

# We are IntechOpen, the world's leading publisher of Open Access books Built by scientists, for scientists

6,900

Open access books available

186,000

International authors and editors

200M

Downloads

Our authors are among the

154

Countries delivered to

TOP 1%

most cited scientists

12.2%

Contributors from top 500 universities



WEB OF SCIENCE™

Selection of our books indexed in the Book Citation Index  
in Web of Science™ Core Collection (BKCI)

Interested in publishing with us?  
Contact [book.department@intechopen.com](mailto:book.department@intechopen.com)

Numbers displayed above are based on latest data collected.  
For more information visit [www.intechopen.com](http://www.intechopen.com)



# Approximation Algorithm for Scheduling a Chain of Tasks for Motion Estimation on Heterogeneous Systems MPSoC

*Afef Salhi, Fahmi Ghozzi and Ahmed Fakhfakh*

## Abstract

Co-design embedded system are very important step in digital vehicle and airplane. The multicore and multiprocessor SoC (MPSoC) started a new computing era. It is becoming increasingly used because it can provide designers much more opportunities to meet specific performances. Designing embedded systems includes two main phases: (i) HW/SW Partitioning performed from high-level (eclipse C/C++ or python (machine learning and deep learning)) functional and architecture models (with virtual prototype and real prototype). And (ii) Software Design performed with significantly more detailed models with scheduling and partitioning tasks algorithm DAG Directed Acyclic Graph and GGEN Generation Graph Estimation Nodes (there are automatic DAG algorithm). Partitioning decisions are made according to performance assumptions that should be validated on the more refined software models for ME block and GGEN algorithm. In this paper, we focus to optimize a execution time and amelioration for quality of video with a scheduling and partitioning tasks in video codec. We show how they can be modeled the video sequence test with the size of video in height and width (three models of scheduling tasks in four processor). This modeling with DAG and GGEN are partitioning at different platform in OVP (partitioning, SW design). We can know the optimization of consumption energy and execution time in SoC and MPSoC platform.

**Keywords:** Motion Estimation, Video Codec, Video and image processing, DAG, GGEN, Localization, Scheduling, Partitioning, H 264/AVC, H 265, MPSoC, SoC, OVP, Mapping, Co-design, Xilinx-SoC-Platform, multiple streams processing

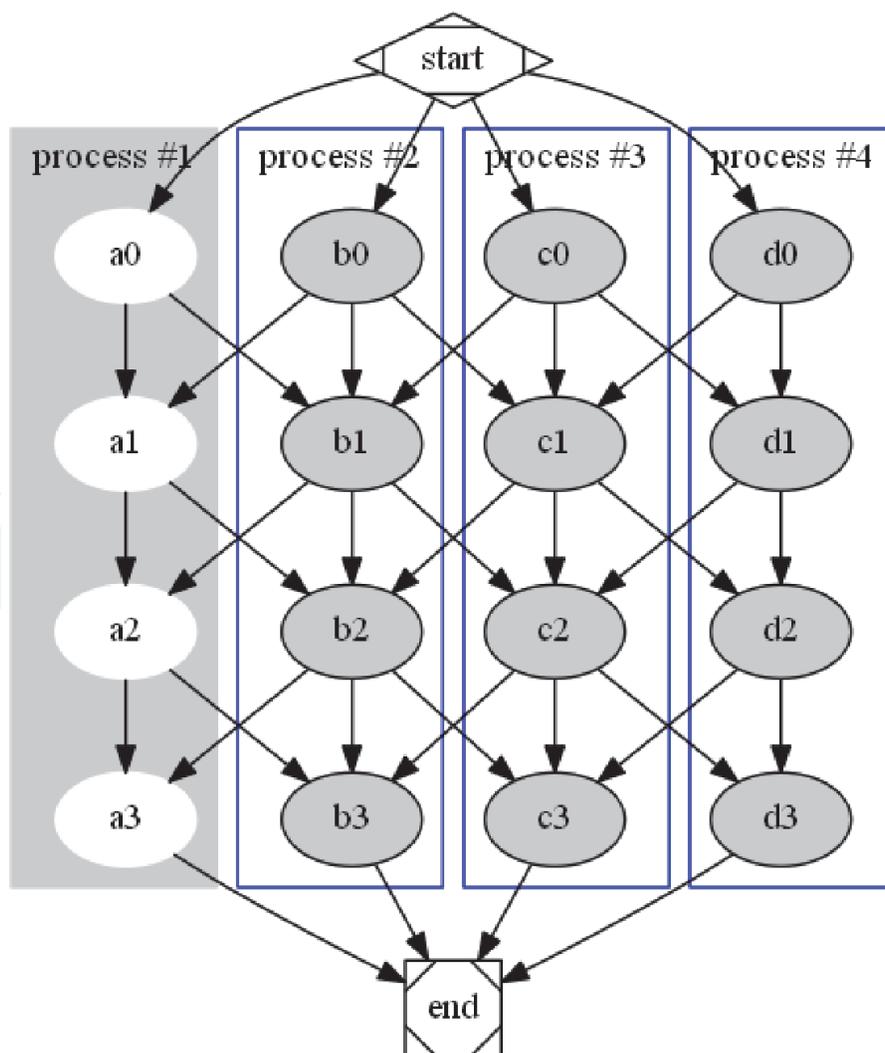
## 1. Introduction

Multi-core and Multi-processor architectures (SoC and MPSoC) started a new computing era. They are becoming increasingly used as they can provide designers with new opportunities to meet desired requirements in embedd system for different application and domain. Multi-media and telecommunication streaming applications are now widely used in several domains such as visio-conference, networking, video cripttage and compression, surveillance, medical services, military imaging and telecommunication applications. These applications are

characterized with stringent delay time of tasks. Model of scheduling tasks in 4 CPUs for Motion Estimation “ME” and DAG algorithm is illustrated in **Figure 1**.

Motion estimation module is very important complex module of a video codec. MPSoCs have the most suitable architecture to meet real time high-definition “HD” encoding requirements. This real time HD coding is based on a block matching algorithm “BMA” which locates matching blocks in a sequence of digital video frames. This technique is used to discover temporal redundancy in the video sequence, increasing the effectiveness of inter-frame video compression. The scheduling of tasks is an important step to accelerate the motion estimation process. The objective is to minimize the execution time of this algorithm, by distributing computing the tasks that describe the algorithm to the various cores of an MPSoC. In this case, we can see various methods to jobs scheduling and partitioning tasks for video codec applications, but are not adapted to our motion estimation block problem.

In this project, we choose the second method to schedule our algorithm based on DAG and GGEN for many reasons. This method minimizes the complexity of NP-complete problems in tasks. This methods are automatic, generic and periodic algorithm. It refine a granularity of tasks in ME block. The true parallelism tasks in SoC and MPSoC system, we start in 4 CPUs as in [1, 2], after we can works 8, 16, ... , 1024 CPUs. There are some works and scheduling in tasks, it is periodic and acyclic. This methods give us an optimal solution for complex scheduling and



**Figure 1.**  
Model of scheduling tasks in CPUs.

partitioning tasks on MPSoC system, as in [3, 4]. In this case, we choose parallelizing tasks and instructions in codec video blocks, there is very interesting step for scheduling and partitioning tasks in embedded platforms. We can cite: FPGA, GPU, DSP target, as in [5, 6]. The standard H 264/AVC is a new video codec configured and released by ITU-T and ISO/IEC [7, 8]. This standard give a very important results in bit rate, quality of image and others criteria in video codec block comparing with others standards [2, 7–9].

This paper is scheduled as follows: In section II, we detail the applied algorithm for block motion estimation with scheduling and partitioning tasks method in SoC and MPSoCs systems, this works are designed with acyclic algorithm (DAG-TPG) and co-designed in OVP platform. After, we present the ME module and importance in video codec standards. In section III, we can describe a new scheduling tasks approach and co-design in platform OVP (partitioning). Then, we synthesis with results, section IV. We finished by our conclusion.

## 2. ME in codec video H 264 and scheduling tasks approach

In this paper, high-quality video encoding imposes unprecedented performance requirements on real time mobile devices. To address the competing requirements of high performance and real time, embedded mobile multimedia device manufactures have recently adopted MPSoC (multiprocessor system-on-chip). Despite the advancements in new technology digital mobile device, computer system and I-Pad, the execution time, energy consumption and quality of image in SoC and MPSoC systems, we needed parallelism and scheduling tasks applied for H 264/AVC video codec. This approach of scheduling can eliminate problems in ME blocks and remains artifacts in image. We can start to minimize a time delays and time execution in ME blocks in video codec. Secondly, video codec blocks very important in structure and function, needed predictive coding blocks. These structures and functions, can be defined and modeled as with acyclic approach (semi-automatic and automatic) directed acyclic graphs (DAG) and generated graphs GGEN. With this approach, we can give a good and important solution for criteria evaluation in video codec, we can site execution time, time delays in tasks, and others [3, 4, 9]. We can finished to describe the delays tasks in some CPUs (SoC and MPSoCs target), there are considered real-tim applications [3, 4, 9]. In other case, video frames and video sequence should meet their deadlines and their estimation, the quality of image is very important. Then, many execution time in SoC and MPSoC targets with scheduling tasks approachs, there are exploit execution time and others criteria evaluations in video codec, we can see [9–12].

