Video Codec Application Scheduling and Optimization Based on DAG and GGEN algorithms

[1]Afef SALHI, [2]Fahmi GHOZZI and [3]Ahmed FAKHFAKH and [4]Alix MUNIER KORDON
[1,4] UPMC-Informatics Laboratory of Paris6 (LIP6), salhiafefge@gmail.com, alix.munier@lip6.fr
[2,3] ENET'Com, University of Sfax, Tunisia,fahmi.ghozzi@enetcom.rnu.tn, ahmed.fakhfakh@enetcom.rnu.tn
[1,2,3]Digital Research Center of SFAX (CRNS), Laboratory of Technology for Smart Systems (LT2S), TUNISIA

Abstract—Several techniques have been recently proposed to adapt video codec H264 applications to existing many core platforms. Among these techniques, the generation and automatic online of DAG algorithm : GGEN methods have been proposed that learn how to adapt at run-time the throughput and resources allocated to the various video codec H264 tasks depending on dynamically changing data video codec characteristics and the desired applications performance (e.g., accuracy). However, most of state-of-the-art techniques consider only one single Motion Estimation "ME" block input in its application model input and assume that the system knows the amount of resources to allocate to each task to achieve a desired performance. To address these limitations, in this paper we propose a new automatic and efficient methodology and associated algorithms for online directed acyclic graph-efficient scheduling of ME block applications with multiple streams on many core systems with resource constraints. Moreover, our scheduler is able to detect overlapping of the tasks, the communications problems between the tasks and to smoothly adapt the scheduling strategy. Our experiments realized on a chain of tasks modeling ME block application demonstrate that our scheduler is able to learn the scheduling policy and to adapt it such that it minimizes the targeted Time To Market TTM as the ME block characteristics in video codec are dynamically changing in Multi-Processor System on Chip "MPSoC" and System on Chip "SoC" system.

Keywords—BMA; ME block; Video Codec application; Embdedd system; SoC; MPSoC; partitioning and scheduling tasks; GGEN; DAG.

I. Introduction

Video codec H264 are now widely used in several domains such as social media analysis, military image and video, video annotation, surveillance and medical applications, systems of intelligent transport. These applications are characterized with stringent delay constraints, the execution time for tasks, increasing parallel computation requirement and a highly variable stochastic input data stream which have direct impact on the application complexity and the Time To Market "TTM". Moreover, they need to adapt to dynamically changing ME block in video codec characteristics and require dynamic data-driven topology graphs of tasks in order to efficiently process them [1], [6], as well as high throughput efficiency which may require parallel tasks processing. For instance, video codec applications, one of the main ME block computing applications, are used to classify a high input of tasks and in general modeled using GGEN algorithm. Different types of tasks in ME block are collected from various MPSoC or SoC and multiple types of classifiers are applied on these tasks to data to uncover hidden instruction or tasks and video codec applications. In order to adapt to the MPSoC or SoC platform of the ME block, each stage may integrate different type of classifiers or quality levels and a selection of the GGEN algorithm is realized at run-time with respect to the tasks type of ME block. The complexity of each task in each stage of the ME block in chain video codec may change at run-time with respect to the type of processed input data which is unknown by the application. Numerous hardware "HW" and software "SW" solutions have been proposed in order to cope with the increasing complexity and computation requirement of video codec applications. At the hardware, several core architectures [1], [7] have been developed to increase the parallelization level and to support the video codec application model. At the software, several platforms MPSoC and SoC based ARM Cortex A9MP in OVP are implemented DAG and GGEN algorithm scheduling adapted with ME block in H264 video codec.

A. Related Work

1) Generate the DAG algorithm: In table I, we summarize the different application methods considered by existing DAG analyzers and we compare them to the DAG approaches of our solution.

TABLE I
THE COMPARAISON OUR SOLUTION WITH DIFFERENT APPROACHS SCHEDULING TASKS.

DAG solution	Types of deadlines	Types of DAG	Analysis
[11]	Independent	Periodic	Offline
[6], [7]	Independent	Periodic	Offline
[14]	Dependent	H264	Online
Our sollution	Dependent	General	Online
[13]	Dependent	General	Online
[1]	Independent	Periodic	Offline

Existing static approaches [1], [6], [7] model the application as the DAG with periodic and dependence tasks as present in figure 1. The solution propose limits to the problem of NP−Complet (No Complet) from scheduling and partitioning. Hence, these approaches are unsuitable in multimedia and telecommunication applications with the dynamic DAG. In fact, this solution of a periodic DAG limits the applicability of static approaches. Proposing to apply the pipelining is not possible, especially for multimedia and telecommunication applications. Moreover, for the H264 video codec, the

pipelining technique buffers delays which are proportional to the Group of Pictures (GoP) size. Finally, static approaches rely on worst-case (granularity), estimation for the execution time. A scheduler is an algorithm allocating a set of tasks resources (processors, machinery, etc), with the objective of optimizing one or more performance criteria. Imagine that you come to develop a new scheduling algorithm.

