the design could support $4Kp30$ video encoding with a reduced hardware cost. In [13] a pipelined $1D$ hardware implementation of the Adaptive Multiple Transform (AMT) for all block sizes up to 32×32 have been proposed. The design has been implemented on various FPGA chips based on several Read Only Memory (ROM) in order to store the matrices of transform coefficients. Despite of its low hardware cost, the design can only support $1D$ AMT design or the transform process is based on $2D$ operations which can be more complex. In [25], a hardware implementation was proposed by investigating two hardware methods with a fixed throughput of 8 pixels/cycle. The first employs separate data paths and the second takes into account reconfigurable data paths for all $1D$ transforms. Although, the efficiency of the proposed implementation is limited to a block size of 8×8. However, larger block size transforms are more complex and require more resources. In [23], a hardware implementation of $2D$ has been proposed but it does not support blocks of size 64×64. In [12] another $2D$ hardware implementations for DCT-VIII and DST-VII has been introduced. Despite being the first implementation that takes advantage of asymmetric blocks, it does not support the DCT-II transformation. There are some approximations for the MTS transforms that were introduced in [33], [34], [31], and [24]. The proposed implementations aim at using the transform DCT-II to approximate DST-VII and DCT-VIII. All optimizations use the same concept, except that the contribution of [24] offers less complexity with almost the same coding efficiency.

In this paper a new software optimization for the MTS concept based on dataflow modeling is introduced. Unlike the state of the art optimizations, the proposed optimization does not focus on reducing the number of computing operations neither referring one transform type from another. Moreover, it considers all transform sizes. It aims at revealing the independency between tasks and explore parallelism in order to speed-up the application performance.

2.2 Dataflow modeling

Dataflow modeling is a powerful tool to capture numerous data dependencies and to explore parallelism between cores, thanks to the additional insight into the application it provides. Besides, dataflow modeling allows targeting different architectures with the same dataflow model. Dataflow is used extensively in Digital Signal Processor (DSP) applications and it has proven to be effective in facilitating the exploration of the various stages of parallelism [3]. In addition, the dataflow

modeling is commonly used in the design of Multiprocessor Systems-on-Chip (MPSoC)[30]-[1]. The complexity of the video coding standards has increased in the last decades in order to increase the video quality and the user experience. Parallel architectures provide suitable solutions to cope with this complexity at the cost of longer software developments. The reduction of the software development effort demands the use of a software tool capable of automatically generating an optimized multicore implementations. In this paper, we evaluate software tools based on dataflow models.
The tool used for creating the dataflow model is called PREESM. PREESM is an Eclipse-based open-source framework that generates C code for embedded multicore systems from dataflow models of the application. Other tools have been employed to create a dataflow model for some previous video coding standards, such as Open RVC-CAL Compiler (Orcc) and SynDEx [38]-[29]. Both PREESM and SynDEx are based on the Algorithm-Architecture Matching (AAM) methodology. Yet, unlike PREESM, SynDEx is not open source. SynDEx is based on a very specific dataflow model, doesn't support schedulability analysis[16] and its mapping scheduling algorithm is too complex to deal with real use cases. Orcc is an open source tool based on a single dataflow language which is RVC-CAL [15]. PREESM and Orcc do not have the same MoC for the algorithm description. In fact, PREESM uses the predictable PiSDF MoC [4] whereas, the Orcc language is not unpredictable providing results that may differ between two runs. In consequence, the code generated by Orcc has no guarantee in terms of deadlocks and memory requirements.
In an attempt to implement the concept of MTS using PREESM, three items must be developed: the algorithm graph, the architecture graph and the scenario. These items are developed using a graphical interface. In fact, this is the same approach used in Simulink for Matlab or in Labview from National Instruments. Another method that could be used to create these inputs is to use the Higher order Coordination Language (HoCL) [35]. The execution is afterwards based on the tasks used in the workflow graph. This section introduces a complete and detailed description of each element.

Two types of algorithm graphs are supported in PREESM which are PiSDF and IBSDF. They are used to model respectively dynamic and static applications. In fact IBSDF is a static specialization of PiSDF. Both algorithm graphs aim at describing the concept of application with the help of actors and First In First Out (FIFO)s. The FIFOs are used to ensure the communications between the actors. FIFOs control the firing rules and the production/consumption rate between the actors. Each type of algorithm graph has its pros and cons. For example, the IBSDF graph improves the predictability of the dataflow model. In addition, it allows the use of some powerful optimizations at compile-time like the memory footprint reduction and the automatic pipelining. On the downside, the IBSDF graph limits the number of application behaviors that can be modeled. For example, a static graph is not suitable for modeling an algorithm where the execution of certain actors can be turned on or off at runtime, depending on the value of a data token provided by a sensor. The PiSDF model enables this kind of dynamic behaviour giving the possibility of changing the parameter's values at runtime thus, allowing the application to be reconfigurable. In fact, the PiSDF semantics provides a range of parameters and parameter dependencies that allows the change of the consumption and production rates of actors [8] at runtime. Figure 1 shows the different semantics for IBSDF and PiSDF models. To run the IBSDF model, the only required tool is PREESM; whereas, to run the PiSDF model, an additional tool called Synchronous Parameterized and Interfaced Dataflow Embedded Runtime (SPIDER) must be used. SPIDER is a dataflow based runtime used to support the execution of dynamic PiSDF applications. SPIDER is based on a master/slave structure in order to facilitate its porting on different multicore architectures [26].
The kind of architecture graph adopted in PREESM is called System-Level Architecture Model (S-LAM). It is a combination of cores related to a shared memory in order to communicate.
The scenario presents the control unit since it contains all the parameters under which the application will perform. The multicore execution is optimized in two steps. In the first one, the scheduling algorithm considers by default that the execution times of all actors are identical. These inaccurate values usually result in an inefficient execution. Therefore, the generated code automatically integrates accurate timing measurements the user report in the scenario for the second step. In the second step, the accurate timing of each actor is used to optimize the multicore mapping scheduling and the generared code.
The workflow is a unit that gathers several tasks. Some of these tasks are responsible of generating the multicore optimized algorithm and others to perform optimizations such as memory optimizations, and automatic pipelining. The automatic pipelining task serves at introducing delays in the critical path in order to increase the throughput of the application on the multicore architecture. [21] details the optimizations provided by the automatic pipelining tool. The memory optimization task provided by PREESM is called the