In **Table 1**, we classify these representative solutions based on their utilized optimization horizons, application models, complexity models, scheduling granularities, and considered sources of execution time. The scheduling tasks approach was chosen due to its efficiency and implementation simplicity for video codec H 264. The main idea of the scheduling tasks in ME is to decompose a sequence video into a frame and a frame into a Macro-block, so we scheduled and partitioned a tasks in parity order, we can see our method [9–12].

### 2.1 Scheduling and partitioning algorithm tasks

In previous works, scheduling and partitioning algorithms in MPSoCs systems have been formulated heuristically due to the inherent complexity of the multi-machine scheduling problem. Such algorithms constitute a very important step in multimedia applications. There is an NP-complete step in co-design. A schedule is

Scheduling and partitioning tasks	Platform	Algorithms	Application	Optimal	Type of granularity
[13]	SoC	DAG	Signal	Yes	Law
[3, 4]	SoC /MPSoC	DAG-GGEN /SDF-CSDF	Treatment	Yes	High
[14]	SoC/MPSoC	DAG	H 264	Yes	Law
Our solution	MPSoC	GGEN	H 265	Yes	High
[15]	SoC	DAG	H 264	No	High
[9-12]	SoC/MPSoC	DAG/GGEN	H 264	Yes	High
[16]	MPSoC	DAG	H 264	Yes	Law

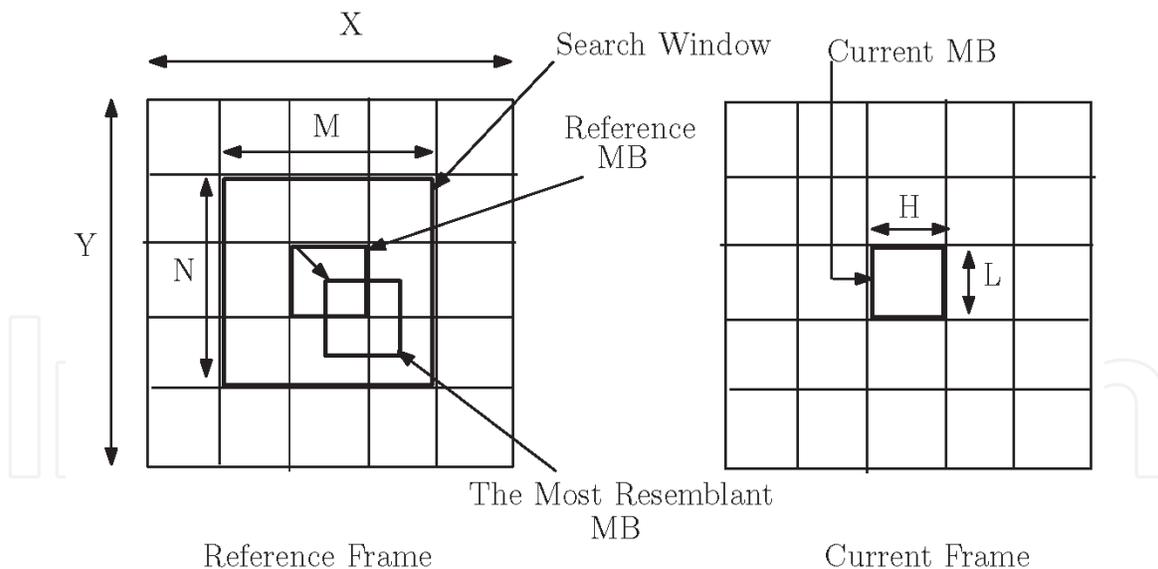
**Table 1.**  
Comparison our solution with different approaches scheduling tasks.

an assignment of each task to a machine. For scheduling, the load  $M_i$  defines the requirement of total processing jobs assigned, and the scheduling length is the load on the busiest machine. We want to find a minimum length for the schedule, as in [3, 13]. There are three distinct approaches to these scheduling problems: the theory of queues for networks scheduling, the deterministic scheduling and the software engineering scheduling. The study of approximation algorithms for NP-complete scheduling and partitioning problems for MPSoC systems has started with the work of Graham in 1996, who has analyzed a simple algorithm. When designing an approximate algorithm to minimize such NP-complete problems, we can evaluate its performance in a different way. One would use an algorithm that approximates with a guarantee of the performance of the deviation of the optimum value of the worst case as in [3, 4]. However, most scheduling and partitioning algorithms make assumptions on the relationship between the task dependencies. We may then classify scheduling algorithms. Scheduling, data partitioning and parallelism identification are three key points for an efficient application deployment. There are several approaches to manage scheduled tasks for real-time applications. The most important approaches are cyclic and acyclic approaches. Cyclic approaches have been used for Static Data Flow “SDF”, Cyclo-Static Data Flow “CSDF” and Petri networks [4, 13]. Acyclic approaches have used DAG, Unified Directed Acyclic Graph “UDAG, Weighted Directed Acyclic Graph “WDAG”, etc.

In this paper, we consider the problem of partitioning and scheduling tasks with a Task Precedence Graph “TPG” which is given as a DAG. A solution based on the DAG representation has the advantage to be generic, simple and usable with a refined granularity. A node in the DAG is a task which is a set of instructions to be executed sequentially without preemption in the same processor. Modeling, terminologies and all the mathematical formalism of the DAG algorithm is presented in [3, 4].

## 2.2 DAG algorithm

DAG algorithm is based on the asynchronous message passing paradigm. The parallel architectures are increasingly popular, but scheduling is very difficult because the data and the program must be partitioned and distributed to processors [3, 17, 18]. The methodology for MPSoC in embedded applications is the Adequacy-Algorithm-Architecture “AAA”. The following issues are of major importance for distributed memory architectures:



**Figure 2.**  
 Principle of SAD function.

- Scheduling, data partitioning and parallelism identification.
- Data mapping and program architecture.
- Scheduling and partitioning the execution of the task.

### 2.3 Motion estimation “ME”: application

ME block is very important in H 264 and H 265 video codec. In various standards, ME needs very complex tasks and instructions, and takes the largest part of video codec [2, 10–12, 19].

#### 2.3.1 Principle of the ME block

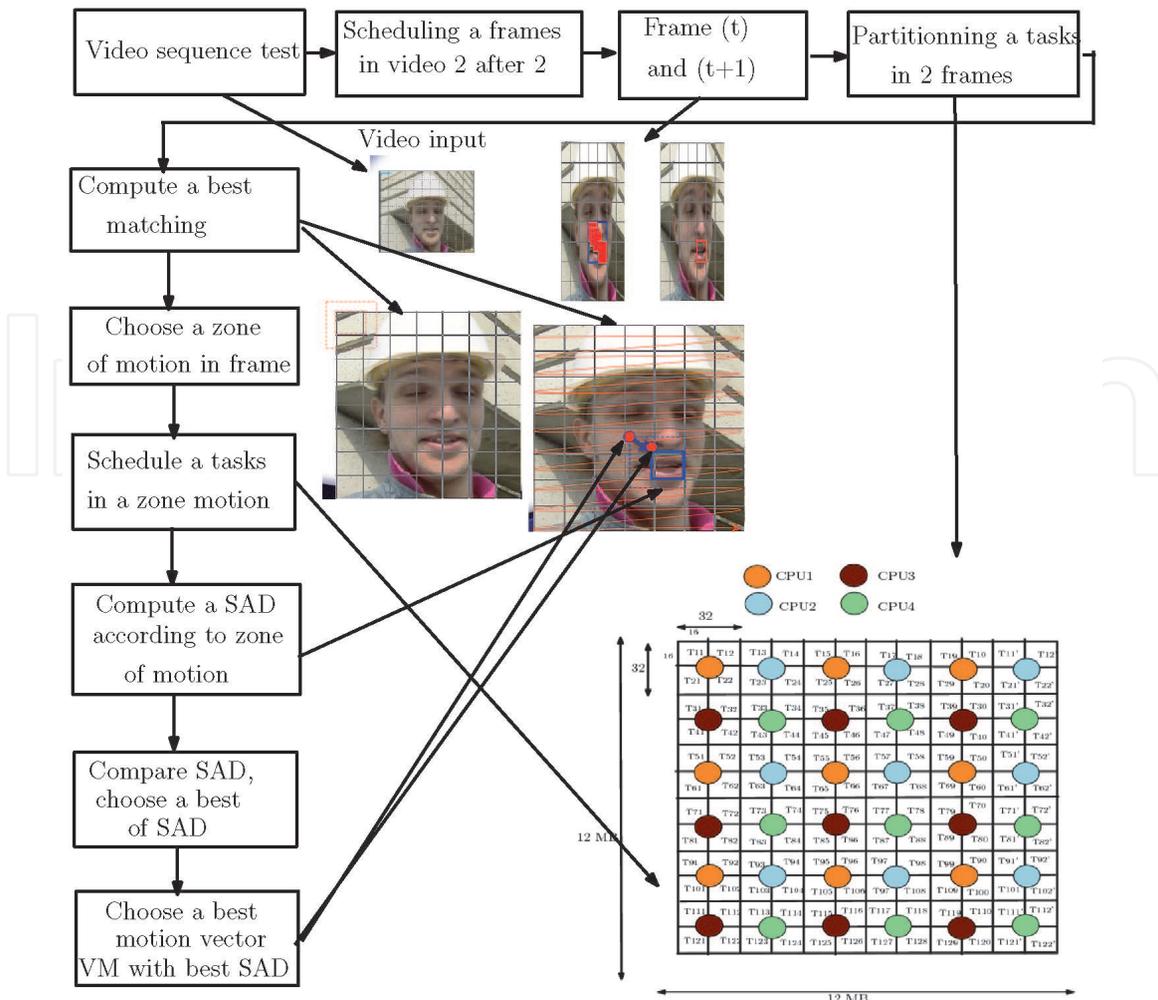
Principle of ME is the following: for a MB in the current frame, we define a search window in the reference frame. There are several evaluation criteria, such as MSE, SAD, BBM, MAD, NCF. We seek in this window the best MB using the Sum of Absolute Difference “SAD” distortion criterion given by Eq. (1) as in [2, 19–21], we describe this function SAD in **Figure 2**. To optimize the complexity level of the ME module, several fast search algorithms have been defined in the literature as DS, FS, TSS, HDS, PMVFAST, LDPS and the block-matching algorithm “BMA”. Inter-coding consists in finding a similar block that is aware of a reference frame block. This process is performed by a BMA. General principle of BMA is to exploit the temporal redundancies between consecutive frames.