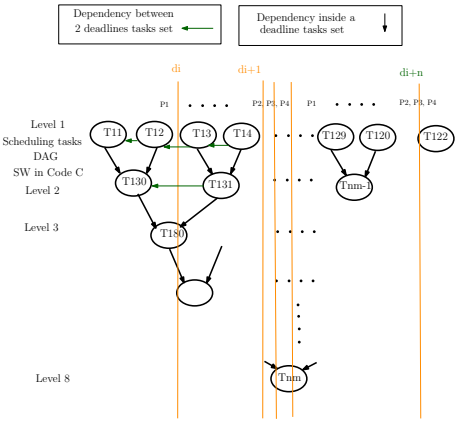

Fig. 1. Dependence tasks in Ggen to TPG-DAG.

To have qualitative information about your algorithm and compare it to others works. This scheduler has given many performances in the characteristics video processing. Then, it limits the problem in dependencies tasks in the same processor or different ones. Hence, this scheduler makes the tracking and monitoring tasks in the different level in TPG graph. The figure 2 present this methodology of scheduling and partitioning tasks, and the dependency from the tasks with different deadline times.

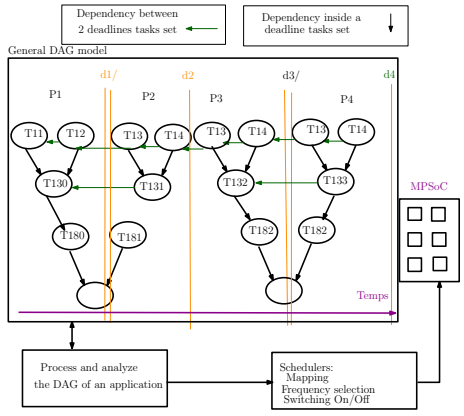

Fig. 2. The blocks of Ggen in TPG-DAG with dependencies tasks.

2) H264 encoder-energy deadlines miss rate and overhead: For many application multimedia and telecommunication benchmark, we have used the code "HM" in codec HEVC/H265 [9], [12]. Then, other researches used software joint "JM" of an H264 codec video [1], [4], [8], [10]. Our application in video codec H264 is ME blocks. Hence, we propose the scheduling and partitioning algorithm tasks (TPG-DAG) and the deadlines configuration, we consider the ME blocks in our application as similar to the one shown in 3 3 with different tasks in two frames, 8 slices per frame and 25 frames per second. For scheduling and partitioning tasks in MPSoC systems, we give 4 sequences S1, S2, S3, S4. These sequences are illustrated in figure 3. We selected 4 deadlines t1, t2, t3, t4.

For scheduling and partitioning tasks in MPSoC systems, we give 4 sequences S1, S2, S3, S4 as illustrated in figure 4. We selected 4 deadlines th1, th2, th3 and th4. So, there is the waiting time of tasks and the processing time of the tasks.

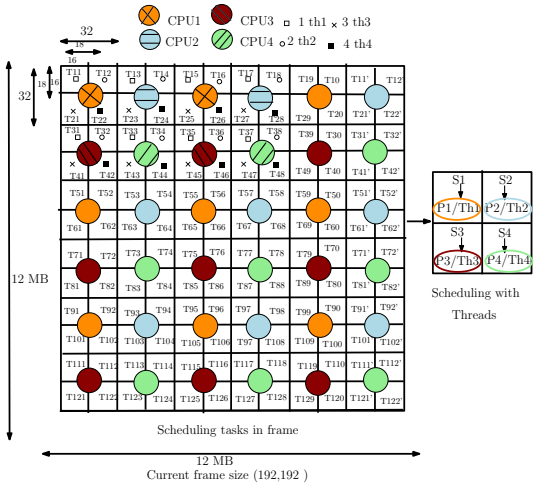

Fig. 3. Scheduling task sequences of ME algorithm.

If the tasks in sequence S1 depends on tasks in sequence S3, we have other deadlines time. So, there is the waiting time of tasks and the treatment time of the tasks. We have other parameter values: cost communication from the parent node ni, i and child node ni, j. It is applicable to other tasks in both sequences S2, S4.