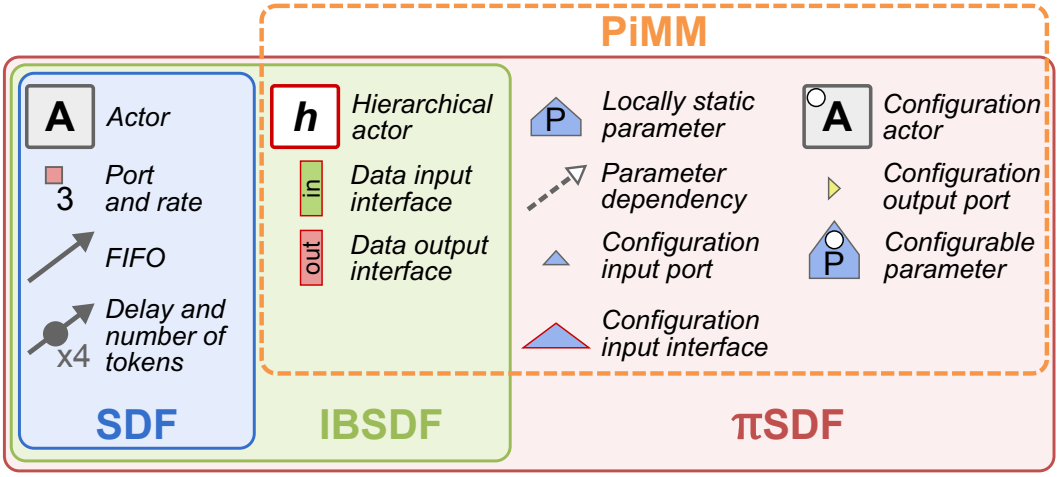


Fig. 1: PiMM semantics [7]

memory footprint reduction. The authors proved in [9] the efficiency of the memory footprint reduction task on a computer vision application, allocating 34% less memory than the state-of-the-art approaches.

3 IBSDF models and granularity impact

In order to study the efficiency of PREESM tool to explore the parallelism of the MTS concept, two models with different granularity have been developed. The granularity is an important parameter in dataflow models as it affects the performance of the execution. The first IBSDF model shown in Figure 2 characterized with fine grain granularity and the second one shown in Figure 3 with a coarse grain granularity. Each model is created based on different levels of parallelism. A first level exposes the parallelism between the MTS concept tasks, and a second level exposes the exploration of the independency between the blocks to be transformed. This section deals with the result given by each model.

3.1 Fine grain granularity IBSDF model

The fine grain model (Figure 2) is composed of two actors with different structure.
The *Blocks_to_be_transformed* actor is a low level actor. On the other hand, the *MTS_concept* actor is a hierarchical actor. It encompasses multiple actors internally as illustrated in Figure 4. Thus, the computational process is divided into several actors. As a result, the execution time of each actor is minimized so that a fine grain granularity model. The fine grain IBSDF model relies on exploring parallelism between MTS concept tasks (DCT-II, DST-VII, DCT-VIII). Running the application on an x86 machine that has 4 cores yielded a lower execution time on a single core than on 2, 3 and 4 as shown in Table 1. This result introduces a contradiction. In fact, the exploration of parallelism aims at reducing the execution time while increasing the number of cores used. There are several possible reasons for this result. In fact the computing time of the actors is too low compared to the time taken by the tasks of the operating system. In fact, for a fine grain IBSDF model, every process that the operating system runs, including thread creation, communication between tasks, synchronization and memory allocation, affects the performance of the application. Indeed, synchronizations in this model becomes very expensive. For example, PREESM calls barriers at the end of each iteration. Each call takes between 400 nanoseconds, which is not insubstantial at the scale of the actors used for this model. Therefore, in order to ensure the efficiency of the model, the computing time of the actors must be greater than the time required for data transfer and synchronization. In conclusion, the fine grain granularity model is not efficient while using x86 architecture but it could be very useful for multicore embedded systems (FPGA, Kalray,...). Actually, according to the study presented in [19] the fine-grained model has proven its efficiency using Kalray MPPA architecture. In fact, at the intra-cluster level, a fine granularity model is necessary because a large number of tasks must be executed in parallel. Similarly, fine granularity is needed on the FPGA to feed its large number of cores fine granularity explodes the parallelism and gives different levels of parallelism.
The fine grain model could leads to a better performance if the MTS concept is applied on multiple blocks

