International Journal of
Science and Engineering Investigations vol. 7, issue 74, April 2018
ISSN: 2251-8843

Scheduling Tasks in MPSoC System for Motion Estimation in H26x Video Codec

Afef Salhi[1], Fahmi Ghozzi[2], Ahmed Fakhfakh[3]
[1,2,3]LT2S Laboratory, Digital Research Center of Sfax, Tunisia, B.P. 275, Sakiet Ezzit, 3021 Sfax, Tunisia
([1]salhiafefge@gmail.com, [2]fahmi.ghozzi@enetcom.usf.tn, [3]ahmed.fakhfakh@enetcom.usf.tn)

Abstract- Multi-core processors are becoming more and more popular in embedded and real-time systems, while fixed-priority scheduling tasks in real-time systems are widely applied. Numerous Directed-Acyclic Graph (DAG) scheduling tasks have been developed to improve the execution time of various multi-core systems. The problem of scheduling tasks applications on multi-core chips is notoriously difficult. This difficulty is compounded with the fact that the graphs describing the applications of interest are becoming very large. This paper develops a scheduling tasks method for the motion estimation of codec video H26x. The application graphs considered in this paper are acyclic directed graph and the architectures are multi-processors. The key idea is to apply a ME block for H26x video codec with DAG algorithm. This permits a formulation of the scheduling tasks problem as an Motion estimation block whose objective is the minimization of execution time and whose real time is fast in the application and architecture.

Keywords- *OVP, DAG-TPG, Motion Estimation, H26x, Video Codec, Scheduling and Partitioning Tasks, SoC and MPSoC*

I. INTRODUCTION

Nowadays, we see the emerging real-time video and frame processing application such as video timing data, video search, and streaming multimedia and telecommunication (e.g HEVC and H264). H26x is the newest video coding software, compared to previous H265, H264 and MPEGxx standards, H26x superior compression efficiency and high scalability make it suitable for different scenarios. Numerous Hardware and Software (HW/SW) systems have been proposed in these applications. In the short term, it is expected that system-on-chip designs, with Multiple processors and assorted hardware on a chip, will match the immediate needs of H.264/AVC or H265/HEVC codec HD [1], [3], [4]. We will explore in this paper the best architecture capable of executing this algorithm in real time using Virtual Prototype "VP" based on ARM cortex A9. This architecture are defined in Open Virtual Platform [6] [7]. OVP enables simulating embedded systems running real application code. Digital video is suited to many applications and numerous dissemination media: digital TV, video conferencing, satellite TV, video telephony and digital cameras. The problem is how to transmit video streaming on a network where we store them in a good quality and low throughput. The solution is to reduce the size of data to be transmitted to the *storage.* Hence, we require a prior compression of the files. Using new approaches has become necessary. OVP represents a new approach for prototyping MPSoC systems. It contains an assembly of reused components. We are interested in this work to methods of design/validation based on simulation at high level. So, the verification and validation time can be relatively very short. An embedded system can be complex, for example in multimedia and telecommunication application. The solution consists of implementing VP to meet the needs. We have many constraints and the choice of an architecture must be a compromise between: flexibility, reduced consumption, performance, low cost and Time-To- Market (TTM). The TTM is very important constraint. A rigorous design methodology should be used. Hence, we are interested in this works to implement the DAG applied to ME for H26x video coding with MPSoC systems. We can model it as DAG scheduler. We define the scheduling and partitioning tasks, a DAG monitoring and tracking solution is used to process and analyze the DAG code for scheduling tasks. This code is semi-automatic and statistic. The code gives a model from scheduler tasks and it is responsible for finding the parallelization tasks in MPSoC systems. The task is partitioned with parity tasks in quadripole architecture. Numerous scheduling tasks in [2], [5] have been proposed from the scheduling and partitioning tasks for the multimedia and telecoms applications. Despite the improvements in multi-media device technology, energy-efficient multi-core scheduling tasks in ME block for video coding remains a challenging problem for several reasons. First, the ME block in video coding application has an intense and time-consuming workload, which has execution times that are significantly larger than the average case. Secondly, they have sophisticated dependency structures due to ME blocks. Then, we can improve the granularity levels in TPG graph. In this paper, we purpose a novel semi-automatic scheduling tasks solution for ME block in H26x video codec.