$$SAD(x,y) = \begin{cases} \sum_{i=0}^{H-1} \sum_{j=0}^{L-1} |R(i,j) - F(x+i,y+j)| \end{cases} \quad (1)$$

## 3. Scheduling based on DAG formalism: approach

### 3.1 Block matching algorithm “BMA”

Inter-coding consists in finding a block similar to the current block of a reference frame. This process is performed by a block-matching algorithm. The general principle



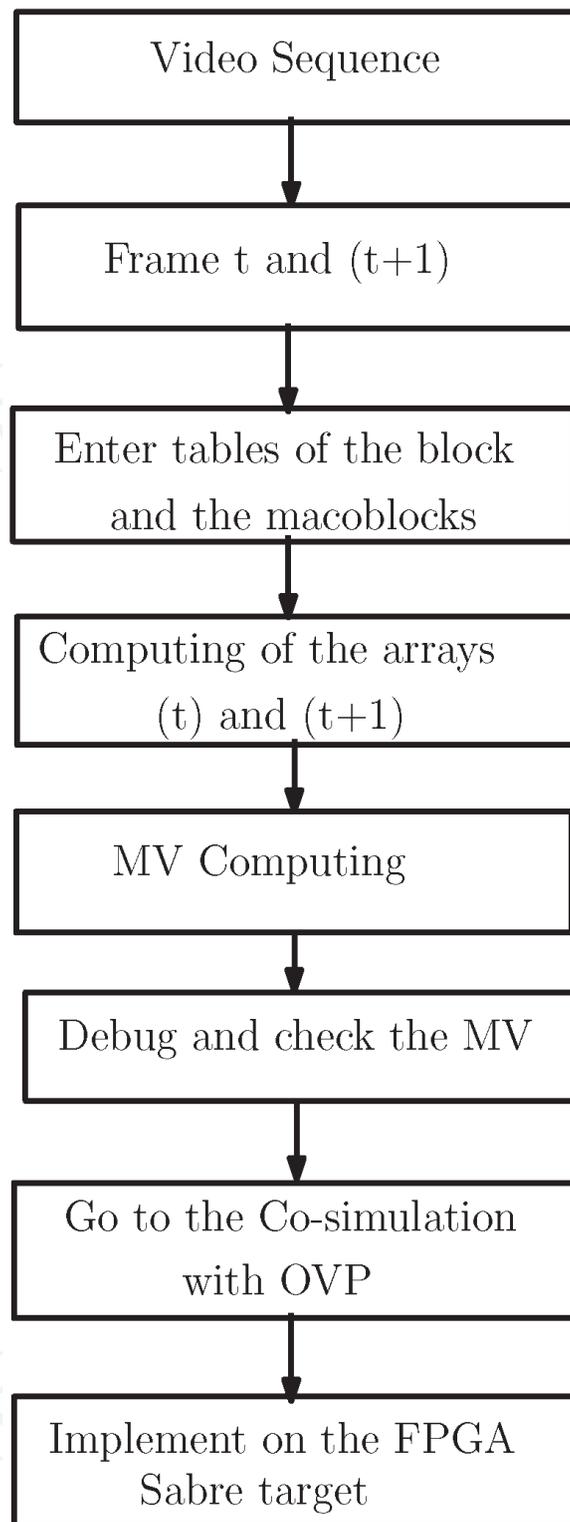
**Figure 3.**  
Principal of block matching algorithm “BMA”.

of the BMA is to exploit existing temporal redundancies between consecutive frames. This method involves searching for each point of the frame of interest  $I_t$ , the point of the frame  $I_{t+1}$  USD which maximizes a correlation score. The search is performed in a search block [7, 10–12, 19]. We describe a principal of this method in **Figure 3**.

Object of interest is determined when the number of corresponding blocks in the previous and current frame is higher than the value of a certain threshold. The threshold value is obtained experimentally [22]. We define the principle from SAD function in Eq. (1), where  $(R(i; j))$  denotes the pixels of the reference MB and  $(F(x + i; y + j))$  denotes the pixels of the current MB. **Figure 4** below shows the flow chart of the ME block of H 265 video codec. We use a padding method in order to enforce a whole number of packets. The scheduling methodology is defined in the **Figure 5**, which illustrates the flow chart of the MB (16 \* 16) for H 265 the video codec. The new idea for the flow chart of H 265 is to work with the padding technique rather than the affinity of the granularity (1/2), (1/4) and the (1/8) pixel method. The padding added to the end of a packet in order to enforce a whole number of packets.

### 3.2 DAG applied to ME

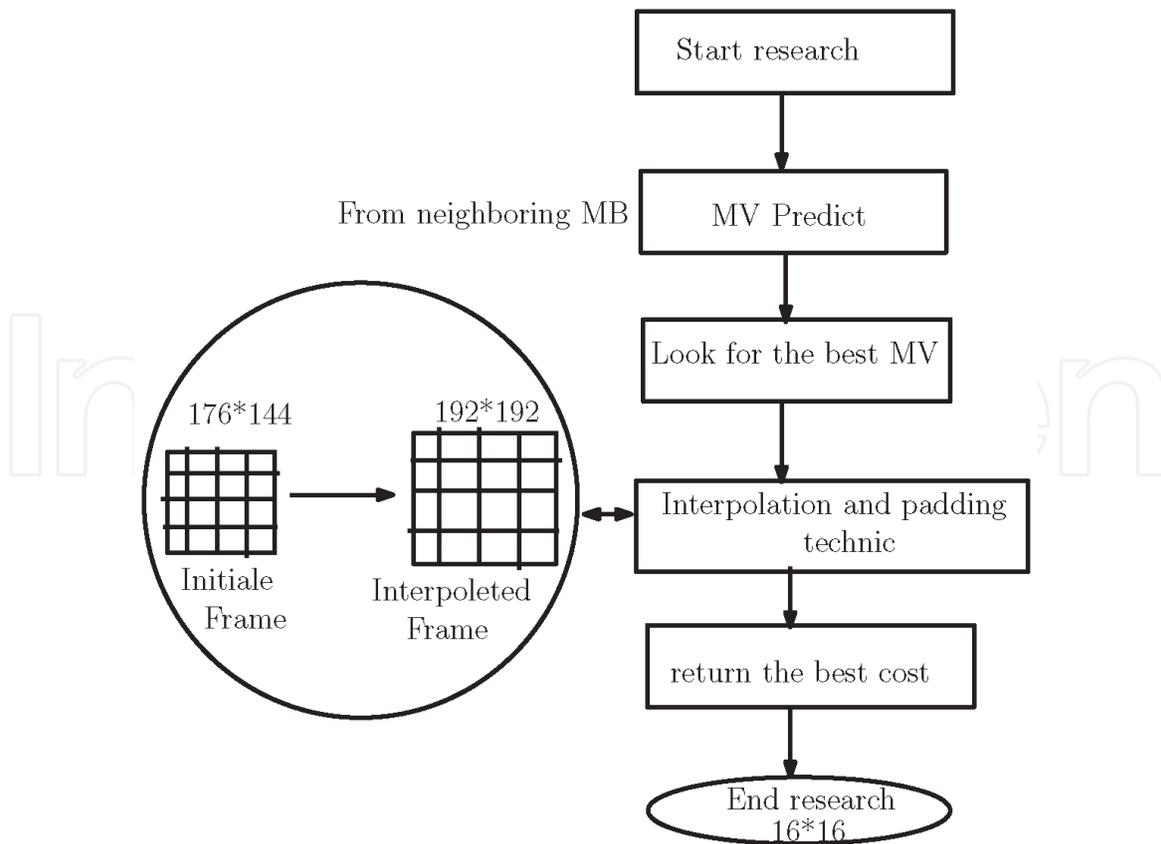
In this section, we describe our new approach for the scheduling tasks DAG. Firstly, we should to make the method generic, we can validated for any frame size. Then, we chose to work on block parity in scheduling tasks for ME blocks. Four combinations are possibles: odd odd, odd even, even odd, even even. The different