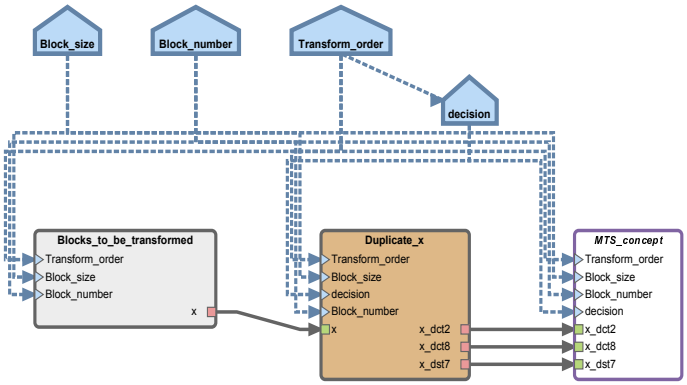


Fig. 2: MTS fine grain IBSDF model

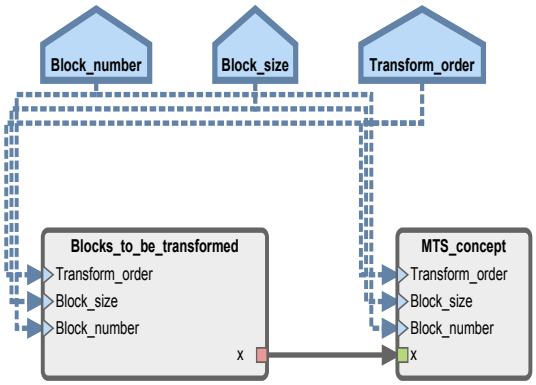


Fig. 3: MTS coarse grain IBSDF model

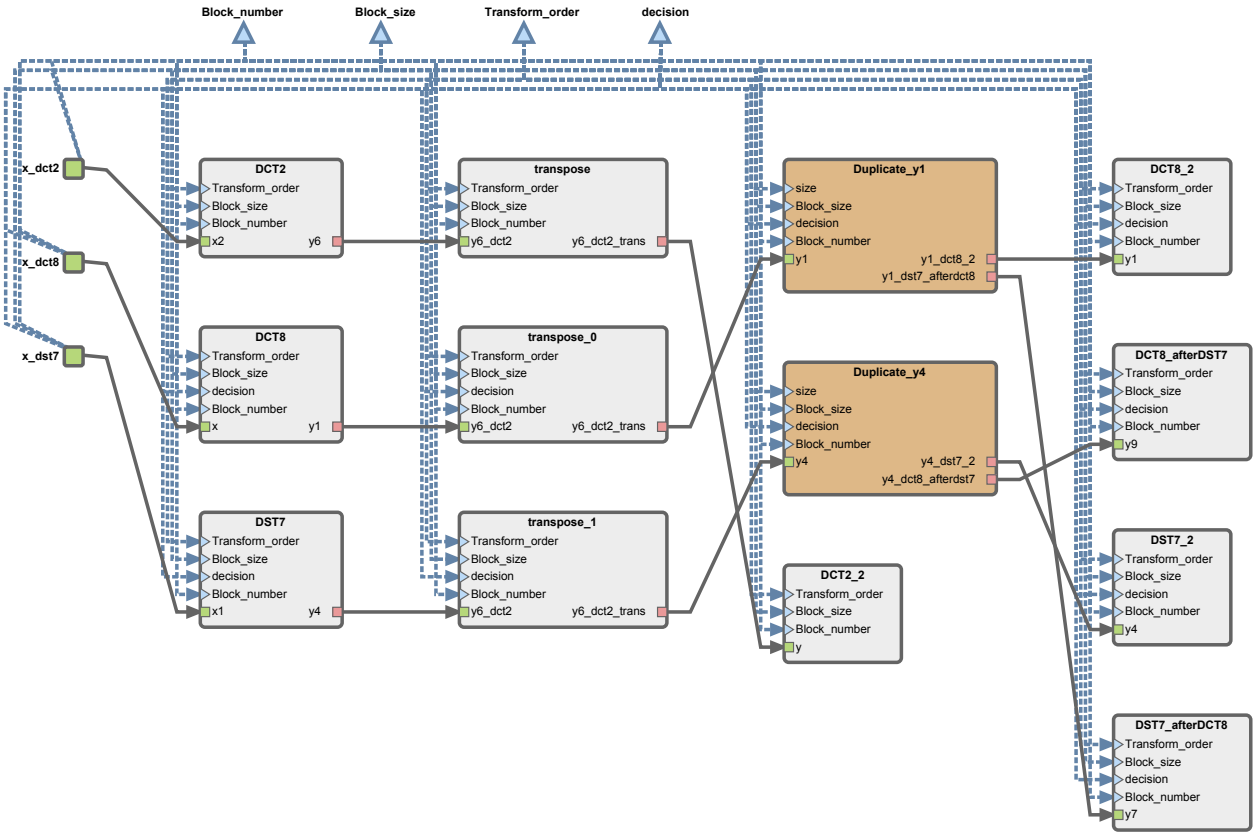


Fig. 4: The inside of the *MTS_concept* actor

at the same time, which is a realistic case as a frame contains a huge number of blocks to be transformed and those blocks are independent. Figure 5 illustrates the results of applying the MTS concept while increasing the number of blocks to be transformed, the number of used cores as well as, applying memory footprint reduction and automatic pipelining optimizations. For memory footprint reduction, a memory script has been added to the transpose actors as their input and output buffers can be allocated in a common memory space. The broadcast actors (the duplicate actors) are automatically associated with memory scripts. The two optimizations boost the speed-up of the application. While increasing the number of blocks to be transformed the speed-up of the application is getting closer and closer to the theoretical case so that the possibility of creating an efficient model for the MTS concept while employing a coarse grain granularity. The theoretical result is plotted according to Amdahl's law for

Table 1: Execution time according to the number of used cores

Cores number	Execution time
1	0.796 ns
2	4.269 ns
3	6.287 ns
4	7.235 ns

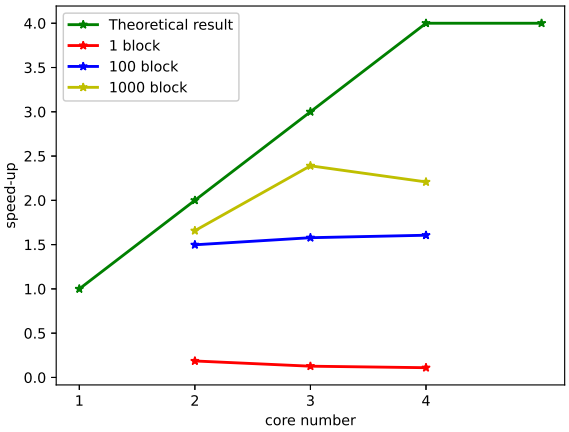


Fig. 5: MTS IBSDF model acceleration according to the number of used cores and while increasing the number of blocks to be transformed

an application that could be parallelized at 100%; this is why the theoretical result is linear $y = x$.