The paper is structured as follows: section 2 provides the DAG algorithm. Section 3 details the motion estimation algorithm in H26x video codec. Section 4 presents the adaptation od DAG algorithm with ME block. Then, Section 5

describes the results scheduling and partitioning tasks for ME block of H26x video codec. In Section 6 we discuss the speedup results of DAG algorithm with ME block. Section 7 concludes our findings.

II. DAG ALGORITHM

It is the method which guaranteed the equity of the application. it is designed to authorize the scheduling tasks in heterogeneous platforms. It is the method which consists in authorizing the parallel and sequential tasks in a heterogeneous system SoC and MPSoC. The algorithm of DAG bases itself on an acyclic graph directed by task which serves to make the scheduling and the partitioning task of the treated application. In general, a Task Precedence Graph (TPG) G(V;E;W;C), is an important graph in DAG algorithm, where V is the set of nodes, E is the set of edges, W is the computational cost of nodes and C is the cost of communication directed edges. A node in the DAG is a task which is in turn a set of instructions to be executed sequentially without preemption in the same processor. Many proofs and theorems useful to multimedia applications are studied and defined in [3], [6]. The flowchart for DAG algorithm is shows in figure 1.

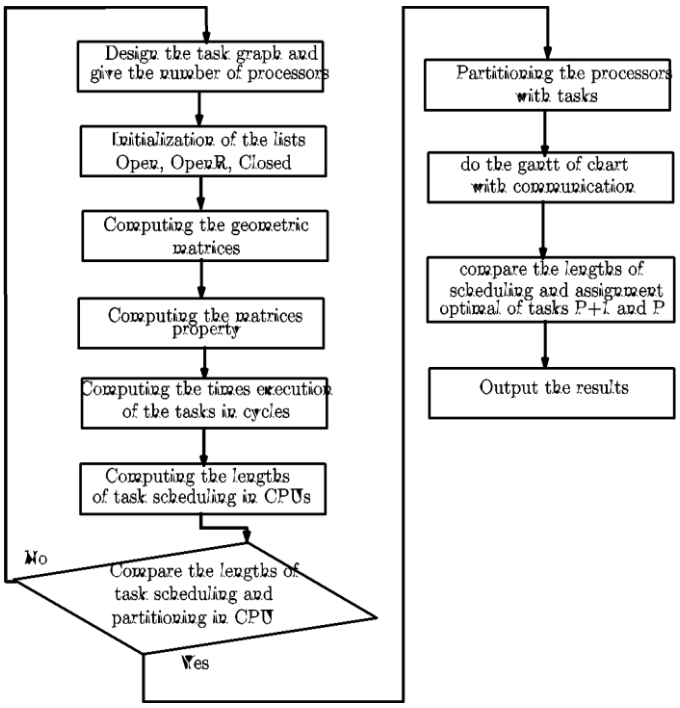

Figure 1. Flowchart for DAG algorithm

III. MOTION ESTIMATION ALGORITHM

The algorithm of motion estimation "ME" for the H26x video codec consists of several modules and under module which are programmed in C/C ++. This algorithm is based on the older algorithm for a long time named Block Matching. This approach is based on a sweeping of all the frame to find the best Vector Motion “VM” and the Sum of Absolute Differences SAD. Every module of ME's block defines a set of function. The selection of the best MV and SAD in the first frame of the sequence was made via a stage and a loop of check of parameters with eclipse C/C ++. A module of ME algorithm consists of several under modules and modules. This module allows to select the exact position of the motion, such as the best motion vector and the SAD of the object in the two first frames, afterward in all the video sequence. It consists of the extraction of the parameters of initialization of the module of ME who are the position, the size, the width, the MB, the MV, the SAD and the length of the window search for the object in the two first frames of the sequence, as well as the other necessary parameters for ME block. The flowchart for ME block is presented in figure 2.