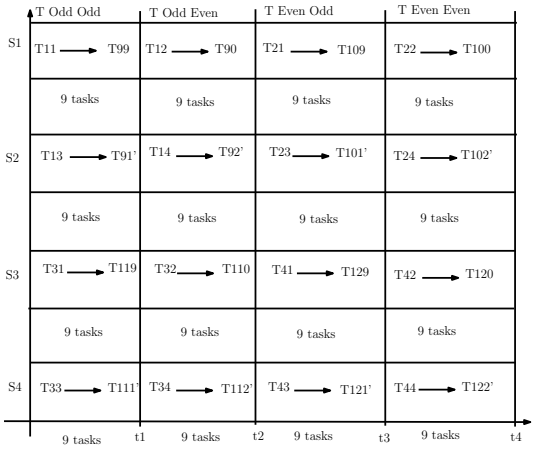

Fig. 4. Sequences tasks of ME algorithm from Akiyo-qcif.

There are many reasons to generate the DAG model. Firstly, the tracking and monitoring deadlines time from the tasks in sequences in the MPSoC systems. Secondly, this methodology limits the problem NP-hard in scheduling and partitioning tasks with buffers in memory platform MPSoC. Finally, this technique improves the granularity from the tasks in different levels in TPG-DAG. Figure 5 describes the time deadline in

sequences tasks of ME algorithm from Akiyo-qcif. These time deadlines depends on tasks in the same CPU1, and tasks in other CPU$(2, 3, 4)$. Hence, we receive many parameters and notations which are defined in table II.

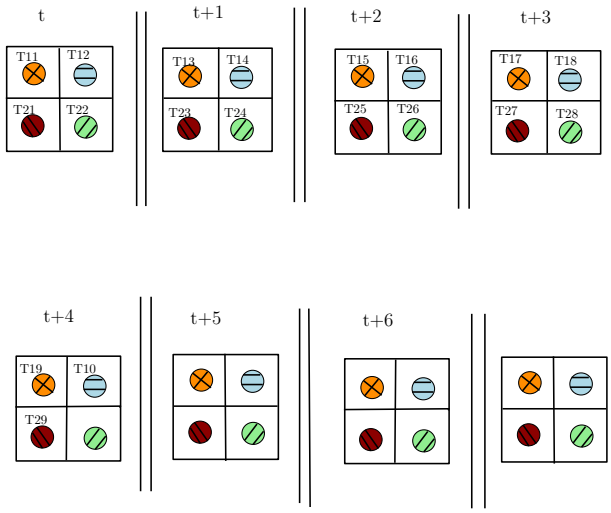

Fig. 5. Time deadlines in sequences tasks of ME algorithm from Akiyo-qcif.

TABLE II
THE SIMULATIONS PARAMETERS OF VIDEO SEQUENCE TEST "AKIYO".

Symbols	Explanations
s	Site, imply also pixel with (x,y) coordinate
Frame current	It
Frame	It+1
MB(t)	Macro-block at t
MB(t+1)	Macroblock at t+1
E	The set of the frame sites
$\Delta(t)$	Deadline time at t
$\Delta(t+1)$	Deadline time at t+1
W	Length
H	Height

B. Parameters setting in Ggen

a) Definition and Goal: : Definition 1 : we have a "n" number of vertices, the G(n; p) method generates graph where each element of the possible edges and tasks are presented with independent probability "p".

b) The G(n;M) method: : Hence, this model can be considered as the most appropriate method for generating random graphs with a number of edges and number of tasks.

Definition 2: for a given number of nodes n and a given number of edges "M", the G(n;M) method is as the method that constructs the graph by choosing uniformly "M" edges from the list of edges of the complete DAG on "n" vertices. This is equivalent to say that the method chooses uniformly a graph from the list of all possible DAGs with "M" edges and "n" nodes.

Notations

C. Multiprocessor scheduling

We first consider the problem of scheduling n jobs independent $(J_1, ..., J_n)$, m identical machines $M_1, ..., Mm$.

TABLE III
THE SETTING CO-SIMULATION OF VIDEO SEQUENCE TEST "AKIYO".

Symbols	Explanations
Nbr CPU	$1, 2, 4, 8$
Frequency set "F"	$25, 50, 125, 166, 250, 500$ MHz
Sequence Akiyo	(300 frames), Silent (100 frames)
Resolution QCIF	$(144x176)$
GOP Structure	"IBPB"
Frame rate	25 frames per second
Time slot duration	$1/90$ s
No. current frame sets	12
No. slices per frame	8

Each job J_j, $j = 1, ..., n$ should be treated exactly on a machine, and requires processing time pj. A schedule is such an assignment of each task on a machine. For scheduling, load Mi is the requirement of total processing jobs assigned, and the scheduling length is the load on the machine busiest. We wish to find a minimum length schedule. While working with a particular case of this problem in which there are two machines [6], [11]. There are then three distinct approaches to these scheduling problems: scheduling networks theory of queues, deterministic scheduling and scheduling software engineering.