3.2 Coarse grain granularity IBSDF model

Compared to the fine grain IBSDF model, the coarse grain model is constructed with two non-hierarchical actors in order to ensure that the computing time of the actors is greater than the time required for synchronization and data transfer. The coarse grain model is based on exploring the independency between the blocks to be transformed. In other words, several blocks to be transformed are sent at the same time and the MTS module will be applied to all the blocks simultaneously. For the purpose of studying the efficiency of this second IBSDF model, several tests have been carried out. The tests were performed on x86 architecture which contains 4 cores and another one contains 24 cores. For both tests, the experimental results are close to the theoretical performance (See Figs. 6 and 7). Thus, proving the efficiency of the coarse grain IBSDF model on the x86 architecture. In nutshell, the dataflow modeling of the MTS concept turns to be very beneficial while using a coarse grain model on x86 architecture.

To prove the efficiency of the coarse-grained dataflow model over the state-of-the-art approach used to explore parallelism among the system cores. A comparison between the OpenMP approach and the coarse-grained IBSDF model was performed. OpenMP is commonly used, in shared memory systems, such as the architecture used in the presented model. Therefore, it is interesting to carry out such a comparison. OpenMP often offers specific features, such as preprocessing directives or specific instructions via intrinsics, to target specific architectures. When developing with OpenMP, it is necessary to adapt #PRAGMA to the architecture. Often, this is empirically done, by testing the impact of the #PRAGMA parameters on the execution time of the application, which is long and not easy to do. Thus, programming with OpenMP requires a deep understanding of the application, the hardware and the runtime libraries, which takes months to master [27]. However, the coarse grain IBSDF model needs less development time. Also, once the algorithm graph of the application is done, it is easy to switch from one architecture to another without rewriting the code. Figure 8 presents a comparison between the result of running a multicore algorithm created using OpenMP and a multicore algorithm-generated automatically while using PREESM. The results show that the speed-up obtained using the algorithm automatically generated by PREESM is superior to that based on the OpenMP approach. Thus, compared to OpenMP, PREESM builds more efficient model with less development time.

4 MTS PiSDF model

In order to implement the multicore optimized MTS concept based on a coarse grain dataflow model in a real encoder or decoder, some modifications must first be made. In fact, in the dataflow models detailed in Section 3, the parameters that control the transform order (4,8,16,32,64), and the size of the blocks to be transformed ($width \times height$), are modified manually. These parameters cannot be changed at runtime as the presented IBSDF model is a static model. As already detailed in Section 2.2 the static property enhances the predictability of the dataflow model and gives the way to perform powerful application optimizations at compile-time, such as memory optimization. While the static property is an important benefit for compile-time optimizations, this property limits the amount of application behavior that can be modeled. On the encoder and decoder sides, there are several blocks to be transformed with different sizes and different transformation orders. In addition, the data dependencies change over

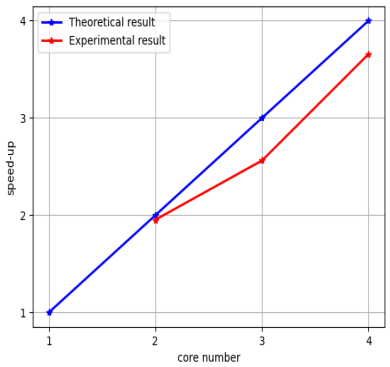


Fig. 6: MTS IBSDF model acceleration according to the number of used cores while using x86 architecture with 4 cores

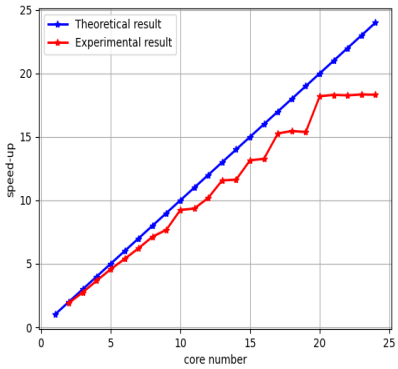


Fig. 7: MTS IBSDF model acceleration according to the number of used cores while using x86 architecture with 24 cores

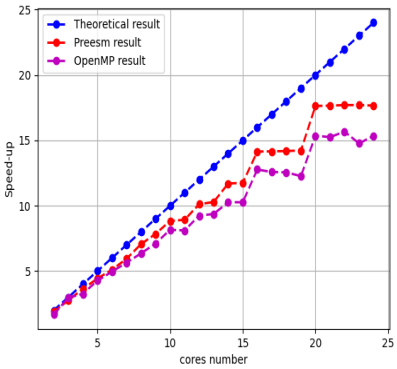


Fig. 8: MTS IBSDF model acceleration compared to OpenMP result

time. Thus, the behavior of the application to be implemented must be dynamic and not static. The coarse grain IBSDF model can be implemented in a real encoder/decoder, but for each combination of block size and transform order, an executable must be created. Then, these executables are controlled via another large one, which is not very convenient. However, by using a PiSDF model, only one executable is needed for all possible combinations.