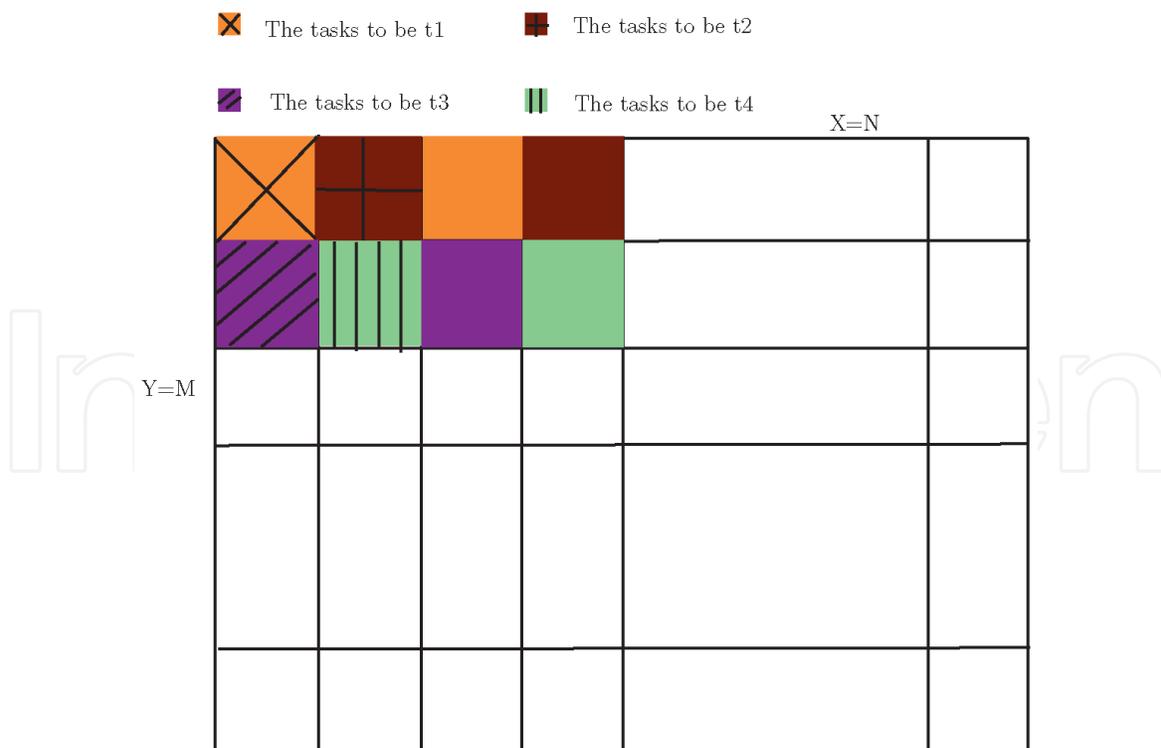
**Figure 4.**  
*Flow chart of ME algorithm.*

sequence test in our works are modeled in three models for the scheduling tasks methods. The generic frame size is (“X = N”, “Y = M”) where “X” represents the pixels for lines and “Y” denotes the column pixels. After scheduling a three models, we can see a problem in border of image or frame. This problem, we can apply padding method to add rows and columns to the frames by adding empty pixels. We notice that “N” has to take an even value that is divisible by 4. **Figure 6** shows the work-flow of an entire frame with size (N\*M).

For each node in the task graph, one must compute the start and end times of execution cycles by using the weight of each node. The “Gantt of chart” represents

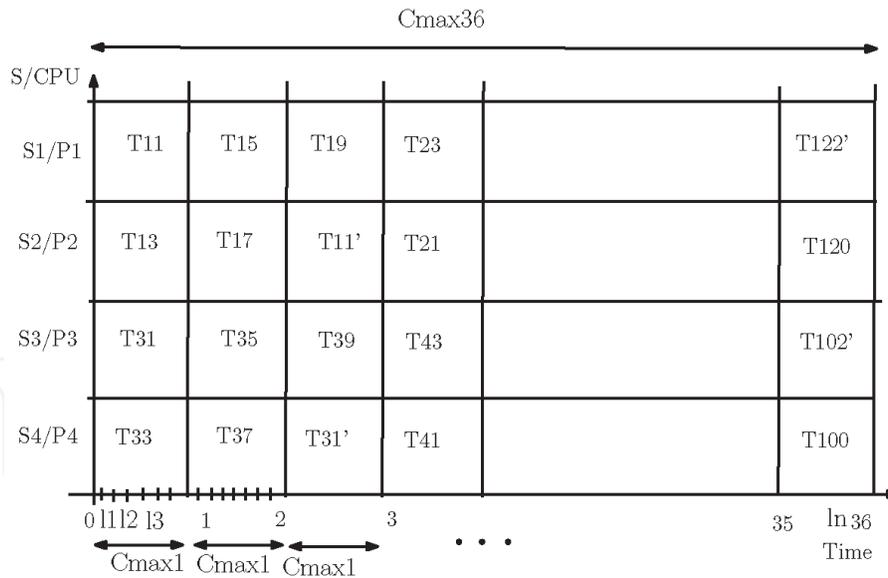


**Figure 5.**  
Flow chart of the 16 \* 16 for ME block in H 264 video codec.



**Figure 6.**  
Scheduling of an frame of size (N \* M).

the task scheduling in processors in terms of time and gives the order of the tasks of the ME block. This Gantt of chart is illustrated in **Figure 7**. Clustering is the placement of the various tasks of our application on different clusters. It depends on



**Figure 7.**  
 Gantt of chart from ME block.

the number of processors in the implementation platform. In our case, we chose a platform with four processors. We need four groups, four instants of time and then we have four outputs.

### 3.3 Parameters setting

For a set P of processors, each node must be assigned to a single processor. For all the the addressable memory, we give a portion of the same space to each case. When setting up parameters, we have to take in consideration the time constraints and the constraint of targets HW/SW. The classical assumptions made on the target MPSoC are:

- The tasks are non-preemptive.
- Local communication is costless.

Despite, the following equations show the scheduling tasks of a classic frame. First, we treat the odd tasks and the odd nodes in the TPG graph,  $ts(n_{ii})$  and  $w(n_{ii})$  presenting the time and weight in odd nodes.

$$tf(n_{ii}) = ts(n_{ii}) + w(n_{ii}) \quad (2)$$

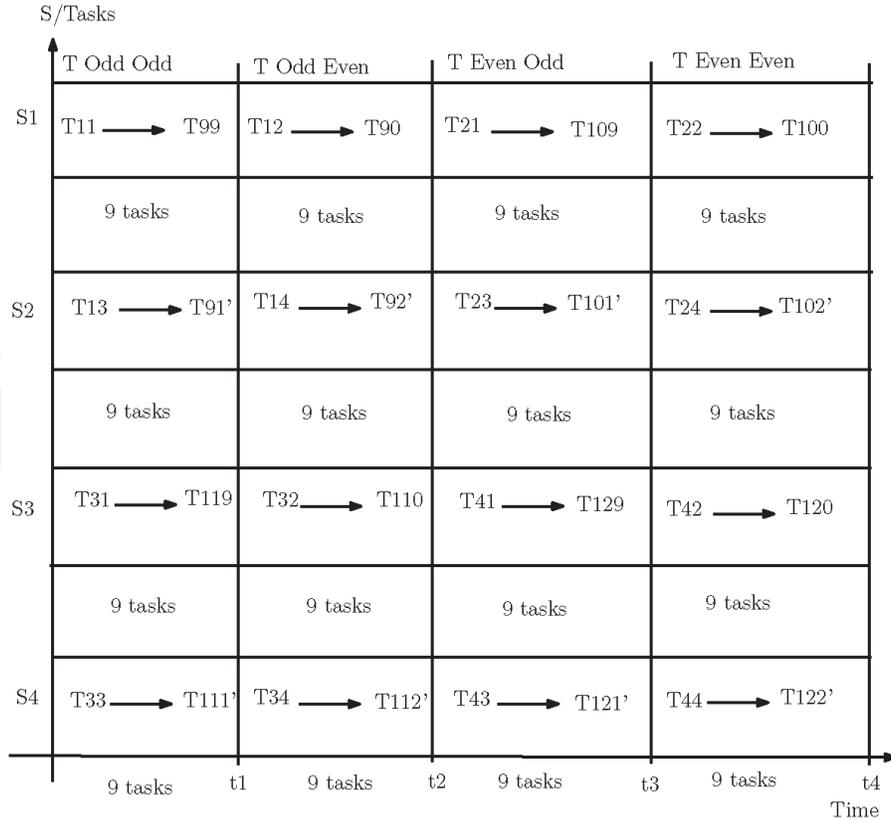
At the end, we compute the equations of even tasks, where  $ts(n_{jj})$  is the time of the node (jj) and  $w(n_{jj})$  is the weight in this node.