c) Jobs by Flux: : An endless stream of jobs arrive at time $0 <= A_1 <= A_2 <= ...A_n$. We suppose that $A_1, A_2, ...A_n$, from a Poisson process with rate lambda. This assumption mean that the arrival times are unpredictable, and no information on hours preceding arrival can prepare for the next arrival. All we know is that it will happen in time planned $1/\lambda$. Where it will take place in the next interval $\triangle$ some time with a probability of $\lambda \times \triangle$ [6]. The jobs that arrive necessitate processing time $(X_1, X_2, ...)$. We suppose that identical are independent based on the length of processing distribution F with average $m1$. Another parameter for F is B given by equation 1 [6], [7]:

$$B = supx : F(x) < 1 \tag{1}$$

B is the longest time jobs from F are never likely. With partial information on the average waiting time. This formula is based on arguments of heavy traffic and that $\rho - > 1$, when the average number of job queue processor for years the average waiting time is very large. The exact formulas for all $0 < \rho < 1$ are derived in Miller (1968), but they are less easy to interpret. The following formula has heavy traffic 2 [1], [6]:

$$E(waitingtime|SPT) \div E(waitingtime|FiFo) = m1 \div B \tag{2}$$

D. Scheduling parallel machines

For j jobs, the makspan or time cost of communication is presented by the following equation:

$$C_{max} = max(C_j) \tag{3}$$

- theorem.2:

If the processing time jobs are random variables then stochastic and comparable Sept. It minimizes the flow time schedule [6].

How we choose the order of M-Machine?

The order is very important in the task scheduling and partitioning between different working machine according to our need in our application. In an M−machines, for j jobs requires M stages of processing, with processing time X_{ij}, $i = 1, ..., M$. Given a batch of n tasks, we must then to do well for order to minimize waiting times [1], [6], [7].

- theorem.3:
  Joint distributions of arrivals and departures $P(a_1, ..., a_n|d_1, ..., d_n|c_1, ..., c_n)$ are independent in the order of machine [6].

E. Training of GGEN from DAG algorithm

Because of the method for DAG algorithm is semi-automatic, this method give a many problems and done more Time To Market (TTM) for our application. The training method considered here give a solution for this problems. The code generic generation automatic from DAG algorithm is optimal. To prepare the list of DAG nodes to compute (that is, the list for which code is to be emitted), start at the root of the right-most subtree. Put this node on the list , and continue by adding a left-most node to the list after all its parents are already on the list. Then generate code for the nodes in by starting at the end of and proceeding to the beginning.

II. Optimization results of the GGEN generic code

In this section, we chose to optimize the various parameters of our application and task scheduling algorithm. The task graph of this approach with both techniques (interpolation and padding) is shown in figure 6.

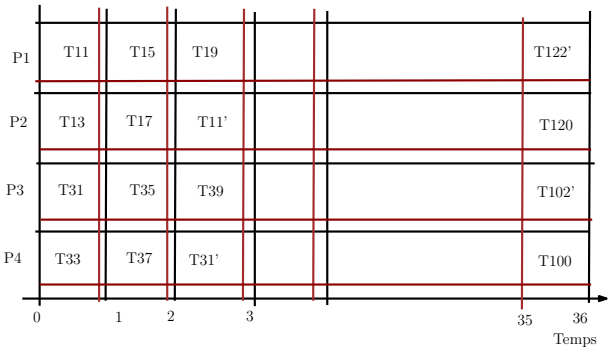

Fig. 6. The task graph of the single-level EM block with 4 CPU via the GGEN algorithm.

We do it is a simulator that will add deadlines between parent nodes and child nodes, later between high and low tasks in the task graph DAG-TPG or the GGEN task graph. In this part, we give the GGEN scheduler to do this automatically. We have two modes of operation in line (beta = 1) where well off-line (beta = 0). Any mathematical formalism and modeling via the DAG-TPG algorithm of the video coding EM algorithm will be adapted to the GGEN algorithm. Our work will be compared with the programmer developer "Perarnau Swann" with a student working with the same video coding application but the H264 standard. The ME block in H264 video codec will be classified as an internship with the automatic compiler, we present the task graph in the figure 7.

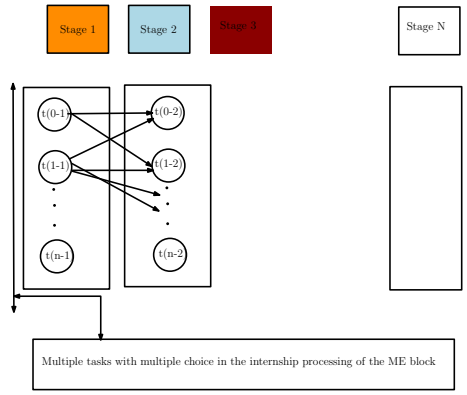

Fig. 7. The stage classification graph of the EM block via the GGEN algorithm.