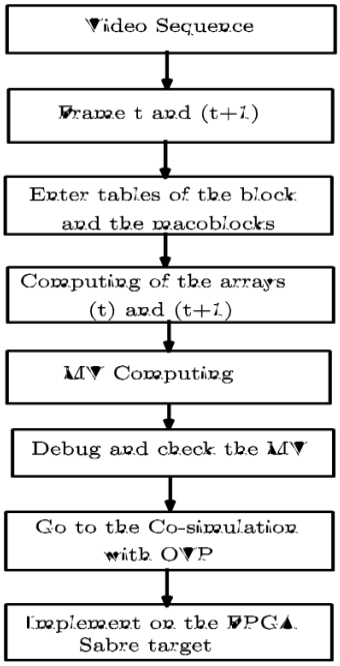

Figure 2. Flowchart for ME algorithm.

IV. ADAPTATION OF DAG ALGORITHM WITH ME BLOCK

In this part, we detail the methodology of the DAG approach that we proposed, with this method and how we adapted the approach with our needs for a generic image (size $N*M). It is for it, and as a first originality of our works, we chose to realize the approach for any size of frame. As a second originality, the method of tasks scheduling is applied for ME block of the standard of video codec H26x. It is a scheduling by parity (even even, even odd, odd even, even even). For X=N, Y=M, X for pixels lines of the generic frame and Y for pixels column. Every time, we have to make the method of interpolation, it is a question of adding lines and columns for embellish with frames, thus to add empty pixels and full {0,255}. We chose this technique to eliminate the problems of the border of frame. After modelling of various sequences of test, we noticed that N has to take a value odd and divisielle by 4. Furthermore, Y must be equal in X. The graph of task TPG-DAG presents the various nodes with the various levels and the paths. The tasks of the graph are illustrated in a geometrical partitioning in paths and levels. The paths are sorted in the order of parity for the tasks and the number of processors. We have three models of video test sequence {qcif, sif and cif, HD} = {8, 10, 16}. In the algorithm DAG, we read the video sequence. Every time, we take the reference frame and the current frame, we search for the best MV "Motion Vector". The scheduling and the partitioning are created for two successive images. We look for the best SAD then for the best VM. This task is made, every time on two successive images (a reference frame and a current frame). This method eliminates numerous problems. The algorithm DAG is applied for the entire video sequences test, we obtain three models: {1, 2, 3} which have {8, 10, 16} for the levels and the paths. The methodology of tasks scheduling and partitioning with DAG for ME block is presented in figure 3.

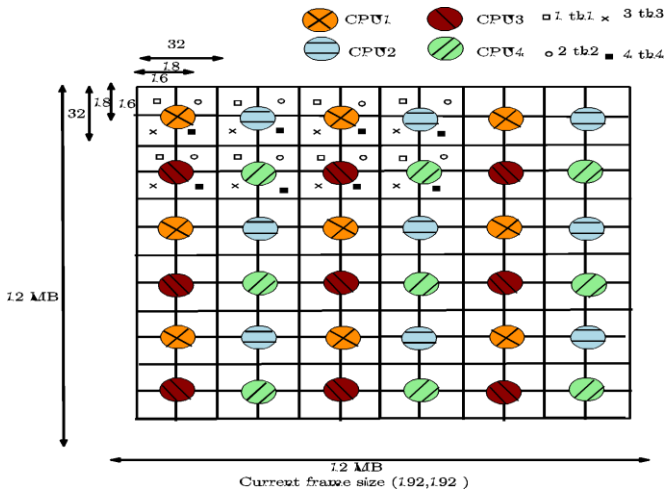

Figure 3. Methodology of tasks scheduling with DAG algorithm

The sequence tasks from the ME block in CPU are defined in figure 4. These sequences represent the gantt of chart for tasks sequences by CPU.