$$tf(n_{jj}) = ts(n_{jj}) + w(n_{jj}) \quad (3)$$

For the time end calculation “tf” in the node  $(n_i; n_j)$ . The sequence tasks from the ME blocks are defined in **Figure 8**.

In general, the nodes and tasks in TPG graph are made of nodes  $n_{ii}, n_{ij}, n_{ji}, n_{jj}$ . They are illustrated in Eqs. (4)–(7).

$$N_{ii} = (n_{ii} + n_{ij}) \quad (4)$$



**Figure 8.**  
Gantt of chart for tasks sequences by CPU.

$$N_{ij} = (n_{ij} + n_{ji}) \quad (5)$$

$$N_{ji} = (n_{ji} + n_{ii}) \quad (6)$$

$$N_{jj} = (n_{jj} + n_{ji}) \quad (7)$$

Execution time for the node of the test sequence is presented by Eq. (8) below.

$$t_f(n_{ii}, n_{ij}) = \begin{cases} t_f(n_{ii}) \\ n_{11}, \dots, n_{ii} \text{ in the same processor} \end{cases} \quad (8)$$

For the nodes or the tasks executed on different processors, the execution period is computed in the output frame for the current reconstruction frame. The nodes are spread by parities over each processor. In the first level, we have 16 nodes. In Level II, we add nodes to the neighbor to reconstruct the macro-blocks. The same is done for all levels, the goal being to reconstruct the frame. In our case, we have four groups since both architectures test four processors. We compute the execution period in Eq. (9) below.

$$t_f(n_i, n_j) = \begin{cases} t_{f_{n_i}} + C(n_i, n_j) \\ \text{Otherwise } n_i, n_j \text{ in different processor} \end{cases} \quad (9)$$

Hence, the set of end times is computed in the three models for scheduling and partitioning tasks. This algorithm is applied to ME blocks for the test video sequence in H 265 video codec:  $t_f(n_{ii}, n_{ii}), t_f(n_{ij}, n_{ij}), t_f(n_{ij}, n_{ij}), t_f(n_{ji}, n_{ji})$ . It is applied to all nodes in the graph found for the task with the ME block using test sequence “Akiyo”. Two processors  $p_i$  and  $p_j$  are isomorphic if read times are equal. Then the weight odd

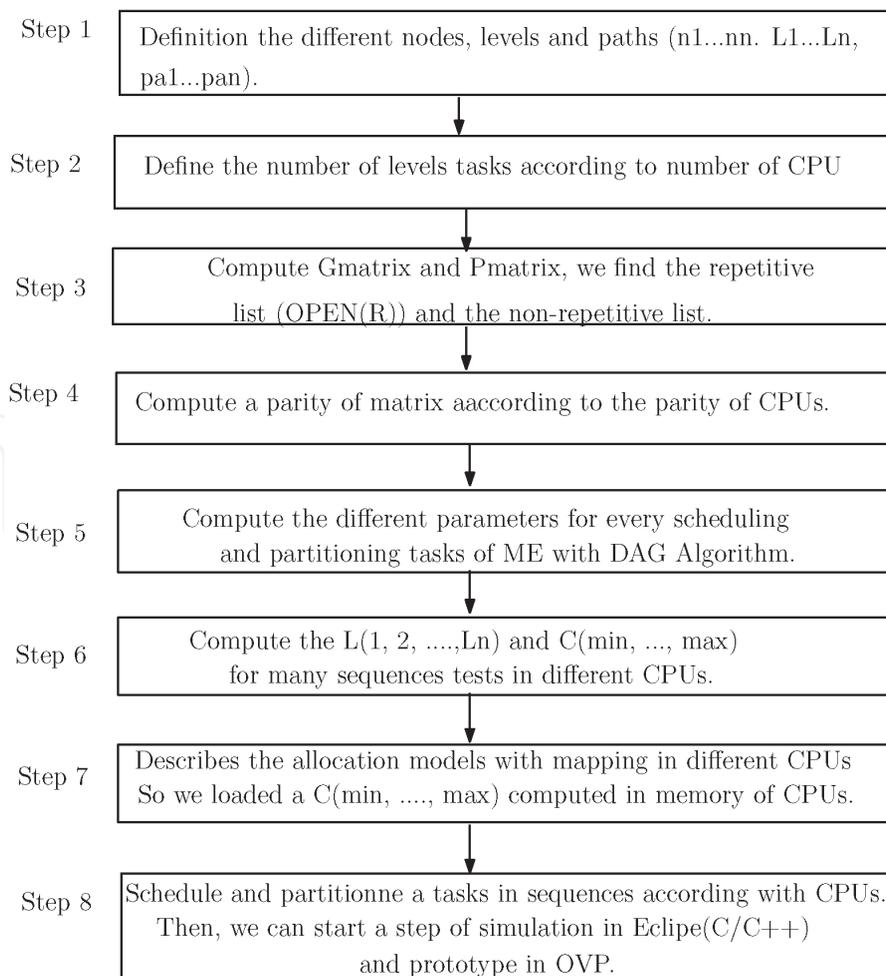
nodes or tasks  $w(n_i)$  are equal to the weight nodes or tasks  $w(n_j)$ . Thus, the set of nodes is  $n_{ii}, n_{ij}, n_{jj}, n_{ji}$  and the set of tasks is  $T_{ii}, T_{ij}, T_{jj}, T_{ji}$ . Succeeding in scheduling and partitioning the tasks with DAG is equivalent to the following conditions::

- $\text{Pred}(n_i) = \text{pred}(n_j)$ .
- $W(n_i) = W(n_j)$ .
- $\text{Succ}(n_i) = \text{Succ}(n_j)$ .

In our case, the platforms contain four processors, therefore we seek other processors satisfying the conditions. Basing on the assumptions above, we can deduce other parameters. The scheduling length for tasks and partitioning problems of processor  $(P + 1)$  is always less than or equal to the processor  $P$ .

$$SL(S_{opt}(P + 1)) < = SL(S_{opt}(P)) = \max(t_f(n_{xx})) \quad (10)$$

Where  $n_{xx}$  is the set of nodes  $n_{ii}, n_{ij}, n_{ji}, n_{jj}$ . This equation is applied to the three models, where “SL(S)” contains the scheduling graph length “Sopt(P)” which has the optimal schedule of P processors. For our application, we have four processors. To apply the hypothesis, we must know the maximum execution time for all nodes (“X” nodes = “X” tasks).



**Figure 9.**  
 Flow-chart of steps DAG algorithm with ME.

Steps	Model of sequences	Nbre of frames	Nebre levels	Val of paths	Nbre of tasks	Nbre of L (1, ..., n)
1	Q-cif	100	8	2/./288	144	36/36 + c
2	Q-cif	495	8	2/./288	144	36/36 + c
3	Cif	150	8	2/./480	240	144/144 + c
4	Cif	280	8	2/./480	240	144/144 + c
5	Sif	300	8	2/./480	240	144/144 + c
6	Q-cif	995	8	2/./480	240	144/144 + c
7	HD	25	16	2/./3840	1920	4096/4096 + c
8	HD	25	16	2/./3840	1920	4096/4096 + c

**Table 2.**  
Parameters for steps of tests videos sequences.

### 3.4 Applied DAG algorithm with ME in video codec

Flow chart of the global steps is illustrated in **Figure 9**.  
So, we can resume the scheduling steps are as follows in **Table 2**.