To create an efficient multicore algorithm for the MTS concept based on PiSDF model is highly required to ensure the efficient dispatching of tasks according to the available processing elements while considering dynamic behavior. For this reason, this section aims to provide a dynamic model of the MTS concept. Also, studying the efficiency of this new model. SPIDER runtime tool has been used to support the execution of this model. When using SPIDER, the application developer only has to provide a dataflow graph and SPIDER takes care of the rest since it is based on pthread. In fact, the developer's work is limited to high-level analysis and manipulation of the application graph (choice of hierarchy level ie. granularity, production/consumption rate, and adding delays between actors). Another powerful aspect of SPIDER is that its optimization leads to high performances. For example, in [20] SPIDER has proved to be able to reduces the execution latency by up to 26% and allows handling multiple executions.

The new model illustrated in figure 9 has an additional actor compared to the two IBSDF graphs already detailed. The additional actor named *parameters_control* as its name indicates, controls the value of the parameters and makes possible their change at runtime, so that a dynamic reconfiguration. The reconfigurable dataflow model promotes a unique trade-off between application dynamicity and predictability, which can be leveraged by a runtime manager (SPIDER) to check application properties or to perform runtime optimizations, such as actor computation mapping [20]. Reconfigurable dataflow models allow actor firing rules to be reconfigured in a non-deterministic way at specific times during the application's execution [28]. The reconfiguration in the PiSDF model is based on parameters. Following the PiSDF execution rules [10], an actor can invoke a reconfiguration of the graph topology and intrinsic parallelism by defining a new parameter value at runtime.

The parameters $Transform\_order$ and $Block\_size$ affect the performance of the application both during compile-time and runtime. Defining a new value for a parameter in a graph at runtime has a strong impact on the way the actors in that graph will be executed. For example, increasing the value of a parameter used

as a production rate by an actor will have an impact on the amount of memory allocated for the output buffer of this actor. By dynamically changing the consumption and production rates of the actors, the number of executions of these actors can be modified, which requires a new mapping of these executions to the processing elements of the architecture. For those reasons an investigation on the impact of the presented PiSDF graph must be performed in order to ensure whether the dynamic coarse grain model still performing on x86 architecture.

The *parameters_control* actor is realized in a manner that provides all possible combinations between the size of the block to be transformed and the order of transformation to be applied. Adding this process to the dataflow model facilitates the implementation of the optimized multicore MTS algorithm in an encoder/decoder in the future. Figure 11 illustrates the result of performing the MTS concept on 80 blocks of different sizes. For all block sizes the execution time on 2, 3, 4 cores are lower than on one core. This result proves the efficiency of the PiSDF model on x86 architecture. Figure 10 illustrates the SPIDER overhead time compared to the execution time of the application according to the number of used cores on an x86 architecture with 4 and 24 cores, respectively. Results show that the SPIDER overhead is too small compared to the execution time for all block sizes. It represents only 0.05% of the entire process. This fact proves the efficiency of the coarse grain model on the x86 architecture and confirms the fact that using a coarse grain model on x86 architecture guarantees that the computation time of the application is higher than the time taken by the operating system. The PiSDF model was also tested on an x86 architecture with 24 cores and with the use of 480 blocks. As shown in figure 12 the experimental results for all block sizes were close to the theoretical result (blue curve). In fact, by increasing the number of cores used, the speed-up of the application increases, which corresponds to the objectives of the proposed PiSDF model.

5 Conclusion

This paper introduced an in-depth case study about the efficiency of applying dataflow modeling on the MTS concept using PREESM. The study was based on two types of algorithm graphs, a static graph IBSDF and a dynamic graph PiSDF. The execution of the PiSDF model requires the use of the SPIDER tool. For both models (IBSDF and PiSDF), the use of dataflow modeling has proven to be effective on the x86 architecture while using a coarse grain model. However, compared to

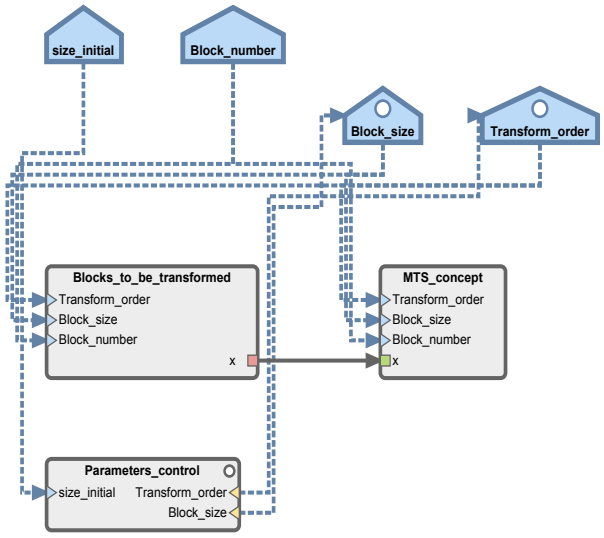
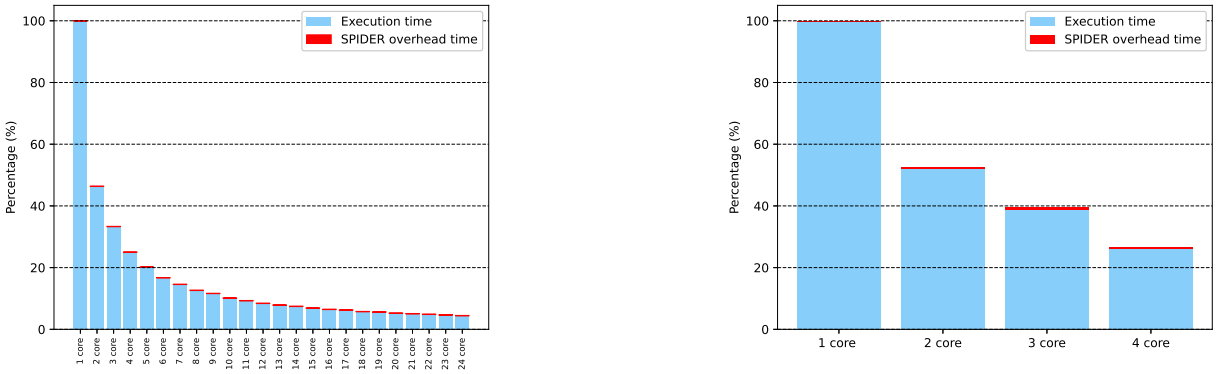