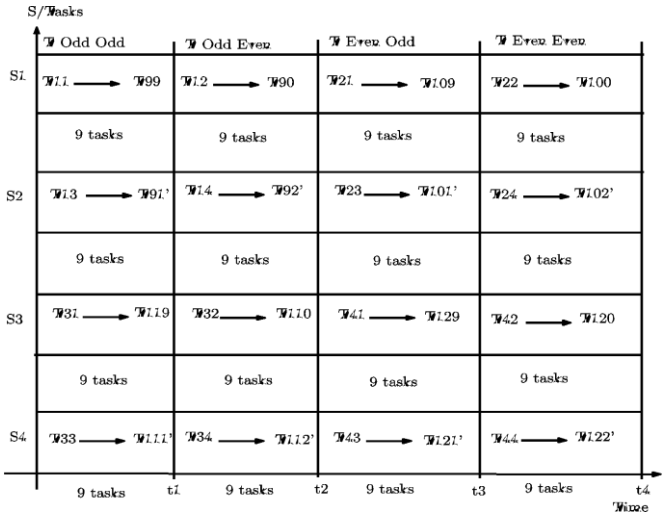

Figure 4. The graph gantt of chart for tasks sequences by CPU.

V. THE SCHEDULING AND PARTITIONING TASKS ALGORITHM "DAG" FOR THE ME BLOCKS

The DAG algorithm is modeled with three models in video codec H26x. Each one has a different frame size: model 1 (176*144), model 2 (352*240), model 3 (1920*1080). The graph of this DAG algorithm is show in figure 5.

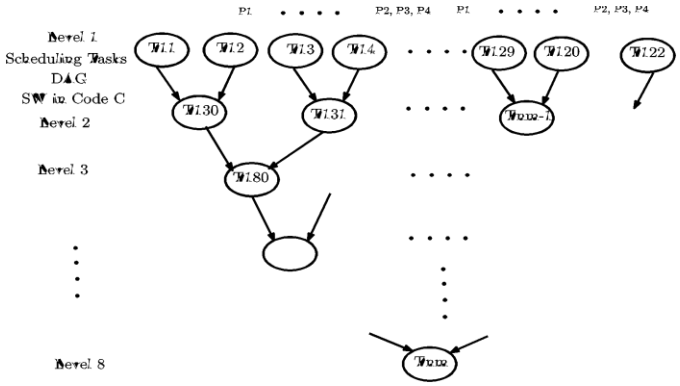

Figure 5. The DAG task scheduling ME block.

VI. RESULTS OF SCHEDULING AND PARTITIONING TASKS ALGORITHM DAG FOR ME BLOCK

In table 1, we present the modelling of the scheduling and partitioning algorithm DAG.

TABLE I. THE MODELING SEQUENCES VIDEO TEST IN DAG ALGORITHM.

Sequence	Format	Nbr frame	Nbr tasks/F
Akiyo-qcif	qcif(176*144)→(192*192)	300	144
Miss-amcif	qcif(176*144)→(192*192)	150	144
Forman	qcif(352*288)→(384*384)	300	576
Mobile	sif(352*240) →(384*384)	112	576
Bus1	Cif(352*240)→(384*384)	100	576
skin1	sif(352*240)→(384*384)	900	576
claire	qcif(176*144)→(192*192)	494	144
Seq-HD[1]	(1080*1920)→(2048*2048	24	16384
Seq-HD[2]	(1080*1920)→(2048*2048	24	16384

Then, we present the results of execution times from ME block with DAG algorithm in table 2.

TABLE II. THE RESULTS OF EXECUTION TIMES FROM ME BLOCK WITH DAG ALGORITHM.

Sequence	Tt(cyc)	Tex Fr(cyc)	Tex Seq (cyc)	Tex seq (s)
Akiyo-qcif	25	3600	1080000	7.2
Miss-amcif	25	3600	540000	5.4
Forman	25	14400	4320000	8.8
Mobile	25	14400	1612800	10.75
Bus1	25	14400	1440000	9.6
skin1	25	14400	12960000	8.64
claire	25	3600	1778400	11.85
Seq-HD[1]	25	409600	9830400	7.9
Seq-HD[2]	25	409600	9830400	7.9

We notice that the theoretical execution time is closer than the execution time we found in co-simulations in OVP. Then we can say that our algorithm "DAG-TPG" is optimal in the approximation formalism and modeling.