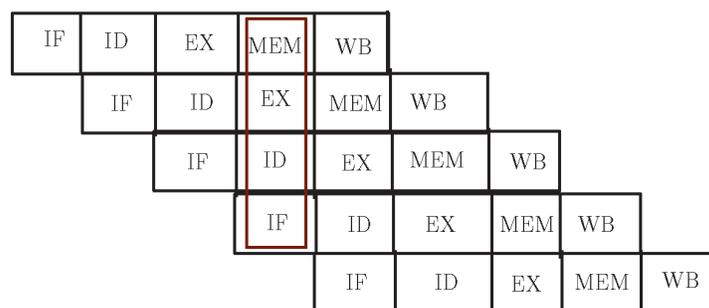
#### 3.4.1 Proposed architecture and performance of the DAG algorithm applied to ME blocks in OVP

##### How to compute the execution time?

To calculate the processing time of an instruction, since the message transfer time depends sent to the calculation of instructions executed by a processor flows. Each instruction requires multiple clock cycles, the instruction is executed in as many cycles steps. Sequential microprocessors run the following statement when they finish first. In the case of instruction parallelism, the microprocessor can process several of these steps at the same time for several different instructions because different internal resources are mobilized. In other words, the processor executes instructions in parallel and sequentially at various stages of completion. This execution queue is called a pipeline. This mechanism has been implemented for the first time in the 1960 by IBM. **Figure 10** describe the canonical example of this type of pipeline, under the form of a RISC processor, in five stages. Sequencing instructions in a processor with a 5-stage pipeline needs 9 cycles to execute 5 instructions. At  $t = 5$ , all floors have solicited the pipeline, and 5 operations occur simultaneously.

##### How to compute the transfer time?

Assume that the data transfer time is proportional to the size of the data exchanged (between processors), this transfer time is the time of sending data within



**Figure 10.**  
The canonical example of this type of pipe is that of a RISC processor (5 stages).

tasks in between processors. Then “Tt” is the transfer time and “Td” is the data size.  $k(p1; p2)$  is the transfer duration between p1 and p2. Tt is defined by Eq. 11:

$$Tt = td \times K(p1, p2) \quad (11)$$

The execution time of an instruction and message transfer between processors, and tasks between processors processed by the ME block are presented in **Table 3**.

#### Virtual prototyping with OVP environment

We work with a project virtual prototype in OVP simulator. The strategies from a project and simulations in OVP are illustrated in [23]. We describes the co-simulation in **Figure 11**. This figure is composed the implementation and prototype in OVP (with thread method in CPUs).

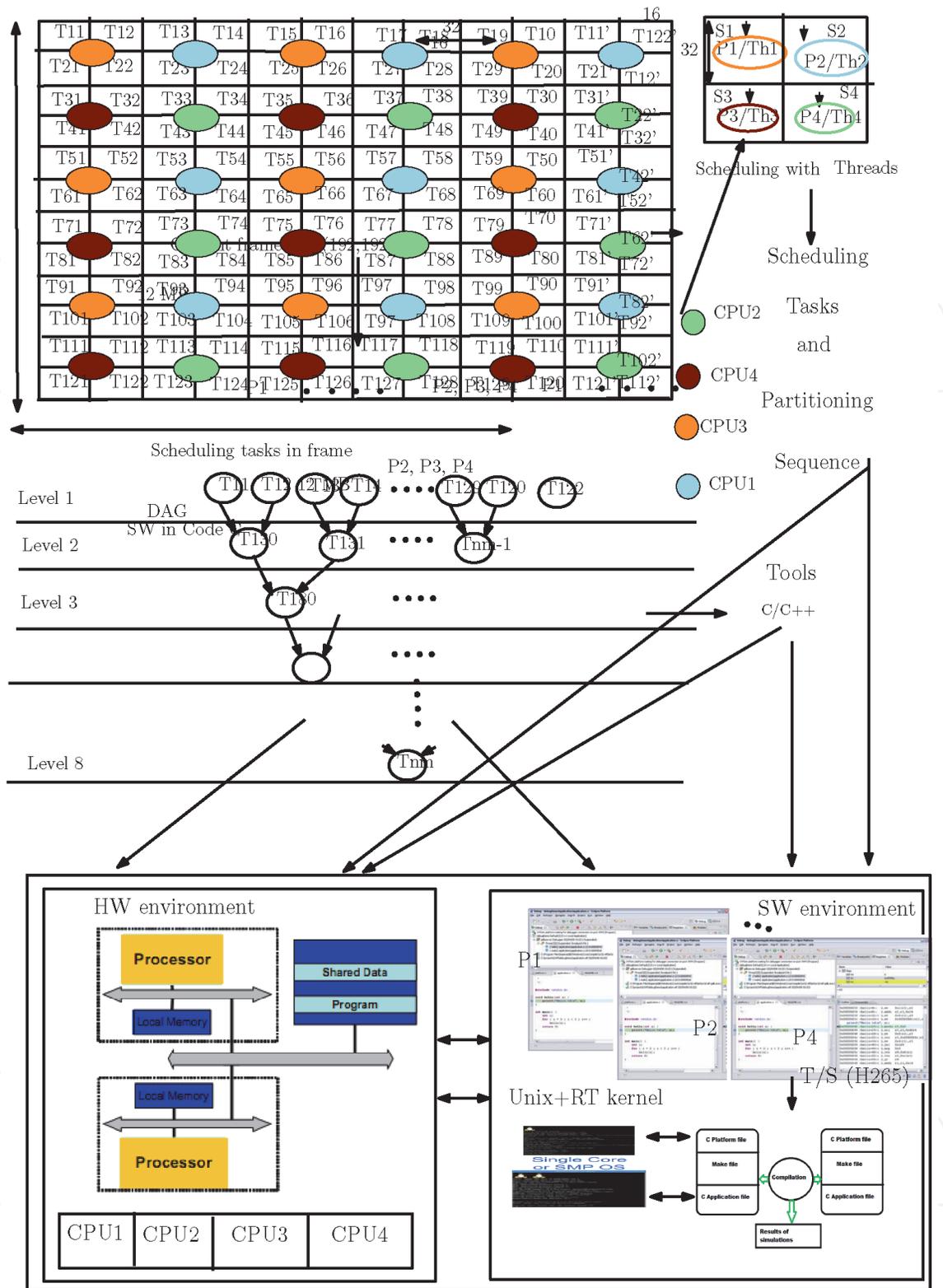
#### Affinity scheduling granularity of the ME algorithm

The following diagram shows the flow chart of mode (16 \* 16) for the video coding in H 264/AVC. The difference with the flowchart of H 264/AVC standard is the interpolation technique, used instead of the affinity method granularity 1/2, 1/4 and 1/8 pixels. Using the interpolation technique minimizes the number of jobs and the number of level task graphs. Thereafter, the execution time of block ME is optimized for the standard video codec HEVC/H 265 and the standard H.264/AVC old report time. Thus, we have one level graph TPG-DAG. Both graphs were improving accuracy and frame quality. In this section, we consider time communications between different tasks, as independent tasks in the same processor itself or in different processors as shown in **Figure 12**. We define the communication delays between tasks and scheduling tasks lengths. **Figure 12** presents the partitioning and scheduling algorithm for the ME block in the H 265/AVC video codec and the different communications and mappings for the different processors.

**Table 4** shows the test video sequences with theoretical results. We can see that the theoretical execution time is close to the practical execution time observed in

L	Nbr T(1)	Nbr T(2)	Nbr T(3)	Tdt (1)	Tdt (2)	Tdt (3)	k (p1,p2)
L1	144	576	16384	3600	14400	409600	2/.135462
L2	72	288	8192	1800	7200	204800	2/.67736
L3	36	144	4096	900	3600	102400	2/.33868
L4	18	72	2048	450	1800	51200	2/.26844
L5	9	36	1024	225	900	25600	2/.18422
L6	5	18	512	112.5	450	12800	2/.9216
L7	2	9	256	56.25	225	6400	2/.4608
L8	1	5	128	28.12	112.5	3200	2/.2304
L9	0	2	64	14.6	56.25	1600	2/.1152
L10	0	1	32	7.3	28.12	800	2/.576
L11	0	0	16	3.8	14.6	400	2/.256
L12	0	0	8	1.9	7.3	200	2/.128
L13	0	0	4	0.95	3.8	100	2/.64
L14	0	0	2	0.475	1.9	50	2/.32
L15	0	0	1	0.24	0.95	25	2/.16