Fig. 9: MTS coarse grain PiSDF model

the IBSDF model, the PiSDF model can be easily implemented in a real encoder or decoder. In fact, PiSDF model can handle different combinations of block sizes and transformation orders using the same generated executable, which is not the case for the IBSDF model. The IBSDF model requires the generation of a new executable for each combination. While running the PiSDF model, the result showed a negligible SPIDER overhead time on both architectures (x86 architecture with 4 cores and with 24 cores). It represents 0.05% of the complete process. This fact proves the efficiency of the PiSDF model on both cases and also proves the possibility to switch from one architecture to another without modifying the model and without rewriting any code, unlike classical approaches like OpenMP or OpenCL. The modeling of the MTS concept revealed two important points. The first one is the importance of choosing the level of granularity according to the architecture used, and the second point is to show the difference between IBSDF and PiSDF graphs and their advantages. In future work, we aim to create a PiSDF model for a complete decoder. Also, to use the dataflow models to generate optimized solutions for heterogeneous and embedded architectures (big little Qualcomm SXR2130P processor, Kalray, ...).

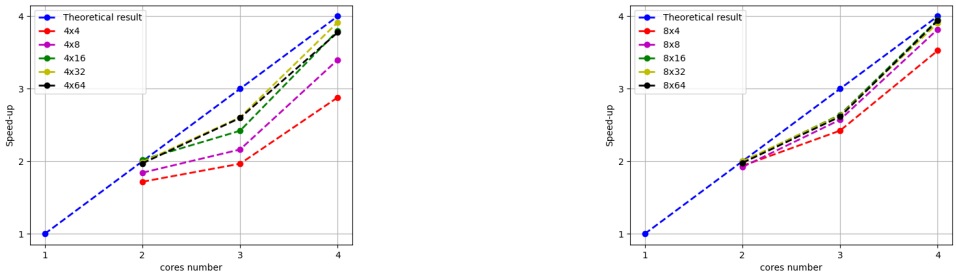
Acknowledgement

This work is supported by the PHC Maghreb project, and within a co-supervised thesis between Institute of Electronics and Telecommunications of Rennes (IETR), and Electronics and Information Technology Laboratory (LETI) of Sfax.

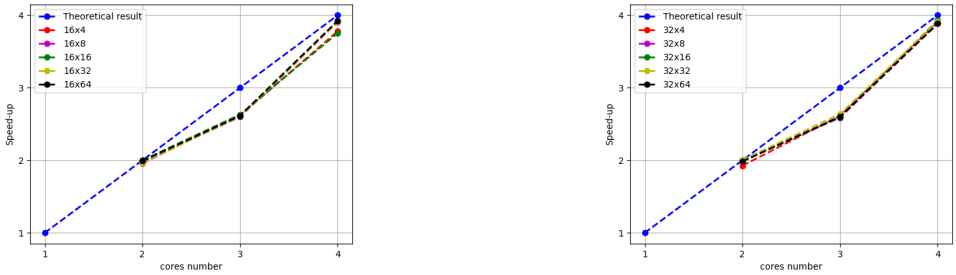


(a) x86 architecture with 24 cores (b) x86 architecture with 4 cores

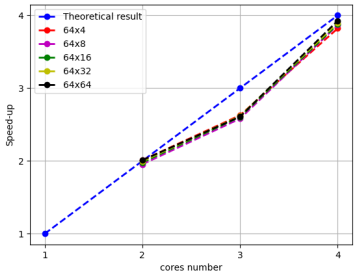
Fig. 10: SPIDER overhead time and execution time according to the number of used cores.



(a) blocks of sizes 4×4, 4×8, 4×16, 4×32,and 4×64 (b) blocks of sizes 8×4, 8×8, 8×16, 8×32,and 8×64

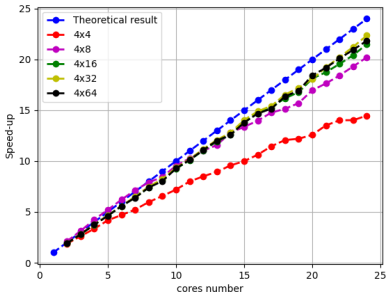


(c) blocks of sizes 16×4, 16×8, 16×16, 16×32,and 32×64 (d) blocks of sizes 32×4, 32×8, 32×16, 32×32,and 32×64