A. *Comparaison with results in the literature*

We chose to make a comparison of the results of a set of DAG generated randomly by size going from 5 to 100 tasks, among which each generates a random number of deadlines of communication for the graph of the DAG-TPG. We compared our results with other statistical algorithms of scheduling "((LMBC) Load Balancing with Minimized Communications, (LPTDC) varying Largest Processing Time of DeClustering, (DC) DeClustering)" in the results presented in [2], [5]. The code was tested at the level of tasks few imports the application. We compare simply our works with other statistical approaches of scheduling with the optimal algorithm at the level of tasks.

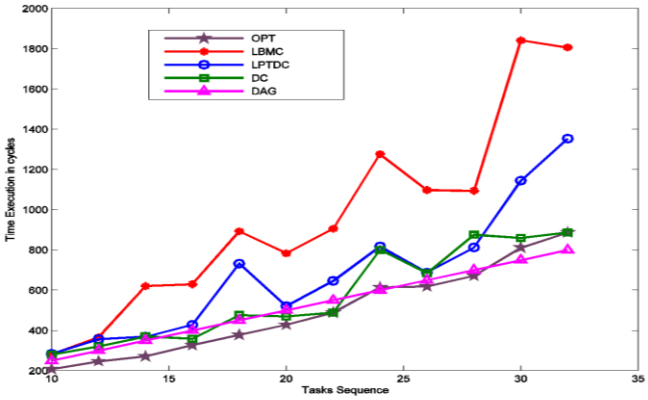

Figure 6. Comparative results for our approach with other algorithm in scheduling and partitioning tasks.

We can observe the results for the DAG algorithm, they are better than another algorithm that scheduling and partitioning tasks.

VII. RESULTS OF PROTOTYPE WITH OVP:

The main metric is execution time. In table 3, the formulations for the scheduling of dependent tasks on a homogeneous multi-processor architecture of a DAG structure in order taking into account the communication delays are proposed. The modeling of the DAG algorithm is used in the ME for H26x video codec, to reduce the execution time required by ME in the H26x video codec to optimality. Table III summarizes the different execution time results for the different platform architectures that we used during our co-simulations with OVP.

TABLE III. THE RESULTS OF EXECUTION TIMES FOR THE DIFFERENT ARCHITECTURES USED UNDER OVP.

Sequence	T1(C/C++)	T2(OVP/SoC)	T2(OVP/P1)	T3(OVP/P2)
Akiyo-qcif	10.1	9.5	8.8	7.9
Miss-amcif	8.6	7.1	5.6	4.9
slamen	10.6	8.4	6.25	5.8
Forman	9.2	8.0	7.45	6.9
Mobile	29.7	27.2	25.08	24.6
Bus1	16.3	14.6	12.5	9.8
skin1	60.8	54.7	49.2	47.8
claire	45.8	42.3	38.5	36.6
Seq-HD[1]	6.5	5.1	4.4	3.6
Seq-HD[2]	6.7	5.9	4.7	3.8

The results that we found in our co-simulations show that the DAG algorithm is optimal compared to the results of existing work without scheduling. We can deduce that the step of scheduling tasks for any block in the H26x video codec chain is very important. This step is complex, takes time to achieve but gives us good results in virtual prototyping. T1 is execution time for one processor, T2 is the execution time for many processor. The gain of application is computed in equation 1.

$$Gain = [(|T1 - T2| \div T1) \times 100 \quad (1)$$

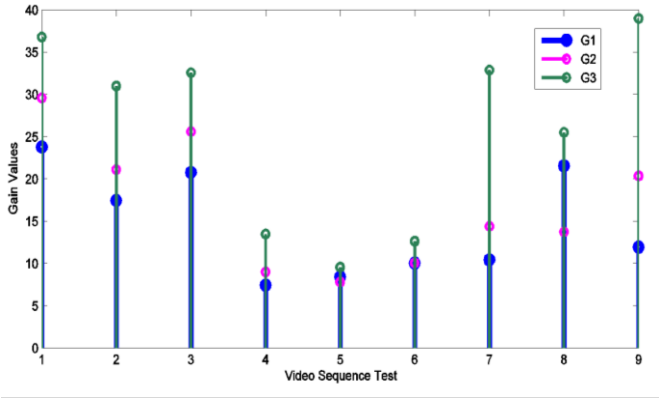

Figure 7. The results of gain for the video sequence test with the DAG algorithm.