**Table 3.**  
 The results of scheduling and partitioning levels of the model for the sequence test 1.

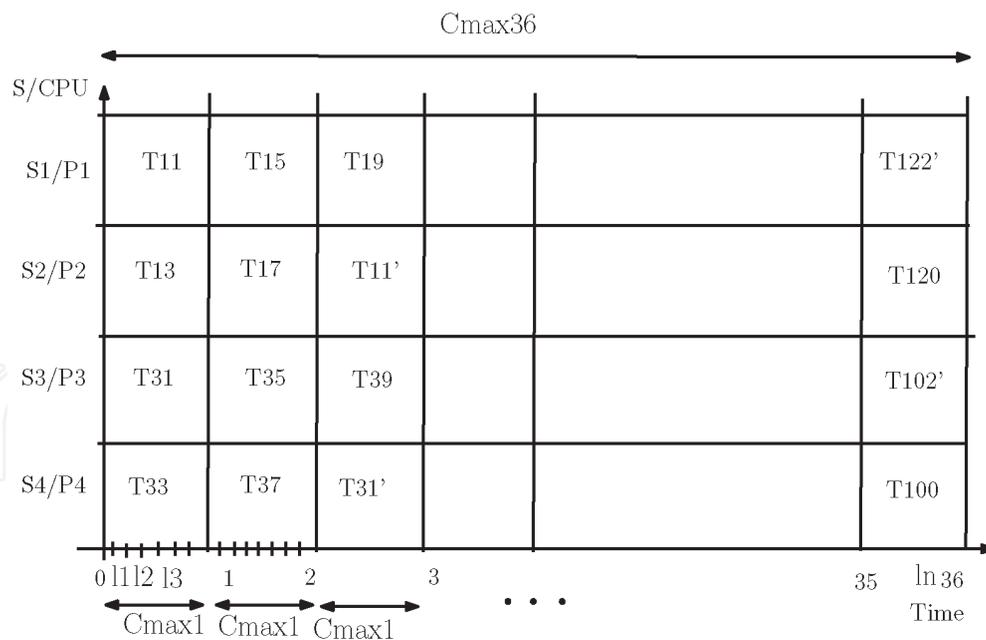


**Figure 11.** Prototype and implementation in OVP.

co-simulations of OVP. Also, we see that our scheduling and partitioning algorithm is optimal within the approximation and modeling formalisms considered.

### 3.5 The results of the ME block applied to scheduling tasks approach

Some experimental curves of execution time for the ME block applied to the DAG algorithm are presented in **Figure 13**, obtained in OVP for SoC and MPSoC based ARM Cortex A9MP. The important metric in our works is execution time.



**Figure 12.**  
 Independence tasks in sequences with CPU.

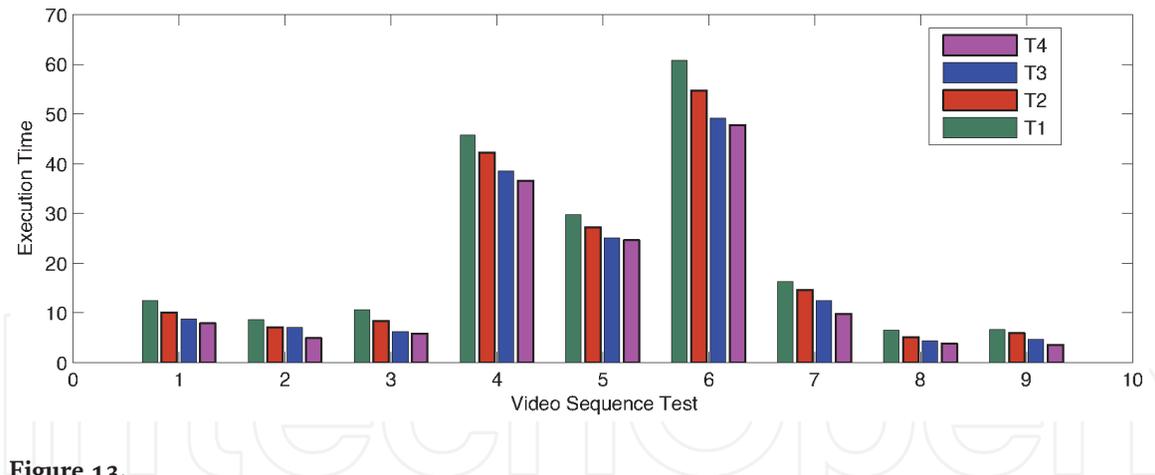
Seq	Tt (cycls)	EX-T in Fr(cycls)	EX-T in Seq(cycls)	EX-T in seq(s)
Qcif	25	3600	1080000	7.2
Qcif	25	3600	540000	5.4
Qcif	25	3600	1616400	10.76
Cif	25	14400	4320000	8.8
Sif	25	14400	1612800	10.75
Cif	25	14400	1440000	9.6
Sif	25	14400	12960000	8.64
Qcif	25	14400	1778400	11.85
HD [9]	25	409600	117964800	7.9
HD [10]	25	409600	117964800	7.9

**Table 4.**  
 Modeling test sequences using the DAG algorithm.

After, execution time simulation, we chose co-simulation in OVP. In this virtual platform, we can try variety SoC and MPSoC targets in OVP. The execution time is ameliorated compared with other results in literature with respect to the number or frames in test video sequences. For more details we can see [9], in our works we presents a different values in three models in scheduling and partitioning tasks co-designed in platform OVP (SW/HW). Our scheduling algorithm DAG applied to ME block in video codec is optimal because  $(0 < = C < = 0 : 5)$ .  $T_1$  is the execution time for one processor and  $T_2$  is the execution time for many processors. The gain for the application is computed in Eq. 12.

$$Gain = [(|T_1 - T_2| \div T_1) \times 100] \quad (12)$$

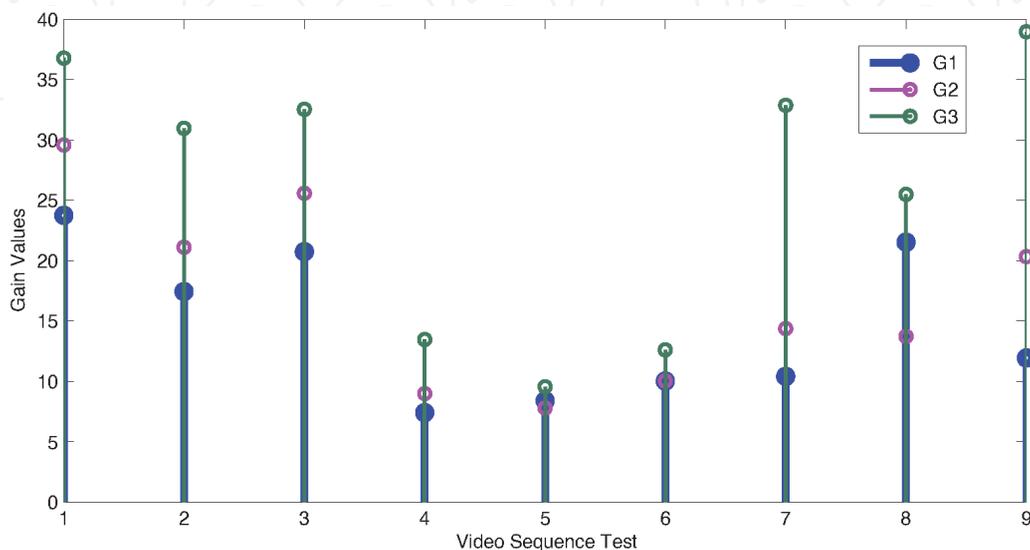
Also, our parameters for DAG and GGEN algorithms applied to ME blocks in codec video is very important if you compared with others approaches in scheduling tasks. In **Figure 13**, we give the formulations for scheduling dependent tasks into



**Figure 13.** Results execution time in SoC and MPSoCs targets in OVP with DAG algorithm.

homogeneous multiprocessor architectures of an arbitrary DAG and GGEN, taking into account communication delays. The time execution is in seconds.  $T_1$  presents the execution time for simulations with C/C++. Then  $T_2$  is the execution time for simulations with SoC platform in OVP.  $T_3$  and  $T_4$  present the execution times when scheduling and partitioning tasks with the MPSoC system in OVP.  $P_1$  is the platform Versatile ARM CortexA9MP\*4,  $P_2$  is the platform Ukernel arm Cortex A9MP\*4.

We observe our results in simulation for execution time are very more high compared with our results in co-simulation in Virtual Platform OVP, we can see the important minimization. **Figure 14** shows the results obtained for the gain of the entire test frame with the DAG algorithm. From **Figure 14**, we can see that using our technique reduces the execution time of the ME block. The experimental results illustrates the substantial enhancement of execution time. The scheduling and partitioning tasks algorithm DAG is give a true parallism with 4 processors in SoC and MPSoCs targets. For those comparisons, the metric value is the latency time of executions “t” for the SoC system. We selected SOC and tow platforms MPSoC (the models of MPSoC are: Platform including ARM Cortex-A9MPx4 to run ARM MPCore Sample Code and Versatile Express booting Linux on Cortex-A9MP Single, Dual and Quad Core in OVP). We remark that the execution time decreases when changing the platform, as in SoC and MPSoC systems. We deduce that the



**Figure 14.** The results of gain for the entire test frame with the DAG algorithm.

Sequences	F(sc)/Frame	F(sc)/Video
Akiyo	525533184	157659955200
Car(Avnet)[HD]	42998169600	2579890176000
Avion(Avnet)[HD]	42998169600	2149908480000
Forman	1751777280	525533184000
Miss-America	525533184	78829977600
Claire	525533184	259613392896
Mobile	1751777280	196199055360
Bus1	1751777280	175177728000
Skin1	1751777280	1576599522000
HD <sub>seq</sub> [9]	42998169600	1031956070400
HD <sub>seq</sub> [10]	42998169600	1031956070400

**Table 5.**  
 The results of the “F(sc)” function of the ME block for the different test sequences.

scheduling methodology has beneficial results for the execution times in OVP for the various test video sequences.