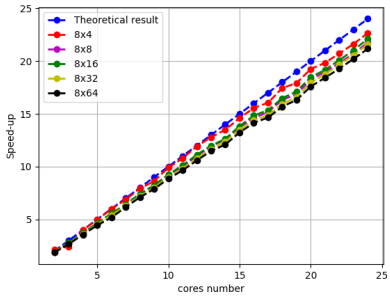


(e) blocks of sizes 64×4, 64×8, 64×16, 64×32, and 64×64

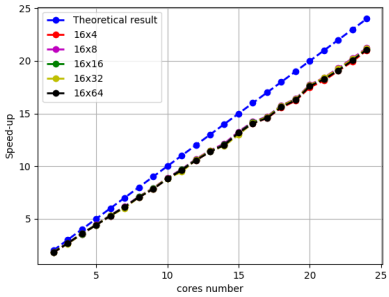
Fig. 11: MTS PiSDF model acceleration according to the number of used cores for all block sizes from 4×4 up to 64×64 while using x86 architecture with 4 cores



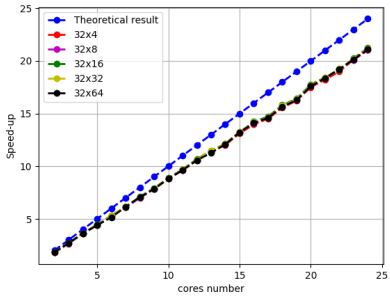
(a) blocks of sizes 4×4, 4×8, 4×16, 4×32,and 4×64



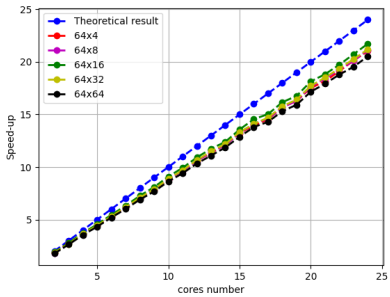
(b) blocks of sizes 8×4, 8×8, 8×16, 8×32,and 8×64



(c) blocks of sizes 16×4, 16×8, 16×16, 16×32,and 32×64



(d) blocks of sizes 32×4, 32×8, 32×16, 32×32,and 32×64



(e) blocks of sizes 64×4, 64×8, 64×16, 64×32, and 64×64

Fig. 12: MTS PiSDF model acceleration according to the number of used cores while using x86 architecture with 24 cores

References

1. Aguilar, M.A., Leupers, R., Ascheid, G., Murillo, L.G.: Automatic parallelization and accelerator offloading for embedded applications on heterogeneous mpsocs. In: Proceedings of the 53rd Annual Design Automation Conference, pp. 1–6 (2016)
2. Ahmed, A., Shahid, M.U., et al.: N point dct vlsi architecture for emerging hevc standard. VLSI design (2012)
3. Bhattacharyya, S.S., Deprettere, E.F., Leupers, R., Takala, J.: Handbook of signal processing systems. Springer (2018)
4. Bhattacharyya, S.S., Levine, W.S.: Optimization of signal processing software for control system implementation. In: 2006 IEEE Conference on Computer Aided Control System Design, 2006 IEEE International Conference on Control Applications, 2006 IEEE International Symposium on Intelligent Control, pp. 1562–1567. IEEE (2006)
5. Bossen, F., Li, X., Norkin, A., Sühring, K.: Jvet-o0003. JVET AHG report: Test model software development (AHG3) (2019)
6. Chen, M., Zhang, Y., Lu, C.: Efficient architecture of variable size hevc 2d-dct for fpga platforms. AEU-International Journal of Electronics and Communications **73**, 1–8 (2017)
7. Desnos, K.: Memory study and dataflow representations for rapid prototyping of signal processing applications on mpsocs. Ph.D. thesis, INSA de Rennes (2014)
8. Desnos, K., Heulot, J.: Pisdf: Parameterized & interfaced synchronous dataflow for mpsocs runtime reconfiguration. In: 1st Workshop on MEthods and TOols for