In our simulations with the GGEN scheduler, we will change the number of processors $(4; 8; 16; 32; ...; N)$ each time to increase the performance of the processed application, which will validate the optimality of the algorithms of ordering and partitioning tasks called PYRROS in [6]. We do the work of the EM block with the different levels and with only one level via the two interpolation and padding techniques, to show that our works give more reliable, robust, optimal and fast performances. The complete task graph for the 8 levels of the H264 video codec EM block is shown in figure 8.

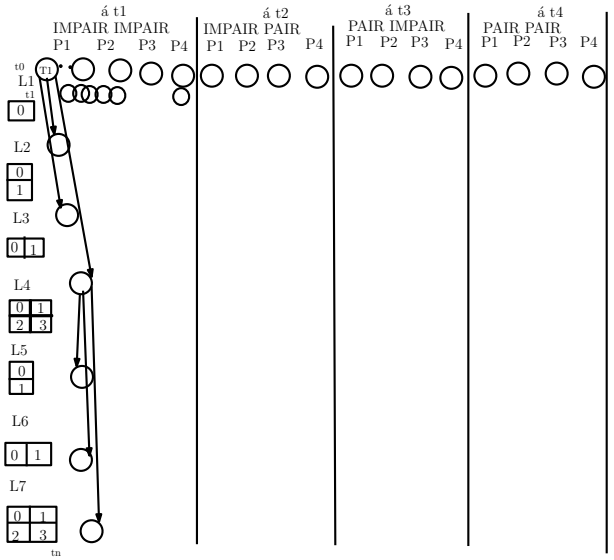

Fig. 8. The 8 ME block task graph with 4 CPU via the GGEN algorithm.

Despite, we illustrate the complete flow of the algorithm offers in figure9. First, a state is observed and then an initial action is selected in depending on the scheduling policy or intensive method if the state is unknown.

The initial action could then be adjusted during the exploration phase according to the observed DAG and the dynamics of the environment. The scheduling tasks is updated when

a concept drift is detected. This section explains this task sequence flow, and the tasks of the test video sequence.

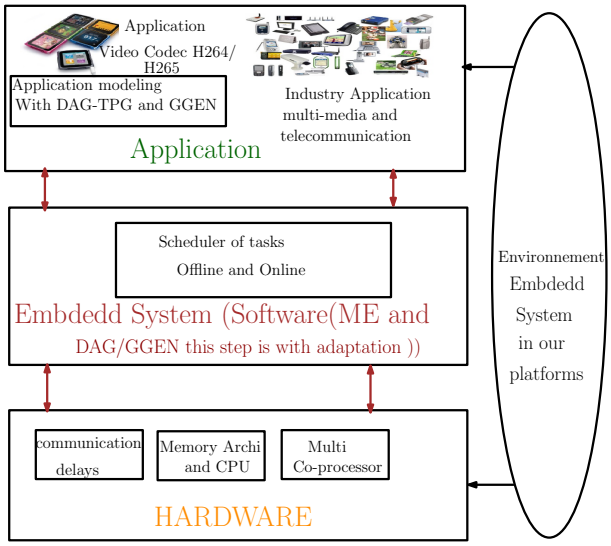

Fig. 9. Diagram of abstraction layers of the complete system.

To summarize, this figure resumes the different abstraction layers of the system (the application, the operating system, and the hardware layers) and their interaction with the external environment. Most of the existing optimizations for running modern broadcast applications were implemented in the layer application that is often unaware of the system architecture, the resources of the system available or the execution time is minimized. In addition, optimizations implemented at the hardware layer of the target, a specific set of tasks with a specific type of input data. So, it is essential that the system layer operating (instead of the application layer or the hardware layer), responsible mapping of the application on the platform, integrates a new planning solutions to bridge the gap between layers and exploit the features and comments received from both the application layer and the hardware layer in order to trade online throughput, quality and energy consumption. The results of modeling of the different sequences of tests in automatic and generic scheduling via the GGEN algorithm are illustrated in the tables IV, IX, VI.

From the results presented by the table IV, we notice that this automatic method gives satisfactory results for the test sequences. However, the method is complex to implement and adapt with the EM algorithm especially in the online operating state. This can be explained by the complexity of the sequence of tasks scheduled and partitioned on the scheduler and the complexity of the test video sequence (change of lighting, change of background and shooting). According to the table IX, we notice that the number of tasks is increased when working on the different pixel precision levels in the EM block, so the number of instructions will be increased as well. The code takes more time in execution, hence the importance of two techniques used in the results of prototyping and modeling with DAG-TPG (interpolation and padding).

When we implemented and adapted the DAG algorithm to the H264 video codec ME block, comparing to the GGEN

TABLE IV
MODELING TEST SEQUENCES USING THE DAG ALGORITHM.