The time execution is in seconds. T1 presents the execution time for simulations with C/C++. Then T2 is the execution time for simulations with SoC platform in OVP. T3 and T4 present the execution times when scheduling and partitioning tasks with the MPSoC system in OVP. We can see that the execution time of DAG algorithm applied to ME block with two MPSoC platforms in OVP decreases when compared the execution time in C/C++ simulation and in OVP co-simulation with SoC platform. The figure 6 present the results of gain of video sequence test with DAG algorithm. From the figure, we can deduce the results with DAG algorithm reduces the execution time of ME block. The results of execution time are varied between 8% and 40%. We choose SoC and tow MPSoC platforms in co-simulation (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 observe that the scheduling methodology has beneficial results for the execution times in OVP for the various test video sequences.

VIII. CONCLUSION

In this paper, we have proposed a new methodology of scheduling and partitioning tasks algorithm DAG applied to ME block of H26x video codec. The key contributions of this work were as follows: The adaptability of semi-automatic scheduling and partitioning task for H26x video codec, the performance of scheduling task with respect granularity, the accuracy and the short time execution for our application. We achieved satisfactory performance virtual prototyping under OVP of the task scheduling and partitioning approach applied to the H26x video codec ME block. As well as performing a task scheduling semi-automatically, robust and fast in real time (20 frames per second). The DAG model solution for ME block as a low-complexity compared with other algorithms of scheduling tasks. This solution reduces the execution time by up to 40%.

In future work, we would like to increase the performance of the task scheduling DAG algorithm, we decided to work with the GGEN algorithm to make scheduling automatic. We will also implement in a real target.

ACKNOWLEDGMENT

We thank Alix Munier Kordon for the Task Scheduling Algorithm and Ahmed Fakhfakh and Fahmi Ghozzi for inspiring discussions.

REFERENCES

[1] S. Rethinagiri, B.A. Rabie, N. Smaiel, E. Senn, and J. Dekeyser, “Fast and accurate hybrid power estimation methodology for embedded systems,” In Design and Architectures for Signal and Image Processing (DAS1P), vol. 10, pp. 1-7, 2011.

[2] Ch. Philipe, G.C. Edward, K.L. Jr Jan, and L. Zhen, “Scheduling Theory and its Applications”, 3rd ed., by Jhon Wiley ans Sons Ltd, England, 1995.

[3] K.. Li, K. Jia, J. Xie, and J. Wang, “Design and Optimization of H.264 Video Encoder on DSP Platform,” IEEE, vol. III, 2011, pp. 1-4.

[4] S. Nuno, N. Tiago, R. Nuno, F. Paulo, and S. Leonel, “Application Specific Programmable IP Core for Motion Estimation: Technology Comparison Targeting Efficient Embedded Co-Processing Units,” 11th EUROMICRO CONFERENCE on DIGITAL SYSTEM DESIGN Architectures, Methods and Tools, vol. 6, pp. 182-189, 2008.

[5] K. Atasu, “HARDWARE/SOFTWARE PARTITIONING FOR CUSTOM INSTRUCTION PROCESSORS,” Svizzera, italiana, 2002.

[6] OVP,http://www.ovpworld.org , 2008-2017.

[7] M. Marios, G. Jordi, and B. Mark, “Mixed Simulation Kernels for High Performance Virtual Platform,” Forum on Specification Design Languages, pp. 1-6, 2009.

[8] S.G. Maxiwell, S. Marcelo, and S.O. Marcio,” VIPRO-MP: a Virtual Prototype for Multiprocessor Architectures Based on the SimpleScalar”, 17th IFIP International Conference on Very Large Scale Integration (VLSI-SoC), vol. 10, pp. 41-46, 2009.

Afef 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'm currently working toward the Ph.D. degree in the Laboratory of Technology for Smart Systems (L.T.2.S). My research interests include implementation codesign in very high level MPSoCs for real time application (image processing and video applied in multimedia application).

Fahmi Ghozzi: received his PHD degree in electronics from IXL laboratory, university, Bordeaux1, France, in 2004. He is member in the Laboratory of Technology for Smart Systems (LT2S). 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.

Professor Ahmed Fakhfakh received 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.