Dataflow PrOgramming (METODO) (2014)
9. Desnos, K., Pelcat, M., Nezan, J.F., Aridhi, S.: Buffer merging technique for minimizing memory footprints of synchronous dataflow specifications. In: 2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 1111–1115. IEEE (2015)
10. Desnos, K., Pelcat, M., Nezan, J.F., Bhattacharyya, S.S., Aridhi, S.: Pimm: Parameterized and interfaced dataflow meta-model for mpsocs runtime reconfiguration. In: 2013 International Conference on Embedded Computer Systems: Architectures, Modeling, and Simulation (SAMOS), pp. 41–48. IEEE (2013)
11. Domínguez, H.O., Rao, K.R.: Versatile Video Coding Latest Advances in Video Coding Standards, pp. i–xxx (2018)
12. Fan, Y., Zeng, Y., Sun, H., Katto, J., Zeng, X.: A pipelined 2d transform architecture supporting mixed block sizes for the vvc standard. IEEE Transactions on Circuits and Systems for Video Technology **30**(9), 3289–3295 (2019)
13. Garrido, M.J., Pescador, F., Chavarrias, M., Lobo, P.J., Sanz, C.: A high performance fpga-based architecture for the future video coding adaptive multiple core transform. IEEE Transactions on Consumer Electronics **64**(1), 53–60 (2018)
14. Garrido, M.J., Pescador, F., Chavarrías, M., Lobo, P.J., Sanz, C., Paz, P.: An fpga-based architecture for the versatile video coding multiple transform selection core. IEEE Access **8**, 81887–81903 (2020)
15. Gorin, J., Raulet, M., Prêteux, F.: Mpeg reconfigurable video coding: From specification to a reconfigurable implementation. Signal Processing: Image Communication **28**(10), 1224–1238 (2013)
16. Grandpierre, T., Lavarenne, C., Sorel, Y.: Optimized rapid prototyping for real-time embedded heterogeneous multiprocessors. In: Proceedings of the seventh international workshop on Hardware/software codesign, pp. 74–78 (1999)
17. Haggui, N., Belghith, F., Hamidouche, W., Masmoudi, N., Nezan, J.F.: 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 (2021)
18. Hamidouche, W., Philippe, P., Mohamed, C.E., Kammoun, A., Menard, D., Déforges, O.: Hardware-friendly dst-vii/dct-viii approximations for the versatile video coding standard. In: 2019 Picture Coding Symposium (PCS), pp. 1–5. IEEE (2019)
19. Hascoët, J., Desnos, K., Nezan, J.F., de Dinechin, B.D.: Hierarchical dataflow model for efficient programming of clustered manycore processors. In: 2017 IEEE 28th International Conference on Application-specific Systems, Architectures and Processors (ASAP), pp. 137–142. IEEE (2017)
20. Heulot, J., Pelcat, M., Desnos, K., Nezan, J.F., Aridhi, S.: Spider: A synchronous parameterized and interfaced dataflow-based rtos for multicore dsps. In: 2014 6th European Embedded Design in Education and Research Conference (EDERC), pp. 167–171. IEEE (2014)
21. Honorat, A., Desnos, K., Dardaillon, M., Nezan, J.F.: A fast heuristic to pipeline sdf graphs. In: International Conference on Embedded Computer Systems, pp. 139–151. Springer (2020)
22. JVET: Algorithm Description of Joint Exploration Test Model 7(JEM7), MPEG document N17055. ITU-TVCEG (Q6/16) and ISO/IEC MPEG (JTC 1/SC 29/WG 11), July 2017
23. Kammoun, A., Hamidouche, W., Belghith, F., Nezan, J.F., Masmoudi, N.: Hardware design and implementation of adaptive multiple transforms for the versatile video coding standard. IEEE Transactions on Consumer Electronics **64**(4), 424–432 (2018)
24. Lorcy, V., Philippe, P.: Ce6: Further simplification of amt with adjustment stages (test ce6. 1.6 b). In: Document JVETL0135-v1 12th JVET Meeting: Macao, CN (2018)
25. Mert, A.C., Kalali, E., Hamzaoglu, I.: High performance 2d transform hardware for future video coding. IEEE Transactions on Consumer Electronics **63**(2), 117–125 (2017)
26. Miomandre, H., Hascoët, J., Desnos, K., Martin, K., de Dinechin, B.D., Nezan, J.F.: Demonstrating the spider runtime for reconfigurable dataflow graphs execution onto a dma-based manycore processor. In: IEEE International Workshop on Signal Processing Systems (2017)
27. Miomandre, H., Hascoët, J., Desnos, K., Martin, K.J., de Dinechin Kalray, B.D., Nezan, J.F.: Embedded runtime for reconfigurable dataflow graphs on manycore architectures. In: Proceedings of the 9th Workshop and 7th Workshop on Parallel Programming and RunTime Management Techniques for Manycore Architectures and Design Tools and Architectures for Multicore Embedded Computing Platforms, pp. 51–56 (2018)
28. Neuendorffer, S., Lee, E.: Hierarchical reconfiguration of dataflow models. In: Proceedings. Second ACM and IEEE International Conference on Formal Methods and Models for Co-Design, 2004. MEMOCODE'04., pp. 179–188. IEEE (2004)
29. Nezan, J.F.: Prototypage rapide d'applications de traitement des images sur systèmes embarqués. Ph.D. thesis (2009)
30. Pelcat, M., Desnos, K., Heulot, J., Guy, C., Nezan, J.F., Aridhi, S.: 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 (2014)
31. Philippe, P., Lorcy, V.: Further simplification for amt complexity reduction (ce6. 1.2). In: Document JVET-K0299 11th JVET Meeting: Ljubljana, SI (2018)
32. Reznik, Y.A.: Relationship between dct-ii, dct-vi, and dst-vii transforms. In: 2013 IEEE International Conference on Acoustics, Speech and Signal Processing, pp. 5642–5646. IEEE (2013)
33. Said, A., Egilmez, H., Seregin, V., Karczewicz, M.: Complexity reduction for adaptive multiple transforms (amts) using adjustment stages. In: Document JVET-J0066 10th JVET Meeting: San Diego, CA, USA (2018)
34. Said, A., Egilmez, H., Seregin, V., Karczewicz, M., Seregin, V.: Efficient implementations of amt with transform adjustment stages. In: Document JVET-K0272 11th JVET Meeting: Ljubljana, SI (2018)
35. Sérot, J.: Hocl: High level specification of dataflow graphs. In: IFL 2020: Proceedings of the 32nd Symposium on Implementation and Application of Functional Languages, pp. 11–22 (2020)
36. Shen, S., Shen, W., Fan, Y., Zeng, X.: A unified 4/8/16/32-point integer idct architecture for multiple video coding standards. In: 2012 IEEE International Conference on Multimedia and Expo, pp. 788–793. IEEE (2012)
37. Sidaty, N., Hamidouche, W., Philippe, P., Fournier, J., Déforges, O.: Compression Performance of the Versatile

Video Coding: HD and UHD Visual Quality Monitoring. Picture Coding Symposium (PCS) (2019)
38. Yviquel, H.: From dataflow-based video coding tools to dedicated embedded multi-core platforms. Ph.D. thesis, Rennes 1 (2013)
39. Zhang, Z., Zhao, X., Li, X., Li, Z., Liu, S.: Fast adaptive multiple transform for versatile video coding. In: 2019 Data Compression Conference (DCC), pp. 63–72. IEEE (2019)
40. Zhao, X., Zhang, L., Ma, S., Gao, W.: Rate-distortion optimized transform for intra-frame coding. In: 2010 IEEE International Conference on Acoustics, Speech and Signal Processing, pp. 1414–1417. IEEE (2010)