Sequence	Format	Nbr Frame	Nbr Tasks/F
Akiyo -qcif	$qcif(176*144) \rightarrow (192*192)$	300	826
MissA cif	$qcif(176*144) \rightarrow (192*192)$	150 150	826 826
Forman	$qcif(352*288) \rightarrow (384*384)$	300 300	3304 3304
Mobile	$Sif(352*240) \rightarrow (384*384)$	112	3304
Bus1	$Cif(352*240) \rightarrow (384*384)$	100 100	3304 3304
skin1	$Sif(352*240) \rightarrow (384*384)$	900 900	3304 3304
claire	$qcif(176*144) \rightarrow (192*192)$	494	826
HD-seq[1]	$(1080*1920) \rightarrow (2048*2048)$	24	93982
HD-seq[2]	$(1080*1920) \rightarrow (2048*2048)$	24	93982

TABLE V
MODELING TEST SEQUENCES WITH DAG.

Seq	Nbr tasks/CPU	L (P1,3)	L(P2,4) (P1,3)	TE (P2,4)	TE
Akiyo-qcif	144	144 +c	144	3600 +c	3600
claire	144	144 +c	144 144	3600 +c	3600
missA	144	144 +c	144	3600 +c	3600
Forman	576	576 +c	576	8400 +c	8400 8400
Mobile	576	576 +c	576	8400 +c	8400
Bus1	576	576+c	576	8400+c	8400
skin1	576	576+c	576	8400+c	8400
HD-seq[1]	16384	16384 +c	16384	409600 +c	409600
HD-seq[2]	16384	16384 +c	16384	1409600 +c	409600

automatic scheduler, the number of tasks will be increased to increase the number of choices. So each task will be done in two deadlines, between two tasks we add tasks, instructions and nodes. For example, the task 1 (T1) will be performed on two deadlines at time t0 and time $t0$ + delta1. So for the different tasks of the image or at the level of the video sequence, until we finish the different strings of the task sequence with 4 CPU can any platform used VI.

We simulated in "C/C++" and in virtual prototype OVP, the two previous algorithms (ME and DAG) using different task graphs generated with GGen in OVP. Using the different sequence test video, we used these algorithms to solve the NP-hard problem known as ("Pj—$pi = 1$"; "P1+j—Cmax"). In other case, we want to schedule a set of tasks in ME algorithm with unit size and arbitrary precedence constraints through a set of parallel identical machines in order to minimize the completion time of the last task to finish. Each algorithm was executed against graphs produced by each generation method available in GGen. We experimentally evaluated both GGEN and DAG approaches with the video coding EM algorithm for

TABLE VI
MODELING TEST SEQUENCES WITH GGEN.

Sequence	Nbr-tasks /CPU	L(P1,3) (P2,4)	L (P1,3)	TE(P/I)	Cmin(P1,3) Cmax(P2,4)
Akiyo -qcif	414	414 +c	414	10738 +c/10738	414/ 414+c
claire	414	414+c	414 +c	10738 +c/10738	414/ 414+c
missA	414	414+c	414 +c	10738+c /10738	414/ 414+c
Forman	1656	1656 +c	1656	43132 +c/43132	1656 /1656+c
Mobile	1656	1656 +c	1656	43132 +c/43132	1656 /1656+c
Bus1	1656	1656 +c	1656	43132 +c/43132	1656 /1656+c
skin1	1656	1656 +c	1656	43132 43132+c	1656 /1656+c
HD-seq[1]	48024	48024 +c	48024	192096 +c/192096	48024 /48024+c
HD-seq[2]	48024	48024 +c	48024	192096 +c/192096	48024 /48024+c

the different test sequences. We will use in our work the term TTM and video sequence complexity as evaluation criteria since they are easy to interpret and can be applied for all the two methods of scheduling GGEN and DAG tasks used. The table IX shows the results of simulation in "C/C++" and "OVP" using the same platforms (three platforms based on ARM-Cortex A9 MP) with use of the ME block threads in only level. We present the execution time (TE) in seconds, the final execution times of each platform in virtual prototyping via the three platforms that were used in the DAG-TPG algorithm.

TABLE VII
RESULTS OF EXECUTION TIME WITH GGEN ALGORITHM OF TEST SEQUENCES.

Sequence	TE(C/C++ en s)	TEf(P1 (s))	TEf(P2)	TEf(P3)
Akiyo-qcif	8.7	7.6	5.43	4.23
missA	7.3	6.4	4.67	3.34
Salmen	8.9	7.4	5.56	4.67
Claire	33.8	32.15	30.6	29.23
Forman	8.3	7.23	5.21	4.12
Mobile	27.7	26.4	24.31	23.78
Bus1	15.2	14.43	12.62	11.76
Skin1	50.5	49.45	46.5	43.43
HD-seq[1]	5.4	4.23	3.14	2.12
HD-seq[2]	5.6	4.34	3.21	2.25

In this part, we make a comparison to show that we have reached our goals. We evaluate our work with researchers working on minimization execution time for the EM block on SoC and MPSoC platforms, this comparison is shown in Table VIII, given that the results obtained correspond the different sequence video test. So, SA is Semi-Automatic and A is Automatic. We can resume in this comparison, the optimality and precision in GGEN algorithm.