We show in this part the evaluation of the criteria of our application treated in our research work. We calculate the complexity of the H 264 video codec for ME block in the Eq. (14). **Table 5** present a results for the function F(sc) of ME blocks. The function (Fsc) is the function for calculating the complexity of the EM block; such as “(w, h)” frame size of the test sequence, “p” is the maximum authorized displacement, “N” is the size of MB processed, “h” is the factor and w is the width.

$$F_{sc}(Im) = \left( (h \times w)(2 \times p + 1)^2 \times N^2 \right) avecp = 4 \quad (13)$$

$$F_{sc}(Im) = \left[ \left( (h \times w)(2 \times p + 1)^2 \times N^2 \right) \right] \times (NF) \quad (14)$$

We notice that the sequences that we use in our research are very complex, so we need a very optimal, precise, generic and automatic approach. With this we get good results in all H 264/AVC video codec.

#### 4. Conclusion

In this paper, we presented a new scheduling and partitioning tasks algorithm DAG applied to ME for MPSoC platforms in OVP. The main contributions of our approach are the following ones: the semi-automatic scheduling and partitioning tasks, the performance with respect to granularity, the high quality, the accuracy and the short time execution for ME blocks in H 265 video codec. Stemming from complexity and profiling analysis, the DAG algorithm with ME blocks for three architectures platforms are presented in OVP (Soc and MPSoC system). The prototype SoC and MPSoCs system in OVP results highlighted that the processors have interesting performances, complexity, and execution times compared to other published solutions. This is visible in the tables and figures. The co-design HW/SW high level is also presented in SoC and MPSoC systems in OVP with IP of the DAG algorithm applied to ME blocks in H 265 video codec. Our scheduling and partitioning algorithm DAG is able to handle the execution time efficiently with

very limited resources, dropping the appropriate tasks in order to reduce the deadline time and the execution time in ME blocks. Our scheduling and partitioning algorithm DAG is fully semi-automatic to the characteristics of each application and it does not require any offline profiling data. The DAG model solution as a low-complexity applied to ME blocks for H 265 compared with other approaches. Besides, our results for the H 265 video codec have demonstrated that our proposed low-complexity solution for the general DAG model reduces the execution time by up to 70 %.

The execution times computed theoretically are almost the same that are found in OVP. We have also shown how our solution efficiently adapts with respect to the DAG type, and scales well with the number of cores and the number of deadlines considered in the buffer. The design of the bus and the interface of SoC and MPSoC platforms based on Arm Cortex A 9 MP is also described, allowing direct integration of the IP cores on-chip communication used in MPSoC for H 265 video codec. In a future work, we will try to minimize other metrics in the H 265 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 A 9 MP, which is composed of four processors and named Embest SABER Lite, Target from Development SABER Lite-i.MX 6 Quad.

## Acknowledgements

This research was financially supported by the “Laboratory of Technology for Smarts systems in Digital Research Center of Sfax “CRNS””, We would like to thank all the team for Pr. Ahmed FAKHFAKH, phd and master students, for his ambition and support in this project.

## Author details

Afef Salhi<sup>1\*</sup>, Fahmi Ghozzi<sup>1,2</sup> and Ahmed Fakhfakh<sup>1,2</sup>

1 Digital Research Center of SFAX (CRNS), Laboratory of Signals, Systems, Artificial Intelligence and Networks (SM@RTS), Sfax, Tunisia

2 ENET'COM, University of Sfax, Sfax, Tunisia

\*Address all correspondence to: [salhiafefge@gmail.com](mailto:salhiafefge@gmail.com); [afef.salhi.ge@enis.tn](mailto:afef.salhi.ge@enis.tn)

## IntechOpen

© 2021 The Author(s). Licensee IntechOpen. This chapter is distributed under the terms of the Creative Commons Attribution License (<http://creativecommons.org/licenses/by/3.0>), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited. 

## References

- [1] L. Rainer, S. Frank, M. Grant, K. Tim, P. Roman and V. Martin, *Virtual Platforms: Breaking New Grounds* DATE12/2012 EDAA.
- [2] B. Elias, S. Hassan and N. Smail, *H.264 Color Components Video Decoding Parallelization on Multi-Core Processors* 2010 13th euro-micro conference on digital system design: Architectures, Methods and Tools.
- [3] 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.
- [4] M.A. Kordon, *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.
- [5] 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.
- [6] E. Wajdi, D. Julien, M. Johel, A. Mohamed, *An efficient low-cost FPGA implementation of a configurable motion estimation for H.264 video coding* J Real-Time Image Proc, IF 2, n.10, 2012, pp.1-12.
- [7] W. Thomas, G. Sullivan, G. Bjntegaard, A. Luthra, *Overview of the H.264/AVC Video Coding Standard* IEEE Tran. on Circuits and systems for video tech, Vol 13, n.7, 2003, pp.560-576.
- [8] JVT and ITU-T, *Draft ITU-T recommendation and final draft international standard of joint video specification (ITU-T Rec. H 264-ISO/IEC 14496-10 AVC.)*
- [9] A. Salhi, F. Ghozzi, and A. fakhfakh, *Toward a Methodology for Object Tracking System in computer Vision* GAMMMART TUNISIA, TJASSST'2017, colloque Japonais-Tunisia, section "manegment and innovation", pp:185-189.
- [10] A. Salhi, F. Ghozzi, and A. fakhfakh, and M.A. Kordon, *Video Codec Applications Scheduling and Optimization Based on DAG and GGEN Algorithm* 2018 2nd European Conference on Electrical Engineering and Computer Science (EECS), Bern-Suisse, November-2018, 2018-IEEE, pp: 236-241.
- [11] A. Salhi, F. Ghozzi, and A. fakhfakh, *Scheduling Tasks in MPSoC System for Motion Estimation in H26x Video Codec* International Journal of Science and Engineering Investigations, Volume 7, Issue 74, April 2018,ISSN: 2251-8843, pp:72-77.
- [12] A. Salhi, F. Ghozzi, and A. fakhfakh, *Modeling and Scheduling with DAG used for tasks in ME with OVP*, International Conference on Smart Applications, Communications and Networking, SmartNets 2019, Sharm El Sheik, Egypt, December 17-19, 2019, IEEE 2019, ISBN 978-1-7281-4275-3, pp:1-6.
- [13] 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.
- [14] K. Kanoun, and M. van der SchaaK, *Big-Data Streaming Applications Scheduling with Online Learning and Concept Drift Detection*, 3rd ed. 978-3-9815370-4-8/DATE15/c 2015 EDAA, DATE15/IEEE, pp: 1547-1550.
- [15] K. Kanoun, M. Nicholas, A. David, and M. van der Schaar, *Online Energy-Efficient Task-Graph Scheduling for Multicore Platforms*, 3rd ed. IEEE TRANSACTIONS ON COMPUTER-AIDED DESIGN OF INTEGRATED CIRCUITS AND

SYSTEMS, VOL. 33, NO. 8, AUGUST  
2014, pp:1194–1204.

[16] R. Ben Atittallah, S. Niar, A. Grainier, *Estimating Energy Consumption for an MPSoC Architectural Exploration*, 3rd ed. International Conference on Architecture of Computing Systems, 298-310, 2011.

[17] M. Ruggiero, D. Bertozzi, L. Benini, M. Milano, and A. Andrei, *Reducing the abstraction and optimality gaps in the allocation and scheduling for variable voltage/frequency MPSoC platforms*, 3rd ed. IEEE Transactions Computer-Aided Design Integr. Circuits Syst, vol. 28, no. 3, pp. 378391, Mar. 2009.

[18] J. Luo and N. K. Jha, *Power-efficient scheduling for heterogeneous distributed real-time embedded systems*, 3rd ed. IEEE Trans. Computer-Aided Design Integr. Circuits Syst, vol. 26, no. 6, pp. 11611170, Jun. 2007.

[19] S. Heiko, D. Marpe and W. Thomas, *Overview of the Scalable Video Coding Extension of the H.264/AVC Standard*. IEEE Transactions On Circuits And Systems For Video Technology, VOL. 17, NO. 9, 2007.

[20] J. Rainer G.J. Sullivain, *High efficiency video coding: The next frontier in video compression*. IEEE Signal Processing Magazine, 2013.

[21] 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.

[22] B. Sugandi, 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.

[23] Open Virtual Platform and Imperas, <http://www.ovpworld.org> Imperas-OVP, 2008.