The table shows that run time results are optimized in C/C++ simulation and in co-simulation with the three SoC and MPSoC platforms, which are based on ARM-Cortex A9 MP. These results from the GGEN algorithm of the ME block in H264 video coding show that the execution time decreases

TABLE VIII
COMPARISON OF EXECUTION TIMES.

Sequence	TE(C/C++ in s(SA/A))	TEf(P1 (s)SA))	TEf(P2 (SA/A))	TEf(P3 (SA/A))
Akiyo-qcif	10.1/8.7	9.5/7.6	8.8/5.43	7.9/4.23
miss-A	8.6/7.3	7.1/6.4	5.6/4.67	4.9/3.34
Salmen	10.6/8.9	8.4/7.4	6.25/5.56	5.8/4.67
Forman	8.8/8.3	5.9/7.23	6.12/5.21	4.66/4.12
HD-seq[1]	5.4/6.5	5.1/4.23	4.4/3.14	3.6/2.12

in simulation and co-simulation, ie with and without task scheduling. But we note that these times become smaller when scheduling tasks, it shows that the scheduler is optimal.

d) Comparison with other work results and evaluation environment: In this section, we presented GGen, a unified and standard implementation of random task graph generation methods used in the scheduling and partitioning tasks for ME algorithm of video codec H264 in OVP platform. So, we compare our solution with other work in literature. The three platform are based in ARM-Cortex-A9MP, this table resume the performance in our platform (SoC and MPSoC) in OVP.

TABLE IX
EVALUATION FOR ENVIRONMENT IN OVP OF TEST SEQUENCES.

Processor/	P1 (SoC)	P2 (MPSoC)	P3 (MPSoC)
Architecture	ARM-Cortex-A9MP	same Platform	same Platform
Nbr core	4Co-processor	4 processor	4 processor
Performance	High-Performance	High-P	High-P
Memory	2 DDR	8 DDR	8 DDR
STREAM performance	22 Frame/s	20 Frame/s	20 Frame/s

III. CONCLUSION

In this paper, we presented the optimization algorithm and automatic generation of GGEN task scheduling: definitions, different algorithms that are studied by Perarnau Swann and his team, the different modules and sub-modules of the algorithms used in this automatic approach of task scheduling, modeling of various platforms SoC and MPSoC. Since the blocks and sub-blocks processed in our H264 video codec application implemented and scheduled on real and virtual prototyping platforms, we have shown the complexity and computational capacity of the blocks processed in the application. We decide another phase of optimization and automatic generation of the scheduling approach, this is the step of using an automatic and generic scheduler, which works in two modes of operation online or offline. This new scheduling approach is programmable and robust. Since there is no change in the mathematical model when changing the sequence, it has been characterized, automated and generated. A parametric modeling was then proposed from the results of simulation and co-simulation, proving the optimization of the various parameters, the processing time and the execution time. In a future work, we will try to minimize other metrics in the H264 video codec. We will use an automatic scheduling and partitioning algorithm DAG. We will also implement this work in a real target, based on the ARM Cortex A9MP, which is

composed of 4 processors and named Embest SABRE Lite, Target from Development SABRE Lite-i.MX6Quad.

ACKNOWLEDGMENT

The authors would like to thank SWANN Perarnau for valuable feedback on an early software GGEN install path of this manuscript.

REFERENCES

[1] S. Afef, G. Fahmi, F. Ahmed, *Scheduling tasks in MPSoC system for Motion Estimation in H26x video codec* International Journal of Science and Engineering Investigations, vol. 7, issue 74, April 2018, ISSN: 2251-8843.

[2] L. Rainer, S. Frank, M. Grant, K. Tim, P. Roman and V. Martin, *Virtual Platforms: Breaking New Grounds* DATE12/2012 EDAA.

[3] Amit, K. and , S. Akash, K. and Henkel, J. K. Amit, S. Muhammad, K. Akash, J. Henkel, *Mapping on Multi/Many−core Systems: Survey of Current and Emerging Trends* DAC13/2013 DAC.

[4] S. Muhammed, M. Bastian, J. Henkel, *An HVS−based Adaptive Computational Complexity Reduction Scheme for H.264/AVC Video Encoder using Prognostic Early Mode Exclusion* Date10/2010 DAte.

[5] B. Elias, S. Hassan and N. Smail, *H.264 Color Components Video Decoding Parallelization on Multi-Core Processors* 2010 13th Euromicro Conference on Digital System Design: Architectures, Methods and Tools.

[6] Ch. philipe, G.C. Edward, K.L Jr Jan and L. Zhen, *Scheduling Theory and its Applications* 1995 by Jhon Wiley ans Sons Ltd, England.

[7] M.K. Alix, *A graph-based analysis of the cyclic scheduling problem with time Constraints: schedulability and periodicity of the earliest schedule*, 3rd ed. Springer Science+Business Media, February 2010.

[8] B. Sebastien, K. Jabran, B. Ccile, K.B. Muhammad,*Effectiveness of power strategies for video applications: a practical study* J Real-Time Image Proc, IF 2, n.10, 2014, pp.1-10.

[9] P. Helle, S. Oudin, B. Dross, M. Oguz, O. Kemal, J. Joel, C. Gordon and W. Thomas, *Block Merging for Quadtree-Based Partitioning in HEVC* IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY, VOL. 22, NO. 12, DECEMBER 2012.

[10] JVT and ITU-T, *Draft ITU-T recommendation and final draft international standard of joint video specification (ITU-T Rec. H264–ISO/IEC 14496-10 AVC)* .

[11] C. Hanen, *Study of a NP-hard cyclic scheduling problem: The recurrent job-shop*, 3rd ed. European Journal of Operational Research 72 (1994) 82-101 North-Holland.

[12] Z. Pan, J. Lei and Y. Zhang, X. Sun and S. Kwong, *Fast Motion Estimation Based on Content Property for Low-Complexity H.265/HEVC Encoder*. IEEE TRANSACTIONS ON BROADCASTING, VOL. 62, NO. 3, SEPTEMBER 2016.

[13] K. Karim, H. Kim, J.K. Tan, and S. Ishikawa, *A Block Matching Technique for Object Tracking Based on Peripheral Increment Sign Correlation Image* Image and Vision Computing, 2008.

[14] K. Karim, A. David, M. Nicholas, and V.S Michaela, *A Unified Online Directed Acyclic Graph Flow Manager for Multicore Schedulers* IEEE, 2014, pp:714–719.

[15] Q. Hu, X. Zhang and J. Sun, *Analysis and Optimization of x265 Encoder* IEEE, 2014.

[16] Open Virtual Platform and Imperas, *http://www.ovpworld.org* Imperas-OVP, 2008.

[17] ARM, *http://www.arm.com/products/processors/cortex-a/cortex-a9.php* ARM, 2011.

[18] M. Eric, I. Lahoucine, N.C Marcian, B. Imene, T. Alin, and N. Mohamed Wissem, *fpgas in Industrial Control Applications*, 3rd ed. IEEE Transactions On Industrial Informatics, 2011.

[19] Davis P and S. Marikkannan *Implementation of Motion Estimation Algorithm for H.265/HEVC*, 3rd ed. International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering, Vol. 3, Special Issue 3, April 2014.

PLACE PHOTO HERE

SALHI I received my engineer's and Master's degrees in electrical engineering from national engineering school of sfax, Sfax University, in 2009 and 2011 respectively, Tunisia. I obtained my doctorat thesis in electrical engineering option electronic/telecommunication (Embededd System) at ENIS with cooperation UPMC. He is member of the Laboratory of Technology for Smart Systems (L.T.2.S) in the Digital Research Center of Sfax (CRNS). My research interests include implementation co-design in very high level MPSoCs for real time application (image processing and video applied in multimedia application).

Fahmi GHOZZI R his Ph.D. degree in electronics from IXL laboratory, university Bordeaux1, France, in 2004. He is member in the Laboratory of Technology for Smart Systems (L.T.2.S) in the Digital Research Center of Sfax (CRNS). He is currently an assistant professor at the University of Sfax in Tunisia. His research interests include specific architectures for image processing and video coding, and hardware/software implementation on FPGA target.

Ahmed FAKHFAKH R the engineer degree in electrical engineering from National Engineering School of Sfax (ENIS), Sfax University, in 1997, the master and the PhD degrees in Electronics both from IXL laboratory, Bordeaux, France, in 1998 and 2002 respectively. In 2009, he obtained the HDR diploma in Electrical Engineering from ENIS. From 2002 to 2016, he was member of the Laboratory of Electronics and Information Technologies (LETI) in ENIS. Since 2016, he is member of the Laboratory of Technologies for Smart Systems (LT2S) in the Digital Research Center of Sfax (CRNS) and head of the research team Design and implementation of communicating systems. His research interest deals with the development of electrical smart systems for Smart Grid and e-health applications

Alix MUNIER KORDON F Professor of mathematic and operational research in University of Pierre and Marie Curie "UPMC", research member in Laboratory of Informatic in Paris 6 "LIP6"/SoC department, 4, place Jussieu 75252 Paris cedex 05.